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.

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.

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 %set the option variable wrapcol .

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

Options dialog
ICS dialog
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.

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 the (vacant) home square of the type of piece you want. 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.

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 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 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.

`%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 made invisible. Or if ARG is equal to the single character +, it is made visible. 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 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.

%icswin ***ARG***
Controls the visibility of the ICS control 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.)

%reverse
Flip the orientation of the board display.

%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.)

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.

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.

icschatautowrap (default: **1**)
Controls whether the ICS Chat textport will automatically change its wrapcol when resized. If non-0, the textport will automatically re-calculate the column number at which to wrap long lines, in order to keep incoming text within the window borders.

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. So by default, your games, incoming challenges, and incoming and outgoing chats are saved to the games file.

icswhocmd (default: **who v \\n**)
The command string sent to ICS when you click the "who ref" button in the ICS Arena dialog. Don't change it.

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

icswrapcol (default: **80**)
The column at which CHESSBD will automatically break ICS output lines in the ICS control window textport.

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.

weakplayersep (default: **/-**)
A list of characters that may be used to separate player names in an implicit game header of the weak form. Eg, **Karpov-Kasparov** or **Karpov/Kasparov** .

wrapcol (default: **30**)
Textport lines are automatically wrapped at this column number. Ignored if <=0. You should change this if you change the textport font.

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. **(:-)**

queryindex
querysavegame
silentpunct
sloppycheck
strict
weakplayersep
wrapcol

CHESSBD Use as an ICS Interface via Winsock TCP/IP

You can use CHESSBD to play chess on the Internet Chess Server (ICS) or the Free Internet Chess Server (FICS) via Winsock. I assume you already know how to use ICS and Winsock. This document only explains how CHESSBD fits in.

Throughout the 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 the Script Commands section for more info.

Quick Start with ICS

Using CHESSBD with ICS is easy. First start up your Winsock software and login to your SLIP/PPP server. Then run CHESSBD and click on the ICS item in the main menu bar.

If this is the first time you've used CHESSBD's ICS feature, the ICS ConnectInfo dialog will pop up asking you to supply ICS some related information. CHESSBD wants to know the host name of your ICS server, and your ICS handle. (If you don't have an ICS handle, don't worry, that information is only for parsing challenges.) Enter the correct values, then click on the OK button. If you know you will be using these same values over and over again, you can bypass this dialog next time you login to ICS, by checking the "skip this dialog" checkbox, then clicking the "save" button. If you ever want to use different parameters, you can call up the ICS ConnectInfo dialog again by selecting "File|Connect Info" from the ICS control window menu.

If all goes well, in a few seconds the word "connected!" will appear in the main CHESSBD textport, and the ICS login greeting will appear in the ICS control window. Use the ICS control window to login to ICS, just as you would with telnet.

You can also create an optional login script "_ics.rc" to make subsequent logins faster. I recommend that you use the example "ex_ics.rc" file supplied in the distribution as a template. Use your favorite text editor to modify "ex_ics.rc" appropriately, then rename it to "_ics.rc". Next time you login to ICS, your login sequence will be run automatically.

NOTE: If you don't have an "_ics.rc" file, you must issue the ICS "style 12" command manually after logging in to the ICS server. (CHESSBD will remind you to do this.) If you've set up ICS for a particular terminal type (such as vt100) you may also have to issue the commands "set highlight 0" and "set width 255" to the ICS server.

Once you've logged in to ICS, the ICS control window can be used as a very minimal telnet. Click on the control window and you can type in any ICS command. You can also call up the ICS Chat dialog to converse with other ICS users, or the ICS Arena dialog to arrange matches. The main chessboard display will be used for playing your games or examining a game. Extra board windows will be created for games that you are only observing. (NOTE: at this time, CHESSBD provides only minimal support for "examining" games; "wild" games are not supported.)

ICS Connect Info dialog
The ICS Connect Info dialog allows you to easily change CHESSBD variables whose values are host dependent. You can specify the server hostname, the port number, your ICS handle (login name), an optional login script (default _ics.rc).

You can also specify whether it is a FICS (Free ICS, as opposed to commercial ICS or ICC) server. Due to minor incompatibilities between FICS vs ICS, some CHESSBD features will not operate correctly unless this variable is set (checked) appropriately.

You can specify each of these parameters individually and manually, or you can instead choose from a list of canned profiles from the optional icsprofs.dat file.

Click the OK button to accept the currently displayed values. Click the Cancel button to exit the info dialog. Click the Save button to save the currently displayed values as defaults in your chessbd.ini file.

If you only use one ICS host, you can check the "skip this dialog" checkbox, then click Save. The next time you connect, the current values will be used without popping up the info dialog. If you later need to change any of these variables, simply select the

The dialog also has an experimental section for direct modem connections instead of winsock connections. If you click the "Modem" radio button instead of the "Winsock" radio button, you can fill in com port info and connect without using Winsock. But for the moment, you still need to have winsock.dll installed. 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 ICS|Mode|Full Duplex mode. This may work, or it may cause other problems. Your best bet is probably to just get used to the extra echo. ICS File|Connect Info menu item from the ICS control window menu.

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 Connect Info dialog. Since you can enter the same info manually, this file is optional, just for convenience.

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 A-ICS for the so-called American ICS 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 A-ICS .)

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 Connect Info dialog. That dialog only displays the "standard" host dependent variables, but you can put others in if you wish.

ICS Control window
The ICS control window consists of a menu bar and a textport. The textport functions as a very stripped-down telnet. All ICS output is echoed to the textport; and everything you type in the textport is sent to ICS (on a line-by-line basis). You can copy text from the textport to the clipboard simply by selecting it with the left mouse button.

The menu bar provides a File sub-menu, Chat item, and a Arena item.

While the ICS control window is open, some of the normal CHESSBD features are suspended. This includes most of the commands under the main window's File sub-menu. You can still type commands in the (main) textport, but the navigation dialog and the game selection dialog are disabled. Hint: even some of the normal CHESSBD commands which are available, may not work as expected, because while the ICS window is open, the board is partially under the control of the ICS server.

ICS File sub-menu

The ICS File sub-menu provides the following commands.

- ICS|File|Connect
- ICS|File|Disconnect
- ICS|File|Connect Info
- ICS|File|Logging
- ICS|File|Log flush
- 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 section.)

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

ICS File|Connect Info
Pops up the ICS Connect Info 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|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 control window, but then the output would scroll off as later messages came in.)

ICS File|Save layout
Saves the current layout of the ICS windows (the control window, the chat dialog, and the arena dialog). Next time you invoke the ICS feature, the ICS windows will be in the same locations as when you saved.

ICS File|Tty Mode
This dialog lets you set various tty-like options relating to the ICS control window. The Raw Output option enables/disables "raw output" mode. When Raw Output mode is OFF, the program will try to suppress the printing of style 12 messages. This will only work properly if you have set the "icspromptpat" variable properly. Otherwise the program won't know when to print partial lines (such as after a system prompt). Turn this option on if you want to see the style 12 output from the server, or if you are talking to an intermediate UNIX host (not the chess server). The Full Duplex option enables/disables full duplex mode. Turn this option on if you are talking to an intermediate UNIX host (not the chess server). The Local Echo option enables/disables local echoing. Turn this option off if you are talking to an intermediate UNIX host (not the chess server). The Line Wrap option enables automatic wrapping of long lines in the ICS control window. Purely a matter of taste.

The normal settings are Raw Output off, Full Duplex off, Local Echo on.

The Brk button causes a break to be sent if you are using a direct modem connection. (Not applicable if you are using winsock.)

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

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 control window textport). Everything you type in the ICS Chat textport is sent, prepended by the contents of the "Chat prefix" box. The default 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 control window textport. 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.)

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 control window.

ICS Arena
The ICS Arena dialog implements a convenient way to select opponents for challenges, or to select games for observing.

The ICS Arena dialog provides a main listbox of users, a "who ref" button, a bank of 16 "function key" buttons, and a challenge listbox.

The main listbox displays the output of the ICS "who v" command. Initially it's empty. To update it, press the "who ref" button.

The challenge listbox shows who has issued challenges to you during the current ICS session.

Double-clicking on an entry in either the main arena listbox or the challenge listbox, calls up a "match" dialog that lets you initiate or respond to a challenge. 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 the button.

The function bank buttons 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 issue a flag command, click "flag".

You can customize the binding 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 _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 "gameno" expands to the game number (in the seleced 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 %gameno%\n

The label for button 00 is "obs".
The action for button 00 is "%%icsoutput observe %gameno%\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 %gameno%\n".

The variable expansion substituted the substring "%gameno%" with the actual game number 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 "%gameno%" is 16 (and "%user% is Lippy). The net effect is to send "observe 16" 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.

ICS user sub-menu

The ICS control menu 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 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 ARENA buttons, the user definable menu item actions are subject to %variable% expansion (which is why the % symbol must be 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. Your ini file MUST begin with the tag

```
[chessbd]
```

in order for the settings to be recognized.

ICS Script commands
When you connect to the ICS server, CHESSBD automatically executes the script file "_ics.rc" if one is present. It can contain any CHESSBD commands, but There are several CHESSBD commands that are ICS-specific and particularly useful in the ICS login procedure. (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 control window.

The "%icssend" does the same thing as "%icsoutput", only without echoing locally.

The "%icsinput" command

```
%icsinput N string
```

monitors output from the ICS server and waits up to N seconds for the STRING to appear. As with "%icsoutput", C style escape processing is performed on the string.

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*.

ICS Main Chessboard
The main chessboard works as described in Board Display except that when you are playing a game, your opponent's moves appear without your doing anything. When it's your turn, you can move by clicking the left mouse button on the piece you want to move, and dragging it to the destination square.

The clock display is updated automatically. Of course, the running clock time shown between moves is only an estimate. The server is the ultimate authority on time left for both sides. (NOTE: when you make your move, the program updates the board position accordingly, but continues to decrement the estimated time of your clock, rather than your opponent's, until your move is acknowledged by the ICS server. This is a good way to tell if you are experiencing lag.)

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.

ICS mode variables

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

- icschatautowrap
- icsficsmode
- icsgamefile
- icsmatchdefaults
- icsnoslip
- icspromptpat
- icsquerylogout
- icsrawout
- icsrcfile
- icssave