CHESSBD Help Index

CHESSBD Quick Introduction

Check it out! Use the File|Open command and select the file "demofile.txt" provided in the CHESSBD distribution. You could instead try any other ascii text file containing chess games.

After the program has scanned the file, you can start playing through the first game with the +Move button. Or you can select a different game with the Select Game button. See the GUI Usage help topic for more information.

If you have a big file with hundreds of games, it can take a long time to open... so i recommend that you create an "index" file with the File|Create index dialog. Then the next time you want to look at those games, you can "open" the index file instead. It'll be a lot faster than opening the original games file again.

If you don't like the looks of the CHESSBD user interface, you can change the fonts and colors using the Options dialog. You can also control various CHESSBD operational features.

If you are using CHESSBD as a graphical client for an Internet Chess Server (ICS), then see the section ICS Mode and the section ICS Tutorial .

Good luck!

CHESSBD GUI Usage

The CHESSBD display provides 3 interaction areas: the board display, the game tree navigation dialog, and the textport; plus the menu bar .

Board Display

The board display is drawing of a chessboard and pieces. Initially the pieces are set up in the standard starting position. You can move the pieces around by dragging them with the left mouse button. When you select a piece by holding down the left mouse button, all its legal destination squares are briefly highlighted. (This feature is controlled by the `flashmoves` option variable.) Illegal moves are not permitted.

If you are playing a "wild" game on the chess server, you can make a "drop" move by clicking the right mouse button on a vacant square. This calls up a dialog that lets you select the piece to be dropped. (This is an experimental feature.)

You can let the program make the moves for you as you play through a games file, using the navigation dialog. Even if you're in the middle of playing through a game from a file, you can mouse the pieces around to see what would have happened if a different move had been made, then return to the real game using the -Move button in the navigation dialog . Any moves you make with the mouse are added to the current line of the current game tree.

Textport

The textport, at the upper right quadrant of the CHESSBD window, displays status messages and provides a command-line interface to the program. The textport will accept typed in commands or chess moves. This interface is mostly redundant since all major program features are available via the GUI instead. (See the Script Commands and Option Variables help topics for more information.)

You can scroll the textport vertically with the scrollbar. In addition, you can "auto-scroll" the textport horizontally if there are any lines wider than the textport width. If you want wide lines to wrap around automatically instead of having to scroll to see them, you can type Alt+M to pop up the TtyMode dialog.

Navigation Dialog

The navigation dialog is a set of controls at the lower right of the CHESSBD window. You can use these controls to move through the game tree one step at a time; go backward or forward several moves; or explore variations. To begin with, you are positioned at the root of the tree, corresponding to White's first move. As you move the current node, the board position display, the move number display, and the variation display to the left are updated accordingly. The most useful controls are the +Move button and the arrow buttons on the ends of the scrollbar.

The following controls are provided.

- Scrollbar
- +Move and -Move buttons
- +Var and -Var buttons
- << Mark and >> buttons
- Move list box
- +Game and -Game buttons
- Select Game button

Menu Bar

The menu bar provides:

- **File Sub-menu**

- Open dialog
- Add dialog
- Save index dialog
- Close
- Save game dialog
- Edit game dialog
- Save all dialog
- Run dialog
- Log open/close
- Exit

Commands Sub-menu

- New Game
- Reverse
- Setup dialog
- Varboard .

Options dialog

ICS Sub-menu

- Manual Login
- ...profiles...

User sub-menu

Help sub-menu

Navigation Scrollbar
The scrollbar control: move the current node forward or backward through the tree of moves, *in the order in which the moves were found in the file.* Clicking on the ends of the scrollbar is the easiest way to walk through the current game.

Navigation +Move and -Move buttons
The +Move and -Move controls: move the current node forward or backward *through the main line* of the current branch of the tree, or *along the same path* if the branch has already been explored.

Navigation +Var and -Var buttons
The +Var and -Var controls: move the current node inward to the next branchpoint, or outward to previous branchpoint.

Navigation move list box
The move list box displays all branches (moves) available at the current node. The first entry is always the main line (if any). You can move the current node to another branch by clicking the corresponding item in the move list box. If one of the items is already highlighted, you can select that move by clicking the +Move button.

<< Mark and >> buttons
You can "mark" the current position in the current game tree with the Mark button, and return to it later using the "<<" button. This feature may be useful for examining and re-examining key positions while playing thru a game. If you create multiple marks, you can cycle thru them with the "<<" and ">>" buttons. The "<<" button goes to the "previous" marked position, and the ">>" button goes to the "next" marked position. NOTE, when you use the "<<" or ">>" buttons to go to a marked position, the current position is also marked so you can return to it easily.

Navigation +Game and -Game buttons
For convenience, the traversal dialog contains game selection controls. The +Game and -Game buttons switch to the next or previous game in the current games list.

Navigation Select Game button
The Select Game button calls up a dialog that lets you select any game in the current games list. When you select a game, the program parses the game and builds a tree of all moves, comments, and variations in the game. This becomes the current game and game tree until you choose another one.

I just added an experimental "search" feature to this dialog. You can search for a game title (substring), with the +Search or -Search buttons. Type the substring in the small text window between the +Search and -Search buttons. If you want the search to be case-sensitive, check the "c.sens" box (default is case-insensitive). If you want to do a regular expression search, check the "r.expr" box (default is plain string matching). (I don't know exactly what kind r.e. patterns are supported; it's whatever is in the borland String class.) Then click the +Search button to search forward from the current position, or -Search to search backward.

Click OK to select the currently selected item, or Cancel to abort the dialog.

CHESSBD File Sub-menu

File|Open
Read a file of games and replace the current games list. Each game in the file consists of a header followed by a body (containing move text). When you open a games file, the entire file is scanned to determine the number and location of all game headers in the file. All the games are added to the current games list until you close (File|Close) the list. The File|Open command is identical to the File|Add command except that the former clears the current games list beforehand, and the latter doesn't.

File|Add
Read a file of games and add to the current games list. (See also File|Open and File|Close .)

File|Create index
Create an index file for the current games list. The index file can be opened (with File|Open) instead of the original game file(s), next time you want to look at the same set of games. Opening the index file is faster because it is smaller. You still have to keep around the original games file(s), though. Do not try to give the index the same filename and extension as the original games file! At least make the extension different.

File|Close
Clear the current games list.

File|Edit game
I just added an experimental feature to edit the current game. This writes the current game to a temporary file and calls up a dialog with an edit control so you can make little changes. Click "OK" to accept the changes (writing them to the temp file). Click "Cancel" to abort the edit. There's a checkbox for "rewriting". If you check this box, any changes will be written back to the original games file, and the temp file will be deleted. If you don't check the box, the original games file will be left unchanged, and the temp file will not be deleted. In this case, it is your responsibility to deal with the temp file.
Note, this feature does NOT currently permit adding or deleting games; it only makes it easier to correct mistakes within a game. To add or delete games, use your favorite text editor and then re-open the file.

File|Save game
Calls up a dialog to save the score of the current game to a file. The score is written out in a cleaned-up or "canonical" form. Among other things: every White move is numbered; Black moves are not numbered unless necessary; all implicit commentary and variations recognized by the program's heuristics are written out as explicit comments and variations. Variations are positioned just after after the corresponding main line move.

File|Save all
I just added an experimental "save all" feature. Calls up a dialog to save all games in the games list to a file. Like File|Save game except it saves all games, not just the current game. This has a side-effect of changing the current game to the last game in the list.

File|Run
Dialog to run a script file. A script file can contain commands or chess moves. (See the Script Commands help topic for information.)

You can run a script file (or a game file, for that matter, although it is preferable to use File|Open to read game files) using the Run feature. The program will read and execute the contents of the file.

NOTE: Script files and game files are almost the same. You can "Open" a script file or "Run" a game file. But there is a difference of intention. The "Open" feature is intended for files containing multiple games, possibly with a few commands to help the parser. The "Run" feature is intended for files containing mostly

commands, or at most one game. You can nest script files, but not game files. The File|Run command does *not* add games to the current games list.

File|Logging
Opens/closes the log file. When the log file is open, all textport messages are copied to the log file.

File|Save layout
Saves the current size and position of the main board window. If the ICS window is currently available, then the size and position of the ICS window and dialogs are also saved.

File|Exit
Quits the program.

CHESSBD Commands Sub-menu

Commands|New Game
Starts a new game tree at move 1; resets the board to the initial position; and resets the game title and player name displays.

Commands|Reverse
Flips the orientation of the board display.

Commands|Setup
Calls up a dialog that lets you set up an arbitrary position on the board display. In setup mode, you can move a piece to any square by dragging with the left mouse button. Remove a piece by clicking the right mouse button on it. Add a piece by clicking the right mouse button on a vacant square, then select a piece type from the dialog. You can also change the current game state, consisting of castling flags, en passant column (a-h), move number and to-move flag, through the Setup dialog. Click the OK button in the setup dialog box to exit setup mode and accept the currently setup position; click Cancel to exit setup mode and revert to the position and game state from before setup mode was entered.

Commands|Varboard
Creates a new "variation" board window, setup to the current position. You can move the pieces around in this window without affecting the main board window. Note that although the variation board window does not have a command menu, still all the accelerators can be used on it. (Alt+1 to reset the board to the initial position, Alt+B to flip the board, Alt+S to setup, and Alt+V to create another variation board window.)

CHESSBD Options dialog

A dialog that lets you customize the appearance and operation of CHESSBD. The FONTS listbox, at the lower left, shows what fonts you can change. Double-clicking on a FONTS item calls up a font change dialog. The COLORS listbox, at the lower right, shows what colors you can change. Double-clicking on a COLORS item calls up a color change dialog. The variables window shows other variables you can change. If you want to change one, just click on its value box and type in a new value. See the Option Variables help topic for information on what the variables do.

After you've made all the changes you care to make, click OK to accept the changes for now. Click Save if you want to make the changes permanent. (saved to the initialization file **`chessbd.ini`** .) If you decide not to make the changes, click the Cancel button.

CHESSBD The User Sub-menu

This is a menu of up to 50 user-defined commands. The commands are numbered from 00 to 49. The value of the option variable **`lm00`** specifies the label and action for the user menu item *00*. The value is a string consisting of the label, followed by a vertical bar ("|"), followed by the action. CHESSBD does variable expansion on the action before executing it. See the %eval command for a description of variable expansion. For example,

lm07=OPENIT|%%open myfile.pgn

creates menu item number 7 with a menu label "OPENIT" and action "%open myfile.gpn". NOTE: you must number your menu items consecutively starting from 00, with no gaps.

You can put these definitions in your **`chessbd.ini`** file. But i strongly recommend you put them in a different file and then change your program manager command line for CHESSBD to have that file as an argument. NOTE: If you do put your definitions in a different file, it must have the extension ".ini". ALSO NOTE: line 1 of the file must be **`[chessbd]`** .

CHESSBD Help Sub-menu

The Help sub-menu consists of:

Help|Index
Displays CHESSBD's windows help file. (You're soaking in it.)

Help|Using help
Displays windows help on help file.

Help|About Chessbd
Displays information about CHESSBD.

CHESSBD Tty Mode dialog

The Tty Mode dialog allows you to set various internal parameters of the textport. The dialog pops up when you type Alt+M in the textport. Some of the settings only apply to the ICS Telnet window, and can be ignored when applied to other textports.

Raw Output checkbox
Check the Raw Output box to *disable* the suppression of style12 messages in ICS mode. Normally you want this to be OFF (unchecked) while in ICS mode. You may wish to turn it ON temporarily if you login thru a firewall. This option applies only to the ICS Telnet window. #

Full Duplex checkbox
Check the Full Duplex box to *enable* full duplex operation while in ICS mode. Normally you want this to be OFF (unchecked) while in ICS mode. You may wish to turn it ON temporarily if you login thru a firewall. This option applies only to the ICS Telnet window. #

Local Echo checkbox
Check the Local Echo box to *enable* local echo while in ICS mode. Normally you want this to be ON (checked) while in ICS mode. You may wish to turn it OFF temporarily if you login thru a firewall. This option applies only to the ICS Telnet window. #

Line Wrap checkbox
Check the Line Wrap box to *enable* automatic wraparound of long output lines. #

UX button
This button is provided as a convenience to set various parameters to ``UNIX'' compatible settings. It sets Raw Output ON, Full Duplex ON, Local Echo OFF. #

LCL button
This button is provided as a convenience to set various parameters to ``LOCAL'' compatible (i.e., ICS-compatible) settings. It sets Raw Output OFF, Full Duplex OFF, Local Echo ON. #

OK button
Click the OK button to accept the displayed parameter settings. #

Cancel button
Click the Cancel button to dismiss the Tty Mode dialog without changing the actual parameter settings.

CHESSBD Input File Syntax

1. INTRODUCTION

The CHESSBD parser attempts to handle a wide range of input styles. It would be difficult, and not necessarily useful, to describe exactly the syntax used. This document attempts to at least list its basic elements, and give examples to illustrate (in an admittedly imprecise way) the scope of the language. This document describes the syntax of "clean" input. But note that the parser also attempts to handle "dirty" input, using a variety of heuristics. See the Parser heuristics help topic for more information.

2. CONCEPTUAL MODEL

The following grammar illustrates the program's basic concepts and assumptions about input structure. It must be stressed that this grammar is only a conceptual model, and the program deviates from it in order to accommodate a wider variety of sloppy input styles. Details have been suppressed in favor of clarity. NOTE: in reality the parser is implemented with ad hoc code, not based on this or any other CF grammar.

2.1. THE FILE MODEL

Input files are ascii, line-based files. There are 2 types of input files: game files and script files. The difference between game files vs script files is mainly one of intended use, rather than content. Game files contain one or more games, and are intended to be "opened". Script files contain commands, and are intended to be "run". However, in reality game files can contain commands and script files can contain games.
Files are "line-based": the gross elements of a file consist of a whole number of adjacent lines, although "move elements" may be split across lines. For simplicity, the grammar (which is anyway only approximate) does not reflect these constraints.

```
InputFile ::== GameFile | ScriptFile
GameFile ::= Game*
Game ::== GameHeader GameBody
```

2.2. GAME MODEL

A game consists of a header and a body. The game header can be explicit or implicit. The game body is essentially a series of moves.

```
GameHeader ::= ExplicitGameHeader | ImplicitGameHeader
ExplicitGameHeader ::== PGNHeader | GameCmdHeader | InitGameHeader
```

2.2.1. GAME HEADER MODEL

A game header can be explicit or implicit. There are 3 basic kinds of explicit game headers: PGN, %game, and %init.

A PGN header is a series of PGN taglines. A PGN tagline contains one or more PGN tagpairs. See the PGN Standard document for details.

```
PGNHeader ::== PGNLine*
PGNLine ::== PGNPair +
PGNPair ::== '[' #TAGNAME# '"' #VALUE# '"' ']'
```

Example:

```
[Event "Mar del Plata 1960"]
[White "Spassky,B"]
[Black "Fischer,R"]
```

A %game header is a %game line followed by 2 information lines, the first line giving the players' names,

and the second line giving the game title.

```
GameCmdHeader ::== GameCmdLine PlayersLine TitleLine
GameCmdLine ::== '%game'
PlayersLine ::== #WHITEPLAYER# / #BLACKPLAYER#
TitleLine ::= #TITLETEXT#
```

Example:

```
%game
Spassky,B/Fischer,R
Mar del Plata 1960 King's Gambit
```

A %init header is a %init line followed by a %players command and a %title command.

```
InitCmdHeader ::== InitCmdLine PlayersCmdLine TitleCmdLine
InitCmdLine ::== '%init'
PlayersCmdLine ::== '%players' #WHITEPLAYER# / #BLACKPLAYER#
TitleCmdLine ::== '%title' #TITLETEXT#
```

Example:

```
%init
%title Mar del Plata 1960 King's Gambit
%players Spassky,B /Fischer,R
```

An implicit header is a series of lines recognized by the parser's "header recognition heuristic". See the Parser heuristics help topic for more info.
An approximate grammar is:

```
ImplicitHeader ::== ImplicitHeaderLines*
ImplicitHeaderLine ::== PlayerLine | TitleLine
PlayerLine ::== 'White:' #WHITEPLAYER#
        | 'Black:' #BLACKPLAYER#
        | #WHITEPLAYER# / #BLACKPLAYER#
        | USCFPlayerLine
TitleLine ::== #TITLETEXT#
USCFPlayerLine ::== USCFPlayer '-' USCFPlayer
USCFPlayer ::== #PLAYERNAME# [ '(' #RATING# ')' ] [ '[' #AFF# ']' ]
```

Example:

```
WHITE: Spassky, B BLACK: Fischer, R
Mar del Plata 1960
King's Gambit
```

2.2.2. GAME BODY MODEL

A game body is a series of lines containing commands or moves. A command line begins with a percent (%) character, in accord with the PGN spec. A move line contains 0 or more "moves".

```
GameBody ::== GameLine*
GameLine ::== CmdLine | MoveLine
```

CmdLine ::== '%' #COMMAND# #ARGS#
MoveLine ::== MoveElement*

NOTE: Move elements MAY BE SPLIT ACROSS LINE BOUNDARIES.

2.2.2.1. MOVE ELEMENT MODEL

A move is a "normal chessmove", a comment, one of a variety of special descriptive strings, or a parenthesized variation.

MoveElement ::== NumberedMove | Comment | DescString | Variation

NumberedMove ::== [MoveNumber] MoveGuts
MoveNumber ::== #NUMBER#
| #NUMBER# '.'
| #NUMBER# '...'
| '...'
MoveGuts ::== AlgebraicMove MoveModifiers | DescrptiveMove MoveModifiers
| CastlingMove MoveModifiers

CastlingMove ::== 'oo' 'o-o' | 'ooo' ... ETC

MoveModifiers ::== (EnPassant | Promotion | #NULL#) Check* Evaluator*
EnPassant ::== 'e.p.' | 'ep' | '(ep)' ... ETC
Promotion ::== PromoPiece | '=' PromoPiece | '(' PromoPiece ')'
PromoPiece ::== 'N'|'B'|'R'|'Q'
Check ::== '+' | '#'
Evaluator ::== '?' | '!'

Examples:

44...0-0-0+!
38 Nf7#
17. ef
pxp(ep)

2.2.2.1.1. ALGEBRAIC MOVE MODEL

This grammar is only approximate. The program accepts most ordinary forms of algebraic notation. You can use either 'x' or ':' for captures; the separator '-' is optional; abbreviated forms like 'cd' are accepted. But patently illegal inputs such as "eh" or "be" are rejected. Piece names if present must be uppercase. Column names must be lowercase. Internationalized piece names are not supported, only PNBRQK.

AlgebraicMove ::== [AlgPiece] [AlgFrom [AlgSep]] AlgTo
AlgSep ::== 'x' | ':' | '-'
AlgFrom ::== AlgCol | AlgCol AlgRow
AlgTo ::== AlgCol | AlgCol AlgRow
AlgCol ::== 'a'..'h'
AlgRow ::== '1'..'8'
AlgPiece ::== 'P'|'N'|'B'|'R'|'Q'|'K'

Examples:

1 e4
8 cd
11 e7e5
12 N1e2

2.2.2.1.2. DESCRIPTIVE MOVE MODEL

This grammar is only approximate. The program accepts most ordinary forms of descriptive notation. Piece names and column names may be in either upper- or lowercase. Internationalize piece names are not supported, only PNBRQK. The program does NOT accept "kt" for "n".

```
DescriptiveMove ::== DescPiece ['/' DescLoc] '-' DescLoc
        | DescPiece 'x' DescPiece ['/' DescLoc]
DescPiece ::== 'p'|'n'|'b'|'r'|'q'|'k'|'krp' ... ETC
DescLoc ::== DescCol | DescRow | DescCol DescRow
DescCol ::== 'r'|'n'|'b'|'k'|'q'|'kr'|'kn' ... ETC
DescRow ::== '1'..'8'
```

Examples:

```
10 qn-kn5
11. pxnp
5...n/1-q2
```

2.2.2.1.3. COMMENTS AND DESCRIPTIVE STRINGS

A comment is text enclosed within braces, either '[]' or '{}'.

```
Comment ::== '[' #COMMENTTEXT# ']' | '{' #COMMENTTEXT# '}'
```

Comments may be nested. (Option variable commentnesting .) The program also supports end-of-line comments, but this feature is disabled by default. (Option variable eolcomment .) Example of a comment:

```
{Overlooking the Black's crushing reply}

DescString ::== ResultString | ClockString | SilentPunct
ResultString ::== '1-0' | '0-1' | '1/2-1/2' ... ETC
ClockString ::== #NUMBER# ':' #NUMBER# ... ETC
```

Example of a clock string:

```
4:03
```

Some writers put redundant commas, periods, semicolons, etc, after moves. They would confuse the parser unless it had some way to discard them. (Option variable silentpunct .)

```
SilentPunct ::== ',' | ';' | '.'
```

2.2.2.1.4. VARIATIONS

A variation is a parenthesized list of moves, possibly including nested sub-variations. The first move within the list "anchors" the variation in the game tree. If the first move is numbered, the variation is anchored at that move number. Otherwise, the variation is anchored at *the move number of the move just before the variation.* Alternative branches from the same anchor point can be represented using a vertical bar '|'. You can use either parens or angle-brackets for variations.

```
Variation ::== '(' VarList ')'
        | '<' VarList '>'
VarList ::== MoveElement* | MoveElement* '|' VarList
```

Example:

1 p-k4 p-k4 2 p-kb4 pxp (2...b-b4 [is the KG declined]
|p-q4 [is a good counter gambit])

2.3 SCRIPT FILE MODEL

Again, the difference between a script file and a game file is in the the intended use, not in the syntax. One practical difference, not embedded in the grammar, is that the program supports nesting of script files, but not game files.

ScriptFile ::== ScriptLine*
ScriptLine ::== CmdLine | MoveLine

Example:

%set eolcomment=;
%set pgnout=1

3. SEMANTIC CONSTRAINTS

Correct input must obey semantic as well as syntactic constraints. The usual chess rules apply.

3.1 MOVE NUMBERS

Move numbers, if present, must be correct. The program uses a variety of heuristics based on move numbers. The program has NO algorithm to determine the correct move number if it is not the writer intended. Move numbers ascend in the usual sequence except where variations are present.

3.2 ILLEGAL MOVES

The program has NO algorithm to determine the correct move if it is not what the writer intended.

3.3 AMBIGUOUS MOVES

The program has NO algorithm to determine the correct move if there are 2 or move legal moves corresponding to an input string.

3.4 MODIFIERS

Check and mate (+ and #) indicators, if present, must be correct. For example, the move "pxp+" is illegal unless the pawn takes with check. (Controlled by option variable sloppycheck .) The program uses check, mate, and en passant indicators to disambiguate moves. For example, the move "pxp(ep)" matches ONLY en passant captures.

4. EXTENDED FORSYTH NOTATION

The program uses an extended forsyth notation to represent chess positions as ascii strings. In forsyth notation, a position is listed by rows from top to bottom (8th rank to 1st rank), and left to right, with upper case letters PNBRQK representing resp a White pawn, knight, bishop, ..., etc; and lower case pnbrqk representing a Black pawn, knight ..., etc. Empty squares at the left of a row, or between pieces in a row, are represented by a digit equal to the number of empty squares. For example, "5PPP" is the representation for a row with White's 3 kingside pawns side by side. Rows are separated by a "/" (slash). The starting position is "RNBQKBNR/PPPPPPPP/////pppppppp/rnbqkbnr" in forsyth.
The forsyth extensions used in this program are all based on the "*" (asterisk) character.
A * at the start of the forsyth string prefixes the current move number. A move number followed by a . (dot) indicates White to move; a - (dash) indicates Black to move. For example, "*27-" at the start indicates that the position is at move 27, Black to move.
A * after a rook (R or r) indicates that preceding rook cannot be used for castling, even if the rook and king are still on their home squares.
A * after a pawn (P or p) indicates that the preceding pawn is vulnerable to en-passant capture by the opposing side.

Examples:
The first game of the 1993 women's world championship Ioselani - Xie Jun, at move 31, Black to move. Note that the White queenrook cannot be used for castling.

*31-r*1b2r1k/1pp3b/6p/pPPPp2p/4Pn/B1N2P/P3BR/R*2QK1Nq

The first game of the 1992 Fischer - Spassky rematch, at move 19, Black to move can capture en passant.

*19-r*2qr1k/1b1n1pb/p2p1npp/1p1Pp/PP*p1P/2P2NNP/2BB1PP/R*2QR1K

CHESSBD Parser Heuristics

1. INTRODUCTION

CHESSBD's parser attempts to handle a wide variety of input styles. Basically, input falls into 2 categories: "clean" and "dirty". The Input File Syntax help topic describes approximately the program's syntax for clean input. The parser handles clean input (including PGN headers and straight game scores) reliably.

But the goal of CHESSBD is to handle "dirty" input: input with *implicit* headers, comments, and variations. To meet this goal, the program uses a set of empirical, ad hoc heuristics that seem to work "pretty well" but fall far short of perfection. The heuristics sometimes fail in surprising ways.

When this happens, the fix is usually to hand edit the input file. You have to help out the parser, by "cleaning up" the input. (It may also help to tweak some of the parser's parameters, described in the Option Variables help topic.) Often, it is not necessary to completely convert the file to a clean form: it suffices to clean up the point where the the parser first choked, just to bring it into the domain where the heuristics operate correctly. Hopefully, only minimal hand editing will be required. After fixing the file, simply re-open it. If the errors haven't disappeared, at least the portion of the file that is handled correctly should have increased.

There are 2 kinds of heuristics, and 2 corresponding failure modes: (1) implicit header recognition heuristics; (2) implicit analysis recognition heuristics. See the sections on implicit header recognition failure and implicit analysis recognition failure .

1.1. GENERAL REMARKS

This document provides only an approximate description of the heuristics implemented in CHESSBD. The actual algorithms are empirical, ad hoc, and hard to explain. In general, the heuristics are "greedy", "peephole" type algorithms. They make a decision based on a small amount of local context, and once that decision is made, it is never re-examined no matter the consequences.

Another point to keep in mind is that CHESSBD has NO KNOWLEDGE of natural language. The heuristics are based on superficial features of chess games.

Future releases of this program may not use the same heuristics. Hopefully the coverage will only increase, but with heuristics it is hard to be sure.

2. IMPLICIT HEADER RECOGNITION

The header recognition heuristic models a header as a *paragraph* of text, containing some recognizable form of "White player" and "Black player" notation. Other lines within the header paragraph are concatenated to form the game title. The header paragraph consists of a block of non-blank lines, preceded by a blank line (or the beginning of the file), and followed immediately by a line starting with move 1 of a game (possibly with intervening blank lines). The header paragraph may not contain any legal moves. The number of lines in a header paragraph must not exceed 20 (option variable easyheaderwindow).
Example: "White: Black:" form

> White: Spassky, Boris Black: Fischer, Robert
> Mar del Plata 1960 King's Gambit
> 1. p-k4 p-k4 2. p-kb4 pxp

Example: weak form

> Spassky, Boris - Fischer, Robert
> Mar del Plata 1960 King's Gambit

It is conceivable that the header recognition heuristic might inappropriately recognize commentary text as a header, but i have never seen it happen. The current heuristic apparently errs in the other direction. If it should happen, it is most likely to occur with the "weak" form of implicit header. The option variable weakplayersep can be set to null to disable weak headers.

2.1. NAME RECOGNITION

The header recognition heuristic depends on a name recognition heuristic. Not well developed at this time. Essentially a series of capitalized words containing limited punctuation. There's a special hack that makes a parenthesized number (rating) part of the name.

2.2. IMPLICIT HEADER RECOGNITION FAILURES

The usual symptom of failure is that a game does not show up in the game selection box. Because the program failed to recognize its header. Or, the game might show up but have incorrect information for the "White player", "Black player", or "title" displays.

Some common problems are:

(1) The names could not be recognized, either due to use of funny punctuation, or the presence of extraneous text next to the names.
Fix A: put the names in a simpler form; move extraneous text to a separate line.
Fix B: add to a %game, %init, or PGN header.
Example:

```
Spassky, Boris - Fischer, Robert (Mar del Plata 1960 King's Gambit)
```

Fix:

```
Spassky, Boris - Fischer, Robert
Mar del Plata 1960 King's Gambit
```

(2) Intervening text between the header paragraph and move 1. Fix: Delete blank lines between the header paragraph and move 1, making it all one big header paragraph. Or delete it.
Example:

```
White: Boris Spassky
Black: Robert Fischer
Mar del Plata 1960
King's Gambit

OLD WINE IN A NEW BOTTLE

Here is the second ...

1. p-k4 p-k4
2. p-kb4 pxp
```

Fix:

```
White: Boris Spassky
Black: Robert Fischer
Mar del Plata 1960
King's Gambit
-
OLD WINE IN A NEW BOTTLE
```

1. p-k4 p-k4
2. p-kb4 pxp

3. IMPLICIT ANALYSIS RECOGNITION
The parser's model is alternating blocks of "clean" and "dirty" input. The clean blocks contain only legal moves, and taken together constitute the main line of the game. The dirty blocks contain unrestricted commentary and variations (analysis). The parser has 2 states: parsing a clean block, and parsing a dirty block.

A key issue here is what constitutes a "block". *The program's heuristics model a block as a sequence of lines.* Therefore, transitions occur at line boundaries.

3.1. THE CLEAN STATE
In the clean state, moves, explicit commentary, and explicit variations are parsed (approximately) as described in the Input File Syntax help topic. Moves occur in sequence only, except within explicit variations.

3.2. TRANSITION FROM CLEAN TO DIRTY
When parsing a clean block, any line starting with a legal move continues in the clean state. Any line *starting* with an illegal or ambiguous move, or urecognizable text, shifts the parser to the dirty state.

If an illegal or ambiguous move or unrecognizable text occurs *in the middle of* a line, the parser's action is harder to describe. It may or may not go the the dirty state. Basically it tries to recover by skipping over the "bad part", treating it as implicit commentary. A key point is that recovery can only occur with *numbered* moves.

3.2. THE DIRTY STATE
In the dirty state, the parser models input as an implicit variation possibly containing implicit commentary (ie, text not recognizable as legal moves even in a variation). The first legal numbered move "anchors" the implicit variation at that move number. To be legal in the context means that the move number must be equal or prior to the expected move in the main line, and within 30 (option variable easyvarwindow) plies of the expect move. Subsequent moves in the variation are expected to be in sequence from this anchor move number. Within the sequence of moves in a variation, move numbers may go backward from the current move, as far back as the anchor move, triggering an implicit tail variation.

Again, any illegal or ambiguous moves, or unrecognizable text is taken as implicit commentary. The parser's exact handling of comments (explicit or implicit), variations (explicit or implicit), etc, within the dirty state is hard to describe. The assumption is that the input is "correct", but merely missing the commentary and variation tokens.

3.3. TRANSITION FROM DIRTY TO CLEAN
The parser transitions from the dirty to the clean state when it finds a line *starting with a numbered legal move with the expected move number of the last clean block*. (Ie, the next move in the main line.)

3.4. IMPLICIT ANALYSIS RECOGNITION FAILURES
The usual symptom of failure is everything becomes a giant comment, or the moves of what should be the main line are taken as part of an implicit variation. This could happen because the input just plain does not match our model, or because of unrecoverable problems within the block. (Or because of BUGS in the parser.)

You should look carefully at the start of the giant comment, or at the move number BEFORE the start of the variation. Input "errors" that can mess things up are:

(1) Unbalanced parentheses. The parser attempts to recover from unbalanced parentheses, but in some cases it can't. The best thing to do is figure out where the matching close paren should go, and put it

there. Or delete the offending text.

(2) Ambiguous moves. The parser has NO algorithm to guess genuinely ambiguous moves. The fix is for you to figure it out and replace the ambiguous move with what it should be.

(3) Extraneous text within "clean" block. The parser attempts to recover from this, but if it fails, you have to get rid of the junk or turn it into an explicit comment.

(4) If the formatting of the implicit variation is such that it coincidentally contains a line *starting with move number that is the same as the expected move number of the main line*, the program could prematurely end the implicit variation, with confusing effects. the fix is to reformat it to eliminate the unlucky "coincidence".

(5) Moves and text interspersed without move numbers. Remember, when unrecognizable text is encountered, recovery only occurs when a *numbered* move is found.

4. CONCLUSION

I'm getting burned out on documentation. GOOD LUCK.

CHESSBD Script Commands

`%.` ***FILE***
Read a script ***FILE*** (containing commands). Example: **`%. cmds.txt`**

`%a` ***PXX***
Add piece ***P*** at location ***XX*** (algebraic). (This command is provided for setup purposes.) Example: **`%a Nd5`**

`%cd` ***DIR***
Change current working directory to ***DIR***. Example: **`%cd c:\games`**

`%close`
Close (clear out) the current games list (deleting all games).

`%d` ***XX***
Delete piece from location ***XX*** (algebraic). (This command is provided for setup purposes.)

`%echo` ***ARGS***
Print ***ARGS*** to screen.

`%eval` ***STRING***
Perform variable expansion on STRING and execute the resulting string as a command. With the STRING, the form **`%NAME%`** is substituted by the value of the option variable *NAME*. the form **`%%`** is substituted by a single "%". For example,

%eval %%set A=%B%

sets variable A equal to the value of variable B. This is mainly intended for use in login scripts, so you can use a single login script with different hosts by changing the variables.

`%game`
Force start of a %game header .

`%get` ***POSN***
Set current board position to ***POSN*** (in extended forsyth). Example:

`%get *16-1r5r*/p1pRnkbp/4p1p/5p//1NP3P/PP2bPBP/R*1B3K`

`%h` ***[CMD]***
Print a list of all commands and how to use them.

`%icsarena` ***ARG***
Controls the visibility of the ICS Arena dialog. If ARG is absent or equal to the null string, the ICS Arena dialog is miniaturized. Or if ARG is equal to the single character +, it is made expanded to its "normal" size. Otherwise, ARG is expected to be a geometry specification string of the form WWWxHHH+XXX+YYY where WWW, HHH, XXX, and YYY are respectively the width, height, x-position, and y-position of the dialog; and the dialog is made visible with the given size and position. See also the icschat , icstelnet , and icswin commands. This is an experimental feature.

`%icscapture` ***CMD***
Sends CMD to the ICS server and captures the resulting output in a separate window of the appropriate size. The output is captured up to but not including the next prompt. (See the icspromptpat option variable.) This is an experimental feature.

%icschat ***ARG***
Controls the visibility of the ICS chat dialog, similar to the icsarena command. This is an experimental feature.

%icsinput ***NSEC***
Wait up to ***NSEC*** for ***STRING*** to appear the output from the ICS server. C style backslash escape processing is performed on ***STRING***. (See the description of the **icsoutput** command). If ***STRING*** does NOT appear within ***NSEC*** seconds, the execution of current script file is aborted.

%icsoutput ***STRING***
Send STRING to the ICS server after performing C style backslash escape processing. (Ie, \n becomes newline, \t becomes tab, \ooo becomes octal character ooo.) This command is for use in ICS button bindings and ICS login scripts. The ***STRING*** is also echoed locally in the ICS control window.

%icssend ***STRING***
Like **icsoutput** , except no local echo is performed.

%icstelnet ***ARG***
Controls position of the ICS telnet window, similar to the icsarena command. This is an experimental feature.

%icswho ***ARG***
Sends the ICS command ARG to the ICS server and sends the resulting output to the ICS arena listbox. ARG is subject C-like escape (backslash) character processing. This is used in the implementation of the "who refresh" button in the arena dialog. The default binding of the "who refresh" button is "%%icswho who v". (Note, this feature will not work correctly unless ARG is a variant of "who v". You could do "who av" or something similar.)

%icswin ***ARG***
Controls the position of the ICS main window, similar to the icsarena command. This is an experimental feature.

%init
Start a new game. This resets the board to the initial piece setup, and clears the game title and White and Black player names. (See also the %players and %title commands.)

%logging ***FLAG***
Turn logging on (***FLAG***=1) or off (***FLAG***=0). When logging is on, textport output is written to the logfile. (See the logfile option variable.)

%m ***XXYY***
Move piece from location ***XX*** to ***YY*** (algebraic). (This command is provided for setup purposes) Example: **%m a2a4**

%moves
Print a list of all legal moves from the current position.

%natural
Sets parameters for natural (as opp to PGN) input (See also the %pgnmode command.)

%open ***FILE***
Open a ***FILE*** of games (discarding current game list). (See also the %scan command.)

%p
Print an ascii representation of current position.

%pgnmode
Sets parameters to PGN-compatible (as opp to natural input) values See also the %natural command.)

%players ***WNAME/BNAME***
Set player names for current game. For the duration of the game, ***WNAME*** will be displayed next to the White side of the board, and ***BNAME*** will be displayed next to the Black side. (See also the %title command.)

%put
Print current position (extended forsyth) on screen. (See also the %get command.)

%scan ***[FILE]***
Add all games in ***FILE*** to current list. (This differs from the %open command by adding to, not replacing, the current games list.)

%score ***[FILE]***
Print score of current game to ***FILENAME*** (or screen if none).

%set ***[NAME=VALUE]***
Set internal variable ***NAME*** to ***VALUE*** Or just print values of all variables on the screen if no ***NAME***=VALUE is present (See also the Options help topic.) (See also the %unset command.)

%title ***STRING***
Set title for current game to ***STRING***, which will be displayed atop the board for the duration of the game. (See also the %players command.)

%unset ***NAME***
Unset internal variable ***NAME***. (See also the %set command.)

%winexec ***[-m] DIR CMD [ARGS]***
Launches a command "CMD ARGS" from directory DIR. If the optional -m is given, the command is started minimized.

CHESSBD Script Commands Index

The following is a partial list of CHESSBD's script commands. Script commands can be put in script files or typed in through the textport. All the major features can be accessed via the GUI instead of running commands directly, so you should rarely have to use them. **(:-)** The %h command lists all commands, documented or no. **(:-)** The most useful commands are %init, %title, %players, %game, and %set. Most of the commands available through the menu are also available in some form as %-commands.

CHESSBD Option Variables

animationspeed (default: **30**)
Controls the speed of move animation on the board display. A speed of 30 causes pieces to be redrawn approximately on every square passed during a move. If =0 move animation is disabled, and the new position is drawn almost instantaneously. NOTE, if you want the animation to look right, animationspeed must be at least 3.

badwarn (default: **1**)
Controls whether a warning will be issued for bad moves. If !=0, ENABLES bad move warning.

commentnesting (default: **1**)
Controls whether comments within {} are treated as nested (balanced). If !=0, a comment extends to the *matching* end brace.

dragmove (default: **1**)
Controls the style of interaction when moving pieces with the mouse. If =1 (drag-and-drop), you move by holding down the left mouse button and dragging the piece in question. If =0 (click-from-and-to), you move by clicking the left mouse on the piece to be moved, then on its desired target square. If =-1 (click-without-drag), it is the same as =0, except that the piece symbol does NOT track with the cursor.

easyheaderwindow (default: **20**)
The maximum number of lines in an implicit game header. Set to 0 to DISABLE implicit header detection.

easyvarwindow (default: **15**)
The maximum number of plies (half-moves) back that will be considered a plausible implicit variation. Ie, implicit variations must start with a move number within easyvarwindow half-moves of the current move number. Set to 0 to DISABLE implicit variations.

eolcomment (default:)
The character which introduces end of line comment in input. Set to ';' for PGN.

exitautosave (default: **0**)
Controls whether changed option variables will be automatically saved on exit. If !=0, they are saved without asking the user.

fileoverwrite (default: **0**)
Controls whether an output file will be overwritten or appended to. If !=0 the File|Save game command will overwrite an existing file; otherwise it will append (if you are saving the current game), or ask for permission (if you are creating an index file).

flashmoves (default: **1**)
Controls the highlighting of legal destination squares when you move a piece with the mouse. If !=0 the feature is enabled.

icsassesscmd (default: **assess %opponent%\\n**)
The command string sent to ICS when you click the "assess" button in the ICS Arena dialog.

icsautoflag (default: **0**)
Controls whether CHESSBD will automatically flag your opponent when his or her time remaining goes below the threshold icsautoflagthresh . Set to non-0 to enable autoflagging. This applies only to ICS mode.

icsautoflagthresh (default: **0**)
The threshold in seconds for autoflagging.

icsautoqueen (default: **0**)
Controls whether CHESSBD will automatically promote to a queen when one of your pawns reaches the 8th rank. If 0, CHESSBD will pop up a dialog asking what piece you want to promote to. This applies only to ICS mode.

icsficsmode (default: **1**)
Enables various FICS compatibility features.

icsfingercmd (default: **finger %opponent%\\n**)
The command string sent to ICS when you click the "finger" button in the ICS Match dialog.

icsgamefile (default: **icsgames.pgn**)
CHESSBD automatically appends your ICS games to this file. See also the option variable icssave

icshandle (default: **?**)
Your ICS login name.

icshost (default: **chess.lm.com**)
Your ICS server's hostname.

icslogfile (default: **ics.log**)
Name of the log file used by the "ICS File|Log open" command.

icsmatchdefaults (default: **r b 3 12**)
The default ICS match parameters that are used when you click the default button in the ICS Match dialog

icsnoslip (default: **0**)
Whether or not to use SLIP (vs a direct modem connection).

icsport (default:)
The IP port number to use when connecting to the ICS service.

icspromptpat (default:)
A list of strings that can appear as (the right end of) a prompt. Patterns are separated by a '|' character. This variable is used if and only if you have set icsrawout to 0. If so, incomplete lines of output from the server will not be echoed unless the right end of the line matches one of the strings. This is done so that you can see the prompt. You should ALWAYS include the strings 'assword: ' and 'ogin: ' so the password and login prompts will be echoed properly. Example:

assword: |ogin: |ics% |fics% |FICS %
(Experimental feature, use at your own risk.)

icsquerylogout (default:)
(Experimental feature.) Set this variable to 1 if you want the program to pop up a confirmation dialog before closing the ICS control window.

icsrawout (default:)
(Experimental feature, use at your own risk): set this to 0 to suppress the echoing of style 12 move messages in the control textport. If you enable this feature, you must also set the icspromptpat variable to match the server's prompt string.

icsrcfile (default: **_ics.rc**)

The default ICS login script to run when you connect to ICS.

icssave (default: **07**)
CHESSBD automatically saves your games, and related information, to the file named by the option variable icsgamefile The icssave value determines what information gets saved: If icssave mod 1 is not 0, the game header, moves, and game result are saved. If icssave mod 2 is not 0, game-related information such as incoming challenges are also saved as comments in the file. If icssave mod 4 is not 0, incoming and outgoing chats are also saved as comments. If icssave mod 8 is not 0, incoming c-shouts and "-->" messages are also saved as comments. So by default, your games, incoming challenges, and incoming and outgoing chats are saved to the games file.

icstmstampflag (default: **0**)
If non-0, CHESSBD will execute a suitable command to start the timestamper program, based on the values of icshost , icsport , icstmstampport and icsficsmode . It will do the right thing. Do NOT set this if you are using the "manual launch" method of starting the timestamper program.

icstmstampcmd (default: **-m . tmstamp**)
Obsolete.

icstmstampport (default: **5000**)
The client port to use when starting up the timestamper program. (TMSTAMP or TMSEAL). If you need to run multiple simulataneous copies of CHESSBD with TMSTAMP/TMSEAL, you can do so - but only if each session is running with a different client port.

icswhocmd (default: **%icsoutput who v \\n**)
The command string executed by CHESSBD when you click the "who ref" button in the ICS Arena dialog. You can change it, but it won't work right unless you set it to something with the "v" option set, e.g., "% %icsoutput who av\\n". (Note: it is a CHESSBD command, and it undergoes both variable expansion and escape (backslash) character processing.)

icswinmax (default: **800x768+356+0**)
The size of the ICS control window when maximized.

innotation (default: **3**)
Controls move notation used for game input. If =1 input must be in english descriptive notation; if =2 input must be in algebraic notation; if =3 input may be in either english descriptive or algebraic. See also the outnotation variable.

logfile (default: **chessbd.log**)
The name of the log file. See the File|Log command.

mainwin (default: **646x570+40+40**)
The geometry for the main window. The format is WWWxHHH+XX+YY where WWW is the width, HHH the height, XX the x offset, YY the y offset.

materialupdate (default: **1**)
If !=0, enables automatic display of the material balance after each move. The difference in material is displayed as part of the to-move strong.

maxblackdots (default: **20**)
The maximum number of .'s (dots) that will be recognized on input as the token signifying Black's move.

maxcomment (default: **4096**)
The maximum allowable comment length. Longer comments are truncated.

maxplayerchars (default: **50**)
The maximum number of letters in a player name, for implicit game header recognition.

minplayerchars (default: **3**)
The minimum number of letters in a player name, for implicit game header recognition.

nullheader (default: **0**)
Controls recognition of degenerate implicit game headers. If !=0, ENABLES recognition of implicit headers with NO header info: just a line starting with a legal move 1.

outlinelen (default: **75**)
The maximum line length of output lines, used when writing a game score with the File|Save game command.

outnotation (default: **0**)
Controls move notation used for output by the program. If =1 moves are output in english descriptive notation; if =2 moves are output in algebraic notation. if =0 moves are printed in the same notation that was used for the last input move. See also the innotation variable.

pgnout (default: **1**)
Controls algebraic output move notation style. If !=0, moves are output in PGN/SAN style.

pgntitlefmt (default:)
The format string used to synthesize a game title from PGN tags. %NAME% is replaced by the value of PGN tag NAME.

piecechars (default: **PNBRQK?pnbrqk?**)
List of characters representing, respectively, White Pawn, White Knight, ... etc, ... Black Pawn, Black Knight, ... You must put a question mark (?) after the White King and Black King characters.

promptstr (default: **>**)
The string used to prompt for user input in the textport window.

queryindex (default: **1**)
Controls whether to query for a filename when you create an index. If !=0, a dialog is used to get the name to create. Otherwise, a default filename is used: the last-opened games file, with the extension changed to ".i" . See also the fileoverwrite variable.

querysavegame (default: **1**)
Controls whether to query for a filename when you save the current game to a file. If !=0, a dialog is used to get the name to create. Otherwise, the default name "chessbd.sav" is used. See also the fileoverwrite variable.

silentpunct (default: **,.;**)
A list of characters (usually punctuation characters) which may be silently ignored on input.

sloppycheck (default: **1**)
Controls check verification on input moves. If !=0, input moves will be accepted with incorrect check indications (otherwise it's an error).

strict (default: **0**)
Controls parser heuristics for implicit game headers, implicit comments, and implicit variations. If !=0, DISABLES the recognition of implicit features.

varbdwin (default: **316x474**)

The initial size XXXxYYY of variation board windows, or ICS game observation windows.

`weakplayersep` (default: **`/-`**)
A list of characters that may be used to separate Eg, **`Karpov-Kasparov`** or **`Karpov/Kasparov`** .

CHESSBD Option Variables Index

The following is a partial list of CHESSBD's option variables. Option variables can be set using the %set command. or via the Options dialog, which also allows you to change the fonts and colors used by the program. You can examine and change all option variables --- even undocumented ones --- with the **%set** command. **(:-)** The program as distributed has reasonable defaults for all option variables, so you should rarely have to change them. **(:-)**

outnotation
pgnout
pgntitlefmt
piecechars
promptstr
queryindex
querysavegame
silentpunct
sloppycheck
strict
varbdwin
weakplayersep

CHESSBD Command

CHESSBD recognizes 2 kinds of command line arguments: "ini" files, and "game" files.

ini

Any command line argument with the extension ".ini" is read in as an extension of **`chessbd.ini`** , before the main window is created. This feature allows you to put your custom settings such as an alternate piece set, or user-defined menu items, in a file separate from **`chessbd.ini`** . NOTE: if you do put your custom settings in a separate ini file, be sure the first line of that file is **`[chessbd]`** . Also be sure to put that file in the same directory as the CHESSBD executable. Example:

chessbd toups.ini

If you change the command line as above in your program manager item for CHESSBD, then the next time you start CHESSBD it will use the alternate piece set designed by Harry Toups. (Be sure you have set the directory in the program manager item, to the directory where CHESSBD and all its files are stored.)

NOTE that you can add ANY option settings to extra .ini files, and they will work, but IF you then use the Options dialog to change them, the changes will only be saved in **`chessbd.ini`** , not your extra .ini files. Then the next time you start CHESSBD, the changes will be over-ridden by the old values in your extra .ini file(s). Thus, if you use the Options dialog to make changes to any variables that you are also setting in an extra .ini file, go to the **`chessbd.ini`** file and move the corresponding entries to your extra .ini file. Use any text editor to do this.

game

Any command line arguments with an extension other than ".ini" are treated as "game" files, and are read in as if you had opened then with File|Add . This feature allows you to use CHESSBD as a NETSCAPE helper application. (In NETSCAPE, use the Options|Preferences dialog.)

CHESSBD ICS Mode

You can use CHESSBD to play chess on an Internet Chess Server. CHESSBD works with both the Internet Chess Club (ICC) and the Free Internet Chess Servers (FICS). Click on ICS|ManualLogin in the main menu. This activates CHESSBD's ICS Main window and pops up the ICS ConnectInfo dialog. You can also bypass the ICS ConnectInfo dialog by selecting one of the profile entries instead of ManualLogin from the ICS sub-menu. (You must first set up the profile entries in the file **`icsprofs.dat`** with the correct information. See ICS Profiles file .)

ICS Help topics

Here is a list of the major ICS mode help topics.

ICS Main window
ICS mode creates a new window --- the ICS main window --- which contains the ICS menu bar , the ICS Board display , plus 3 specialized child windows: the ICS Telnet window , the ICS Chat dialog , the ICS Arena dialog .

Basically, the ICS menu bar provides miscellaneous program management features; the ICS Board display is used to play or examine games; the ICS Telnet window is used to login and communicate with the server; the ICS Chat dialog is used to talk to other users; the ICS Arena dialog is used to arrange matches.

Each of the 3 child windows (ICS Telnet, Chat, and Arena) can exist in 2 different forms, "miniaturized" and "expanded". A child window in its "miniaturized" form occupies a fixed size and location within the ICS main window. A miniaturized child window is functional, but may be less convenient to use than its expanded form. You can move a miniaturized window (by dragging its title bar), or expand it by resizing (dragging the sizing handles) or by maximizing it (using the maximize button). The "maximize" button in a miniaturized child window only restores it to its normal "expanded" size, rather than truly maximizing it to full screen size. But in an "expanded" child window, the maximize button has the normal maximizing effect. The minimize button in an "expanded" child window will miniaturize it again. (NOTE: each miniaturized window "remembers" its normal "expanded" size and position. To customize these remembered sizes and positions, see the ICS File|Save layout topic.)

Throughout this document, there are many references to CHESSBD "option variables". These are internal CHESSBD variables that you can set with the Options dialog, or in the **`chessbd.ini`** file. These control the operation and appearance of the program, includings things like colors, fonts, and the bindings for programmable buttons. See the Option Variables topic for info on what option variables are there and what they do. There are also references to CHESSBD "script commands". See Script Commands and ICS Script Commands for more info.

ICS Menu Bar

The menu bar contains the ICS File sub-menu, the ICS Commands sub-menu, the ICS User sub-menu, the ICS Window sub-menu.

ICS|Manual Login
This menu item, in the main menu bar, enters ICS Mode pops up the ICS ConnectInfo dialog, and tries to make a connection using the parameters specified in that dialog.

ICS|...profiles...

The other items in the ICS sub-menu (besides ManualLogin) are based on entries in the **`icsprofs.dat`** file. That file is like the dialling directory in a comm program. Each entry assigns a "profile name" to a set of parameter values. When you select the corresponding menu item, CHESSBD uses those parameter values and tries to make a connection. You must edit **`icsprofs.dat`** to contain the correct information for YOUR various accounts, before attempting to use these profile entries. See ICS Profiles file for more information.

ICS ConnectInfo dialog
In order to connect to and successfully use an ICS server, you need to specify values for a handful of server-dependent option variables. The ICS ConnectInfo dialog allows you to set them all at once.

For Modem Connections
If you are connecting via a raw modem connection (as opposed to using SLIP/PPP), select the Modem radio button and fill in the COM field with the number of your modem COM port (e.g., 2); fill in the baud field with the baud rate you wish to use (e.g., 19200); and select the correct parity, databits, and stopbits options.

For SLIP/PPP Connections
If you are connecting via SLIP/PPP, select the Winsock radio button and fill in the ICS host field with the hostname of the ICS server (e.g., chess.onenet.net); and fill in the port field with the port number used by the ICS server (e.g., the default is 5000).

ICS handle
Fill in the name (handle) by which you will enter the ICS server (e.g., ButtHead). CHESSBD does not use this information to log you in, but it is used to parse challenges and such. If you do not supply the correct information, various CHESSBD features will not be fully functional.

ICS login script
Fill in the name of your login script (e.g., _ics.rc). CHESSBD will run your login script as soon as the connection is made. You can put any CHESSBD script commands in your login script, including commands to send arbitrary strings, and wait for prompts. (See Script Commands for more information.) Leave this field blank if you do not have a login script. (It is highly recommended that you create a login script. The file **`ex_ics.rc`** can be used as a template.)

FICS mode
Check the FICS mode checkbox if you are connecting to a Free Internet Chess Server (FICS) server; uncheck it if you are connecting to the ICC server. You must supply this information. If you check FICS mode and connect to ICC, or vice versa, various CHESSBD features will not be fully functional.

Save profile button
Click the save button to save the displayed parameters in in a *profile*. The parameters are saved under the name currently shown in the Profile control. The next time you want to connect using the same parameters, you can just select that profile under the ICS menu or from the Profile control in the ConnectInfo dialog.

To create a new profile or modify an existing profile, just type its name in the Profile control, fill in the other dialog parameters. and click "save profile".

The ConnectInfo dialog allows you to create new profiles, but it has no provision for deleting obsolete profiles. To do that, just use your favorite text editor to modify the file **`icsprofs.dat`** . Each profile corresponds to a section in the file. You can also add other, arbitrary option variable assignments to any profile by editing the corresponding section.

OK button
Click OK to accept the displayed parameters.

Cancel button
Click Cancel to dismiss the ConnectInfo dialog without changing the actual parameters.

Profile
The Profile combo box allows you to select "canned" settings for all parameters from a list in the **`icsprofs.dat`** file. You must first modify **`icsprofs.dat`** to contain your correct settings. (See ICS Profiles file .)

ICS Telnet window
The ICS Telnet window contains a textport that simulates a stripped-down communications program. Output from the server appears in the ICS Telnet window, and everything you type in it is sent to the server. You can login to a machine, or enter an arbitrary command to the remote machine, using the ICS Telnet window. Normally, the ICS Telnet window operates in Half Duplex Local Echo mode, and suppresses style12 messages. You can change these settings via the TtyMode dialog (Alt+M).

You can copy text from the ICS Telnet window to the clipboard simply by selecting it with the left mouse button. This has the side-effect of setting the variable **`%clipboard%`** to the selected text. You can paste text from the clipboard into the ICS Telnet window by clicking the right mouse button.

ICS Profiles file

The file **icsprofs.dat** contains host-dependent variable settings (profiles) of the different ICS servers you can login to. This is the source of the data that is used in the "profiles" combo box in the ICS ConnectInfo dialog. Since you can enter the same info manually, this file is optional, just for convenience. You must edit this file (in the obvious way), to have the correct information for YOUR various accounts.

It is a plain text file. You can add or change entries using your favorite text editor. There is one entry per host. Each entry starts with a line of the form

```
[NAME]
```

where NAME is a mnemonic name for the entry. (Eg, you might choose ICC for the Internet Chess Club at chess.lm.com .) Subsequent lines in the entry must be of the form

```
VAR=VALUE
```

where *VAR* is the name of an option variable (see ICS mode variables) and *VALUE* is the desired value for that variable under the *NAME* entry. (Eg, **icshost=chess.lm.com** for ICC .)

If you don't understand this, you can just modify the sample icsprofs.dat file from the distribution, in the obvious way.

You are not limited to specifying ICS mode variables: you may specify any CHESSBD option variable that you wish to associate with the NAME entry, and it will be set when you select that entry from the ICS ConnectInfo dialog. That dialog only displays the "standard" host dependent variables, but you can put others in if you wish. One reason you might wish to do so is so you can have a single login script for several servers whose login procedure differs only in the specific login name and password required; you can use a profile variable for the login name (or password) and use the %eval script command to do variable expansion in your login script. See **dial_ics.rc** for an example.

The special variable "defaults" can be used to simplify your profiles file. You can use it in one profile entry to specify that default values are to be taken from another entry. I added this feature because many profile entries tend to differ from each other by only 1 or 2 values. Any explicit values in the current entry take precedence over the defaults.

Example:

```
[B-FICS]
defaults=A-FICS
icshost=crocus.csv.warwick.ac.uk
icstmstampflag=0
[A-FICS]
icsficsmode=1
icshandle=dfong
icsnoslip=0
icsport=5000
icsrcfile=_ics.rc
icstmstampflag=0
icshost=fics.onenet.net
icstmstampflag=1
```

In the above example, profile entry B-FICS gets its default values from the entry A-FICS, and overrides the values of icshost and icstmstampflag.

ICS File sub-menu

The ICS File sub-menu provides the following commands.

- ICS File|Connect
- ICS File|Disconnect
- ICS File|ConnectInfo
- ICS File|Logging
- ICS File|Log flush
- ICS File|Tty Hangup
- ICS File|Tty Break
- ICS File|Capture
- ICS File|Save layout
- ICS File|Close ICS

ICS File|Connect
Connects to the ICS server. You must have first set the option variable *icshost*. (See the Options section.) Upon connection, CHESSBD searches for and executes (runs) a script file **`_ics.rc`** if there is one. (See the ICS Script Commands section.)

ICS File|Disconnect
Disconnects from the ICS server (assuming you are already connected).

ICS File|ConnectInfo
Pops up the ICS ConnectInfo dialog, the same dialog as you get by default when you start a connection.

ICS File|Logging
Turns ICS logging on/off. When on, all output from the ICS server is written to the file "ics.log" (option variable *icslogfile*). If the file already exists, all new logging information is appended to the end. NOTE: the logfile is internally buffered, so if you want to look at the contents of the logfile while you're still running the program, you should close it first (ICS File|Log close) to flush it to disk. If you want, you can open it again (ICS File|Logging) right afterward. You can also use the ICS File|Log flush item.

ICS File|Log flush
Flushes the ICS log file, if logging is on. Also flushes the ICS games file.

ICS File|TtyHangup
Issues a hangup on the com port. Has no effect unless you are connecting with a raw modem connection.

ICS File|TtyBreak
Issues a break on the com port. Has no effect unless you are connecting with a raw modem connection.

ICS File|Capture
Pops up a dialog that asks you for an ICS command. The command is then sent to the ICS server, and the resulting output is saved in a separate window. All ICS output is diverted to the window until a line matching your icspromptpat appears in the output. The window persists until you close it. The window is sized to fit the output. (This feature provides a handy way to keep the output of a command (for example, "finger") visible during a game. You could just enter the command in the ICS Telnet window but then the output would scroll off as later messages came in.)

New feature: when you select text in the capture window using the left mouse button, it is copied to the clipboard. You can then access the selected text in the programmable menu items or arena buttons with the **`%clipboard%`** variable. And, you can paste text from the clipboard into the chat or telnet textports by clicking the right mouse button.

ICS File|Save layout
Saves the current size and position of the ICS main window, and the child windows (the ICS Telnet window, the ICS Chat dialog, and the ICS Arena dialog). The saved size and position of the ICS main window will be used next time you start ICS mode. The saved size and position of the child windows will be used when you "maximize" a "miniaturized" window. NOTE: if a child window is currently miniaturized when you save the layout, the size and position are NOT saved. Therefore, before saving, you should "expand" any child windows whose size and position you want to affect.

If you have an observation board visible, the size of that board will be saved and used as the default size for future observation boards windows.

ICS File|Close
Closes the ICS windows. If you were connected to the ICS server, this disconnects you.

ICS Commands sub-menu
The ICS Commands sub-menu is the same as the Commands sub-menu.

ICS Chat
The ICS Chat dialog provides a convenient way to keep track of shouts, tells, kibitzes, etc. All shouts, whispers, s-shouts, etc, are echoed to the ICS Chat textport (in addition to the ICS Telnet window). Everything you type in the ICS Chat textport is sent, prepended by the contents of the "Chat prefix" box. The default prefix is "say".

For example, if you've changed the "Chat prefix" to "tell Butthead", and you then enter "Huh, huh, huh" in the textport, this is equivalent to entering "tell Butthead Huh, huh, huh" in the ICS Telnet window. Try it.

Notice that the "Chat prefix" box is a drop-down listbox. If you need to do more than one kind of chat in a single session, you can select a previously used chat prefix by clicking on the down-arrow. And when an incoming chat message arrives, the program automatically enters the prefix you need to respond. That is, if user "Beavis" tells you something, the prefix "tell Beavis" will be appended to the drop-down listbox for you to select if you wish.

If you don't like having your typing interrupted by incoming messages, you can type in the one-line edit box at the lower right of the Chat dialog. When you press enter, whatever is in the box will be copied into the Chat textport AND sent to the server. (This feature is just for convenience. You can type directly into the Chat textport if you want.)

You can copy text from the ICS Chat textport to the clipboard by selecting the text with the left mouse button. This has the side-effect of setting the variable **`%clipboard%`** to the selected text. You can paste text from the clipboard to the ICS Chat textport by clicking the right mouse button.

NOTE: the Chat and Arena dialogs provide convenience features relating to challenges and conversing with other users. They are based on interpreting messages from the ICS server, and they thus have limitations based on the lack of documentation and standardization of ICS messages. If the dialogs don't do what you want, you can always resort to typing ICS commands in the ICS Telnet window.

You can cause long output lines to be automatically wrapped (or not) thru the TtyMode dialog (Alt+M).

ICS Arena
The ICS Arena dialog implements a convenient way to challenge or respond to challenges, to select games for observing, or to quickly execute a pre-programmed command. The ICS Arena dialog contains a user listbox and a refresh button to fill in the listbox with the latest information; a challenge listbox ; a button to clear all challenges; ; a bank of 16 user-programmable function keys ; an autoflag checkbox ; an autoqueen checkbox ; an autofocus checkbox .

The user listbox and refresh button
The user listbox can be used to select opponents for challenges, or to select games for observing. To fill (or "refresh") the list with the latest information, click the "who ref" button. This sends the command "who v" to the server and records the resulting output. Since this is a listbox, you select an entry by clicking on it. This has the side-effect of setting the internal CHESSBD option variable **`%clipboard%`** to the name of the user in the selected entry. The variable **`%user%`** is also set to the same value. (NOTE: the variable %gameno% is no longer supported.) These internal option variables can then be used by variable expansion in the programmable function keys. To issue a challenge, simply double-click on the corresponding entry. This will pop up the ICS Match dialog.

The challenge listbox
Incoming challenges from other users are stored in this listbox. When you select an entry, it has the side-effect of setting the internal CHESSBD option variable **`%challenger%`** to the user who issued the selected challenge. This variable can then be used by variable expansion in the programmable function keys. To issue a challenge, or respond to a challenge, simply double-click on the corresponding entry. This will pop up the ICS Match dialog.

ICS Autoflag
This checkbox, when checked, enables "autoflagging" your opponent. That is, CHESSBD will automatically send the "flag" command when its estimate of your opponent's time remaining goes below a the value icsautoflagthresh (default 0 seconds). If you have bad lag, you may wish to set icsautoflagthresh to a positive number, to send the "flag" command in anticipation. NOTE: because of lag effects, it is possible that CHESSBD will autoflag when your opponent still has time left. In that case, CHESSBD will try again to autoflag when your opponent's estimated time left crosses the threshold, but not until another update comes back from the server. ALSO NOTE: CHESSBD assumes your side of the game is the side on the bottom of the board diagram, and will "autoflag" whichever side is on the top side.

ICS Autoqueen
This checkbox, if checked, tells CHESSBD that you wish to automatically promote to a queen when one of your pawns reaches the 8th rank. If unchecked, CHESSBD will pop up a dialog asking you what to promote to.

ICS Autofocus
This checkbox, if checked, tells CHESSBD that you wish to have the "focus" automatically switch to whatever window shows activity. For example if you have it chedcked and you are observing several games, CHESSBD will automatically popup the board whenever someone moves.

The user-programmable function keys
The ICS Arena dialog contains a bank of 16 user-programmable function keys. They have the following default bindings:

* The "match" button issues a challenge to the user designated (by clicking on the appropriate line) in the main listbox.
* The "obs" button issues an observe command for the game number selected in the main listbox.
* If someone's challenged you and you want to issue a counter-challenge, click "neg" (negotiate).
* To accept a challenge, click "acc". (This accepts the challenge from the line selected in the "who v" listbox.)
* To move forward (back) in an examined game, click "fwd" ("back").
* To issue a takeback request, click "tkback".
* To call flag, click "flag".

You can customize the bindings of these buttons. It's not that difficult to do, but it is messy to explain. Use your favorite text editor to edit the **chessbd.ini** file.

The buttons are numbered 00-15 from left to right, top to bottom. The label and action for button *00* are specified by the value of the option variable **ics_lu00** . The label and action are separated by a vertical bar (|) character. You can issue any CHESSBD command from a button. (See the Commands section for details on general CHESSBD commands, and the ICS Script section for details on ICS-specific CHESSBD commands.) When you click a button, the value of the corresponding variable is interpreted as a CHESSBD command. CHESSBD does a simple form of variable expansion before executing the command.

Variable expansion substitutes %VAR% with the value of variable VAR. You can use as VAR any CHESSBD variable described in the As an important special case, two consecutive %'s is substituted with one % (to allow literal %'s to occur in a value). Options section. In addition, CHESSBD recognizes a few extra variables specific to the ICS Arena dialog: the variable "user" expands to the user name (in the selected line) in the main listbox; the variable "challenger" expands to the user name of the selected line in the challenge listbox.

In addition to variable expansion, some commands (like **%icsoutput**) do backslash escape processing. This lets you put special characters such as newline and tab in the button bindings.

An explained example will hopefully make things clear. See the default **chessbd.ini** file for more examples.

```
ics_lu00=obs|%%icsoutput observe %user%\n
```

The label for button 00 is **obs** .
The action for button 00 is **%%icsoutput observe %user%\n** .
Again, this action is a string which is interpreted as a CHESSBD command. The CHESSBD command here is **%icsoutput** . (Note the two consecutive % signs; the variable expansion converts them to a single % .) The **%icsoutput** command takes its argument as a string, performs C style backslash escape processing, and sends the result to the ICS server, as if you had typed it in. In the example, the argument to the **%icsoutput** command is

```
"observe %user%\n".
```

The variable expansion substituted the substring **%user%** with the actual selected user from the selected line in the main listbox. Then the **%icsoutput** command turns \n into a newline. Assume for the moment that the selected line is

```
 16    Lippy             2306        2306        1759         2:02
```

Then the **%user%** is **Lippy** . The net effect is to send **observe Lippy** followed by a newline to the server.

The most useful command to use in button bindings is **%icsoutput** , but you could also use **%.** to execute a CHESSBD script, or ? use your imagination.

NOTE: This method of specifying the label and content of a button in one variable, is different from the method used in previous versions of the program, which used a separate variable for the label and content of each button. The old method will still work, but i encourage everyone to switch to the new method because it is less hassle.

NOTE: If you do customize the menu or the arena buttons, i recommend that you put the changed variables in a separate ini file --- as described in the distribution file **readme.txt** --- and add that file to CHESSBD's command line under the program manager. Your ini file MUST begin with the tag **[chessbd]** in order for the settings to be recognized. It must also have the extension ".ini".

ICS Match dialog
The ICS Match dialog lets you issue or respond to challenges. It pops up when you double-click on an entry in the ICS Arena dialog user listbox or the challenge listbox.

Within the match dialog, the "match" button issues an ICS challenge to the currently displayed opponent, with the currently displayed match parameters. The "accept" button issues an ICS accept command to the currently displayed opponent. The "decline" button issues an ICS decline command to the currently displayed opponent. The "cancel" button closes the match dialog without further action. The odd button labeled "r b 3 12" below the "opponent" value box sets the match parameters to "rated blitz 3 12". This is just the default value of the CHESSBD option variable "icsmatchdefaults". You can change it to your favorite set of default match parameters, using the Options dialog from the main CHESSBD menu bar. Then when you want to set all the controls to your defaults, just click that button.

ICS User sub-menu

The ICS control menu bar also provides a User-definable sub-menu. You can define up to 50 menu items. The contents of the menu are determined by the option variables **ics_lm00** - **49** . These work just like the user-definable buttons in the ARENA dialog. The label and action for the menu item *00* are defined by the variable **ics_lm00** . The label and action are separated by a vertical bar (|). For example, the line

```
ics_lm00=thanks|%%icsoutput say thanks\n
```

in your **chessbd.ini** file sets the first User menu item (numbered 00) to be **thanks** , which will execute the command **%icsoutput say thanks\n** when the item is selected. As with the user-programmable function key buttons in the ICS Arena dialog, the user definable menu item actions are subject to %variable% expansion (which is why the % symbol must be doubled to yield a single % character); and escape processing (which turns \n into a newline).

NOTE: If you do customize the menu or the arena buttons, i recommend that you put the changed variables in a separate ini file --- as described in the distribution file **readme.txt** --- and add that file to CHESSBD's command line under the program manager. Doing so will make it easier to maintain your custom settings when the next version of CHESSBD is released. Your ini file MUST begin with the tag **[chessbd]** in order for the settings to be recognized. It must also have the extension ".ini".

ICS Window sub-menu
Selecting an item in the ICS Window menu pops the corresponding window to the top. (ICS Telnet, ICS Chat, or ICS Arena.)

ICS Script Commands
When you connect to the ICS server, CHESSBD automatically executes the login script file that you specified in the ConnectInfo dialog. The script file can contain any CHESSBD commands, but there are a handful of ICS-specific CHESSBD commands that are particularly useful here. (These commands are also briefly described in the Commands section.) The **%icsoutput** command

```
%icsoutput STRING
```

sends *STRING* to the ICS server, as if you had typed it in. But first it does C style backslash escape processing to allow you to send special characters like newline (**\n**), tab (**\t**), and octal (*ooo*). The string is also echoed locally to the ICS Telnet window.

The **%icssend** does the same thing as **%icsoutput** , except it does not echo the *STRING* locally. (You might wish to use it for sending your password.)

The **%icsinput** command

```
%icsinput N string
```

monitors output from the ICS server and waits up to *N* seconds for the *STRING* to appear in the output stream from the server. As with **%icsoutput** , C style escape processing is performed on the string. If *STRING* fails to appear within *N* seconds, then the enclosing script is aborted.

See the sample **_ics.rc** file in the distribution for an example login script. NOTE the presence of the command

```
%icsoutput style 12\n
```

at the end of the file. You *must* use ICS *style 12* in order to use CHESSBD. You can put it in your **_ics.rc** file, or you can do it manually, but *you must use style 12*.

Note that the above commands do NOT do %variable% expansion. If you want to use %variable% expansion in your script, you can use the %eval command, which first performs %variable% expansion on the argument string, then executes the result as a CHESSBD script command. You can use the %eval command in conjunction with the **icsprofs.dat** file to create a single login script that will work on different machines. See the sample **dial_ics.rc** file in the distribution for an example login script using the "%eval" command.

A list of script commands that you may wish to use:

- %eval
- %icsinput
- %icsoutput
- %icssend
- %icsarena
- %icschat
- %icstelnet
- %icswin

ICS Board display
When you enter ICS mode, CHESSBD creates a new board window --- the ICS board window --- that overlays the CHESSBD main window. The main CHESSBD window still exists, and all its usual chess reader functions can be used, simply by moving the ICS board window aside or minimizing it.

The ICS board window works very similarly to the main CHESSBD window. (For more information about the main CHESSBD window, see Board Display .)

The ICS board window is used to play or examine games on ICS. It contains a chess diagram, plus 7 informational message areas, described below.

The chess diagram is the "heart" of the board window. It is normally oriented with "your" side at bottom, your opponent's side at top. (NOTE: if the board appears upside down, it's probably because you failed to supply your correct ICS handle in the ConnectInfo dialog. You can correct this by clicking on ICS File| ConnectInfo and entering the correct information, then click OK.)

The 7 message areas convey a lot of information, if you look carefully. They are:
* Your name (or ICS handle), which appears next to "your side" of the chess diagram.
* Your opponent's name (or ICS handle), which appears next to "the other side" of the chess diagram.
* Move number and side to move. If there is a material imbalance, that is shown in parentheses. The imbalance is shown relative to the side to move.
* Your clock display, which appears next to your name. The clock display shows the number of minutes and seconds you have left on your clock, in the format "MM:SS". When it is your move, CHESSBD updates your clock display every second, in the format "mm:ss >> MM:SS" The left "mm:ss" is your official time remaining as given in the most recent update from the server. The right "MM:SS" is the CHESSBD estimate of your time remaining. The numbers are preceded by a minus sign ("-") when your time remaining goes negative. It must be emphasized that this is only an estimate. This estimate could be a second too high or too low, because of granularity effects; or it could be too high by an arbitrary number of seconds due to lag.

When there is lag, you may notice that your clock display is still ticking even after you have made your move, and even though the "to move" message shows it is your opponent's move. That's because the server hasn't acknowledged your move yet. Until the acknowledgement appears, CHESSBD assumes the server may still be ticking down your real clock. In such cases, the display format changes to ">> MM:SS (lag=mm:ss)". Your time before the move is no longer shown. The left "MM:SS" is the estimated time left when you sent your move. (It does not include your "Fischer clock" time increment if any.) The right "mm:ss" is an estimate of the lag, ie, how long the server is taking to acknowledge your move. If you are using one of the timestamper programs, your time on the server should be close to the "MM:SS". Otherwise, your time could be as low as "MM:SS" minus the lag.
* Your opponent's clock display, which appears next to your opponent's name. When it is your opponent's move, CHESSBD updates your opponent's clock display in the same way as described above for your clock. Again, it must be emphasized that the right "MM:SS" is only an estimate.
* Miscellaneous server-supplied information, which appears at the top of the board window. This miscellaneous information includes the time control, the last move played, and the time taken for that move.
* Status alerts, which appear at the bottom of the board window. This includes notifications such as draw offers, resignation, and takeback requests. (NOTE: these alerts are erased when you make a move.)

If you are playing a game, and it is your move, you can make a move simply by clicking on a piece in the diagram and dragging it to the desired destination square. Your opponent's moves will appear on the board without your doing anything. (NOTE: if this does not work, it's probably because you forgot to enter the "style 12" command to the server. You can correct this by entering "style 12" in the telnet window at any time.)

Making moves
Making moves is done in the obvious way. Simply click on a piece in the chess diagram, and drag it to the desired destination square. Your move will be automatically sent to the server.

CHESSBD will not allow you to make illegal moves. When you first click on a piece, CHESSBD will highlight all its legal destination squares. Once you begin to drag the piece, those highlights will disappear. Instead, as you drag the piece, CHESSBD will highlight the square that it thinks you are moving to. There are some special cases:

* Capturing -- captured pieces are removed automatically.
* Castling -- just drag the King.
* Queening -- just drag the pawn to the 8th rank. This will pop up the promotion dialog. (NOTE: if you want to automatically promote to a queen instead of being prompted, then check the "autoqueen" checkbox in the ICS Arena dialog.)

All your games played on the main board are appended to the file **`icsgames.pgn`** (option variable icsgamefile). If you don't want this to happen, set option variable icssave to 0. NOTE: this game saving feature is very crude. The format is not 100% PGN compliant. And things like ICS takeback aren't handled. But at least it's a start. By default, other information such as challenges and chats are also recorded in the gamesfile. This is also controlled by the icssave option variable.

Connecting thru a firewall

If you have to connect thru a firewall machine, here is now to do it. Every situation is a little different, so all i can do here is give the general outline. The basic idea is that you will establish a telnet session to your firewall machine, login, and then execute on the firewall machine whatever command is needed to start a telnet session from the firewall machine to the ICS host.

(a) Start up CHESSBD and click on ICS|ManualLogin to bring up the ConnectInfo dialog.
(b) Fill in the ICS host field to the hostname of your firewall machine.
(c) Fill in the ICS port field to 23 (instead of 5000).
(d) Fill in your ICS handle and other fields as you usually would.
(e) Click OK. This will open a telnet connection to your firewall machine.
(f) Type Alt+M to pop up the TtyMode dialog. Click UX then OK. This step is just to make the echoing and prompting less confusing. You may skip this step if you are writing a login script.
(g) Use the Telnet window to login to your firewall machine.
(h) Once you have logged in to your firewall machine, enter firewall command to telnet to the ICS host, e.g., "telnet chess.onenet.net 5000" (or whatever is appropriate).
(i) Type Alt+M to pop up the TtyMode dialog. Click LCL then OK. This is to undo the effect of step (f). You may skip this step if you are writing a login script.
(j) Type control-E (^E) to turn off the telnet command's local echoing. This may or may not work. If it doesn't work, you may notice that everything gets echoed twice. Tough luck, try to ignore it. Things will work anyway.
(k) Use the Telnet window to login to the ICS server.

That's all there is to it!

Using ICC Timestamp
You can use ICC's anti-lag timestamp protocol with CHESSBD. There are two methods. The first method and the preferable one, is to use TMSTAMP, the Windows version of timestamp. You can do this if you are using SLICS over a SLIP/PPP connection. The other method is to use the UNIX version of timestamp, which will work if you must use SLICS over a straight modem connection.

Using Windows Timestamp
To use TMSTAMP, you must first install it. Simply move the TMSTAMP files into the same directory as CHESSBD. Then connect to " . Now click on "Winsock" and set the ICS host to **`localhost`** (instead of **`chess.lm.com`**). Fill in the rest of the fields and click OK. Soon you should see the beginning of the normal login screen in the ICS Telnet window.

Automatically Launching TMSTAMP
You can also setup a Profile Entry that will automatically start TMSTAMP. You can simply modify the sample ICC-TMSTAMP entry in the **`icsprofs.dat`** file in the distribution to contain your login information. Use your favorite text editor. The variable of interest is: **`icstmstampflag`** (which tells CHESSBD whether to launch the timestamp command). In old versions of CHESSBD, prior to 2.1g, you had to set a few other variables **`icstmstampcmd`** , **`icsficstmstampcmd`** , and set **`icshost`** to **`localhost`** . These are now obsolete, please delete them.

I decided to make it simpler. Now all you need to do is set **`icstmstampflag=1`** and the program will figure out the rest. If you wish to use a non-default client port (the default is 5000), just set the variable **`icstmstampport`** to whatever value you want. You might need to do this if you want to run multiple simultaneous instances of CHESSBD and TMSEAL or TMSTAMP. CHESSBD is now smart enough to know that if you have the timestamp option checked, it needs to connect to localhost and pass the value of **`icshost`** (and **`icsport`**) to the timestamper. It is now smart enough to know you need "tmseal.exe" if you are in FICS mode, otherwise "tmstamp.exe". In short, CHESSBD will now do the right thing. This includes passing the value of **`icstmstampport`** if any. You no longer need to mess with **`icstmstampcmd`** .

Problems
If you are using the manual launch method, but cannot connect to "localhost", try the IP address 127.0.0.1 instead. If you are using the automatic launch method, but cannot connect to "localhost" or 127.0.0.1, try setting the option variable "localhost" to your IP address.

Using UNIX Timestamp
The alternative method of using timestamp requires that you have a shell account on a local UNIX machine. The UNIX machine must be internet accessible and must be one of the supported types, ie, it must be capable of running one of the precompiled timestamp binaries.

The underlying concept is the same for all the clients: the timestamp program runs on your local UNIX machine and acts as an intermediary between your client program and the ICC. Effectively, timestamp emulates the ICC server software. Your client program, eg CHESSBD, can then connect to your local UNIX machine as if that machine were the actual ICC. Behind the scenes, the timestamp program communicates with the real ICC using a protocol that compensates for lag between it and the ICC host.

There is no absolute requirement that your local UNIX machine be truly "local" for this to work. However, the timestamp protocol is only able to compensate for lag between your local UNIX machine and the ICC; it is not able to compensate for lag between your PC and your local UNIX machine. Thus, for obvious reasons it is desirable for the local UNIX machine to be as local as possible.

First obtain the appropriate binary (as described in help timestamp). Then, whenever you wish to use the timestamp feature, you must perform some additional steps during the connection process. Essentially, you will treat your local UNIX machine like a "firewall". Read the section Connecting thru a firewall .

Follow the directions for logging in thru a firewall, but instead of executing the "telnet" command on the UNIX machine, execute the timestamp program.

Using FICS Timeseal

You can use FICS anti-lag program with SLICS. The instructions are the same as for ICC Timestamp, except for a few details. The FICS version is called Timeseal; the Windows version of timeseal is called TMSEAL; the profile entry for automatically launching TMSEAL is A-FICS-TMSEAL. (See Using ICC Timestamp above.) You can specify a different client portnumber with -p PORTNO. The default is 5000. The default server hostname for TMSEAL is "fics.onenet.net". You can specify a different host as the first command line argument to TMSEAL. You can specify a different host portnumber as the second command line argument. The default is 5000.

Example:

tmseal -p 5001 fics.onenet.net 5000

The above example uses 5001 as the client (local) portnumber. You would then connect to localhost port 5001 in the SLICS ConnectInfo dialog.

Using a Direct Modem Connection

SLICS was designed to be used with a SLIP/PPP connection, but you can also use it with a direct modem connection if you wish. You will have to enter the correct modem commands (eg, ATZ, ATDTnnn-nnnn, etc) to dial in to your service provider. If you're not sure how to do this, take a look at the settings in your comm program, or look at the sample "dial_ics.rc" script in the SLICS distribution. The "dial_ics.rc" script has commands that are typical for a Hayes-compatible modem. If you use this feature to connect, you may observe that your input is echoed when it shouldn't be. You can try toggling the FullDuplex checkbox in the TtyMode dialog. This may work, or it may cause other problems. Your best bet is probably to just get used to the extra echo. ICS File|ConnectInfo menu item from the ICS control window menu.

Getting started on ICS

The purpose of this section is to help rank beginners get started playing online chess. This is admittedly sketchy. I will describe "typical" steps you can follow, although the precise details will vary from user to user and server to server. CHESSBD has special support for a subset of the server's features. But remember this: you can access ANY and ALL server commands by typing in the ICS Telnet window . It is easiest to learn the commands that way to begin with, then once you know what is going on, you can use the CHESSBD interface with more confidence.

The first step is to fire up CHESSBD. If you are going to connect via SLIP or PPP, the next step is to start up your Winsock software and login to your ISP. Next you need to choose one of the chess servers and tell CHESSBD to connect to it. See ICS|ManualLogin . This pops up the ICS ConnectInfo dialog. Fill in the dialog then click OK. Assuming everything is working, this will establish a connection. (If you are using a raw modem connection, this will only connect to your modem! and you need to enter the modem initialization and dialing commands, in the ICS Telnet window.) Once the connection is established, you talk to the remote computer by typing in the ICS Telnet window. If you have to connect thru a firewall, see the section Connecting thru a firewall .

Now you can login to the chess server. If you are logging in to the ICC (chess.lm.com), enter your ICS handle and password as you are prompted. (Once you are comfortable with this aspect of things, it is highly recommended that you setup a login script. See ICS Script Commands , and look at **_ics.rc** for an example login script.)

If you don't have a registered ICS handle on ICC, you can enter "g" as your ICS handle, and enter as a "guest". (This only works on the ICC, not on FICS servers.) Then the server will assign you a unique handle once you have logged in. Once you get this handle, click on ICS File|ConnectInfo which brings back the ConnectInfo dialog, and fill in the ICS handle field with your assigned handle.

The first thing to do is enter the command "style 12". You must do this in order to play or observe games.

You are now ready to explore the server. The second command to learn is "help". "help" by itself gives you a list of server commands. "help COMMAND" gives you help for the given COMMAND.

Here are some commands you should look into.

help as above.

quit ends your session with the server.

set changes your profile on the server.

Example:

```
set open 0
```

means you are not open to receiving match requests. "set open 1" means you are open to match requests. There are many other profile variables you can set. Use help to learn what they are and what they do.

who tells you who is available to play a game.

Example:

```
who a
```

tells you who is available to play a game.

match issues a challenge. You specify WHO and what time control. WHO is given a challenge notification, which s/he may either accept or decline. That is the way games get started. You issue a

challenge with match and someone accepts, or someone else challenges you and you accept. (The ICS Arena has buttons to make these operations simpler.) Once the game gets started, the playing is easy. See the help topic ICS Board display .

Example:
    match XXXYYY 2 12
challenges XXXYYY to a game with 2 minute time control, 12 second increment.

accept accepts a challenge.

Example:
    accept YYYZZZ
accepts a challenge from YYYZZZ.

flag to claim a win (or loss) on time, during a game. The default CHESSBD setup has a "flag" button in the ICS Arena dialog.

abort during a game, asks your opponent's permission to abort.

draw during a game, sends a draw offer to your opponent, or if your opponent has offered you a draw, "draw" also accepts the offer.

observe lets you watch a game being played by others. You can see which games are currently being played by doing a "who v", or by pressing the "who ref" button in the ICS Arena dialog.

tell sends a communication to another logged in user.

Example:
    tell AAABBB let's play
sends the message "let's play" to user AAABBB. You can do the same thing with the ICS Chat dialog.

Good luck!

ICS mode variables

The following variables affect ICS mode. You can read about them in the Option Variables section.

- icsautoflag
- icsautoflagthresh
- icsautoqueen
- icsficsmode
- icsgamefile
- icsmatchdefaults
- icsnoslip
- icspromptpat
- icsquerylogout
- icsrcfile
- icssave
- icstmstampflag
- icstmstampport