Contents

Introduction

PIRCH is a one-man project that resulted from my desire to learn winsock programming. I usually write network based accounting applications and although it pays the bills, it can become tedious at times. PIRCH was a way to renew my programming interest and explore new areas. PIRCH was not originally intended to be released publicly, in fact it did not start as an IRC client, it started a desired to build a Bot from the ground up.

The name PIRCH (PolarGeek's IRC Hack) is sort of a misnomer in that no technical hacking actually took place, rather, knowing very little about how IRC functioned (wish I would have known about the RFC specs earlier), I approached this sort of like a hack watching how messages were sent between client and server and building upon that information.

The PIRCH project started in the early November of 1995 and I have enjoyed myself immensely as it progressed. I would like to thank the few alpha version testers for their participation, suggestions and patience.

Supporting This Project

PIRCH is not freeware. It is being released as shareware, so if you continue to use this software, and/or appreciate the effort put into PIRCH and would like to see it continue, I ask that you send the paultry sum of $10 US (check or money orders) to:

Northwest Computer Services
309 North Labree Suite #9
Thief River Falls, MN 56701

Failure to send in your ten bucks will result in absolutely nothing, however, I may take a greater effort in releasing new and improved versions faster if the development costs are paid. Like all user supported software, if users don't support it, it will cease to exist. Registered users who have difficulties using PIRCH may send E-mail questions to nwcomp@thiefriverfalls.polaristel.net. I will not answer questions from non-registered users.

License Agreement

I hate boring legal mumbo jumbo so I will try to keep this simple.

Warranty

This software is provided AS IS and isn't even guaranteed to work. Furthermore, if you use it, you can't hold me or Northwest Computer Services responsible for anything, either real or imagined. Period.

If you prefer to have a warranty, please send $50,000.00 to the address below. This will entitle you to a Limited Warranty in which you may recover damages NOT to exceed $2.50 for damages to your neighbors goats caused by PIRCH on any February 31st in which a full solar eclipse occurs at midnight. What a deal eh?

These are the only warranties either expressed or implied.

Supporting This Project

PIRCH is not freeware. It is being released as shareware, so if you continue to use this software, and/or appreciate the effort put into PIRCH (barring any lame jokes in this file) and would like to see it continue, I ask that you send the paultry sum of $10 US (check or money orders) to:

Northwest Computer Services
309 North Labree Suite #9
Thief River Falls, MN 56701

Your support will be greatly appreciated. Failure to send in your ten bucks will result in absolutely nothing, however, I may take a greater effort in releasing new and improved versions faster if the development costs are paid. Like all user supported software, if users don't support it, it will cease to exist. Registered users who have difficulties using PIRCH may send E-mail questions to nwcomp@thiefriverfalls.polaristel.net. I will not answer questions from non-registered users.

Internet Relay Chat (IRC) Overview

Internet Relay Chat (IRC) is network client/server system on which Internet users can 'talk' together using their computers. People gather in individual areas called channels. Some channels have a basic topic of discussion or are special interest groups. Many channels are simply dedicated to small talk or chat. And then there are those that can be used to get help on finding places of interest on the Internet, getting technical assistance with software problems, etc. You would be surprised at the number of people you can find on IRC that basically donate their time to help others.

IRC is basically divided into 2 large networks and several smaller networks. The two most popular networks are known as EffNet and Undernet. Both of these network systems have servers located throughout the world in various countries, maintaining hundreds to thousands of channels, and several thousand users at any given time; with EffNet being the larger of the two.

You login to an IRC network using an IRC server host and a nickname. This nickname is the name by which people on the network will refer to you. The PIRCH connection dialog box contains a number of server names for you to choose from and allows a place to enter your preferred nickname (and an alternate nickname in the event you first choice has already been taken by someone else). Note: It is generally best to connect to the server located nearest you.

Once you have connected, you can list the available channels on the network and join a given channel that looks interesting to you. When you join a channel, you will see a list of nicknames... these are the people who are already in the channel. Once you have entered... you can join in on the discussion or chat.

Channels are generally under the control of channel operators. Operators are experienced IRC users and their nicknames will be preceded by an @ symbol in the names list. If you have questions regarding the channel, you can usually direct them to the channel's operator(s).

How you behave on a channel depends on the type of channel you have joined. Many channels, like those dedicated to new users or a younger audiences prohibit profanity. It's up to you to find out what a given channel's rules are and to follow those rules. If you do not... the channel's operators have the power and ability to kick you out of the channel, and even prevent you from ever entering the channel again.

Many channels are controlled by special programs called bots (short for robots). The bots have the same powers as channel operators. These bots will show up on your channel names list just like any other IRC user. On the Undernet network, many channels are controlled by one of two bots that go by the names of X and W. If you see these user names, you can be assured they are bots and there really is no point in attempting to talk to them.

Connecting to IRC

The Connection Dialog Box allows you to select the desired IRC server and setup basic user information about yourself before logging in.

Selecting a nickname

There are also two fields allowing you to enter the nickname you want to use. PIRCH will always attempt to log into IRC using the name you provide in the Nick Name field, however, if PIRCH is unable to login with this name, generally because it is already being used by someone else on the network, it will use the alternate nickname. In the event both your primary and alternate nicknames are already being used by other people, PIRCH will display a dialog box prompting you to enter another choice.   Once you are logged into IRC, you can still change your nickname using the /nick command.

Selecting a Server

Click on the drop down list and select, or type in the name of the server you want to connect to. You can also edit the main server list by clicking on the Edit Servers button. All server names must be in the format of server.name:port, for example us.undernet.org:6667 where us.undernet.org is the name of the server and 6667 is the number of the port to connect to. (Most IRC servers use port 6667 or IRC connections)

PIRCH allows you to also specify a network ID in label following the server name. At this time, this ID label is purely informational for the user.

Real Name & E-mail Address

Enter your name and E-mail address in the fields provided. Your E-mail address ident (the part before the @ symbol) will be used to log into the IRC server.

Initial User Mode

Select the desired user modes. See the /MODE command for more information on individual modes and what they do. When you log into IRC, PIRCH will automatically Change your user mode to the selected preferences.

Loading & Saving Profiles

PIRCH allows you to have multiple profiles, which contain the user's preferences such as font settings, and other user settings. If there is more than one person who will use PIRCH in your household, you can each have your own profiles. Also, if you switch frequently between IRC networks such as EffNet and Undernet, you may want to setup an individual profile for each network.

To load a new profile, click on the Load Profile button and select the desired profile name from the file list. All PIRCH profiles end with the extension .ini. All changes you make to the user options will apply to this profile, and are automatically saved for you.

Editing The Server List

To permanently add or remove server names from the list, click on the Edit Servers button

Once you have entered the desired connection information, click on the Connect button. This action will display the server/status window for the connection and show the connection progress. As PIRCH logs you into IRC, the server will display the MOTD (Message of the Day) file.

The Server/Status Window

The Server/Status window display the server status connection, and a variety of information returned by various IRC commands. Each pane of the Server/status window has a popup menu that can be accessed by moving the mouse pointer over the pane on the window and clicking the right mouse button.

Server Command Bar

The top of the server/status window has a series of command buttons as follows...

Connect/Cancel

This button can be used to connect to the selected server. During a connection attempt the text on this button will change to Cancel, and you can abort the attempt by clicking on the button again. Once a connection is fully established, this button will be grayed out and disabled.

Disconnect

This button will allow you to disconnect from the network and is the same as issuing the /QUIT command. When selected, the system will send your Default Sign Off Message to the channels to which you are connected.

Channels

Use this button to retrieve a full channel listing from the network. This command is equivalent to the /LIST command. The channels names/topic & number of current users will be displayed in the channel list pane of the server/status window and can be filtered.

Join

The Join button allows you to join a select channel in the channel list and is equivalent to the /JOIN command.

Mode

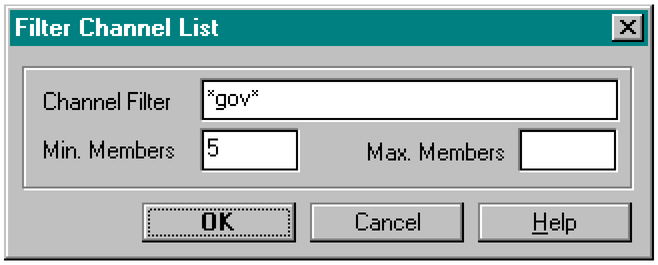
This button will display a dialog box allowing you to change your user modes and is equivalent to the /MODE command.

Server Drop Down Combo

This is a field showing the current server and contains a drop down list of the all servers you have set up. In addition, you may type in a new server name:port into this area, however it will not be saved to the main server list. To permanently modify the server list use the Server List command from the main menu.

Filtering the Channel List

You can filter the channels list so that channels you may be interested in are highlighted and grouped together at the top of the list. To select a filter, select the Channel List Filter command from the Channel List Popup menu.



Channel Filter: *gov*
Min. Members: 5
Max. Members:

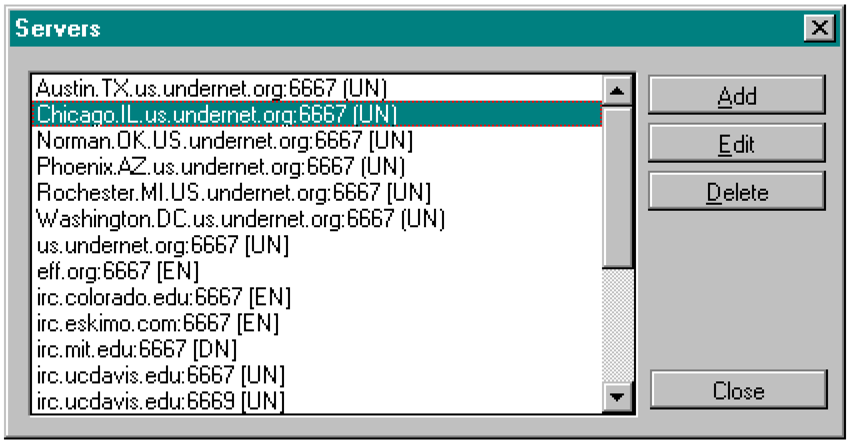
Channel Filter
This field can be used to filter the list according to text contained in the channel name. You may use the wildcard (*) character in the filter text. For example: *gov* would match any channel name the contains the text ‘gov’, ie. #govern, #government, etc...

Min. & Max. Members
These fields can be used to include/exclude channels from the filter based on the number of people in the channel. For example, enter 5 in the Min Member field and 25 in the Max. Members field means that the channels must have at least 5 people but not more than 25.

Each field in the filter dialog is optional; if all fields are left empty the channel list will simply be sorted alphabetically. Otherwise, all channels matching the filter criteria are sorted to the top of the channels list and are highlighted. You can filter the channels list as often as you like, searching for different criteria each time. Bear in mind however, that the number of members in the channels may change at anytime, and are only valid at the time the list was originally retrieved from the server.   The /LIST command also has filtering capability.

Editing the Server List

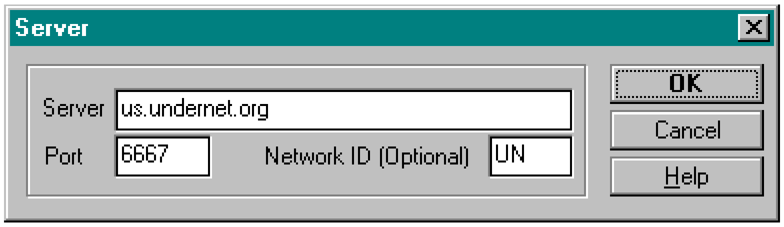
You can customize the list of available IRC servers and add your own list of new servers or IRC networks.



Austin.TX.us.undernet.org:6667 [UN]
Chicago.IL.us.undernet.org:6667 [UN]
Norman.OK.US.undernet.org:6667 [UN]
Phoenix.AZ.us.undernet.org:6667 [UN]
Rochester.MI.US.undernet.org:6667 [UN]
Washington.DC.us.undernet.org:6667 [UN]
us.undernet.org:6667 [UN]
eff.org:6667 [EN]
irc.colorado.edu:6667 [EN]
irc.eskimo.com:6667 [EN]
irc.mit.edu:6667 [DN]
irc.ucdavis.edu:6667 [UN]
irc.ucdavis.edu:6669 [UN]

Adding a new server

Click on the Add button to display an blank server entry window and enter the server name, port and optional network ID. Most IRC servers use port 6667, but many are now making additional ports ranging from 6666 to 6669 available.



Server: us.undernet.org
Port: 6667
Network ID (Optional): UN

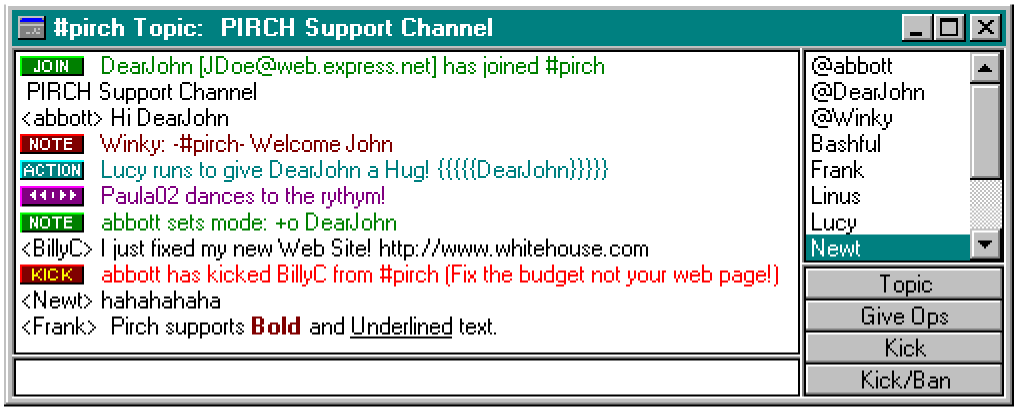
Editing and removing servers

To edit or remove a sever name from the list, highlight the desired name and click on the appropriate button.

Joining & Chatting in Channels

You can join a channel by either using the /JOIN command, or by double clicking the channel's name in the channel list of the server/status window. Each pane of the channel window has a popup menu that can be accessed by moving the mouse pointer over the pane on the window and clicking the right mouse button.

When the IRC server says its ok for you to enter the channel, PIRCH will open a new channel window. The windows caption will show the channel mode flags (see /MODE for information on mode flags) and the channel topic if any.


#pirch Topic: PIRCH Support Channel

```
JOIN    DearJohn [JDoe@web.express.net] has joined #pirch
PIRCH Support Channel
<abbott> Hi DearJohn
NOTE    Winky: -#pirch- Welcome John
ACTION  Lucy runs to give DearJohn a Hug! {{{{{DearJohn}}}}}
◀◀ ▶▶   Paula02 dances to the rythym!
NOTE    abbott sets mode: +o DearJohn
<BillyC> I just fixed my new Web Site! http://www.whitehouse.com
KICK    abbott has kicked BillyC from #pirch (Fix the budget not your web page!)
<Newt> hahahahaha
<Frank>  Pirch supports Bold and Underlined text.
```

```
@abbott
@DearJohn
@Winky
Bashful
Frank
Linus
Lucy
Newt
```

The command line is where you type what you want to say or enter commands. Text you type here will be sent to the entire channel unless it is a command. All commands in PIRCH begin with the / key, such as /msg or /notice. No text is sent until you press the <ENTER> key.   The command line retains a history of the last 25 lines of text or commands you have entered. You can recall items from the history list by using the up & down arrow keys.

Special Text Attributes

PIRCH also supports special text attributes, such as bold and underlined text. To use a special text attribute, use the keys <ctrl-b> for bold or <ctrl-u> for underlined text. Press the keystroke once to turn the effect on, and again to turn it off. For example: Typing: This is <ctrl-b>bold<ctrl-b> text. Will display as: This is **bold** text. Not all IRC clients can see special text attributes.

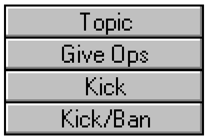
The Names List

Channel windows also have a list of names (by default located on the right side the channel window). These are the names of all other people who are in the channel. The names list has a popup menu that can be used to issue commands, and also contains any special aliases you have created. Generally, command accesses from this menu affect only the people whose names you have highlighted in the list. To highlight a name, simply click on it. To highlight multiple names, hold down the <ctrl> key and click on any desired names. You can also use the <shift> key to select a series of consecutive names.

Channel operators may also have the Operator Control Panel visible directly under the names list.

Operator Control Panel

The Operator Control panel is located directly below the names list in the channel window and allows channel operators to quickly issue commonly used commands.



With the exception of the Topic button, the control panel buttons will be disabled unless a name in the Channel's name list is selected. These buttons will affect ALL the people whose names are selected, so be careful.

Topic
The Topic button allows you to change the topic for the channel and corresponds to the /TOPIC command.
TOPIC

Operator/Give Ops/Take Ops/Toggle Ops
This button will allow you to change the operator status of the select people in the names list. The text on the button will change depending on whose names are selected in the list. This button corresponds to the /MODE +o command.

Kick
This button ejects the selected people out of the channel and corresponds to the /KICK command.

Kick/Ban
This issues the same action as the Kick button, but also issues a Ban (/MODE +b) prior to kicking the person(s).

Channel Operators

Channel operators are generally the people who are in charge of the channel. On IRC, having channel operator privileges allows the use of a number of additional commands.

/KICK	Kicks a person out of a channel
/MODE	Used to change channel modes

PIRCH allows channel operators to also issue common operator commands from a control panel located in the channel window.

Having Private Conversations

IRC allows you to have private one on one conversations with other people on the network. This is done by sending private messages using the /MSG & /NOTICE commands. When another party sends a private message to you, PIRCH will open a new message window. You can then hold the private conversation by simply typing in that window, as any text you enter in that window is sent only to that person. Each private message window will display the nickname of the person to whom the conversation is with in the caption. You can also use the /QUERY command to initiate a private conversation with another party, this command will immediately open a private message window.

You can also have private conversations using the DCC CHAT features. DCC allows a greater degree of insured privacy, since the conversation is not passed through the IRC network. You can start a DCC CHAT conversation by either using the /DCC CHAT command, or by selecting the person name from a channel names list, right clicking the mouse and selecting the DCC CHAT command from the popup menu.

KPersonal Bio

Personal Bios (biographies) are a method by which you can share information about yourself with other people you meet over IRC and vise versa. The Person Bio in PIRCH is a combination of both text & graphics in one small file.

To create a a BIO file for yourself, select the Personal Bio option from the Options menu. This will open up the bio work area. You may include a picture of yourself in the bio. Currently, it is best to crop/resize the picture to 100x125 pixels. Its also recommended that you limited the image type to a 256 color palette image rather than a 24bit true color image. This will allow the best possible picture quality on a greater variety of systems. To insert the picture, simply click on the photo frame area and select the photo you wich to have inserted into the bio from the file list. PIRCH currenlty supports on .BMP (bitmap) files, but JPG/GIF support may be provided in upcoming versions.

The text area of the bio setup form can be used to enter any desired information you want to tell about yourself, such as age, occupation etc. You may include approximately 32k of text in your bio if you so choose.

PIRCH includes a special command for sending your bio to other people call /BIO SEND.

KAutoexec Commands

Autoexec commands are commands that can be execute automatically each time you connect to an IRC server, such as joining a particular channel.

To add autoexec commands, select Autoexec Commands from the Options Menu and enter each command individually... one per line.

KPrinting

PIRCH allows you to print information from channels, private message windows and from DCC chats. To print, go to the IRC menu and select the print option. Depending on the currently active window, it may take a few seconds or so for PIRCH to re-wordwrap and recalculate the number of pages of printed text that the scroll back buffer contains. This is necessary so that you can select a page range, and not be force to print the entire buffer.

You can also print only a selected block of text. To do this... Click and highlight the desired text with the mouse while holding down the ctrl-key. Release the mouse button while the ctrl-key is still down and the selected text will remain hilighed. Now you can go to the IRC | Print menu and the “selected only” option will be available in the print dialog.

KFile Server

The file server allows you to make available files for others to download, without much interaction from yourself.
To set up a file server, goto the tools menu and select the file server options. The first thing you need to do is specify the default options. Click on the Server Setup button and enter the desired values for each of the options.

Idle disconnect - this is the amount of time a user may be in the file server without activity before he is automatically disconnected.

Default Max Gets - This is the maximum number of files a use may download during a single session. When the maximum number of downloads is achieved the user will be disconnected from the file server.

Default Home Dir - Is the root directory a user may access. The user will be allowed to browse through subdirectories of this directory.

Sign On Msg file - This is a standard text file that will be displayed to a user when they log into the file server.

Once the options are complete. You are ready to allow users onto the file server. A user may type HELP and get a list of available commands. When a user requests a file, it will be sent to them using DCC.

In order for someone to access your file server, you must grant them permission using the /faccess command.
Example: /faccess JohnDoe c:\public 5

You want to have more automation, you can use the Events section of PIRCH to automatically handle granting of file server access. For example... to allow anyone access to you file server

```
ON NOTICE:*!fileserver:*:/faccess $nick c:\public 3
```

Then a user can simple type /notice younick !fileserver to gain access. The usage of such an event is entirely up to you.

Setting Options

PIRCH allows you to set a variety of options that determine how PIRCH looks and operates. All options are stored within the active profile .ini file. To change the option settings, select Options|Preferences from the main menu to display the settings dialog box.   This dialog is divided into various different sections indicated by the tabs at the top.

General
Ops
Sound
DCC
CTCP
Fonts
Flood
Logs
Hilite
Notify
Ignore
Auto Op
Protect
Misc

General Options

Private Notice Destination
This option determines where messages sent to you from another person using the /NOTICE command will appear.

Private Messages/Notices
Minimize New Message Windows
This option makes new incoming messages windows minimized on the PIRCH desktop.

Suppress Message Interrupts
Activating this option will prevent a new incoming private/message notice from popping up and becoming active immediately. Instead the window will be created, but the input focus will remain in the currently active window.

Application
Hide Window Tabs
Selecting this option will hide the window tabs at the bottom of the screen and makes more room for windows on PIRCH's desktop.

Connections
Server Connection Timeout
This field determine how long PIRCH will wait for a server to respond to a connection request. Best results are usually obtained with values ranging from 30 to 60 seconds. If PIRCH is unable to connect within this amount of time, you'll receive a message in the server/status window and you can then select a different server and try to connect again.

Operator Options

This section of options deals with setting for channel operators. None of these options have any affect if you do not channel operator privileges.

Show Op Control panel when Ops are granted

Activating this option will cause the channel operator control panel to appear when ever you are given channel operator privledges.

Always confirm kicks

This option will cause PIRCH to display a dialog box asking you to confirm that you want to kick the targeted user(s) from the channel. NOTE: This option affects only kicks issued through the Kick button of the operator control panel, no confirmation will be required when using the /KICK command.

Prompt for kick message

This option will cause PIRCH to prompt for the reason the target user(s) are being kicked from the channel. This message will then override the default kick message.

Deop prior to kick or ban

At times it may be useful to remove operator privledges (if the targeted user has them) prior to kick or banning them from the channel. This is equivalent to setting /MODE -o.

Default Kick Message

This message will be sent to the kicked party stating the reason they were kicked. This can be overridden by setting Always Prompt for message.

Ban Method

This option determines how PIRCH sets a ban against a user. A person is identified on IRC by his nickname and his user@host information. Each part of a ban mask may contain a wildcard (*) that substitutes for any literal text.

Sound Options

Default Sound Dir

This is the directory where PIRCH will look for sound files used in conjunction with the /sound command

Ignore if already playing sound

This option can be used to sounds messages that arrive close together (before the first sound has completely finished playing) from stopping each other.

Note different types of sound files generally do not interrupt each other. For example, a MIDI sound which is currently playing will never be stopped when a sound message for a wave file arrives. The wave file will play right along with the MIDI file.

DCC Options

Auto Accept DCC Transactions

If this option is enabled, PIRCH will automatically accept any DCC transaction requests from other parties.

Auto Minimize DCC Windows

If this option is enabled, PIRCH will automatically minimize DCC windows when connections are established.

Auto Close DCC Windows on Completion

If this option is enabled, PIRCH will automatically close DCC windows when the transactions are complete or the connection is lost.

Packet Size

This field determines the buffer size used to conduct a file transfer. In general, larger buffer sizes may yield faster transactions. If you experience problems transfering files, you should attempt to lower the DCC packet size.

TimeOut

These fields determine the length of time PIRCH will wait when attempting to make a DCC connection to another party. If PIRCH is unable to connect within this length of time, the transaction fails and the window will close.

Default DCC paths

These are the default upload and doanload paths for DCC file transfers.

CTCP Options

Enable CTCP
Enables/disables acknowledgement of messages sent to you client via the CTCP protocol.

Default CTCP Responses
Enter the text here that you wish to have PIRCH send as responses to the CTCP FINGER and CTCP USERINFO commands respectively.

Font Options

The font options part of the options dialog allows you change the default fonts used by PIRCH. To change the default font for a particular window type, click on the sample font pane. This will bring up the font selection dialog where you may select the default font, size & style settings.

Flood Control Options

IRC limits the number of messages that can be sent from your clinet within a specified amount of time. If you send too many messages too fast, the IRC server will disconnect you from the network for 'flooding'. PIRCH has an internal mechanism to control flooding.

Enable Flood Control
Select this option if you want to use PIRCH's internal flood control feature.

Flood Delay Rate
This amount of time PIRCH waits between queued message delivery in milliseconds. The default setting is 500ms or ½ second.

Flood Limit
This number determines if/when PIRCH should wait additional time between sending messages. The default setting is 4.

With the default settings, PIRCH will deliver 4 messages within 2 seconds before any additional delays in message delivery.

Log Options

Auto Log
Select which types conversations you wish to have logged to a text file. The logs will use the name of the channel or person as the filename.

Log Directory
Indicate the name of the directory where you wish to have the log files created.

Highlight Options

The Highlight list is used to determine whether a particular message should be highlighted in the message pane of channel and private message windows.

Most commonly, people add their nick name or variations of their nickname to the highlight list so that messages directed at them are displayed in a different color.

Notify Options

The notify options is used to main a list of nicknames. When a person on this list joins or leaves the IRC network, PIRCH will display a message in the server/status window.

Protect Options

The protect options is used to maintain a list of nicknames which are considered to be protected users. If another user deops, kicks or bans a user on your protect list. You client will react by reversing the action taken. PIRCH never takes action if both persons (the person preforming the action and the person affected by the action) are on your protect list.

Ignore Options

The ignore options is used to maintain a list of nicknames which are considered to be ignored users. If a user is placed on your ignore list, no messages sent by the user to either channels or private messages are displayed to you.

AutoOp Options

The AutoOp options is used to maintain a list of names/addresses that you will automatically given channel operator status when they join channels which you are on.

Misc Options

Hide Server Ping/Pong
This option suppresses the display of PING messages sent to you by the server to which you are connected. Severs do this in order to make sure your connection to the network is still active and working. If this option is active, PIRCH will still respond to all server pings, but simply will not display that it has don so in the server/status window.

Enhance Events Mode
This option enables the display of the events glyphs. If you are on a slow system or prefer not to have the glyphs displayed next to events you may disable this option.

Auto Join Channel on Invite
This option will cause PIRCH to automatically join a channel if an invitation is received.

Auto Rejoin if Kicked
PIRCH will automatically attempt to rejoin a channel if you are kicked from it when this option is active.

Hide Own Nickname in Chat
This option suppresses the display of your nickname in channel/message & DCC chat windows. If selected, the text sent in these windows will simply be prefixed with >.

Auto Tile Windows
This option cause PIRCH to automatically tile the windows on the desktop when a new window is created.

Beep on Inactive Msg
This option will cause a beep each time a message is sent to an inactive window or when a message is received while PIRCH is running in the background.

Whois on Private Msg
Activating this option will automatically do a /WHOIS on any person the sends you a private message.

Auto Rewrap Text
This option will automatically cause text to be rewrapped in a window when the window is resized.

Default Sign Off Message
This message will be displayed to other people in the channels you are on when you click on the Disconnect button in the server/status window.

DCC Extension Map

This option is useful when you want to have your DCC Downloads stored to different directories based on the type of file. PIRCH uses the 3 character extension to determine file type.   Entries in the DCC Extension map should be seperated by a semi colon for similar type files. Extension types not found in the Extension Map will be stored to the default DCC Download directory.

Examples	bmp;gif;tif;jpg	c:\images
	mid;rmi;wav	c:\sounds
	txt;doc;wpd	c:\docs
	zip	c:\archive

Supported Commands

The following is a list of currently supported commands.

- /?
- /ADDUSER
- /ADMIN
- /AWAY
- /ACHAN
- /ASERV
- /AWIN
- /BAN
- /BEEP
- /BIO SEND
- /BIOVIEW
- /CALLBACK
- /CASCADE
- /CHLEVEL
- /CLEAR
- /CLOSE
- /CTCP CLIENTINFO
- /CTCP VERSION
- /CTCP FINGER
- /CTCP USERINFO
- /CTCP PING
- /CTCP TIME
- /DCC CHAT
- /DCC GET
- /DCC SEND
- /DCC RESUME
- /DCC ACCEPT
- /DEFINE
- /DISABLE
- /DISPLAY
- /DNS
- /ENABLE
- /EXECREAD
- /EXIT
- /FACCESS
- /FETCH
- /FINGER
- /FILECOPY
- /FILEDEL
- /FILEMOVE
- /FLUSH
- /HELP
- /IGNORE
- /INFO
- /INVITE
- /ISON
- /JOIN
- /KICK
- /LIST
- /LINKS
- /LOGIN
- /LUSERS
- /NAMES
- /MAP
- /MAX
- /MIN
- /MODE (Channels)
- /MODE (Users)
- /MOTD

/ME
/MSG
/NEWWINDOW
/NICK
/NOTE
/NOTIFY
/NOTICE
/OPER
/OPMSG
/OPNOTICE
/PART
/PING
/PLAYFILE
/PLAYMEDIA
/PLAYPAUSE
/PLAYRESUME
/PLAYSTOP
/PRIVMSG
/QUERY
/QUIT
/REMUSER
/RESTORE
/RUNSCRIPT
/SAVEBUFFER
/SERVER
/SET
/SILENCE
/SOUND
/STATS
/TILE
/TIME
/TIMER
/TOPIC
/TRACE
/UNDEFINE
/USERHOST
/WALLOPS
/WHO
/WHOIS
/WHOWAS
/WRITE
/WRITEINI
/VERBOSE
/VERSION

/?

Usage:	/? [topic]
Example:	/? /who

/? can be used to quickly locate a topic in this help file. If the topic parameter is omitted, PIRCH will display a search dialog box and list of available topics.

/ADMIN

Usage: /admin

Retrieves administration information from the IRC server you are currently connected to. Generally this administrative information will contain E-mail addresses of the persons responsible for running the server.

/ADDUSER

Usage:	/ADDUSER [-q] <Level,[[level],...]> <nick[!user@host]> <maskmode>
Example:	/ADDUSER 100 Billy 2 (adds *!BillG@*.microsoft.com to event level 100)

This command is used to add a user to the user list of an event level. The -Q parameter indicates the command should be done without displaying the change in the server/status window.

Level indicates the level(s) to which the user should be added. The level name may be a partial name, or wildcarded in which the all matching levels are affected. You may add/remove from several levels at one time by listing each level seperated by a comma (do not separate with a space).

nick!user@host is the nickname or internet address of the person you are adding/removing. The address portion of this command is optional, in which case PIRCH will retrive the person's internet address for you via a /WHOIS command.

MaskMode is the address masking/wildcarding options which should be applied to the person's address prior to adding to the list. There are 6 available masking options.

Mask Type	**Resulting address mask**
0 = User	Billy!BillG@ppp01.microsoft.com
1 = IdentPortServer	*!BillG@ppp01.microsoft.com
2 = IdentServer	*!BillG@*.microsoft.com
3 = PortServer	*!*@ppp01.microsoft.com
4 = NickOnly	Billy!*@*
5 = serverOnly	*!*@*.microsoft.com

see also: /REMUSER

/ACHAN

Usage:	/ACHAN <COMMAND>
Example:	/achan /me is away

/ACHAN can be used to sends text or issues a command on all the channels you have joined.

See also:	/ASERV
	/AWIN

/ASERV

Usage:	/ASERV <COMMAND>
Example:	/aserv /away Gone to get something to eat.

/ASERV command can be used to issue a command to each server to which you are connected.

See also:	/ACHAN
	/AWIN

/AWIN

Usage:	/AWIN <COMMAND>
Example:	/awin /me is away.

/AWIN can be used to send text or issue a command for each channel you have join an each private message window that is open.

See also:	/ACHAN
	/ASERV

/AWAY

usage:	/away {message}
examples:	/away I am away.... but don’t panic... I will return shortly
	/away

Marks you as being away from your computer. Use this command to have the server automatically send the away message you entered to any users that send you private messages while you are away from your computer.

To clear the away... use /away again with no message or /unaway.

/BAN

Usage:	/BAN <#channel> <nick[!user@host]> <maskmode>
Example:	/BAN #pirch Billy 2

The /BAN command is used to ban a particular nick, thereby preventing them from joining the specified channel. This is similar to performing a /MODE #channel +b <nick!user@host>

If you specify a nickname only, PIRCH will search for the address information in its internal address list. If not found, PIRCH will retrieve the user@host information by performing a /WHOIS.

MaskMode is the address masking/wildcarding options which should be applied to the person's address prior to perfoming the actual ban. There are 6 available masking options.

Mask Type	**Resulting address mask**
0 = User	Billy!BillG@ppp01.microsoft.com
1 = IdentPortServer	*!BillG@ppp01.microsoft.com
2 = IdentServer	*!BillG@*.microsoft.com
3 = PortServer	*!*@ppp01.microsoft.com
4 = NickOnly	Billy!*@*
5 = serverOnly	*!*@*.microsoft.com

/BEEP

usage: /BEEP

This command simply causes the a simple audio beep sound.

/BIO SEND

usage:	/BIO SEND <nickname> [biofilename]
example:	/bio send jane45

/BIO SEND can be used to send your personal bio information to another person. Note that personal bio's are PIRCH specific feature and both you and the person you are sending to must be using pirch. Other clients will not be able to load/view a bio file.

<Nickname > is the name of the person you are sending to.
The <biofilename> parameter is optional and can be used to deliver a bio you received from someone else to another person.. If you do not include this parameter, PIRCH will attempt to deliver the your default bio file called user.bio.

BIO files are transfered over irc. When a bio is successfully delivered, the PIRCH will automatically display the file to the person it was sent to.

/BIOVIEW

usage:	/BIOVIEW <nickname|filename>
example:	/bioview jane45

/BIO VIEW is used to open the personal bio viewer.   When a bio is received to your computer system, it is generally stored under the nickname of the person to whom it belongs. For example, Jane45's bio would be stored under the name jane45.bio. The bio viewer also contains a drop down combo box which you can use to view other personal bio's stored on your computer.

/CALLBACK

usage: /callback <server rply code> <PIL Script Name>

Use the /callback to install a callback script to handle a particular server reply code.

/CALLBACK is the probably the most advanced, and therefore, unfortunately most complicated command to understand and master. Basically /CALLBACK allows you to install a script to handle a variety of incoming server messages. This is similar to what PIRCH events do, and in most cases use of simple ON XXXXX is recommended. However, not all server messages are accessible in events and thats where use of these server callbacks can become useful.

For example, when you type /whois <nickname>, the IRC server returns up to four (4) distinct messages to PIRCH, and PIRCH will normally display the information for you in the server/status window. But lets assume you want to do something with this specific with this information; and since there is no ON XXXX event which covers the information returned by /whois, you can install a PIL script to handle/manipulate the returned information as desired.

Obviously to make use of this command and implement callback features you need to have information about server replies/messages and the format of these messages, all of which can be found in RFC1459 which is the official RFC protocol specification for IRC. Albeit to some degree out of date, the RFC document is still the most reliable and informative source. This document can be obtained from all complete RFC list sources and at last check was available at ftp.undernet.org

When a callback script is installed, PIRCH passes all the information contained in the server reply/message to the script as a parameter, when PIRCH encounters the specific reply/message code. This information can then be disseminated by the script in whatever fassion you desire.

/CASCADE

usage: /cascade

Use the /cascade command to arrange the windows on the desktop in an overlapping fashion.
Does not affect any windows marked as ‘Keep on Top’.

/CLEAR

usage: /clear

Use the /clear command to clear all the text from the active window.

/CLOSE

usage:	/close [windowname]
examples:	/close
	/close #pirch

Use the /close command to close a window. If the windowname parameter is supplied then the specified window will be closed, otherwise the affectected window will be the active window.

/CTCP CLIENTINFO

usage: /ctcp <nickname> clientinfo

This command is used to retrieve the name and version of the IRC client <nickname> is using. PIRCH automatically responds to ctcp version requests by returning its current version information.

See also: Other CTCP Commands

NOTE: No reply will be sent if CTCP has been disabled in your options.

/CTCP FINGER
usage: /ctcp <nickname> finger

This command is used to retrieve the finger information from another client.   Some clients may not respond to this query. Generally, clients will return a person’s E-mail address, however, PIRCH will return the information you entered in the Finger Reply field of the CTCP options.

NOTE: No reply will be sent if CTCP has been disabled in your options.

/CTCP TIME

usage: /ctcp <nickname> time

This command is used to retrieve the time of day information from another client.   Some clients may not respond to this query. PIRCH will return the date and time according to your computer.

NOTE: No reply will be sent if CTCP has been disabled in your options.

/CTCP USERINFO

usage: /ctcp <nickname> userinfo

This command is used to retrieve some information about the user of another client.   Some clients may not respond to this query. PIRCH will return the information you entered in the User Info Reply field of the CTCP options.

NOTE: No reply will be sent if CTCP has been disabled in your options.

/CTCP VERSION

usage: /ctcp <nickname> version

This command is used to retrieve the version number of the client used by another person.   Some clients may not return a version number.

NOTE: No reply will be sent if CTCP has been disabled in your options.

/DCC ACCEPT

This command functionality is handled automatically within PIRCH.

/DCC CHAT

usage:	/dcc chat <nickname>
example:	/dcc chat jane45

The /DCC CHAT command allows you to establish a Direct Client to Client connection with another user for a private conversation. DCC Chat connections do not pass messages across the IRC network and for this reason are considered more secure than simply using the /MSG command.

In addition, DCC chats are not constrained to the IRC flood limits and therefore are more useful for displaying larger amounts of text i.e. displaying a text file with the /PLAYFILE to another user for example.

/DCC GET

This command functionality is handled automatically within PIRCH. When another user attempts to send you a file using the /DCC SEND command, PIRCH will automatically display at dialog prompting you to accept or reject the file, eliminating the usefulness of this command.

/DCC RESUME

This command functionality is handled automatically within PIRCH. When another user attempts to send you a file using the /DCC SEND command, and the file already exists on your system, PIRCH will automatically display at dialog prompting you to accept, resume or reject the file, eliminating the usefulness of this command.

/DCC SEND

usage:	/DCC SEND <nickname> <filename>
example:	/dcc send Jane45 c:\pirch\readme.txt

The /DCC SEND command is used to transfer files through a DCC connection to another user on IRC. In addition to the /DCC SEND command, you may use PIRCH's DCC Sender dialog or the Drag-N-Drop DCC Sender.

/DEFINE

usage: /DEFINE <variable>=<value>

/DEFINE can be used to create and set user defined variables. All user defined variables are persistent and exists for the duration of the IRC session (until PIRCH is closed) or until the variable is destroyed with the /UNDEFINE command.

Each variable should be undefined whenever the variable is no longer needed to that any memory it occupied is freed.

/DISABLE

usage: /DISABLE [-q] [EventLevel]

Examples:	/disable 0	disables all levels that start with the character 0
	/disable *	individually disables all levels
	/disable	disables the events system

Disables an event level. The level name may be a partial name, or wildcarded in which case all matching levels are affected. If no parameter is supplied, the event system is enabled.

The -q parameter indicates no status message should be displayed.

see also: /ENABLE

/DISPLAY

usage: /DISPLAY [> windowname] <text>

/DISPLAY is used to display information to yourself in the server status window. This command does not generate an IRC message and noone but you will be able to see this message.

You may direct text to a particular window using the > windowname parameters. If a particular window with the supplied name can not be found, the text will be displayed in the server/status window. For example... if you create a window called *WALLOPS* using the command /newwindow *WALLOPS* , you can then redirect WALLOP messages to that particular window, rather than having them displayed in the server/status window, by adding the following event:

```
ON WALLOPS:*:/display > *WALLOPS* \10 $+ $nick $+ *1
```

SPECIAL ENHANCEMENTS

You can include several enhancements for /display that will change the way the text appears on your screen.

Character Attributes

Bold	\-2
Hilighted (special)\-5	
Italic	\-22
Underline	\-31

Message Type Tags

NOTE	\-1
JOIN	\-2
PART	\-3
NOTE	\-4
NOTE	\-5
KICK	\-6
NOTE	\-7
INFO	\-8
NOTE	\-9
OPS	\-10
◀◀ı▶▶	\-11
CTCP	\-12
CTCP	\-13
ACTION	\-14
ERROR	\-15
INFO	\-16
AWAY	\-17
IRCOP	\-18

/DNS

```
usage:   /DNS <nick|[user@]host>
         /dns jane45
         /dns ppp27.server.com
```

The /DNS command can be used to lookup an Internet protocol address or domain name. The results of the DNS will show the name and IP number as reported by the domain name server.

/ENABLE

usage: /ENABLE [-q] [EventLevel]

examples:

```
/enable 0            enables all levels that start with the character 0
/enable *individually enables all levels
/enable              enables the events system
```

Enables an event level. The level name may be a partial name, or wildcarded in which case all matching levels are affected. If no parameter is supplied, the event system is enabled.
The -q parameter indicates no status message should be displayed.

see also: /DISABLE

/EXECREAD

usage:	/execread [-L#] <filename>
examples:	/execread c:\pirch\greets.txt
	/execread -L7 c:\pirch\kicks.txt

/EXECREAD can be used to read and execute a line read from a file composed of aliases. If the -L parameter is not supplied, PIRCH will read a random line, otherwise the line number indicated following -L will be read. The first line number is 1.

/EXIT

usage: /exit

/EXIT closes all server connections immediately and exits the PIRCH application.

/FACCESS

usage: /faccess <nickname> [homedir] [maxgets]

example: /faccess Jdoe c:\public 5

/FACCESS grants a user access to your file server. If homedir is omitted the the default home directory set up in the file server options will be used. Maxgets is the maximum number of files a user may download in a single session. If omitted the default maxgets in file server options will be used.

/FETCH

usage:	/fetch [url]
examples:	/fetch
	/fetch http://cnn.com/index.html

As of the writing the /FETCH specifications have not been completely developed.   /FETCH is a information retrieval command which currently supports HTTP protocols, and is capable of parsing & displaying HTML web pages in PIRCH's internal Hypertext Electronic Noticeboard (HEN).

By default, use of /FETCH without any parameters will retrieve a channel MOTD (Message of the Day) for the active channel from a default WWW server. Not all channels have registered MOTD's with the us. If you would like to have an MOTD for your channel, please contact us for registration information.

/FILECOPY

usage:	/filecopy <sourcefile> <targetfile>
example:	/filecopy c:\pirch\ircfaq.txt c:\documents\ircfaq.txt

/FILECOPY copies a file. Due to windows multi-tasking nature, there is never a gaurantee that another application hasn't changed the system's current directory and PIRCH makes no assumptions about what the current directory should be... therefore, you should always use fully qualified filenames, including the drive and directory in the source & target filename parameters.

NOTE: You may NOT use wildcards, uou must specifiy the exact filenames.

see also: /FILEMOVE
/FILEDEL

/FILEDEL

usage:	/filecopy <filename>
example:	/filedel c:\irc\oldclient.exe

/FILEDEL can be used to erase a file. NOTE: You may NOT use wildcards, uou must specifiy the exact filename.

see also:	/FILECOPY
	/FILEMOVE

/FILEMOVE

usage:	/filemove <sourcefile> <targetdirectory>
example:	/filemove c:\pirch\ircfaq.txt c:\documents

/FILEMOVES a file from a source to a target directory. Due to windows multi-tasking nature, there is never a gaurantee that another application hasn't changed the system's current directory and PIRCH makes no assumptions about what the current directory should be... therefore, you should always use fully qualified filenames, including the drive and directory in the source filename & target directory parameters.

NOTE: You may NOT use wildcards, uou must specifiy the exact filename.

see also: /FILECOPY
/FILEDEL

/FINGER

usage:	/finger <user@host>
example:	/finger jdoe@telco.net

/FINGER can be used to look information about a user, generally by their E-mail address, using the UNIX FINGER protocol. The information returned by this command will depend on what information each individual host wishes to make public about their users.

/FLUSH

usage: /FLUSH

This command can be used to flush any pending messages from PIRCH's internal flood control queue, and can only be used if your flood control option is enabled.

/HELP
usage: /help

Retrieves a list of valid commands from the server to which you are connected. The list of commands will be shown in the server/status window.

/IGNORE

usage:	/ignore [-d] <nick!user@host.mask> <mode>
examples:	/ignore johndoe 2
	/ignore jdoe@*.net.com

If you want to ignore messages sent by some other user or users, it may be done with /ignore command. You can ignore someone by their nickname, or by their user@host data. Wildcards may be used. /ignore ignores both private and channel messages/notices from a party until you remove them. Using the -d parameter will remove a person from your ignore list, or you can see and edit the ignore list through the Options | Preferences | Ignore dialog.

/INFO

usage: /info

Shows basic information about the server you are connected to.

/INVITE

usage:	/invite <nickname> <channelname>
example:	/invite johndoe #newbies

You may invite users to join you on a certain channel using the /invite command. The receiving user gets a message indicating the sender and the invitation.

/ISON

usage:	/ison <nickname [nickname] ... >
example:	/ison bugsbunny tweety sylvester elmer

/ison allows you to check whether one or more nick names entered are currently on IRC. Separate each nickname with a space. PIRCH will display a message in the server window for each nickname that is on IRC.

/JOIN

usage: /join <channelname [,channelname]>
example: /join #pirch

/join <channelname> is the command used to enter a channel. Give the channel name as an argument. You can join more than one channel at a time by separating each channel name with a comma. Your arrival on a channel is announced to the rest of the users already on that channel. Silent, anonymous "lurking" is not supported.

If this is a secret or hidden channel, /who commands will show any users of the channel including yourself channel.

You can join multiple channels at one time by separating the channels names with a comma...
example: /join #channel1,#channel2 Do not add spaces between the channel names.

To leave a channel, use the /PART command or close the channel window.

/KICK

usage:	/kick <channelname> <nickname> [comments]
example:	/kick #newbies jake No vulgar language allowed in this channel

The /kick command forces the client <nickname> to be removed from the channel <channelname>.
The comments parameter is optional , but if included will be sent to <nickname> after he has been ejected from the channel.

Note: This command requires channel operator privileges.

/LINKS

usage: /links [-s] [servername]

The /links commands shows other IRC servers connected to the network. By default, PIRCH will display the linked servers in a network map format showing how the network nodes are linked to one another relative to the servername. If no servername is specified, this commands will retrieve the links relative to the server to which you are connected.

You can use the -s parameter to display the links in the server window rather than the links window. When displayed to the server window, the network map structures are not analyzed or sorted.

/LIST

usage:	/list [#channelmask] [-MIN n] [-MAX n]
example:	/list #new*
	/list -MIN 5 -MAX 25

/list will give a list of active channels, the number of users of each, and the channel topics. Secret channels do not appear and private channels only appear as *. If you use a channel mask, only those channels that match the mask will be listed. If you do not include a channel mask, you will receive a list of all active channels. You can also use the -MIN and -MAX parameters to filter the channel list according to the number of users in the channels.

Filtering the channel list by providing command line parameters will NOT speed up the listing process. Currently only a few servers actually support passing of filter parameters to the server. This means that servers always send the client a full channel list, even if only a few channels match the parameters supplied. For those few servers that do support direct filtering, use the ~~/VERBOSE~~ LIST [params] command according to instructions provided by the server. It is often best to use PIRCH's channel filtering procedure after receiving full channel list.

One of the more common problems in listing channels, is that many servers are sensitive to dedicating time to processing list for slow connections. Servers limit the amount of information that a user can request and if your connection is too slow to keep up with the servers processing of the information, the server may disconnect you. This is done as a preventative measure to keep servers from slowing down and increasing network lag. If the server you are on unexpectedly disconnects you while doing a /list command, you can either try to find another server that is not quite as sensitive, or wait until a later time when the server is less busy.

/LOGIN

usage: /login [-a] [profile]

Use /LOGIN to create a new server connection. Profile is the name of the profile (ini) you would like the new connection to use. If this parameter is not provided, PIRCH will use the currently active profile.

The -a parameter can be used to bypass the login dialog box.

/LUSERS

usage: /lusers

The /LUSERS command list the users currently on the network.

/MAP

usage: /map

The /map commands displays a map of the IRC network layout showing how each server is connected to the network.

See also: /LINKS

/MAX

usage:	/max [windowname]
examples:	/max
	/max #pirch

Maximizes a window. If the windowname parameter is supplied then the specified window will be maximized, otherwise this command will maximize the active window.

See also:	/MIN
	/RESTORE
	/CLOSE

/ME

usage:	/me <action>
example:	/me dances on his desk.

The /me command is used to describe an action. Your name will automatically be inserted and the text will appear differently than normal channel messages.

This command can be used in channels or private message windows... but can not be used in dcc chats.

/MIN

usage:	/min [windowname]
examples:	/min
	/min #pirch

Minimizes a window. If the windowname parameter is supplied then the specified window will be minimized, otherwise this command will minimize the active window.

See also:	/MAX
	/RESTORE
	/CLOSE

/MODE (channels)

usage:	/mode <channelname> {b | I | k | l | m | n | o| p | s | t | v }[<limit>] [<password>] [user] [ban mask]
examples:	/mode #newbies +tn
	/mode #newbies +k newbieskey
	/mode #newbies -l
	/mode #newbies +l 6

This command can be used for altering the various modes of a channel

+/-b Sets/removes a ban mask to keep users out of the channel.

+/-I If set, the channel is invite only. This means outside users must receive an invitation from a channel member before they can join the channel.

+/-k If set, the channel has a key (password) that is required. Users who do not know the channel password can not join the channel.

+/-l If set, the channel is limited to <limit> number of users. If the channel has reached capacity, no additional users will be allowed to join until a current channel member parts the channel.

+/-m If set, the channel is moderated. This means that only channel operators, or users who have been given permission, can send text to the channel.

+/-n If set, the channel can not receive msg's from users who have not joined the channel.

+/-o Gives/takes channel operator privldges.

+/-p If set, the channel is a private channel.

+/-s If set, the channel is a secret channel and does not show up on the channel list.

+/-t If set, only channel operators may change the channel topic.

+/-v Gives/takes the ability to speak on a moderated channel.

Note :You must have channel operator privledges to issue this command.

/MODE (User)

usage: /mode <nickname> {[+|-]|I|w|s|o}

/mode johndoe +I

User modes are typically changes which affect either how the client is seen by others or what extra messages the client is sent. A user mode command may only be accepted if both the sender of the message and the nickname given as a parameter are both the same.

+/-I	Marks a users as invisible
+/-s	Marks a user for receipt of server notices
+/-w	User receives wallops
+/-o	Operator flag.

If you mark yourself as invisible, you will not be visible to users who do a /who or /names command on a server or on a channel. However, you will be visible to all persons in the channels you join.

You can not make yourself an operator using the "+o" flag, as this is can only be done by personas already having operator privledges. however, there is no restriction, on you `deopping' yourself (using "-o").

/MOTD

usage:	/MOTD [servername]
example:	/MOTD
	/MOTD irc.ucdavis.edu

The /MOTD command will retrieve and display the “Message of the Day” for the current server to which you are connected. Generally, this information will contain some basic rules for using the server.

This command can be extended by including the name of a server as a parameter. This will then show you the MOTD of the specified server rather than the server you are connected to.

/MSG and /PRIVMSG

usage:	/msg <nickname[,nickname]> <message>
example:	/msg johndoe Hi there.

Allows you to send a private message to a particular user. No other users in any channel you are on can see the message sent.

This command will actually work with a channel names instead of nick names and would be equivalent to simply typing in the channel window, meaning that all channel members can see the message you sent. Using a channel name is only useful if you want to send a message to a channel that you have not joined, but will only work if the channel mode is -n.

In addition, you can send messages to more than one person. To do this, simply list all of the nicknames separating each name with a comma. Do not include spaces between the nicknames.
For example: /msg Tom,Dick,Harry Hi there will send the message to the 3 people listed.

PIRCH also includes a /QUERY command that is functionally similar to the /msg command, but will open a private message window for the person to whom the message is sent.

/privmsg is identical to the /msg command.

/NAMES

usage:	/names <channelname>
example:	/names #newbies

/names lists all of the nicknames currently occupying the indicated channel. Unlike the /who command, the /names command does not list their user address and other extraneous information.

/NEWWINDOW

usage: /newwindow <windowname>

This command can be used to create a new child window on the PIRCH desktop. The caption of the window will display the windowname. Windows created with command do not have a command/edit line, but are useful for redirecting particular text using events.

/NICK

usage:	/nick <newnickname>
example:	/nick johndoe

You can change nicknames by issuing /nick newnickname. All users on channels you occupy will be informed about the change. The IRC system will not let you change your nickname unless you chose a nickname that is not currently in use.

/NOTE

The /NOTE command has many purposes, including the ability to leave notes for people on an IRC server, which get delivered to them when they sign on. However, this command is server specific, which means it may work on some servers and may not on others. Also due to the potential for abuse, its availability on any servers may be short lived.

/NOTICE

usage:	/notice <nickname | channelname> <message>
example:	/notice #newbies Never give out your passwords!
	/notice johndoe You have been given channel operator privldges

/Notice allows you to a message in much the same way as the /msg command, but notices are reserved for information that would generally require more attention by the receiver(s) of the message.

/NOTIFY

usage: /NOTIFY [?]<nickname>
example: /notify jane45

/NOTIFY can be used to add a user to your notify list. Notification use the /ISON command to routinely query   the server to which you are connected as to whether a particular user is currently logged into the IRC network. The notify list is stored in the Options|Preferences dialog in the notify section and can be modified directly from the dialog.

When a user who is in your notification list logs into IRC, a message will be displayed in the server/status window that the particular now present. When a user in you list logs off, the notification system will display a message stating that he/she is no longer present on the network.

Because the notification system works only with nicknames, PIRCH allows you to automatically do a /WHOIS command on a person's nickname. This can be accomplished by prefixing the person's nickname with a ?, ie. ?Jane45.

/OPER

If you don’t already know how to use this command, there is little chance you are authorized to be a IRC operator.

/OPMSG

usage:	/opmsg <channelname> <message>
example:	/opmsg #pirch Who votes to /kick BillyC?

The /OPMSG sends a private message to all channel operators for the indicated channel. The message is sent in the same format as a regular /MSG command, but is prefix with “-CHANOPS-” indicating that the message is intended for all channel operators. All others in the channel who are not ‘opped’ can not see this message.

/OPNOTICE

usage:	/opnotice <channelname> <message>
example:	/opnotice #pirch Who votes to /kick BillyC?

The /OPNOTICE sends a notice to all channel operators for the indicated channel. The message is sent in the same format as a regular /NOTICE command, but is prefix with “-CHANOPS-” indicating that the message is intended for all channel operators. All others in the channel who are not ‘opped’ can not see this message.

/PART

usage:	/part <channelname>
example:	/part #newbies

Use the /part command to leave a channel. When you leave, all other channel members will be notified of your departure.

/PING and /CTCP PING

usage:	/ping <nickname>
	/ctcp <nickname> ping
examples:	/ping JohnDoe
	/ctcp JohnDoe ping

The /PING and /CTCP PING commands are identical in function. These commands are used to determine the length of time it takes to send a signal to another person on IRC and then back again. The time is reported in seconds and will be displayed in the server/status window.

/PLAYFILE

usage: /playfile <nickname|channel> <filename>
example: /playfile #pirch c:\pirch\readme.txt

The /playfile command can be used to display the contents of a text file in either a channel, private message or dcc chat window.

IRC servers limit the amount of text a user can send within a specific period of time, and if too much information is sent too quickly, the server will disconnect you. In order to prevent you from being disconnected, the /playfile command uses an independent flood control method when the command is used for channel or message windows. In addition, even after a you use this command, you may still type normally, interrupting text from the file being displayed.

For files played within DCC chat windows, the file is delivered as fast as possible.

See also: /PLAYPAUSE, /PLAYRESUME and /PLAYSTOP

/PLAYMEDIA

usage :	/playmedia <filename>
example:	/playmedia c:\sounds\beep.wav

The /PLAYMEDIA command can be used to play a multimedia file on your computer system. PIRCH supports the following multimedia filetypes:

WAVE	Waveform Audio File
MIDI	Musical Instrument Digital Interface Audio File
RMI	A variation on the MIDI file format
AVI	Video file

The ability to play these various mutimedia file formats will depend on your computer system configuration and each requires special device drivers to be installed in windows.

/PLAYPAUSE

The /PLAYPAUSE command can be used to temporarily stop the displaying of text started by the /PLAYFILE command. To restart the text display in the window, use the /PLAYRESUME command.

/PLAYRESUME

The /PLAYRESUME command can be used to restart the displaying of text which had been previously paused with the /PLAYPAUSE command.

/PLAYSTOP

The /PLAYSTOP command can be used to discontinue the displaying of text which had been previously started with the /PLAYFILE command.

/QUERY

usage:	/QUERY <nickname> [message]
example:	/query JohnDoe Hello

The /QUERY command is similar to /MSG command with the exception that /query always opens up a new private message window for the person to whom the message is sent.

/QUIT

usage:	/QUIT [message]
example:	/quit I am not addicted to IRC!

The /quit command can be used to disconnect completely from IRC. When you quit IRC, any channels you did not formally part will be notified of your disconnection. If you entered the optional message, the channel members will also see the message on the notification they receive.

/REMUSER

Usage:	/REMUSER [-q] <Level,[[level],...]> <nick[!user@host]> <maskmode>
Example:	/REMUSER 100 Billy 2
	(removes *!BillG@*.microsoft.com from event level 100)

This command is used to remove a user from the user list of an event level. The -Q parameter indicates the command should be done without displaying the change in the server/status window.

Level indicates the level(s) to which the user should be removed. The level name may be a partial name, or wildcarded in which the all matching levels are affected. You may add/remove from several levels at one time by listing each level seperated by a comma (do not separate with a space).

nick!user@host is the nickname or internet address of the person you are adding/removing. The address portion of this command is optional, in which case PIRCH will retrieve the person's internet address for you via a /WHOIS command.

MaskMode is the address masking/wildcarding options which should be applied to the person's address prior to removing from the list. This should be the same maskmode used to add the user. There are 6 available masking options.

Mask Type	**Resulting address mask**
0 = User	Billy!BillG@ppp01.microsoft.com
1 = IdentPortServer	*!BillG@ppp01.microsoft.com
2 = IdentServer	*!BillG@*.microsoft.com
3 = PortServer	*!*@ppp01.microsoft.com
4 = NickOnly	Billy!*@*
5 = serverOnly	*!*@*.microsoft.com

see also: /ADDUSER

/RESTORE

usage:	/RESTORE [windowname]
examples:	/restore
	/restore #pirch

Restores a windows size & position from a minimized or maximized state. If the windowname parameter is supplied then the specified window will be restored, otherwise this command will restore the active window.

See also:	/MAX
	/MIN
	/CLOSE

/RUN

usage:	/RUN [application.exe]
example:	/run c:\ftp\ftp.exe

The /RUN command is used to run an external application, which maybe a .EXE, .COM, .BAT or .PIF file. The application will become the active program on the windows desktop.

/RUNSCRIPT

usage: /RUNSCRIPT <scriptname> [parameters]
example: /runscript [myscript] this is a test

The /RUNSCRIPT command is used to run a PIL script. The parameters are passed to the script and can be accessed within the script itself using the PIRCH variables *1..*xx, $1 $2 etc. See PIL.DOC for more information on PIL scripting.

/SAVEBUFFER

usage:	/SAVEBUFFER [filename.txt]
example:	/savebuffer funny.txt

the /savebuffer command will save the current window’s text buffer into a text file. If the filename parameter is omitted, you will be prompted for a filename via a dialog box.

/SERVER

usage: /SERVER <servername[:port]>

Forces the current conection to lose and a new connection be made to the indicated server. You may include a port paramer to connected to a specific port number. If no port is specified, the standard port 6667 is used. If a port number is to be included, use a colon (:) between the server name and the port number.

/SET

usage: /SET <option> <value>

/SET is not an IRC command, rather is it a PIRCH specific command that can be used to set a variety of options from the command line or within aliases.

```
/SET AUTOOP   ON|OFF   -enables/disables the autoop list
/SET CMDBAR ON|OFF -shows/hides the command bar in the server/status window
/SET CHANLIST ON|OFF -shows/hides the channel list in the server/status window
/SET CTCP   ON|OFF   -enables/disables the CTCP protocol
/SET NOTIFY   ON|OFF   -enables/disables the notify list
/SET HEADERS ON|OFF -shows/hides the headers in the server/status window
/SET IGNORE   ON|OFF   -enables/disables the ignore list
/SET SOUND     ON|OFF   -enables/disables playing of media for /sound commands
/SET AUTOMIN ON|OFF   -determines whether new msg windows will be minimized
/SET NOPOPUP ON|OFF   -sets the no message interrupt option
/SET TABS ON|OFF -shows/hides the window tabs in the main window
/SET TAG ON|OFF -shows/hides the network id tag for channel/message windows
/SET TOOLBAR ON|OFF   -shows/hides the Toolbar in the main window
/SET STATUSBAR ON|OFF -shows/hides the statusbar in the main window
```

/SILENCE

usage:	/silence [+|-][nick!user@host.mask]
examples:	/silence johndoe!jdoe@net.com
	/silence -johndoe!jdoe@net.com

This command effectively stops private message flooding at the server of the flooder.
You can use "/silence nickname" to get a list of the silence masks of 'nickname'.

To remove a user from you silence list, type /silence -[nick!user@host.mask].

This command is specific to the Undernet network.

/SOUND

Format: /SOUND <#channel|nickname> <filename> [message]
Example: /sound #pirch loudthun.wav makes the thunder crack!

The /sound command can be used to play sounds or media files on other another clients computer system. The command can be sent to an entire channel or individual people. You can append a message to the sound command that will be displayed in the channel/message window much in the same way an action using the /me command is displayed.

PIRCH supports the following media file types for use with the /sound command.

.wav	- Standard Windows Wave file format for sounds.
.mid	- Midi file format
.avi	- Video file format

In order for the command to work properly, the file must exists on the target clients' computer system.

When a .avi file type is played... it will be displayed in a separate popup window.

If no file extion is included on the filename, PIRCH will assume a file type of .wav and append the extension automatically.

/STATS

usage: /STATS [servername]

The /stats command can be used to view statistics for a particular server. If no server name is specified, statistics for the server to which you are connected will be returned.

/TILE

usage: /tile

Use the /tile command to arrange the windows on the desktop in a non-overlapping fashion.
Does not affect any windows marked as ‘Keep on Top’.

/TIME

usage: /time [servername]

/time reports the time according to the server you specified in servername. If no server name is specified, the time reported back to you is from the IRC server you are connected to.

/TIMER

usage:	/timer <timername> <iterations> <interval> <command>
example:	/timer 0 100 30 /me is away

The /TIMER command can be used to automatically execute a command at specific time intervals. Timername may be any name or number you desire but the name may not contain spaces. Iterations is the maximum number of times you wish to have the command executed. Interval is the time delay between each iteration. Command may be any valid PIRCH or IRC command.

You can view status for all timers by typing /timer with no parameters
You can view status for an individual timer by typing /timer <timername>
You can enable/disable a timer by type /timer <timername> <on|off>

/TOPIC

usage:	/topic <channel name> <channel topic>
example:	/topic #newbies Help for new IRC users.

Allows you to change the topic text associated with a channel. This ‘topic’ is generally information describing the purpose of the channel. Channel topics are visible on when you execute the /LIST command, and when you join a channel the topic will appear in the channel windows caption.

Note: If the channel mode flags include +t, you must have channel operator privileges to change the channel topic.

/TRACE

usage: /TRACE

/TRACE can be used to show the network route taken for message delivery to a particular user.

/UNDEFINE

usage: /undefine <variable>

Undefines and frees the memory associated with a user defined variable created with the /define command

/USERHOST

usage:	/userhost <nickname> [nickname]...
example:	/userhost bugsbunny tweety sylvester elmer

The /userhost command takes a list of up to 5 nicknames, each separated by a space and returns a list of information about each nickname that it found, inparticular it returns the address of the user(s).

/userhost also allows a dynamic event handler to be attached. The format for using event a event handler is...

```
/userhost nickname > /command
```

A dynamic event handler is simply a command that processes the information that is returned by /userhost. For the /userhost, PIRCH breaks down the returned information into the following variables:

$nick	nickname of the user
$ident	the username or ident of the user
$port	contains the slip/ppp port to which the user is connected on his/her ISP
$domain	contains the users isp domain name
$address	contains the combined $ident,$port & $domain in IRC format

For example... you can ban someone from a channel using /userhost (the hard way) by doing something similar to the example below:

```
/userhost bob > /mode #mychannel +b *!* $+ $ident $+ @ $+ $domain
```

/VERBOSE

Usage: /verbose <text>

The /verbose command is used to send text directly to the server without any modification by PIRCH. In order to communicate directly with the server, you MUST know the exact format and syntax expected by the server. This can be useful for using command which PIRCH has not implemented or are unique to that server.

/VERSION

usage: /version [servername]

/version is used to query a server and have it report back the server software version. If you do not specify a servername the server your are currently connected to will report its version.

/WALLOPS

Usage: /WALLOPS <message>

The /WALLOPS command is generally used by IRC operators to communicate with each other. The messages are passed publicly through the server system and can be viewed by anyone whose client mode as been set +w.

/WHO

usage:	/who [#channelname_mask | user@host.mask]
examples:	/who #newbies
	/who *gov.us*

/who returns information on who is using IRC.   /who without arguments prints info on all users that can be seen. Users of public channels show up with their channel(s) identified.   Users of private channels appear, but they are specified as being on a private, unspecified channel.   Users of secret channels and users whose user mode is +I (invisible) do not appear at all.

Giving a channel name as an argument to /who returns only those users of the specified channel.   This still doesn't show users of secret channel or invisible users one is actually on the same channel with them. Users of private channels are shown, if an exact channel name is given.

/WHOIS

usage:	/whois <nickname>
example:	/whois johndoe

This returns information about individual users.   Type "/whois nickname" to get information on the login name and host from which the nicknamed user comes, along with a list of channels the user is currently on.

/WHOWAS

usage:	/whowas <nickname>
example:	/whowas johndoe

This command provides the same basic information as /whois but can be used for a limited time after the person has left IRC or has changed their nickname.

/WRITE

usage:	/WRITE [-CDIR#] <filename> [text]
example:	/write -R1 c:\pirch\test.txt This is a test

Parameters:	-C	clears the text file removing all lines
	-D#	deletes line # from the file (if no # the the last line is removed}
	-I#	inserts text at line # (if no # then inserted at end of file)
	-R#	replaces line # in the file (if no # then inserted at end of file)

/WRITE stores a line of text to a text file.

/WRITEINI

usage: /WRITEINI <filename> <section> <key> <value>
example: /writeini c:\pirch32\tracking.ini # $nick $nick last joined # on $day $date at $time (CST)

The structure of an ini file is broken down into sections as diplayed below

```
 [section]
key=value
```

Each inifile may have multiple sections and under each section multiple key=value statements

NOTE: All parameters are required and the function will fail if any are missing. The $readini inline function currently must be the last part of an alias or event handler statement since the last parameter <default> may contain multiple words. The structure of inline functions (ie $read, $readini) will probably change in the near future to allow greater flexibility in parameters and a more consistent structuring of alias/event code.

DCC

DCC is an acronym that stands for Direct Client to Client. Basically this means the two parties communicate directly with each other rather than communicating through the IRC network. DCC allows you to send and receive files and to chat directly to another party.

Sending Files using DCC

PIRCH allows you to send files using either PIRCH's DCC sender, the Drag-N-Drop Sender or by using the /DCC SEND command.

DCC File Sender

The DCC file manager is a file list dialog box that will allow you to select one or more files and send them to another person on IRC. To activate the DCC sender, right click over the Server/Status window and select the option from the popup menu.

You can use the file list section of the dcc sender to navigate directories and select files just as in other windows applications. To select a file for transfer, double-click on the file name, or highlight the file name click on the > button. The file will then be added to the list on the right hand side of the window. You could also click and drag the filename to the right side and drop it into the list.

DCC Drag-N-Drop Sender

The Drag-N-Drop file sender is simply a dialog box containing the directory, drive & file lists. To send a file, click on the desired filename, and while holding mouse button down, drag it to the target person's name in the channel's name list and drop it by releasing the mouse button. The file will then be sent to the select person.

Receiving Files using DCC

Receiving files sent to you from another person is fairly simple using PIRCH. When another party attempts to send you a file, a dialog box will be displayed telling you of the transaction. You can either accept the file or refuse it. If you accept it, it will automatically be saved into the directory you selected in your DCC options.

Resuming Interrupted DCC's

If for some reason a file transfer fails, PIRCH allows you to resume the send/receive process at the point it left off. When a person attempts to resend a file that already exists on your system, the Resume button in the DCC confirmation dialog will be enabled, you may then click on it to have the transfer automatically continue from the point it was last interupted.

DCC Chatting

You can chat directly with another party using the DCC protocol. This method has a number of advantages and a few disadvantages as well. Because the DCC connection bypasses the IRC servers and the IRC network as a whole, DCC connections are generally considered more secure. You also are not limited on the rate at which you send data to your DCC partner by the IRC system. You are limited only by the speed of the parties' network setup and the quality of the network connection you have. The disadvantages are that you can not perform special IRC commands such as ACTIONS within a DCC communication. To initiate a DCC Chat connection you may use the /DCC CHAT command, or select the DCC Chat command from the names list popup menu in a channel window.

CTCP

CTCP is an acronym that stands for Client To Client Protocol. Generally CTCP messages are used to retrieve information about a user such as their Real Name, E-mail address and information about the IRC client they are using. PIRCH does NOT require you to ever divulge personal information about yourself. Use the CTCP options to set   responses to personal information items

The following are CTCP commands which are supported internally by PIRCH

/CTCP CLIENTINFO
/CTCP VERSION
/CTCP TIME
/CTCP DCC CHAT
/CTCP DCC SEND
/CTCP DCC GET
/CTCP USERINFO
/CTCP FINGER

A SIDE NOTE ABOUT FLOODING

Unfortunately, the CTCP protocol has become a method by which some IRC users attempt to interrupt another person's enjoyment of the IRC medium. IRC servers implement a method of flood control which causes a person to be disconnected from the network if they send too much information too quickly. Because of this fact, and the general nature of the CTCP to automatically send a message or reply back to the requesting user, PIRCH has some additional features that attempt to circumvent these potential problems.

For the most part, we hope you are never confronted with this type of problem. But if you see CTCP messages quickly appearing in your server/status window, you can temporily suspend all CTCP replies by entering the followinf command.   /set ctcp off.

I personally recommend you add a function key alias to your alias window...
    example: F11: /set ctcp off | /flush
Then if you see you are being flooded with ctcp messages, you can quickly suspend ctcp activity by pressing the F5 key. the /flush command flushes any pending ctcp replies from the output queue.

Favorite Places

PRICH includes a list box where you can store a list of your favorite channels. To open the Favorite Places dialog, select Favorite Places from the tools menu.

Adding New Items to the Favorite Places List

To add a new entry to the Favorite Places list, click on the Add button, or simply drag the channel name from the main channel list in the server window, and drop it into the Favorite Places list. The channel name will

Word Wide Web Links

PIRCH includes a World Wide Web Links tool that can allow you to capture WWW and FTP addresses from your IRC chat sessions.In addition you can have PIRCH load your web browser and go to a particular web page quickly and easily.

The WWW Links window contains a list box that will hold your URL's.   By double clicking on an item in this list, PIRCH will loaded your web browser (if not already running) and have the browser load the particular url.

As PIRCH inserts new URL's into the list, they will be prefixed with an asterisk (*) indicating that the site has not been visited. When you visit a site by double clicking the item, the asterisk will be removed.

At the end of each session, PIRCH will remove any URL's from the list which you have not visited or cleared.

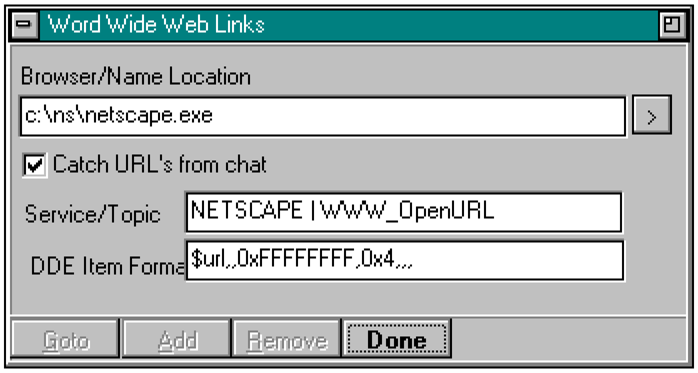
Goto
This button will cause PIRCH to load the web browser and goto the selected web site. Basically the same function has double clicking on a list item.

Add
Select this option to add your own url's to the list.

Remove
Deletes the selected URL from the list.

Options
Select this button for changing your web browser options as outlined below.

Setting up your Browser to work with PIRCH
In order to make your web browser work with PIRCH, you must first tell PIRCH where your web browser is located. Click on the Options button to display the browser setup window.



Browser/Name Location
c:\ns\netscape.exe
Catch URL's from chat
Service/Topic NETSCAPE | WWW_OpenURL
DDE Item Forma $url,,0xFFFFFFFF,0x4,,,

Browser Name/Location
Enter the full path name of your web browser or click on the > button to display a file/directory dialog from with you can select your browser.

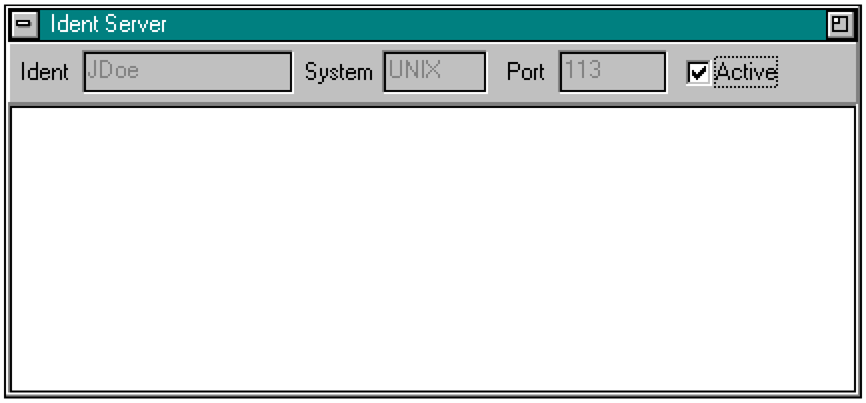
Catch URL's from Chat
When this option is checked, PIRH will scan IRC chat messages for URL's including http and ftp addresses. When found, these addresses will be stored in the URL list.

DDE Options
By default, PIRCH is setup for using the popular Netscape Web Browser and you should not need to change any of the DDE options. If, however, you use a browser other than Netscape, you will need to change the DDE options as outlined in your browser documentation.

The $url variable in the DDE item field will be replaced with the actual url when an item is loaded.

Ident Server

An ident server is basically a means of identification used by some systems, and is required by a few IRC servers. Basically this means that you computer will respond to other systems requesting your identification by returning the information you enter in these fields.



Ident: JDoe
System: UNIX
Port: 113
[✓ Active]

Most public servers do not require you to have an Ident Server, so its usefulness to many people will be limited. If you do decide to have the ident server active, you should leave the System field set to UNIX and the Port field set to 113, unless you have specific reasons for doing otherwise.

How will running an Ident Server affect me?

Generally when you log into IRC, PIRCH uses the ident from the E-mail address (the part before the @ symbol) you entered in the connection dialog box as your user name and the server combines this user name and your IP address to identify you on the network. With an ident server, the IRC server will instead use the Identifier returned by the Ident server. Also, when an ident server is used, your id will not be prefixed with a tilde (~) symbol on the network.

Aliases

Aliases are simply custom commands or macros that you can define for PIRCH. To create aliases, select the Alias option from the main menu. This Alias definition window will be displayed. In order to make use of aliases you must of course already be familiar with IRC and basic IRC commands.

All aliases take the form of <name>:<command>[|command...] and must be entered on a single line in the alias definition window. You should NOT create an alias with the same name as an existing IRC command as PIRCH always gives precedence to IRC commands over aliases. You may also use aliases within your popups and your events & controls.

Defining Simple Aliases

The simplest aliases included only the aliases name followed by the command to be executed.

PIRCH: /join #pirch	typing /pirch will now allow you to join the channel #pirch
P:/part #	typing /p will allow you to leave the current channel

Variables

PIRCH defines the following variables which can be used in your aliases definitions:

$me	your nickname
#	the current channel
$#	the # symbol
$1..$xx	command line parameters
*1..*xx	sequential command lines parameters start with the number provided
%1..%xx	returns the alias token word at the specified position
$+	trims extra space from between parameters
$date	the current date according to your computer's internal calendar
$day	the current day of the week
$time	the current time according to your computer's internal calendar
$version	the version of PIRCH which you are using
$ip	your local ip number
$host	your local host name
$snick	currently selected nickname in a channel
$server	the name of the server you are connected to
$members	returns the number of people in a channel
$url	the currently highlighted url in the WWW window
$?="prompt"	Displays an input dialog box, returning the entered information

Special Variables/Funtions

$read [-L<#>] <filename>

Reads a line from a text file. The -L parameter can be used to specificy a particular line. If no $l param is included pirch will read a random line.

$readini <filename> <section> <key> <default>

Reads a line from a ini file. See /writeini for more information

$file "<caption>" <filemask>

Pops up a file dialog allowing you to select a particular filename to use. You must specify a caption between quotation marks!

example: WAV:/sound # $file c:\pirch\*.wav hopes you like this sound

By default, $file returns a fully qualified filename including the drive and directory. If you wish not to have the drive letter and directory name returned, use $-file. This will simply return the filename and extension.

$addrmask

This variable is only available for aliases executed within private message windows. This variable contains an address mask for the user that the window belongs to. This can be extremely useful for ignoring a user by address.

Example alias: F12: /ignore $addrmask

The address mask is generated by the same mechanism as ban mask and uses the default ban method indicated in the Ops section of the options dialog.

Linking aliases

PIRCH allows you to link aliases to other alias definitions, however, you must be sure to not create a situation that would cause PIRCH to endlessly loop expanding aliases, as this will be sure to cause your system to lockup. In the example below

```
Legal Link of Aliases
RULES:/notice $1 The following is a list of rules on #pirch... | /rules1
RULES1:/notice $1 Failure to follow them will cause you to be kicked or banned! | rules 2
RULES2:/notice $1 No cursing, insults, or sexual references of any kind | rules 3
RULES3:/notice $1 No flooding of channel or our bot | rules 4
RULES4:/notice $1 If you have questions... please ask! We are here to help if we can.
```

Function Keys

You may assign the names of function keys F1-F12 as aliases names and execute these aliases by simply pressing the assigned function key. The examples below are simply to demonstrate valid function key definitions.

```
F2: /me is happy
aF2: /me is angry
acF2:/me is hungry
acsF2:/me is away
asF2:/me is sick
cF2: /me is crazy
csF2: /me is tired
sF2:/me is done with this silly example
```

The prefixes before the function key names can be used to indicate addition keypresses as listed below...

a = alt key
c = control key
s = shift key

You may combine these keys, but they must be in alphabetical order. For example acsF5 means you hold down the control, alt & shift keys simutaneously, then press the F5, and finally releasing all the keys. scaF5 has no meaning to PIRCH because the order of the shift, control, & alt keys is incorrect.

Windows reserves a number of function keys for its own purposes and PIRCH does not make any attempt to override these default definitions...

Ctrl-F4	Closes the active window
Alt-F4	Closes the application
F1	Help (this really isnt a windows reserved function key, but is it is sort of a standard)

Common Aliases & Examples

Join Aliases

J:/join $# $1
Joins a channel without having to enter the # symbol... ie /j pirch will cause you to join #pirch

HOP:/part # | /join #
Cause you to leave the current channel and rejoin it immediately

Sound Aliases

S:/sound # $1 *2
Plays a sound to the current channel (or nickname in a message window)

SND:/sound # $file "Pick a file to play" c:\windows\media\*.mid *1
Allows you to pick a sound file from a file list and play it to the current channel/nick.

Operator Aliases

OPS: /opnotice # *1
Send a notice to all channel operators in the current channel.

Misc Aliases

F2:/tile
CF2:/cascade
Tiles or cascades the desktop windows using the F2 and Ctrl-F2 keys respectively.

F11:/set ctcp off | /flush
CF11:/set ctcp on
Disables/enables the CTCP protocol with F11 and Ctrl-F11 keys respectively.

N: /nick $?="Enter a new nickname"
Displays an input dialog box allowing you to change your nickname.

POPUPS

PIRCH's popups allow you to customize the right-click popup menus. By default, PIRCH includes commands in popup menus that can be accessed by clicking the right mouse button while the mouse pointer is located over either the names list in a channel window, or the main text pane in a channel window/private message or server window.

When you click on the Popups button in the tool bar, or select the Popup command from the Tools menu, PIRCH will display the popup editor. The tabs on the top of the editor window allow you to select which popup menu to edit.

Editing and defining popup commands is almost identical to defining aliases. By default, PIRCH includes a number of the most useful commands in the various POPUP menus. Many of these function can not be easily duplicated using simple aliases, or have special features, like becoming disabled or changing states depending on how the command would perform under a given circumstance. If you want full control over defining the commands in your popup menu, you can remove all the predefined commands by placing the statement $clearmenu as the first line in the popup definition.

Automated Events & Controls

PIRCH includes a set of features which allow you to customize the client and automatically react to a variety of events that occur on IRC, such as people joining and leaving the channel you are on.   This section of this help file is not intended to be a guide for how to implement such features, rather it is only to be used as a reference for what triggers each event, what information is made available to the event handler and the format for implementing specific event handlers.

Creating the user list.

The user list is used to determine which user are affected by the event handlers and have hacees to the custom CTC commands you set up. The user list is sorted alphabetically but the order of the entries in the list are important.
The lowest user levels should be placed at the top of the user list and the highest at the bottom. In many cases, it is simpler to use numbers in the level names to help in insuring your ordering is logical.

For Example:	000-Unknowns
	500-Friends
	999-Enemies

When level names are added, the user list and events list will change to show the users and events associated with that level.

@ and = Event Prefixes

Some events may be prefixed with either the @ symbol or the = symbol (cases both may be used where applicable), which modifies the conditions under which the particular event may be triggered. The @ prefix means only to execute the event if you have channel operator status in the channel the event occurred. The = prefix means only to execute the event if the user who triggered the event has the same exact level as the event was placed in, not a higher level.

Event List

All events begin with the word ON followed by the name of the event.

ON ACTION
ON BAN
ON CLONES
ON CTCPREPLY
ON DEOP
ON IGNORE
ON DCCDONE
ON DCCFAIL
ON INCOMING
ON INVITE
ON JOIN
ON KICK
ON KICKED
ON MODE
ON NOTICE
ON NOTIFY
ON NOSOUND
ON OP
ON PART
ON QUIT
ON SOUND
ON SERVERDEOP
ON SERVERMODE
ON SERVEROP
ON SNOTICE
ON TEXT
ON TOPIC
ON WALLOPS

Any other event names used other than those listed above are considered to be custom CTCP commands. With this feature, you can create you own set of CTCP commands or modify the responses to standard CTCP commands.

Modifying Standard CTCP responses.

Below is an example of modifying the standard reply sent by PIRCH for the CTCP PING command.

```
PING:/notice $nick Your levels are too low to PING me!
```

If you want PIRCH to send the standard CTCP reply in addition to append the :+ symbols to the end of the alias line

```
PING:/notice $nick PONG! :+
```

For Custom CTCP commands, PIRCH provides the additional variables.

    $1..$99 - individual parameters from the ctcp command sent to you

The following command will allow users to have your client send files to themselves at their request.

```
XDCC: /dcc send $nick $2
```

See also: Aliases
Common Event Tasks & Examples

ON ACTION

Format:	[@][=]ON ACTION:<text>:<*>|<?>|<#channel>[;#channel][etc...]:<alias>
Example:	ON ACTION:*kiss*:#pirch:/notice $nick Hey... no kissing in this channel

Available info:	$nick - nick name of the user that sent the action message
	$address - address of the user that sent the action message
	# - the name of the channel

ON ACTION events occur when another user send an action to a channel or in a private message. See then ON TEXT for implementation details.

ON BAN

Format:	[@]ON BAN: <comparemode>:<#channel>[;#channel][etc...]:<alias>[:+]
Example:	@ON BAN:*:#pirch:/mode # -o-b $nick $banmask

Available info:	$nick - nick name of the user that did the ban
	$address - address of the user that did the ban
	$victim - the nickname of person banned (if available)
	# - the name of the channel the event occurred on
	$banmask - the ban mask used to perform the ban

The ON BAN event occurs when a user sets mode +b against another user.

The comparemode parameter compares the user level of the person performing the ban (the banner) & the victim and the event will only execute if the level comparison statement is true.

Compare Mode	Condition
*	No comparison is made.
=	The banner and victim levels are equal
>	The banner’s level is greater than the victim’s level
>=	The banner’s level is greater or equal to the victim’s level
<	The banner’s level is less than the victim’s level
<=	The banner’s level is less or equal to the victim’s level
<>	The banner’s level and the victim’s level are not equal

ON CLONES

Format:	[@]ON CLONES: <clonenumber>:<#channel>[;#channel][etc...]:<alias>[:+]
Example:	@ON CLONES:*:#: /display \-1Clones detected in # $+ !!!! $nick [$clonemask]

Available info:	$nick - nickname of the user that caused the event
	$address - nickname of the user that caused the event
	$clonemask - the address mask of the detected clones

This event detects multiple joins from the same IP address. The duration of time between successive joins does not matter.   The clonenumber parameter must be greater than one (1) and can be used to trigger the event only when a certain number of clones from an IP are in the channel.

ON CTCPREPLY

Format:	[=]ON CTCPREPLY: <text>:<alias>[:+]
Example:	ON CTCPREPLY:*:#: /display > -CTCPs- \-1 *1
Available info:	$nick - nickname of the user that caused the event
	$address - nickname of the user that caused the event
	*1 - contains the full ctcp reply received

This event is triggered when a reply to a CTCP command like /ping is received.

ON DEOP

Format:	[@]ON DEOP: <comparemode>:<#channel>[;#channel][etc...]:<alias>[:+]
Example:	@ON DEOP:*:#pirch:/mode # -o+o $nick $victim

Available info:	$nick - nick name of the user that did the deop
	$address - address of the user that did the deop
	$victim - the nickname of person deoped
	# - the name of the channel the event occurred on

The ON DEOP event occurs when a user set mode -o, (deops) another user. The #channel parameters(s) indicate for which channel(s) this particular event handler will be active. If you include the symbols :+ at the end of the alias and you have enabled the PIRCH's internal protection option, PIRCH will then process protection reactions.

This event occurs individually for each person deopped in the channel and only if PIRCH does not first generate an ON PROTDEOP first.

The comparemode parameter compares the user level of the person performing the deop & the victim and the event will only execute if the level comparison statement is true.

Compare Mode	**Condition**
*	No comparison is made.
=	The deoper and victim levels are equal
>	The deoper's level is greater than the victim's level
>=	The deoper's level is greater or equal to the victim's level
<	The deoper's level is less than the victim's level
<=	The deoper's level is less or equal to the victim's level
<>	The deoper's level and the victim's level are not equal

This event DOES NOT occur if you performed the deop or if a user deops himself.

ON DCCDONE

Format:	ON DCCDONE:<filemask>:<alias>
	ON DCCDONE:*.jpg;*.gif;*.bmp:/run c:\viewer\viewer.exe $filename
Available Info:	$filename - the name of the file as stored on your system
	$filesize - size of the file in bytes as reported by the sender
	$rate - transfer rate in characters per second (same as bytes per second)

The ON DCCDONE event occurs when a file is succesfully tranfered to you computer system via DCC.   This feature allows you to automatically execute a program based on the type of file received.   The above example demonstrates how a graphics viewer application can be lauched to viewed the file you just received.   The filemask parameter is used to determine which types of files are affected by the event based generally based on the 3 character extension of the filename.

If the transfer was incomplete, a DCCFAIL event will be generated.

You may have multiple ON DCCDONE events within a user level of the events/controls setup. Note that if you specifiy a default ON DCC DONE event using the filemask *.*, you must place this after any other ON DCC DONE events.

ON DCCFAIL

Format:	ON DCCFAIL:<filemask>:<alias>
Available Info:	$filename - the name of the file as stored on your system $filesize - size of the file in bytes as reported by the sender $rate - transfer rate in characters per second (same as bytes per second) $received - the number of bytes successfully received $percent - percent completed

This event occurs when a DCC download has failed.

ON IGNORE

Format:	ON IGNORE:<alias>
	ON IGNORE:/notice $nick You are being ignored!
Available info:	$nick - nick name of the user that invited you
	$address - address of the user that invited you

The ON IGNORE event occurs when a user on your ignore list sends you a private message using the /MSG or /NOTICE commands.

ON INVITE

Format:	ON INVITE:<#channel>[;#channel][etc...]:<alias>[:+]
Available info:	$nick - nick name of the user that invited you $address - address of the user that invited you # - the name of the channel you have been invited to

The ON INVITE event occurs when another user invites you to join a channel using the /invite command.

ON INCOMING

Format:	ON INCOMING:<alias>
Example:	ON INCOMING:/playmedia c:\windows\media\ring.wav
Available info:	$nick - nick name of the user that is sending you a message
	$address - address of the user that is sending you a message

An ON INCOMING event will occur when a user is sending you a private message for which PIRCH is about to open a new window.   The example above show how you can have a sound played as the window is opening.

ON JOIN

Format:	[@][=]ON JOIN:<#channel>[;#channel][etc...]:<alias>
Example:	ON JOIN:#pirch:/notice $nick Welcome to # ! Enjoy your visit.
Available info:	$nick - nick name of the user that joined
	$address - address of the user that joined
	# - the channel name

The ON JOIN event occurs when a user joins a channel. The #channel parameters(s) indicates for which channel(s) this particular event handler will be active.

ON KICK

Format:	[@][=]ON KICK:<comparemode>:<#channel>[;#channel][etc...]:<alias>
Example:	ON KICK:*:#pirch: /notice $nick Please don't kick $victim | /invite $victim #
	ON KICK:<=:#pirch:/mode # -o $nick | /notice $nick Your level is too low to kick $victim.

Available info:	$nick - nick name of the user that performed the kick
	$address - address of the user that performed the kick
	# - the channel name
	$victim - the person that was kicked

The ON KICK event occurs when one user kicks another user from the channel. The level at which the event will execute is the user level of the person triggering the event (the kicker).

The comparemode parameter compares the user level of the kicker and the victim and the event will only execute if the level comparison statement is true.

Compare Mode	**Condition**
*	No comparison is made.
=	The kicker and victim levels are equal
>	The kicker's level is greater than the victim's level
>=	The kicker's level is greater or equal to the victim's level
<	The kicker's level is less than the victim's level
<=	The kicker's level is less or equal to the victim's level
<>	The kicker's level and the victim's level are not equal

ON KICKED

Format:	[@][=]ON KICK:<comparemode>:<#channel>[;#channel][etc...]:<alias>
Example:	ON KICKED:*:#pirch: /msg MyBot Ban $nick | /join #
Available info:	$nick - nick name of the user that performed the kick
	$address - address of the user that performed the kick
	# - the channel name

An ON KICKED event occurs when you are kicked from a channel.

ON MODE

Format:	[@][=]ON MODE:<#channel>[;#channel][etc...]:<alias>
Example:	ON MODE:#pirch: /mode # $lastmode | /notice $nick Please don’t change the channel mode!
Available info:	$nick - nick name of the user that changed the channel mode
	$address - address of the user that changed the channel mode
	# - the channel name
	$lastmode - the last active channel mode prior to the change
	$lastkey - the last channel key before the change
	$lastlimit - the last channel limit
	$key - the current channel key
	$limit - current channel limit

The ON MODE event occurs when another user changes the channel mode. The #channel parameters(s) indicate for which channel(s) this particular event handler will be active.

No ON MODE event is generated if the mode changes include only the +o, -o or +b flags. These mode changes generate ON OP/ON DEOP and ON BAN events respectively.

ON NOSOUND

Format:	[@][=]ON NOSOUND:<filemask[,filemask][etc..]>:|<#channel>[;#channel][etc...]:<alias>
Examples:	ON NOSOUND:*.mid;*.wav:#:/notice $nick !DCCSEND $filename
	ON NOSOUND:*.*:#:/notice $nick !DCCSEND $filename

ON NOSOUND events are triggered when someone plays a media file which can not be located on your system in the default sound directory.

ON NOTICE

Format:	[@][=]ON NOTICE:<text>:<*>|<?>|<#channel>[;#channel][etc...]:<alias>
Example:	ON NOTICE:*kiss*:#pirch:/notice $nick Hey... no kissing in this channel
Available info:	$nick - nick name of the user that sent the notice
	$address - address of the user that sent the notice
	# - the name of the channel

ON NOTICE events occur when another user send a notice to a channel or in a private message. See then ON TEXT for implementation details.

ON NOTIFY

Format:	[@][=]ON NOTIFY:<alias>
Example:	ON NOTIFY:/notice $nick Hi there!

Available info:	$nick - nick name of the user that joined IRC

ON NOTIFY events occurs when a user in your notify list joins IRC, and only occurs if your notify option is enabled Unlike many other events, the user address is not made available, and you MUST include the person's nick name in your users list.

ON OP

Format:	[@][=]ON OP:<comparemode>:<#channel>[;#channel][etc...]:<alias>
Example:	ON OP:*:#pirch:/notice $nick Come back soon!
Available info:	$nick - nick name of the user that performed the op
	$address - address of the user that performed the op
	# - the channel name
	$opnick - the person that was opped

This event occurs when another user gives op status (mode +o) to a person.

The comparemode parameter compares the user level of the person performing the op (the opper) & the victim and the event will only execute if the level comparison statement is true.

Compare Mode	**Condition**
*	No comparison is made.
=	The opper and victim levels are equal
>	The opper's level is greater than the victim's level
>=	The opper's level is greater or equal to the victim's level
<	The opper's level is less than the victim's level
<=	The opper's level is less or equal to the victim's level
<>	The opper's level and the victim's level are not equal

ON PART

Format:	[@][=]ON PART:<#channel>[;#channel][etc...]:<alias>
Example:	ON PART:#pirch:/notice $nick Come back soon!

Available info:	$nick - nick name of the user that left
	$address - address of the user that left
	# - the channel name

The ON PART event occurs when a user leaves a channel. The #channel parameters(s) indicate for which channel(s) this particular event handler will be active.

ON QUIT

Format:	[@][=]ON QUIT:<alias>
Available info:	$nick - nick name of the user that quit $address - address of the user that quit # - the channel name

The ON QUIT event occurs when a user quits IRC.

ON SERVEROP

Format:	[@][=]ON SERVEROP:<#channel>[;#channel][etc...]:<alias>
Example:	ON SERVEROP:#pirch:/mode # -o $opnick
Available Info:	$address - name of the server performing the op # - the channel name

This event occurs when a channel operator status is granted to a person by a server.

ON SERVERDEOP

Format:	[@][=]ON SERVEROP:<#channel>[;#channel][etc...]:<alias>
Example:	ON SERVEROP:#pirch:/mode # +o $victim

Available Info:	$address - name of the server performing the deop
	# - the channel name

This event occurs when a channel operator status is taken from a person by a server.

ON SERVERMODE

Format:	[@|=]ON SERVERMODE:<audience>:<alias>
Available info:	$address - name of the server performing the mode change $limit = the current channel limit as defined by ths mode change. $lastlimit = the previous channel key before the mode change $key = the current channel key as defined by ths mode change. $lastkey = the previous channel key before the mode change $lastmode = the previous channel mode before the mode change

This event occurs when a channel mode flag is changed by a server.

ON SNOTICE

Format: [@|=]ON SNOTICE:text[;text...]>:<alias>[:+]

Available info: $address - name of the server issuing the notice

This event occurs when server notice is sent (user mode +s only) The text parameters may be text strings including wild cards See ON TEXT for more info on using text wildcards
If :+ is appended t the end of the line the message is displayed normally in the server/status window otherwise it is suppressed.

ON SOUND

Format:	[@][=]ON SOUND: <#channel>[;#channel][etc...]:<alias>
Example:	ON SOUND:*.wav:#:/display \-11 $filename played by $nick
Available Info:	$nick - nick name of the user that said <text>
	$address - address of the user that said <text>
	# - the channel name
	$filename = filename of the sound to play

This event occurs when a sound command is sent by another person. This event is only generated when the sound command is valid and the file exists. If the sound file does not exist an ON NOSOUND event is triggered instead. If this event is used, and you wish to have normal sound processing to proceed, append the :+ symbols to the end of the event handler.

ON TEXT

Format:	[@][=]ON TEXT:<text>:<*>|<?>|<#channel>[;#channel][etc...]:<alias>
Example:	ON TEXT:*!John*:#pirch:/dcc send $nick c:\pirch\ $+ $2

Available info: $nick - nick name of the user that said <text>
$address - address of the user that said <text>
# - the channel name

The ON TEXT event occurs for any incoming normal text messages, except actions and notices, as they generate the ON ACTION and ON NOTICE events respectively, however, they work exactly as outlined here.

The text parameter may use wildcards (*) as follows...

Text	literal text - user said only this word or words
text*	message starts with the indicated word or words
*text	message ends with the indicated word or words
text	the word or words are anywhere within the message

ON TEXT events can be used for channel or private messages as follows...

#	any channel
?	Any private message
#channel	literal channel name.. like #pirch
name	a private message for nick name
*	any channel or private message

ON TOPIC

Format:	[@][=]ON TOPIC:<#channel>[;#channel][etc...]:<alias>
Example:	ON TOPIC:#pirch: /topic # $lasttopic | /notice $nick Please don’t change the channel topic!

Available info:	$nick - nick name of the user that changed the topic
	$address - address of the user that changed the topic
	# - the channel name
	$lasttopic - the last active channel mode prior to the change
	$topic - the current channel topic

The ON TOPIC event occurs when another user changes the channel topic. The #channel parameters(s) indicate for which channel(s) this particular event handler will be active.

ON WALLOPS

Format: [@|=]ON WALLOPS:text[;text...]>:<alias>[:+]

Available info: $address - name of the server performing the mode change

This event occurs when an IRC Wallops message is received (user mode +w only) The text parameters may be text strings including wild cards See ON TEXT for more info on using text wildcards
if :+ is appended the end of the line the message is displayed normally in the server/status window otherwise it is suppressed.

KCommon Event Tasks & Examples

This section covers some of the more command tasks that can be done with events. This does not mean that this is how it should be done, in fact, event task can be accomplished in many different ways.

A Personal Autogreet

This event will send a welcome message to a person when that join a channel. (A personal note, some people find autogreets particularly annoying) and in not permitted

ON JOIN:#:/notice $nick Welcome to # $nick $+ , I hope you enjoy your visit.

Automated Send & Receive of Sound Files

This task will automatically request another user to send a sound/media file he/she played if it can not be found your system, and is accomplished with two events. In order for this to work, both parties must use the same format for auto send/receive events.

This first event is automatocally sends a request notice to a person that played a media file which you do not have.
ON NOSOUND:*.*:#:/notice $nick !DCCSEND $filename

This event is automatocally responds to a request notice from a person wants the file you just played.
ON NOTICE:!DCCSEND*:*:/dcc send $nick c:\sounds\ $+ $2

Automated BIO Sending

If you wish to make your personal BIO available for others to get on demand, you may set up an event such as the one below:

ON NOTICE:!BIOSEND:*:/bio send $nick

The persona who wants to view you bio simply has to issue the following command...
    /notice yournick !biosend

Redirecting Join/Part/Quit messages to the server window

By default PIRCH displays all join/part/quit messages in the channel window to which they belong. If you want to have PIRCH display them in the server/status window instead, add the following event handlers...

ON JOIN:#:/display > $server \-2 $nick [$+ $address $+] has joined #
ON PART:#:/display > $server \-3 $nick [$+ $address $+] has left #
ON QUIT:/display > $server \-7 $nick [$+ $address $+] has quit IRC ($+ $quitmsg $+)

Be sure to also set the Hide Join/Part/Quit messages option, or the messages will continue to appear in the channel window.

PIL Scripting

PIL (PIRCH Interpreted Language) scripts all you to do a variety of things not normally possible using standard aliases/events. Basically PIL is a programming language that runs within PIRCH itself, and allows you to manipulate information & text, issue commands, run calculations etc etc etc...   the possibilities with PIL are limited only to your imagination.

Introduction to PIL

Future Development

I’ve been bouncing around various ideas of things to add to PIRCH,   at this point, however, since this project is only in beta release, it would be kind of pointless in discussing too much of its future until some user reaction to its present form as been received.

Audio Chat Capabilities - My personal opinion is that audio chatting technology is at best is a novelty given the limitations of the average hardware configuration, and are usually more trouble than triumph. But generally my opinions aren’t worth 2 hoots in an owl barn, therefore, I have been tinkering with this aspect and it may be available in the next full version. (No promises)

I am open to suggestions.

Introduction to PIL

PIRCH Interpreted Language (PIL) is an internal scripting language of PIRCH Internet Relay Chat client software.

PIL's language structure is similar in syntax and structure to PASCAL, combined with the established syntax of PIRCH's alias coding structure.

Using PIL
PIL Scripts are NOT PIRCH aliases, nor are PIRCH aliases PIL scripts. While alias constructs and PIL statements have thier similarities, there are disctinct language differences, and there are key reasons for these differences which I will not cover in this document as it is slightly off topic.

PIL scripts however are installed into the PIRCH alias window/editor, and the contents of scripts are stored within the PIRCH alias file (by default aliases.paf). Its recommended that you adopt a naming convention for PIL scripts, in which the scriptname is enclose in brackets, ie [MYSCRIPT]. This allows your PIL scripts to be sorted to the bottom of the alias editor list, keeping all script together.

PIRCH PIL EXTENSIONS
PIRCH (version 0.82) introduces two new commands that are used specifically for executing PIL scripts. These are covered in detail below and in the PIRCH help file.

/RUNSCRIPT

PIL scripts are executed with the PIRCH command /RUNSCRIPT.
The format for the command is:

```
/RUNSCRIPT <scriptname> <parameters>.
```

Unlike aliases, you can not simply type /scriptname. You can however make an alias to run the script. An example will better demonstrate this point. Lets assume you have a script, called [MYSCRIPT], an you wish to simply type /MYSCRIPT to execute it... add an alias as follows

```
myscript:/runscript [myscript] *1
```

NOTE: typing /[myscript] will NOT work.

Passing information to a script
PIL scripts can access a number of system defined variables, such as $date to retrieve a number of pieces of information. However the most useful information can be the parameter string which you pass to your script when you execute a /runscript command. The parameters can be any information which your script requires or can make use of and can be accessed from within the script using the following variables

$1..$nn	Individual parameters (space delimited)
*1..*nn	Sequential set of parameters (sequential command lines parameters start with the number provided)

For example: executing `/runscript [myscript] My name is Bob`
would make $1 within your script equivalent to the word "My", $2 to the word "name" and so on. *1 would be equivalent to "My name is Bob", *2 would be "name is Bob" and so on. (each without the quote marks)

/CALLBACK

/CALLBACK is the probably the most advanced, and therefore, unfortunately most complicated command to understand and master. Basically /CALLBACK allows you to install a script to handle a variety of incoming server messages. This is similar to what PIRCH events do, and in most cases use of simple ON XXXXX is recommended. However, not all server messages are accessible in events and thats where use of these server callbacks can become useful.

For example, when you type /whois <nickname>, the IRC server returns up to four (4) distinct messages to PIRCH, and PIRCH will normally display the information for you in the server/status window. But lets assume you want to do something with this specific with this information; and since there is no ON XXXX event which covers the information returned by /whois, you can install a PIL script to handle/manipulate the returned information as desired.

The format for the /CALLBACK command is as follows:

```
        /CALLBACK <server rpl code> <scriptname>
```

Obviously to make use of this command and implement callback features you need to have information about server replies/messages and the format of these messages, all of which can be found in RFC1459 which is the official RFC protocol specification for IRC. Albeit to some degree out of date, the RFC document is still the most reliable and informative source. This document can be obtained from all complete RFC list sources and at last check was available at ftp.undernet.org

When a callback script is installed, PIRCH passes all the information contained in the server reply/message to the script as a parameter, when PIRCH encounters the specific reply/message code. This information can then be disseminated by the script in whatever fassion you desire.

Intended Audience

This document is intended for as a reference guide for persons requiring detailed descriptions of the language syntax, structure and components. Much of the document is intended for persons who have at least a fundamental understanding of structured computer languages, and the workings of procedure, functions, conditionals and loops.

This document is structured as a reference guide, and is NOT a step by step guide to writing PIL scripts.

PIL Language Definition and Structure

Embedding Comments in Scripts

Before proceeding with the technical information, a word about commenting your script code. PIL uses the pascal method for placing comments directly within script code, by embedding comments within curly braces { }. The PIL compiler ignores anything it encounters following an open curly brace { until it encounters a closing curly brace } (with the exception of string literals that contain a curly brace character.)

Although for the most part, PIL scripts tend to be readable, and understandable on their own, commenting of script code is a fundamental part of any programming language and should not be ignored.

Commenting can serve multiple purposes, providing the programmer with ability to internally document complicated sections of script code for future reference, possibly indicating why a possible method was chosen over another. If you distribute your scripts for others to use it maybe important for your intended audience to understand what a script does and/or how it does it.

Commenting can also aid in debugging: by commenting out statements, you can see how it changes the result of your script.

Keywords and Identifiers

PIRCH reserves a number of *key words* for special purposes, generally representing built-in commands and language symbols. Identifiers are words/symbols which you can create to identify variables.

BEGIN	END	FOR	TO
WHILE	DO	IF	THEN
ELSE	VAR		

Each of the above keywords are discussed in detail later in this document.

Variables

In PIL as in most other programming/scripting languages, variables hold the data on which the script operates. PIL currently supports two primary variable types:

Strings	Alpha-numeric character arrays
Numbers	Numeric values

(internally stored as a 32 bit signed integer with a value range of -2147483648..2147483647

Declaring Variables

PIL does not currently use an explicit forced variable declaration system, instead PIL will dynamically create and allocate variables as required during the script compilation phase. The disadvantage is that PIL must provide a method of type identification for variables according to the identifier used.

The variable identifier used will explicitly declare it type by the following rule: all string variables will be prefixed with a single $ symbol and numeric variable types my not contain the $ symbol.

$workstr	is explicitly typed as string
numvar	is explicitly typed a numeric variable

Attempting to assign a string expression to a numeric variable will result in a compilation error of “type mismatch”, as will assignment of a numeric expression to a string variable. (see STRTOINT and INTTOSTR function later in this document for type conversion)

Literal Values

A literal value is simply a declared number or string value. String literals must be enclosed within single quotation marks (‘).

50	Declares a numeric literal value
‘hello’	Declares a literal string value

Constants

At its current stage of development, PIL does not support strict constants. However, programmers can achieve almost identical functionality by using variable declarations (with immediate value assignment).

Example: five := 5; would be functionally equivalent to a strict constant declaration. The only difference is that should an assignment attempt be made upon the variable, the assignment would be permitted and the script would continue to execute, whereas attempted assignment to a strict constant would result in a fatal error at compilation time.

Assignment of Values to Variables

The values of variable can be changed through one of two methods:

A. Assignment of an expression
Assignment is achieved by the use of the := symbol, with a variable on the left side and a like type expression on the right.

numvar := 5 * 20;	would assign the value 100 to numvar
$s := 'hello world';	would assign 'hello world' to the $s
$s := 5 * 20;	would result in a 'type mismatch' error

Once an assignment is made, any usage of the variable in an expression would actually be using the value associated with the variable. The value of variable may be changed during the execution of a script as often as you like.

B. Use of a variable as a VAR type parameter to a procedure/function call

Attempting to assign a string expression to a numeric variable will result in a compilation error of "type mismatch", as will assignment of a numeric expression to a string variable. (see STRTOINT and INTTOSTR function later in this document for type conversion)

Global System Variables

Global systems variables contain information that may change according to system state, however are not true PIL variables, meaning that you may not directly values to them, or use these variables as VAR parameters in procedure and/or function calls. In all cases these identifiers begin with the $ symbol

$day	string	Returns the current day of the week
$date	string	Returns the current system sate
$time	string	Returns the current system time
$server	string	Returns the name of the server to which the windows from which the script is called is associated
$host	string	returns your current local host name
$ip	string	returns your IP address in 4 octet form
$me	string	returns your current nickname
$netid	string	returns the server's network id (if assigned)
$audience	string	returns the channel/nickname of the window from which the script is called
$topic	string	returns the channel topic (only if called from a channel)
$mode	string	returns the channel mode (only if called from a channel)
$members	string	returns the channel member count (only if called from a channel)
$version	string	returns the PIRCH version you are running
$input	string	returns a string entered from a dialog box
$snick	string	returns a selected nickname by position
$nicklist	string	returns a nickname from the channel member list by position
$isop	integer	returns 1 if a nickname is a channel operator, otherwise 0
$hasvoice	integer	returns 1 if a nickname has mode +v set

Procedures and Functions

The following is a list of internally support procedures and functions, each of these is discussed later in this document.

BREATHE	CHAR	HALT	RANDOM
STRCOPY	STRDEL	STRINS	STRLEN
STRPOS	STRMATCH	STRUPPER	STRLOWER
STRTOKEN	INTTOSTR	STRTOINT	WRITELN
COMMAND			

Simple Expressions & Statements

Definitions

An expression combines constants, variables and function results into a single result. There are 3 primary expression types recognized by PIL: numeric, string and boolean.

Numeric Expressions & Operators

Numeric expressions may be either simple numeric variables, numeric constants numeric literals, or a mathematical manipulation of numeric identifiers using the following operators.

Mathematical Operators

+	addition
-	subtraction
*	multiplication
/	division
^	exponential
mod	modulus

Bitwise Operators

shl	bitwise shift left
shr	bitwise shift right
bitand	bitwise AND
bitor	bitwise OR
bitxor	bitwise XOR
bitnot	bitwise NOT

Boolean (Logical) Expressions

Boolean expressions result in a return type of either true or false and generally require the use of comparison symbols and an expression on both the left and right side. While there is no explicit boolean type declared within the PIL language structure,   PIL generates ordinal values for boolean expressions, where 0 = false and 1 = true.

Symbol	**Comparison**	**Example**
=	equal	(a = b)
>	greater than	(a > b)
>=	greater than or equal	(a >= b)
<	less than	(a < b)
<=	less than or equal	(a <= b)

Logical expressions may also include logic operators. Logic operators can be used to evaluate two boolean expressions located on the left and right sides of the expression. Currently PIL supports the following logical operators.

Operator	**Comparison**	**Example**
AND	Logical And	(a and b)
OR	Logical Or	(a or b)
NOT	Logical negation	(not b)

The following boolean truth table displays how the above boolean operators affect and expression

A	**AND**	**B**	**=**	**C**
false	|	false	|	false
false	|	true	|	false
true	|	false	|	false
true	|	true	|	true

A	**OR**	**B**	**=**	**C**
false	|	false	|	false
false	|	true	|	true
true	|	false	|	true
true	|	true	|	true

The NOT operator is used to negate a boolean ordinal.
NOT true is equivalent to false
NOT false is equivalent to true

Operator Precedence

PIL interprets all expression from left to right, but applies precedence to operators according to the following table, ranging from highest precedence to lowest:

Parenthetical	()
NOT, unary minus	not, -Expression
exponents	^
multiplication,division,modulus	*, /, mod
addition,subtraction	+.-
AND,OR,XOR	and,or,xor

Examples

2 + 3 * 4 result = 14
multiplication has a higher precedence than addition

4 * (3 + 2) result = 20
parenthesized expression (2+3) is evaluated prior to multiplying by 4.

Single Statements

A statement is either a procedure call or an assignment of an expression or function result to a variable. Each statement must be terminated with a semicolon (;).

Compound Statements

A compound statement is a group of individual statements, each separated with a semicolon (;). You can use compound statements anywhere an ordinary statement is allowed. Because compound and single statements are interchangeable, the term *statement* will be used to mean either a compound or singular statement from now on throughout this document.

Simple compound statement construct

```
begin
    a := b * c;
    $s := ‘answer’;
    writeln(‘$s = ‘,$s,’   a = ‘,a);
end;
```

Repetitive Statements (loops)

PIL normally executes code in a sequential order, top to bottom, left to right, one statement after the other. However, this default behavior can be modified by use of *repetitive* statements which makes loops in a script, repeating operations. and *conditional* statements which make decisions, selecting one statement over others and changing the flow of the script.

Repetitive statements cause one or more statements to repeat. There are two (2) repetitive statement constructs used in PIL, WHILE and FOR loops.

WHILE/DO Loops

WHILE loops cause a statement or statement group to repeat as longs as a given boolean expression is evaluated to be true.

```
SYNTAX: while <boolean expression> do <statement>
```

Simple WHILE loop construct

```
a := 0;
while a <= 10 do
begin
    a := a + 1;
    writeln(a);
end;
```

The above example counts from 1 to 10, displaying each number on your screen.

Its quite possible and legal that the boolean expression within a while statement to be initially evaluated as false, causing the statements controlled by the loop never to be executed. In the above example, assume we initially set the value of *a* to 20, then the *a := a + 1;* statement would never have executed.

FOR/TO/DO Loops

FOR loops cause a statement block to execute a specific number of times.

```
SYNTAX: for <control variable> := <source expression> to <target expression> do
<statement>
```

Simple FOR loop construct

```
a := 1;
for a := 1 to 10 do
begin
    writeln(a);
end;
```

The above example counts from 1 to 10, displaying each number on your screen. This is identical to what the example for the WHILE loops does, but the for loop is much more efficient in performing the task.

FOR loops require a control variable, which must be numeric. The control variable is initially assigned the value of the *source expression*, and subsequently modified by 1 during each iteration of the loop, executing the statement block until it reaches the *target expression* value.

As with the WHILE loop, a FOR loop may never execute its statements, if the *source expression* value is initially greater than the *target expression* value.

Conditional Statements

IF/THEN/ELSE Statement

IF statements allow your script to make decisions about what statements to execute based on a logical expression that evaluates to either true or false.

Syntax: IF <expression> THEN <statement> [ELSE <Statement>]

Simple IF/THEN/ELSE statement construct

```
if a <> 0 then
    writeln('a does not equal 0')
else
    writeln('a does equal 0');
```

In it is not required than an ELSE clause be provided for each if statement, rather this is optional and may be omitted entirely.

Simple IF/THEN statement construct (ELSE clause omitted)

```
if a <> 0 then
    writeln('a does not equal 0');
```

You can chain multiple IF/THEN/ELSE statements together, allowing your script to make a series of decisions, by simply using another IF statement following an ELSE clause.

```
if a = 0 then
    writeln('a is 0')
else if a < 10 then
    writeln('a is greater than 0 but less than 10')
else if a < 50 then
    writeln('a is greater than 10 but less than 50')
else
    writeln('a is greater than 50');
```

Procedures & Functions

PIL currently doesn't not allow user defined procedures and functions to be created within a script, however, this will be implemented in future versions.
Procedures and functions are still an integral part of the PIL definition and the concepts and definitions need to be understood by the programmer.

Since current implementations of PIL support only two primary variable types, this document will address types as either being string or value.

Procedures

PIL defines a procedure as a subroutine or that acts as an individual statement. Procedures may not be used as a component in an expression, as procedures themselves generate not assignable return value.

Functions

Functions, like procedures, are defined as subroutines, however, functions in and of themselves do not constitute a complete statement. Functions may instead be used as a component in an expression, as all functions by definition must return an assignable value.

This document will use the pascal standards for define of procedures and functions.

```
procedure procname(param[,param] : type[; [...]]);
function funcname(param[,param] : type[; [...]]) : type;
```

PIL Procedures and Function Reference

The following is a list of internally support procedures and functions, each of these is discussed later in this document.

BREATHE
CHAR
COMMAND
HALT
RANDOM
INTTOSTR
STRCOPY
STRDEL
STRINS
STRLEN
STRPOS
STRMATCH
STRUPPER
STRLOWER
STRTOKEN
STRTOINT
WRITELN

BREATHE

Declaration
procedure breathe;

Description
Breathe is used to allow the Microsoft Windows system to process messages during lengthy or time consuming script operations. The use of this procedure is never a requirement. When breathe is called, the PIL interpretor returns processor control back to the system and PIRCH will process any pending Windows messages, including processing of incoming data from the server. NOTE: Because a PIRCH event may be triggered during a breathe operation, you must take precautions against the same script being activated via and event and causing recursion.

Example

```
begin
       answer := 0;
       for x := 1 to 5000 do
       begin
                     breathe;
                     answer := answer + 2;
       end;
       writeln(‘the answer is ‘,answer);
end;
```

CHAR

Declaration

function char(index ; value) : string;

Description

Char is returns a 1 character length string holding the character at position index in the system's character table. For ASCII character sets index should be a value ranging from 0 to 255.

Example

```
begin
       $chan := char(35)+'pirch'; { prefix # symbol }
       command('/msg',' ',$chan,' ','hello all');
end;
```

COMMAND

Declaration
procedure command(v1,v2,...,vn);

Description
Use Command to issue IRC or PIRCH specific commands from within a PIL script. Separate multiple items with commas within Command's parenthesis. This items may be of different types, i.e. strings, values or expressions. The resulting syntax of the parameters used for Command must result in a valid command line string which PIRCH can interpret.

Example

```
begin
       $chan := ‘#pirch’;
       command(‘/msg’,’ ‘,$chan,’ ‘,’hello all’);
end;
```

FILEEXISTS

Declaration
function fileexists($filename : string) : value;

Description
Fileexists returns a boolean ordinal (0 = false, 1 = true), if *filename* exists on the system. Fileexists makes no assumptions about the path for the file, therefore, you should use fully qualified filenames with this function.

Example

```
begin
       $filename := ‘c:\pirch\logs\#pirch.log’;
       if Fileexists($filename) then
           writeln(‘#pirch log file was found’)
       else
           writeln(‘#pirch log file was NOT found’);
end;
```

FILEREAD

Declaration
function fileread($filename : string; linenumber : value; VAR $s : string) : value;

Description
Fileread reads a string value from $filename. Linenumber may be 0, in which case fileread will read a random line from the file, otherwise linenumber indicates the specific line to be read, where the first line in the file is line number 1. $s must be a variable string type, which is filled with the information read from the file. The return value is a boolean ordinal indicating whether or not the operation suceeded. (1 means the operation was successful, 0 means it failed) Fileread makes no assumptions about the path for the file, therefore, you should use fully qualified filenames with this function.

Example

```
begin
       $filename := ‘c:\pirch\logs\#pirch.log’;
       if fileexists($filename) then
       begin
           if fileread($filename,-1,$s) then
              writeln($s)
           else
              writeln(‘unable to read a line’);
       end
       else
           writeln(‘unable to find file: ‘,$filename);
end;
```

FILESIZE

Declaration
function filesize($filename : string) : value;

Description
Filesize returns a value indicating the size in bytes of a file indicated by $filename. Filesize makes no assumptions about the path for the file, therefore, you should use fully qualified filenames with this function. If the file is not found, or other error occurs during an attempt to retrieve the file size, this function will return -1.

Example

```
begin
       $filename := ‘c:\pirch\logs\#pirch.log’;
       if Fileexists($filename) then
       begin
          size := filesize($filename);
          writeln(‘#pirch log file size is ‘,size,’ bytes’);
       end
       else
          writeln(‘#pirch log file was NOT found’);
end;
```

FILEWRITE

Declaration
function filewrite($filename : string; linenumber : value; $s : string) : value;

Description
Filewrite reads a string value to $filename. Linenumber may 0, in which case filewrite will append $s to the end of the file, or linenumber maybe any other value indicating the line number: If the value is negative, the existing line at position *linenumber* will be replaced; if the value is positive, the line will be inserted at position *linenumber*. For example, indicating linenumber as -1 would cause the first line to be replaced with the information in *$s*. If linenumber is set to 1, then *$*s would be inserted as the first line in the file and any other lines will be pushed downward. The return value is a boolean ordinal indicating whether or not the operation suceeded. (1 means the operation was successful, 0 means it failed) Filewrite makes no assumptions about the path for the file, therefore, you should use fully qualified filenames with this function.

Example

```
begin
       $filename := ‘c:\pirch\logs\#pirch.log’;
       if fileexists($filename) then
       begin
           if filewrite($filename,0,$s) then
              writeln(‘operation complete’)
           else
              writeln(‘operation failed’);
       end
       else
           writeln(‘unable to find file: ‘,$filename);
end;
```

HALT

Declaration
procedure Halt;

Description
Halt causes a script to terminate, and is useful when you want to stop processing depending on a given condition.

INTOTOSTR

Declaration
function inttostr(n : value) : string;

Description
Inttostr converts a value a string representation of the value.

Example

```
begin
       value := 12345;
       $s := inttostr(value);
end;
```

RANDOM

Declaration
function random(range : value) : value

Description
Random returns a random number ranging from 0 to *range*.

Example

```
begin
       DiceA := random(5)+1;
       DiceB := random(5)+1;
       writeln(‘You rolled a ‘,DiceA,’ and a ‘,DiceB);
end;
```

STRCOPY

Declaration
function strcopy($s : string; index, len : value) : value;

Description
Strcopy is used to return a sub-string from within *$s*, starting at *index* and is at most *len* characters long. If there are fewer characters in *$s* from *index* to the end of *$s* than *len*, strcopy returns only as many characters as it can.

Example

```
begin
       $msg := ‘this is a test’;
       $newmsg := strcopy($msg,1,4);
       writeln($newmsg);
end;
```

STRDEL

Declaration
procedure strdel(var $s : string; index, len : value) : value;

Description
Strdel removes *len* characters from *$s* starting at index. *$s* must be a string variable. If there are fewer characters in *$s* from *index* to the end of *$s* than *len*, strdel removes only as many characters as it can.

Example

```
begin
       $msg := ‘this is a test’;
       strdel($msg,1,4);  {remove the word ‘this’}
       writeln($msg);
end;
```

STRINS

Declaration
procedure strdel($source : string; VAR $target : string; index : value) : value;

Description
Strins inserts *$source* into *$target* at position *index*.

Example

```
begin
       $msg := ‘this is a test’;
       strins(‘new ‘,$msg,11);
       writeln($msg);
end;
```

STRLEN

Declaration
function strlen($s : string) : value;

Description
Strlen returns the length of a string.

Example

```
begin
       $msg := ‘this is a test’;
       len := strlen($msg);
       writeln(‘The length of the message is ‘,len);
end;
```

STRLOWER

Declaration
function strlower($s : string) : string;

Description
Strlower returns the a copy of the string parameter with all letters converted to lowercase.

Example

```
begin
       $msg := ‘This is a test’;
       $msg := strlower($msg);
       writeln($msg);
end;
```

STRPOS

Declaration
function strpos($searchstr, $s : string) : value;

Description
Strpos returns the position at which *$searchstr* is found within *$s*. If *$searchstr* is not found, strpos returns 0.

Example

```
begin
       $msg := ‘This is a test’;
       index := strpos(‘is’,$msg);
       writeln(““is” was found at position ‘,index);
end;
```

STRMATCH

Declaration
function strmatch($pattern, $s : string) : value;

Description
Strmatch returns a boolean ordinal indicating whether or not $s matches a pattern specified in $pattern. $pattern may use question marks (?) and/or asterisks (*) as wildcards.

Example

```
begin
       $msg := ‘This is a test’;
       $pattern := ‘*is*’;
       if strmatch($pattern,$msg) then
           writeln(‘it matches’)
       else
           writeln(‘no match’);
end;
```

STRTOINT

Declaration
function strtoint($s : string) : value;

Description
Strtoint converts a string to a value. The string *$*s must contain only the character ’0’..’9’.

Example

```
begin
       $s := ‘12345’;
       value := strtoint($s);
end;
```

STRTOKEN

Declaration
function strtoken(var $s : string) : string;

Description
Strtoken removes and returns the first word from within a *$s*.

Example

```
begin
       $msg := ‘This is a test’;
       $newmsg := strtoken($msg);
       writeln(newmsg,’ : ‘,$msg);
end;
```

STRUPPER

Declaration
function strupper($s : string) : string;

Description
Strupper returns the a copy of the string parameter with all letters converted to uppercase.

Example

```
begin
       $msg := ‘This is a test’;
       $msg := strupper($msg);
       writeln($msg);
end;
```

WRITELN

Declaration
procedure writeln(v1,v2,...,vn);

Description
Use writeln to write information to a PIRCH window. By default, writeln will use the server window, from which the PIL script is associated, however this may changed by setting the PIRCH system variable DEBUGWIN with the /set command. The following demonstrates on way of redefining the target output window from the PIRCH command line.

```
/newwindow DEBUG
/set debugwin DEBUG
```

Separate multiple items with commas within writeln's parenthesis. This items may be of different types, i.e. strings, values or expressions.

Example

```
begin
      answer := 25 * 4;
      writeln('The result is ',answer);
end;
```

sions¡ŒÎXÊ’ŸGãZ Values to VariablesThe values of variable can be changed through one of two methods:A.Assignment of an expression

€€€

‚ÿ

ñ