BC MANAGER

Version 2.5.0

MANUAL

MOUNTAIN UTILITIES

http://mountainutilities.eu/

CONTENTS

1. Overview

BC Manager is a utility by Mountain Utilities for working with the Behringer BCF2000 and BCR2000 USB MIDI Controllers. It is free, although donations are more than welcome.

A few notes on terminology used in this manual:

- 'BCF' means the BCF2000, 'BCR' the BCR2000.
- 'BC' is sometimes used as a generic term for the BCF2000 and the BCR2000.
- 'BCL' (Behringer's term): this probably stands for 'B-Control Language', the language in which all the BCF2000 and BCR2000's settings are sent, embedded in MIDI System Exclusive messages.
- 'Element': a generic term (also introduced by Behringer) for buttons, encoders and faders.
- *BCMI*: the document BC MIDI Implementation.pdf, available from the Mountain Utilities web site.

BC Manager provides the following facilities (among many others):

- Maintenance of up to 16 BCs simultaneously: you can send and receive global settings and presets.
- Access to the BC's internal global setup history.
- A hardware test mode, allowing you to test whether a BC's buttons, encoders and faders work correctly.
- For the BCF2000 in emulation mode (except bhuI): a window showing the messages which the controlling MIDI device sends to the BCF.
- You can save all BC data as 'syx', 'txt' or 'bc2' files. BC Manager opens and saves all these formats, so no manual, external conversion is needed between syx, txt and bc2. (Actually BC Manager does include a conversion utility, but you never need to use it for BC Manager itself.)
- Display of all global, preset and element settings in on-screen tables.
- Graphical editing of all BC settings. (So you don't have to know BCL.)
- You can edit a particular setting for a range of elements (or even presets) in one go, either to a single value or incrementally.
- Full support for 'custom' ('**`.tx`**') MIDI messages: you can edit these graphically too.
- A 'SysEx preprocessor': for many MIDI devices (e.g. many models by Roland/BOSS, Waldorf and Yamaha) it is possible to use special 'SysEx model definition files': in this way you only have to tell BC Manager which parameter of a specific synthesizer (etc.) you want to assign to a particular button, encoder or fader on your BC, and BC Manager's SysEx preprocessor generates the appropriate MIDI System Exclusive message definition automatically.
- Direct capture of BCF/BCR snapshots.

- Standard copy-and-paste facilities for presets and elements.

- Comparison of presets and elements.

- You can provide labels for elements (and print them — see below). Furthermore, you can add your own comments (of unlimited length) to the global setup, to each preset and to each element.

  Beware: these labels and comments cannot be stored on the BC itself. Therefore it is recommended that you only *send* preset definitions containing labels and/or comments to the BC, but don't attempt to *receive* those definitions back from the BC.

- Various printing facilities:
  - You can print the table window *images* as such.
  - You can print a page containing 'strips' specifying the elements' labels: if you cut these out, you can put them on your BCF or BCR.

- You can export a table to a specially formatted 'txt' file that you can use as an input file to create a table in an external word processor. (In any case this works for WordPerfect; presumably it works for other word processors as well.)

- Many generic MIDI input/output tools, such as a MIDI message recorder, a SysEx editor and a mouse-controlled MIDI keyboard with chord and arpeggio facilities. There are also virtual displays for Mackie Controls (much like those provided by the Huskervu utility).

- For testing purposes, BC Manager incorporates a seamlessly integrated BCL editor: you can load the internally maintained state of a BC (or even a single preset) directly into a BCL text editor window, and you can have BC Manager 'execute' the text from this window or send it directly to the BC via MIDI.

- Visualization of Propellerhead Reason's native BCF2000/BCR2000 parameter mappings.

2. Version history

Version 2.5.0 (2014-02-05):

- A window in which you can perform MIDI data byte conversions.
- SysEx model definition files can now contain labeled parameter sections, which can be nested up to any depth. The 'Insert SysEx parameter' dialog box displays these sections in an expandable tree. (This dialog box has also been improved in several other ways.)
- Keyboard shortcuts to numerical edit boxes with up and down arrows work again. (This was broken in version 2.4.2.)
- Big changes to this manual:
  - Many screenshots of windows and dialog boxes.
  - Several new sections dedicated to particular windows.
  - A tutorial section (§7) explaining how to work with presets.
  - Many small improvements.

Version 2.4.2 (2013-11-21):

- The maximum of *AddressLength* in SysEx model definition files has been raised from 4 to 8. (Make sure you set *ProgramVersion* to 2.4.2 if *AddressLength* is higher than 4.)
- The dialog box in which you can type a parameter knob's value can now also be opened by left-clicking the knob while holding an Alt key. (See §26.)
- The Help pull-down menu of the main window provides access to a window listing the computer keyboard and mouse actions that can be applied to parameter knobs.
- BC Manager uses the new Mountain Utilities web site at http:/mountainutilities.eu/ in links and its update mechanism.
- New in this manual:
  - A discussion of MIDI Yoke's behavior under Windows versions with UAC.
  - A discussion on using multiple BCFs/BCRs via USB. (This discussion was already present in the separate document Windows USB drivers for BCc2000.pdf.)
  - Improvements in the section about the SyxEx model definition file format.

Version 2.4.1 (2013-04-02):

New features:

- The 'MIDI System messages' window, which allows you to send System Exclusive, System Common and System Real-time messages for testing purposes.
- A significant overhaul of the MIDI controllers window, with many new display options.
- The MIDI keyboard window can now switch a key off by a Note On message with Velocity 0 (as before) or by a Note Off message (with customizable 'offset velocity'). Press the 'Velocity options' button to edit all velocity-related settings; all these settings are retained between runs of BC Manager.

Bug fixes:

- In the MIDI keyboard window: a bug (introduced in version 2.4.0) concerning arpeggios and chords was fixed.
- In the 'MIDI System Exclusive message(s)' dialog box (opened from the MIDI System Exclusive messages window): if the sequence contains an invalid byte, the dialog box popping up when you press OK now shows the correct SysEx message number.

Version 2.4.0 (2012-08-23):

New features:

- The MIDI program changer window: this allows you to send MIDI Program Change messages

prefixed by Bank Select MSB/LSB messages, in any combination.

- In the MIDI keyboard window, each note's loudness ('onset velocity') is determined by the mouse's vertical position on the key.
- LoopMIDI has been added to the list of known virtual MIDI ports ('pipes').

  As in previous versions, when you start BC Manager for the first time you must choose to disable or enable all MIDI pipes in BC Manager, and this now includes all loopMIDI ports, or rather those ports that have 'loopMIDI' in their names — as suggested by loopMIDI. However, loopMIDI allows you to change these names completely — if you do so, BC Manager won't recognize them as MIDI pipes, so then you can only enable them manually. i.e. via the MIDI devices dialog box.
- The 'Import Reason map' operation can load Propellerhead Reason 6.5.1's map files for the BCF2000 and BCR2000 (BCF2000.remotemap and BCF2000.remotemap respectively).
- In SysEx model definition files, ManufacturerID can now consist of multiple bytes.
- Seven SysEx model definition files for the Elektron Machinedrum, provided by Bob Svitilla. Many thanks again!

Improvements:

- Windows that were open when the previous session of BC Manager terminated reappear exactly where they were. (In previous versions of BC Manager, even a window previously on a secondary monitor always reappeared on the primary display, even when the secondary monitor was still available.)

  This new behavior has one potentially problematic consequence: when you remove a monitor or reduce the screen resolution, windows may become invisible upon a restart of BC Manager. To remedy this, you can use the new 'Make fully visible' operation in the 'Window list' dialog box (accessible via the main window's View pull-down menu or the Alt+0 key combination).
- The 'Test hardware' dialog box now includes an explanation of what you can do (namely manipulate elements in synchrony and run a 'light show').
- When sending or saving a BC's Global Setup (e.g. via MIDI ⇨ 'Send all data' from the B-Controls window), BC Manager no longer includes a BCL message that sets the BC's device ID (as occurring in the Global Setup). Thus, a BC can no longer suddenly become inaccessible in the middle of a Send operation due to an unintentional change to its device ID.

  Accordingly, the 'Device' column in the 'Global setups' window has been renamed to 'Device (ignored)' and you can't edit the device ID in the Global Setup dialog box any more. For reference purposes only, the 'Device (ignored)' column still shows the device ID *received* from a BC or *read* from an old-style or third-party syx/txt/bc2 file; however, these values are never *sent* or *saved*.

  As before, BC Manager only uses the 'Device' value defined in the 'MIDI options' dialog box (accessible from the B-Controls window via MIDI ⇨ Options) for all its device ID 'fingerprints' in communication with the BC and in syx files.

  So from now on, you can only change a BC's device ID manually on the BC itself via its Global Setup edit mode (EDIT + STORE ⇨ 'EG'); of course you still also have to update the device ID as maintained by BC Manager via the 'MIDI options' dialog box. Note that certain Behringer USB device drivers include the BC's device ID in the BC's MIDI I/O port names; in this case you also have to restart BC Manager and any other running programs still using the old MIDI I/O ports.
- When saving a BC's data (presets/global setup) to a syx file, BC Manager now by default uses the generic device ID ($7F) for each BCL SysEx message's 'fingerprint', rather than the selected BC's device ID (1-16), This ensures that any BC (of the same type, i.e. BCF or BCR) accepts the messages from this syx file when sent via a SysEx-uploader such as MIDI-OX, irrespective of the BC's actual device ID. (For BC Manager the device ID fingerprints stored in syx files don't matter: when opening a syx file, BC Manager ignores these; when sending SysEx data to a BC,

BC Manager uses the BC's actual device ID (1-16) as defined in the 'MIDI options' dialog box.)

If you want to restore the old saving behavior (i.e. use the selected BC's device ID as message fingerprints), open the 'BCL output options' dialog box (accessible from the main window via the Options pull-down menu) and set 'Device to save to syx file' to 'Selected B-Control'.

- The Insert SysEx parameter dialog box no longer displays a definition file's Comment field as a single line (cutting off any characters that don't fit the width of the dialog box), but in a scrollable, word-wrapping multi-line text box. (Note that the Comment field itself can still only be single-line in the definition file.)

Version 2.3.2 (2011-10-21):

Bug fix:

The 'Paste bare hex sequence' operation in the 'Output statement' dialog box now works correctly.

Improvements:

- A verification dialog box pops up whenever you are about to send a global setup to a B-Control, warning you that changing the B-Control's Operating Mode may invalidate its MIDI/USB ports.
- In the BCR's encoder list window, the LEDs column now shows 'INVALID ...' if a 'fancy' setting is chosen for a 'normal' (i.e. non-push) encoder. (A new question in the FAQ section of this manual discusses this phenomenon.)
- Invalid message indexes in the main window's 'Convert syx to txt/bc2' operation are now reported with their most significant byte first, in harmony with the error report generated when the same file is opened in the B-Controls window.
- For clarity, error messages concerning B-Control SysEx messages now add the $ prefix to the hexadecimal command number.
- Minor stylistic improvements to this manual.

New:

Two new SysEx model definition files: one for the Alpha Juno 1 (by Bob Svitilla) and one for the Waldorf Q (by Sven Baus). Many thanks!

Version 2.3.1 (2011-03-10):

The Mackie monitors no longer display garbage.

Version 2.3.0 (2011-03-07):

- Many improvements concerning SysEx model definition files. Most importantly:
  - The format of SysEx model definition files has been widened in several ways. It now supports a much wider variety of manufacturers, such as Waldorf.
  - The BC Manager package contains a few new SysEx model definition files and templates.
  - You can create a CSV report for the parameters in a SysEx model definition file.
  - In this manual the section about the 'Insert SysEx parameter' dialog box (§18) has been revised to reflect all the changes. Also, there are now tables specifying some common SysEx formats (Roland/BOSS, Waldorf and Yamaha).
- For clarity, the name of a *disabled* MIDI I/O device is no longer prefixed by an asterisk (*), but by 'DISABLED ', and the name of an *absent* MIDI I/O device is no longer written between parentheses, but prefixed by 'ABSENT '.
- The 'Check for update' dialog box opens on first startup of BC Manager. (Previously a silent update check was performed in this situation.)

Version 2.2.1 (2011-01-09):

- Fixed a version 2.2.0 regression by which the size of a Mackie monitor window sometimes became incorrect.

Version 2.2.0 (2011-01-07):

New features:

- Unicode support: in most text edit boxes in the program (e.g. for file names) you can now enter any 'international' characters. But of course BCL as sent to/from the BC still only allows ASCII characters (from #32 to #127) in preset names. You *can* include Unicode characters in *comments* and *element names*, but these will only be retained if you save to a *txt* file: syx and bc2 files can't contain Unicode characters (although their file *names can*). (Technical note: BC Manager now saves txt files and its own ini file in UTF-8 format, starting with the three-byte UTF-8 BOM (byte order mark); however, it can still read any plain ASCII/ANSI txt/ini file.)
- As a consequence of its new Unicode support, BC Manager no longer runs under Windows 95, 98 or Me, because these operating systems do not support Unicode. (If you require a version of BC Manager that runs under these operating systems, please send a message to the contact address at the Mountain Utilities web site.)
- The 'Output statement' dialog box can paste a sequence of bare hexadecimals while automatically adding the $ prefixes required in BCL's '**`.tx`**' statements.
- The 'Insert SysEx parameter' dialog box displays the device's manufacturer and version, date and author of the SysEx model definition file. The dialog box also has a button that opens the folder containing the SysEx model definition files in Windows Explorer, and a button that opens the folder containing the templates.
- The 'Import Reason map' operation can load Propellerhead Reason 5's map files for the BCF2000 and BCR2000 (BCF2000.remotemap and BCF2000.remotemap respectively). Since each of these files contains more devices (38) than there are memory presets (32) in a B-Control, BC Manager imports such a file into *two*, *new* B-Controls. Accordingly, BC Manager no longer imports a Reason 3 or 4 map file into the *currently selected* B-Control, but into a (single) *new* B-Control.

  To clarify the above-mentioned altered procedure of 'Import Reason map', the 'Import Reason map' operation has been moved to the 'B-Controls' pull-down menu in the B-Controls window.
- The BC Manager package now includes a SysEx definition file (written by Patrick Manderson) for 4-operator Yamaha FM synthesizers.

Improvements:

- Some error messages concerning MIDI System Exclusive messages now specify the BCL line number in which they occur. This makes it easier to pinpoint the location of the error, e.g. in a corrupt syx file being opened.
- In the 'Output statement' dialog box, the edit box is a lot wider than before, which is useful for long lines.
- BC Manager no longer rejects a SysEx model definition file if it contains one or more lines with only spaces.
- A few cosmetic changes, e.g. to the main window's Help menu.
- Upon first startup, BC Manager now copies *all* the shipped SysEx model definition files from the installation folder to the user's application data folder.
- The questions in the FAQ section of this manual are numbered.
- Miscellaneous improvements and additions to this manual.

Bug fixes:

- The 'Global setup history' operation (in the menu of the B-Controls window) can no longer be executed if no B-Controls are defined.
- The 'Global setup history' window shows up in the middle of the screen.
- Patrick Manderson has provided a corrected version of the TX81Z SysEx definition file.

Version 2.1.2 (2010-11-07):

A blank toolbutton in the BCR's preset list window was removed. This toolbutton was erroneously

included in versions 2.1.0 and 2.1.1. (It was used for testing, but would usually only generate an error message when you pressed it.)

Version 2.1.1 (2010-11-05):

- Several improvements to the BCF and Mackie Control virtual displays: they now stay on top of the windows in other applications consistently and they have a few new settings (which can be customized via their dialog boxes).
- The package now includes SysEx definition files for the Yamaha AN200 and TX81Z synths, written by Patrick Manderson. (See §18 on how to use these files.) Many thanks to Patrick!
- In this manual (particularly in the FAQ section): a few updates and improvements.

Version 2.1.0 (2010-09-23):

New features:

- Up to four virtual displays showing the text messages which certain MIDI applications send to Mackie Controls. (You can use these displays if you have set up your BCF (in standard B-Control mode) or BCR as a Mackie Control.)
- For the BCF2000 in emulation mode (except bhuI): a virtual display showing the text messages which the BCF outputs.
- You can keep BC Manager's main window on top of BC Manager's other windows and other applications.
- A global setup history window (accessible from the B-Controls window's MIDI pull-down menu via Maintenance). This allows you to view and edit the BCF/BCR's internal global setup history.
- A column in the presets window showing the currently used data size.
- This manual's FAQ section (under 'MIDI connection') describes how a BCF/BCR can communicate with a synthesizer and BC Manager simultaneously.

Improvements:

- A hint in the 'MIDI System Exclusive messages' window has been improved.
- In this manual, the subsection on USB drivers has been rewritten completely, to take into account the new Behringer drivers (published December 2009).
- A few minor improvements to this manual and to BC MIDI Implementation.pdf (*BCMI*).
- The link to the Mountain Utilities web site installed in Windows' start menu has been updated.

Bug fix:

- The data size of $FE (Active Sensing) in LEARN/custom output is now counted correctly.

Version 2.0.3 (2009-11-02):

All references to the Mountain Utilities web site have been updated from 'home.hetnet.nl' to 'home.kpn.nl'. Hence the automatic update mechanism will work again.

Version 2.0.2 (2009-08-06):

New features:

- From the System Exclusive window, you can extract the System Exclusive messages from a standard MIDI file and save them to a System Exclusive file.
- Facilities for automatic and manual checking whether an update of BC Manager is available from the Mountain Utilities web site.
- *BCMI* is now included in the BC Manager package, so that you can open *BCMI* from the Windows Start Menu and from BC Manager's Help pull-down menu.
- A request for a 'complex paste' operation is discussed in this manual's FAQ section.

Improvements:

- The package uses a different installer (Inno Setup instead of InstallShield). Consequently, the zip file's size has gone down from 4.5 MB to 1.5 MB, even though it now also includes *BCMI*! You may also note a few minor differences in the installation procedure.

- Every individual data section (global setup, preset specs, button, encoder, fader) that you save to a syx file or send to a BC can now contain up to 16384 BCL lines. Thus, there is hardly any danger any more of getting a 'too many BCL lines' error message, which could previously occur for a batch of very big presets (such as Reason 4's BCR mapping).
- In the table windows, the widths of the File and MIDI asterisk columns are now initialized to accommodate only the asterisks themselves (rather than the headers' 'F' and 'M', which led to different widths for the File and MIDI columns).
- After the 'Test hardware' routine, the LEDs of the LEARN, EDIT and EXIT buttons can no longer remain lit indefinitely. (Previously this occurred if these LEDs happened to be lit when you quit the 'Test hardware' dialog box: even switching to a different preset wouldn't clear these LEDs. In fact, this problem was caused by a bug in the BC firmware that affects *any* transition from '**`.fkeys off`**' to '**`.fkeys on`**'.)
- Some typos and inaccuracies in this manual's FAQ section were corrected.

Version 2.0.1 (2009-04-30):

New feature:

- The *Value 2* parameter can now be **`off`** for standard CC, NRPN, AT and GS/XG button definitions. (This actually only *works* on the BC if the *Mode* parameter is **`toggleoff`**. For more information, see *BCMI* §15.)

Improvement:

- New icons for the MIDI input and output meters.

Bug fixes:

- The 'MIDI input messages: BCL' window no longer generates a spurious MIDI error message when it receives a *non*-BCL SysEx message in Record mode.
- The window logic of the MIDI System Exclusive messages 'Play' dialog box has been corrected.
- The Detach menu item is disabled after it has executed.

Version 2.0.0 (2009-03-01):

New features:

- Several additions to this manual:
  - A section on the setup of BCFs and BCRs, discussing firmware, BCF emulation modes, operating modes and USB drivers.
  - The MIDI setup section (§6) discusses some dangers related to virtual MIDI devices (MIDI 'pipes') such as MIDI Yoke.
  - An extensive FAQ section.
- The new Restart operation terminates BC Manager and automatically starts a new instance of it. This is particularly useful after a BC's operating mode has changed from/to a U-mode, in order to update BC Manager to the new set of available USB-based MIDI I/O devices. There is also a version of Restart that restarts BC Manager with its default setup (i.e. by deleting BCMan.ini\).
- Several procedures to optimize BC Manager's setup of 'MIDI pipes' (i.e. virtual MIDI ports). See §6 for details.
- The MIDI controller window now allows you to send Control Change messages for *any* controller, i.e. from 0-127. (Previously the allowed range was 0-95). (Note: the program change box has been moved to the MIDI keyboard window — see below.)
- A MIDI keyboard window, in which you can send mouse-generated note messages (even chords and arpeggios) to any MIDI output device.
- A window in which you can create, edit, load, save, record and play MIDI System Exclusive messages.
- You can detach a syx/txt/bc2 file from a B-Control.
- BC Manager identifies a BCF2000's 'personality'. i.e. standard 'B-Control' mode or any of the emulation modes. When BC Manager detects that a BCF is in emulation mode, it automatically

restricts the set of available communication operations to the only ones that work in emulation mode, namely connection status detection and firmware dumps.

- The MIDI pull-down menu of the B-Controls window contains a new submenu called Maintenance. From this submenu you can execute the existing 'Send firmware' operation, but also the new 'Test hardware' operation, which allows you to check whether your BCF/BCR's buttons, encoders and faders function correctly.
- You can directly move and copy preset, button, encoder and fader definitions from one position to any other by mouse-dragging. See §27 for instructions.
- The layout windows display the selected element's name and output summary on the status bar.
- The element windows and dialog boxes additionally display the NRPN parameter as two hexadecimal 7-bit values (MSB first).
- In the element windows, the 'Create simple' submenu allows you to quickly set up an element (or range of elements) for 'standard' Program Change and Control Change messages.
- The encoder dialog box contains a button to quickly set the encoder resolutions to the standard value of 96.
- If a custom output definition is invalid (i.e. if it will cause the BCF/BCR to transmit incorrect MIDI byte sequences), BC Manager displays a warning, indicating the exact location and nature of the error.
- BCL editor windows display the current line number and character index on the status bar.
- In the MIDI input messages window, you can save the bytes of *any* MIDI messages (so not just the *SysEx* messages) to a 'flat' binary file. This is mainly useful for specialistic further processing. (Previously you could only achieve this indirectly, namely by first saving the bytes to a text file, then converting that text file to a binary file.)
- The main window's Help pull-down menu contains a link to the Mountain Utilities web site.

Improvements:

- BC Manager and its installer now adhere to the folder and registry access restrictions of modern Windows operating systems (XP and later). As before, you must *install* BC Manager with Administrator privileges, but you can now *run* BC Manager from a limited user account, i.e. one without Administrator privileges. (Previously this could lead to file access violations.)

  In accordance with this new policy, BC Manager's configuration file (BCMan.ini) and the SysEx model definition files are no longer kept in BC Manager's installation folder (e.g. C:\Program Files\Mountain Utilities\BC Manager). Instead, BC Manager now stores these files in your home folder: under Windows XP typically in C:\Documents and Settings*Username*\Application Data\Mountain Utilities\BC Manager, under Windows Vista and later typically in C:\Users*Username*\AppData\Local\Mountain Utilities\BC Manager.

  On its first run, this new version of BC Manager automatically copies any existing BCMan.ini from the *installation* folder to your *home* folder, so you don't lose your previous settings. However, if you have created any additional SysEx model definition files, you should move these manually from C:\Program Files\Mountain Utilities\BC Manager\Models (on a 32-bit operating system) or C:\Program Files (x86)\Mountain Utilities\BC Manager\Models (on a 64-bit operating system) to the Models subfolder at the new location, otherwise BC Manager will no longer see them.
- A sequence of MIDI communication errors (e.g. those relating to a USB port that has been disconnected after BC Manager was started) no longer leads to dozens of error boxes you have to plow through; instead, these errors are gathered in a single window, which you only have to click away once.
- Operations involving SysEx messages from the BC to BC Manager (such as 'Refresh connection status', 'Receive presets' and 'Send firmware') can no longer be spoiled by interference from unrelated incoming SysEx messages on the same MIDI input port.
- The 'Send firmware' operation is more robust and user-friendly.
- The 'MIDI input messages: BCL' window can now record a maximum of 99,999 BCL messages

(previously only 9,999).

- A number of small improvements to the user interface. For instance:
  - Many menu and toolbar icons were modified (in particular to make them look more 3-D), and several new icons were added.
  - In the B-Controls window: after you execute 'New BCF2000' or 'New BCR2000', the table automatically *selects* the row of the new B-Control. (Previously the selection bar stayed where it was.)
  - The 'BCF2000' and 'BCR2000' windows have been renamed to 'layout' windows, and their preset LEDs change color from red to orange for preset 0 (i.e. the temporary preset).
  - When a knob is selected, its position mark changes from green to red.
  - Many message boxes now contain an icon indicating the severity of the message: information, warning or error.

Bug fixes:

- BC Manager now remains the active application throughout startup. (Previously it could become inactive during startup (particularly the very first time it was run). Consequently, BC Manager's intended active window or dialog box could end up hidden behind other running applications, forcing the user to manually navigate back to BC Manager.)
- The Exit operation from the main window's Files pull-down menu now asks you whether you want to save/send any unsaved/unsent B-Control data before termination of BC Manager; this also applies when you press the associated hotkey (Alt+X). (Previously the save/send questions only occurred when you pressed Alt+F4 or clicked on the ✕ icon on the main window's title bar.)
- The mouse options dialog box can no longer be resized.
- The maximum hint width (as set in the 'Hint options' dialog box) now works correctly.
- Upon a restart, BC Manager restores all associations between B-Controls and syx/txt/bc2 files correctly. (Previously, if e.g. the first B-Control did not have a file but the second B-Control did, a restart would wrongly assign the second B-Control's file to the *first* B-Control instead of to the second.)
- The byte display formats in the MIDI input messages window are now retained between sessions of BC Manager.
- After the deletion of a range of recorded MIDI messages, the MIDI input messages window now displays the correct remaining messages.
- It is no longer possible to paste more than 999 bytes from the MIDI message clipboard into a preset's LEARN output or an element's custom output.
- The Compare dialog box no longer reports that two identical items containing LEARN/custom output are different.
- BC Manager no longer produces messy windows and dialog boxes under non-standard DPI settings.
- The default table window widths are no longer too narrow under Windows Vista's default 'Windows Vista Basic' color scheme.

Version 1.5.1 (2008-08-19):

New features:

- If a button or encoder is dysfunctional on the BCF/BCR due to the preset settings (number of encoder groups etc.), its background in the layout window is pink.
- The 'Record MIDI messages' dialog box (opened by 'Record MIDI messages' in the preset/element dialog boxes) can now apply MIDI 'running status'. This may decrease the number of bytes recorded, so that the recorded messages are more likely to fit in the preset's LEARN output or the element's custom output. (The maximum is 125 bytes.)
- In the MIDI input messages window, an extra version of the 'Save MIDI file' operation is now available: this version saves all selected MIDI messages with their times set to zero.

Improvements:

- Previously, the saving and sending verification dialog boxes only showed up when a B-Control was being closed or the program was being terminated. In this version, these dialog boxes also show up during operations that can replace all data of a B-Control (as maintained by BC Manager), namely 'Open' and 'Receive all data'. (They still *don't* appear during 'Import'.)
- In the 'Convert txt/bc2 to syx' operation, the device ID dialog box is more user-friendly:
  - You can simply enter the device ID shown on the BC itself and in BC Manager's global setups window. (Previously, the value you entered was stored in the syx file 'as is', which meant that the dialog box caption was misleading: the caption stated that you had to enter a value from 1 to 16; if you entered 1, the actual value stored was indeed 1, but that wrongly *meant* device 2!)
  - The 'generic' device ID can be chosen by simply selecting the 'Generic' radio button: you no longer have to enter the actual value (127 = $7F).
- The handling of characters above #127 (e.g. 'international' characters with diacritics) in comments and element names has been improved: if you try to save such characters to a syx file, send them to a BC or load them in a BCL editor window, they are automatically converted to dots; this also applies to the 'Convert txt/bc2 to syx' operation. (Previously, a character above #127 would simply trigger an error message and the operation would be cut off, resulting in e.g. a 'truncated', invalid syx file.)

  Note that saving such characters to a txt or bc2 file is (and always has been) unproblematic, so if you use such characters, you should save to txt or bc2, not to syx. Also note that the BC itself doesn't *store* any comments or element names anyway, which is why by default BC Manager doesn't send comments or element names to the BC.
- A typo in the MIDI input messages window has been corrected.

Version 1.5.0 (2008-07-17):

New features:

- The B-Control detection routines identify MIDI feedback loops.
- In the element and layout windows you can paste an element whose type (button, encoder or fader) differs from the target element's type. (Naturally this does involve some data loss.)
- 'Set' operations are now available for *all* standard output parameters, including Mode, Increment, Frame Rate and Location.
- You can compare presets and elements: select Compare from the View pull-down menu of the B-Controls window.
- In the MIDI input messages window, you can set the display formats of the recorded bytes to hexadecimal and/or decimal. Separate settings are available for status and data bytes in both SysEx and non-SysEx messages.
- All saving operations in the MIDI input messages window now only save the *selected* MIDI messages. (Previously, *all* messages were always saved, irrespective of the selected range.)
- You can save the MIDI messages recorded in the MIDI input messages window to a standard MIDI file ('SMF', with extension '.mid').
- This manual contains a separate section on the MIDI input messages window (§22).
- Miscellaneous relevant web links are created in BC Manager's menu group in the Windows Start Menu.

Improvements:

- BC Manager now takes up much less RAM when a lot of MIDI input devices are enabled. Previously, every enabled MIDI input device reserved a total of 2 MB for its System Exclusive message buffers: 64 kB for each of its 32 buffers. By default BC Manager now reserves only 4 kB for each buffer, so a total of 128 kB per enabled MIDI input device. The reduced buffer size is still more than enough for all BC-related SysEx communication. In the unlikely case that you need BC Manager to handle third-party, longer SysEx messages, you can raise the buffer size in

the MIDI devices dialog box, up to the old value of 64 kB.

- Several improvements to the low-level MIDI input handling routines. As a result, the MIDI input buffer is much less likely to overflow, and the B-Control detection routines are more robust. BC Manager now also works with Microsoft's standard USB audio driver ('USB Audio Device'), which Windows XP and later install by default.
- Behringer's B-Control USB driver vs. 1.2.1.3 for Windows 2000/XP contains a bug that can freeze output from a BCF/BCR to a USB-based MIDI input device on the computer when this MIDI input device gets closed and reopened in rapid succession.

  In previous versions of BC Manager this bug in the USB driver sometimes manifested itself when you quit the MIDI devices dialog box by pressing the OK button: the *first* USB-based MIDI input device for a BCF or BCR could then stop sending any MIDI messages to the computer. (Only entering the MIDI devices dialog box for the *second* time and pressing OK again would then unfreeze this MIDI input device.)

  The MIDI devices dialog box in BC Manager 1.5.0 waits 100 msec between closing and reopening the MIDI input devices, which should be more than enough to circumvent the Behringer USB driver bug, so USB-based MIDI input devices should not freeze any more upon exit of the MIDI devices dialog box.

  Note: as far as I could establish, Microsoft's USB driver ('USB Audio Device') and Behringer's older USB drivers (vs. 1.1.1.0 and 1.1.1.1) do *not* suffer from vs. 1.2.1.3's bug. However, vs. 1.2.1.3 is still to be preferred: Microsoft's driver has no multi-client support and creates generic, very confusing MIDI I/O device names, and Behringer's vs. 1.1.1.0 and 1.1.1.1 are confused about the names of the MIDI output A and B device names.
- A MIDI I/O device (typically a USB device) that is temporarily unavailable (as indicated by its appearance between parentheses) now shows up in the MIDI options dialog box. As long as you keep choosing the temporarily unavailable device name, the device will be connected again once available upon a restart of BC Manager. However, if you have selected '[None]' (or any *present* device) when you press 'OK', the reference to the unavailable device disappears and won't be shown the next time you open the dialog box.
- The 'Receive firmware version' dialog box now reports the response time.
- The Comments areas in the dialog boxes now contain vertical scroll bars. (Horizontal scroll bars are unnecessary, since any text gets wrapped.)
- In the element dialog boxes: if *any* standard output is defined (i.e. Type isn't None), the Default's *value knob* is enabled and the Default's *checkbox* is disabled. This matches the BC's actual behavior concerning the Default parameter: any standard output causes a default value to be defined. You can only forgo a default value if only *custom* output is defined; this is relevant for snapshots: an initial snapshot only includes the element's output if it has a default value.
- A positive button increment value is now displayed with '+' in front of the number.
- In the MIDI input messages window, the settings for 'Buffer size', 'Buffer overflow protocol' and 'Scroll to new message' are now editable via a separate dialog box. Moreover, they are retained across BC Manager sessions. (The MIDI message type filters already were.)

Bug fixes:

- When opening a BCL data file, BC Manager no longer skips empty comment lines.
- A bug has been fixed that sometimes caused the MMC frame rate to be set to a wrong value upon entrance of the button dialog box.
- In encoder and fader dialog boxes, the GS/XG parameter list no longer shows duplicates of all items.

Version 1.4.0 (2008-06-23):

New features:

- You can now choose to skip the B-Control detection routine performed when BC Manager is being run for the very first time. (Skipping this routine can avoid certain problems related to

MIDI input/output devices, such as infinite feedback loops.)

- The 'Set' operation can now also be applied to most preset settings (Encoder groups, Request, etc.), to LEARN and custom output items, and to the Value 1 and Value 2 parameters of standard CC, NRPN, AT and GS/XG output.
- You can open a layout window directly from the B-Controls window, via either the View pull-down menu or the toolbar.
- You can 'import' a syx, txt or bc2 file containing BCL data. The only difference with *opening* is that *importing* does *not* copy the imported file's name to the file name associated with the B-Control. [Note: as of version 1.5.1 there is another difference: the Import operation never shows any saving or sending verification dialog boxes.]
- You can 'export' a range of presets to a BCL (syx, txt or bc2) file.
- When importing a Propellerhead Reason remote map file, you can choose to only load the Reason control names: BC Manager then skips its generation of fully functional elements. See §21 for why this is useful.
- You can now also open/save the content of the BCL editor window from/to a syx or bc2 file.
- The BCL editor window now contains scroll bars.

Improvements/bug fixes:

- The preset number or range displayed in the caption of a 'Set' dialog box run from the presets window is now correct.
- BC Manager strictly enforces BCL's allowed range of characters in preset names: characters below #32 and above #127 are converted to dots, just as the BC itself does.
- Changing the width of layout windows works more accurately than in version 1.3.0. (This is particularly noticeable under Windows Vista's default color scheme.)
- BC Manager now maintains *separate* Value 1 and Value 2 parameters for CC (in both 7 and 14 bit versions), NRPN (7 and 14 bit), AT (7 bit) and GS/XG (7 bit). A few obscure problems concerning the ranges of Value 1 and Value 2 have thus been solved.
- The 'Set MMC Device' operation is no longer available in encoder and fader list windows. (Encoders and faders don't have MMC options, so the operation can't apply to them.)

Version 1.3.0 (2008-06-10):

New features:

- The pdf file of this manual includes bookmarks for the document sections, which makes navigation much easier.
- Several new BCF2000/BCR2000 detection facilities are available.

  For instance, when you start BC Manager for the very first time (actually: if BCMan.ini doesn't exist), it automatically finds all switched-on BCs connected to the computer by USB or standard MIDI I/O ports.

  BC Manager is also a bit more strict when it comes to allowing MIDI communication with a BC: a successful 'Refresh connection status' check must have occurred before any MIDI data exchange is allowed. However, by default BC Manager performs this check automatically, so you don't have to do this manually.
- You can open, save and convert 'bc2' files.
- You can import Propellerhead Reason's remote map files for the BCF2000 and BCR2000. (This is mainly useful for visualization and printing.)
- The 'Set' operation now works for several more element parameters, and the step size per row (element) can be set to *any* value.
- The pop-up hints for the paste toolbuttons specify the number of items currently on the pertinent clipboard.
- Many changes to the layout windows:
  - You can widen the window. (This is extremely useful when your element labels are long.)
  - Printing the window image now always uses Landscape orientation, irrespective of the

orientation selected in the main window's printer dialog box. (This practice was already used for printing *element strips*.)
    - You can cut, copy, paste and initialize the selected element.
    - You can move to a particular element group by pressing the associated character underlined in the group's caption.
    - You can choose what is displayed in the element boxes: their names or their output summaries.
    - The pull-down menu allows you to switch to the corresponding preset and element windows.

Bug fixes:

- The 'open' and 'save' dialog boxes for B-Control data now use the extension of the current file as the default file type ('.syx', '.txt' or '.bc2'). (Previously, these dialog boxes always started with '.syx', even when you were working with a '.txt' file.)
- In the B-Controls window it is no longer possible to select more than one B-Control. (Previously you *could*, but subsequent operations would only apply to one B-Control anyway.)
- In the B-Controls window and the global setups window: double-clicking on the table when no B-Control has been defined no longer produces an error message.
- Copying an element or preset to its clipboard now correctly enables the Paste operation in the pertinent windows of *all* B-Controls (not just the *current* one).
- In black & white prints of layout window images, all grid borders are now equally thick.
- The tab order of the grids in a BCR2000 layout window is now correct.

Version 1.2.2 (2008-05-13):

- In the MIDI message recording dialog box you can request a snapshot from the connected BCF or BCR.
- A tiny bug concerning the Record button in the preset and element dialog boxes has been fixed.

Version 1.2.1 (2008-05-11):

- In the preset and element dialog boxes, you can now record MIDI messages directly as LEARN and custom output.
- Quite a few changes to the user interface. Among others:
    - Some buttons have been added to the main window's toolbar.
    - Custom output's 'Min' and 'Max' settings have been renamed to 'Value 1' and 'Value 2' in the element dialog boxes, and are now displayed in the 'Value 1' and 'Value 2' columns in the element windows.
    - A button's custom output increment value is now only shown in the button table if the mode is Increment.
    - In the BCL editor window, all references to 'temporary preset' have been renamed to 'preset 0', and you can execute 'Receive preset 0' from the toolbar.
    - Several changes to the MIDI controller window:
        - When BC Manager is restarted, the window is restored to its last position.
        - The window is more compact.
        - The controller names are available via hint boxes linked to the controller numbers.

Version 1.2.0 (2008-04-30):

- You can now express your appreciation of BC Manager and support its further development by making a donation via the 'Donate' dialog box (accessible via the main window's Help pull-down menu).
- In the input message window you can copy recorded MIDI messages to the new 'MIDI message clipboard', and you can then paste these messages as LEARN and custom output in the preset and element dialog boxes.

- For a button sending only *custom* output messages, you can now set the Mode parameter to 'Increment' in the button dialog box. (This uses the recently discovered **`incval`** parameter.)
- Negative values for a button's Increment parameter are now displayed correctly in the Inc column of the button window.
- 'Receive firmware version' and 'Send firmware' now also work correctly when the BC is in bootloader mode.
- In MIDI communication dialog boxes, the reported number of items transferred is now correct at all times. (In previous versions this number sometimes lagged one item behind.)

Version 1.1.1 (2008-04-13):

- Sending an individual element of preset 0 to the connected BC now tells the BC to initialize the element if all its parameters are at their default values. (In version 1.1.0 such an 'empty' element wasn't sent at all, so that the BC didn't clear it.) The same applies if you load such an element into a BCL editor window.

Version 1.1.0 (2008-04-09):

Major changes:

- This manual is now included in the BC Manager package itself and can be opened from the Windows Start Menu or from BC Manager's Help pull-down menu.
- Closing the B-Controls window no longer leads to questions whether you want to save and/or send a B-Control's data if it has been modified. These questions now only occur when you try to quit BC Manager as a whole (i.e. the *main* window).
- A memory preset can be copied directly to the temporary preset ('preset 0'), and the connected BCF or BCR can be ordered to follow this change immediately.
- Individual buttons, encoders and faders in the temporary preset can be sent to the BCF and BCR, and loaded in a BCL editor window.
- A bug has been fixed that prevented element names from being copied to other elements, and from being cleared in the 'Initialize' operation. This bug was also responsible for the disappearance of element names from memory presets when a previously saved syx or txt file was being opened.

Minor changes:

- BC Manager no longer terminates on startup if the syx or txt file associated with a B-Control cannot be found.
- The presets window can now be maximized.
- In the preset dialog box:
  - The caption now displays the correct preset number.
  - Hints are available for the toolbuttons on the 'LEARN output' tab.
- In the layout windows:
  - The buttons for decreasing/increasing the preset number are now aligned horizontally.
  - There are menu items and keyboard shortcuts for increasing and decreasing the preset number.
  - The preset name is displayed on the status bar.
  - The preset's general settings can be edited.
  - The window image can be saved or printed with an ink-saving white background.
  - Under Windows 95/98/Me, spurious scrollbars no longer mess up the element labels.
- The order of a few table columns has been changed to increase consistency.
- The default width of the mode column in the button window has been increased to accommodate the mode parameter names.
- The new 'Speed levels' setting simplifies the manipulation of encoder resolutions.
- A LEARN or custom output statement can be edited as a single text line.
- The initial status of the 'Stop' menu item in the 'MIDI input messages: BCL' window has been

corrected to ‘disabled’.

- The BCL interpreter accepts the **`incval`** parameter for the button ‘**`.mode`**’ statement, just as the BCF and BCR do.

Version 1.0.0 (2008-02-27):

First published version.

3. Computer requirements

To run BC Manager, your computer must comply with the following requirements:

- Processor: Any Intel 80486- or Pentium-compatible CPU. Processor speed is relatively unimportant. The number of cores is irrelevant, since BC Manager only uses one core.

- Operating system: starting with version 2.2.0, BC Manager only runs under a Windows operating system that supports Unicode, i.e. Windows 2000 or later.

- An SVGA-compatible graphical card and monitor:
  - The screen size should be at least 800 × 600 pixels. (However, some of the bigger windows and dialog boxes may be cut off if the screen is too narrow, so 1024 × 768 is the *practical* minimum.)
  - For best results, the color depth should be at least 16 bits. (At a depth of only 256 colors, some colors aren't rendered as intended.)

- A mouse.

- Free hard disk space: about 4 MB.

- When running, BC Manager normally occupies roughly 6-10 MB of RAM.

- Unlike Behringer's BC-EDIT, BC Manager does *not* require Java Runtime Environment (JRE). (By the way: it doesn't require Microsoft .NET Framework either.)

- BC Manager runs perfectly without a MIDI link to an actual BCF2000 or BCR2000, but if you want to establish a real-time connection between BC Manager and a BC, the BC must be linked to the computer via MIDI input and output ports; these can be either 'standard' MIDI I/O ports or the BC's USB-based MIDI I/O ports. See the next section for some tips on setting up your BCF/BCR.

4. BCF2000/BCR2000 setup

It is highly advisable to set up your BC correctly on your computer before installing BC Manager. You should consult the relevant documentation for the full details on these matters, but here are some guidelines:

1. **The BCF2000's emulation modes:**
   For a BCF2000 you should first make sure that it is in standard 'B-Control' mode (at least at this stage), not in any emulation mode. The BCF is in B-Control mode if its display shows 'P–' followed by a preset number after power-on.

   To change to B-Control mode from an emulation mode, you have to switch the unit off, then keep the top left button on the top panel pressed while switching it back on by pressing the POWER ON button at the back. (See the Behringer documentation for further details on this procedure.)

   B-Control mode has several advantages (at least at this stage):
   - In B-Control mode, all the BCF's global settings can be changed at any moment *after* power-on, but the emulation modes provide fewer global settings, and these setting can only be changed 'during' the power-on process. Thus, in B-Control mode it's a lot easier to set up the BCF concerning I/O ports, USB drivers and application software like BC Manager.
   - The BCF's 32 memory presets are completely inaccessible when the BCF is in any emulation mode. Consequently, BC Manager (which is mainly concerned with these presets) is of limited use when the BCF is in an emulation mode: many of BC Manager's features are then dysfunctional.

   Note that the *BCR2000* doesn't *have* any emulation modes, so it is always in B-Control mode by definition.

2. **The BCF2000/BCR2000's firmware:**
   This is the operating system of the BCF and BCR themselves. The data of this firmware resides on a flashable memory chip in the BCF/BCR.

   You should ensure that your BC's firmware is version 1.10, as indicated by the number briefly shown in the display immediately after power-on. If the version number is lower than 1.10, you *may* be able to proceed with the installation as described below, but it is advisable to upgrade as soon as possible, preferably via BC Manager's 'Send firmware' operation (q.v.). (Beware: at least in my experience, Behringer's firmware utility BCUPDATE.exe doesn't work.)

3. **MIDI connections:**
   You can connect a BCF or BCR to your computer either via USB or standard MIDI I/O ports (typically on a soundcard).

   For proper communication, BC Manager requires a *bidirectional* connection. If you use the BC's USB connection, you can set the BC's Operating Mode to U-1, U-2, U-3 or U-4. If you use the standard MIDI ports, S-4 (MIDI OUT A) is recommended, though S-3 (MIDI OUT B/THRU) is possible too; however, S-1 and S-2 should *not* be used, since these modes lead to inherent feedback, messing up communication between BC Manager and the BC.

   BC Manager can even communicate with two chained BCs, where one BC is connected to the computer directly via USB (in mode U-4), and the other (in mode S-3 or S-4) is connected to the first BC's standard MIDI sockets, as described in Behringer's User's Manual.

   The advantage of the U-modes is that more additional connections to other MIDI equipment are possible than in the S-modes, because you don't have to sacrifice any MIDI hardware ports

to the bidirectional connection between BC Manager and the BC.

However, the USB connection also has some pitfalls:

- Most extremely, several people have reported cases where the USB controller of their BCF or BCR was defective. Very nasty.
- You may experience USB driver problems, both concerning installation and later operation. See point 4 below.
- To get a working USB connection between the BC and any computer program (such as BC Manager), you must always start up the BC *before* the computer program. And switching the BC's Operating Mode to *any* other value (U-1/2/3/4 or S-1/2/3/4) while a computer program is running *completely* disconnects any existing USB connections between the BC and the computer program, and any newly set USB connections only start working after you have terminated and restarted the computer program.

4. **USB driver:**

If you wish to use a USB connection between your BC and your computer, you must install an appropriate USB driver. You have the following options:

- **Microsoft's 'USB Audio Device'**:

  Advantage:

  Available under any 'modern' Windows operating system. Windows XP and later install this driver by default.

  Drawbacks:

  1. The resulting device names of the USB-based MIDI ports are truly horrible: the driver only assigns incremental *numbers* to any simultaneously available ports. The situation gets particularly confusing when you connect more than one B-Control via USB.
  2. The driver has a *single-client* nature: it can only connect to one computer program at the same time. On the other hand, all Behringer's USB drivers (see below) allow you to connect a virtually unlimited number of computer programs simultaneously; so for instance you can work with BC Manager and a DAW program (Ableton Live, Cubase, Sonar etc.) at the same time.

- **Behringer's BCF2000/BCR2000 USB driver version 1.1.1.0**:

  Advantage:

  The resulting device names of the USB-based MIDI ports are much more informative than under Microsoft's 'USB Audio Device'. (However, see drawback 2 below.)

  Drawbacks:

  1. To install this driver under the 32-bit version of Windows Vista, 7 or 8(?), you need to set the installer's compatibility mode to 'Windows XP (Service Pack 2)' on the Compatibility tab of the file's Properties dialog box.
     Even worse, this driver can't be installed *at all* under the 64-bit versions of Windows XP and later.
  2. The driver provides very confusing MIDI port names in operating mode U-4.

  Note that Behringer no longer offer this driver on their web site.

- **Behringer's BCF2000/BCR2000 USB driver version 1.1.1.1**:

  This has the same advantage and drawbacks as version 1.1.1.0. However, it fixes one or two small bugs in version 1.1.1.0. Behringer no longer offer this driver on their web site either.

- **Behringer's BCF2000/BCR2000 USB driver version 1.2.1.3** (dated October 19, 2005):

  Advantage:

  This driver consistently uses transparent device names for its USB-based MIDI input

and output ports (in particular in operating mode U-4).

Drawback:

The same installation restrictions as version 1.1.1.0.

- **Behringer's generic USB MIDI driver version 1.0.10** (dated December 15, 2009):

  Advantage:

  No more installation problems. As stated on the Behringer web site, this driver comes in two versions:

  1. BEHRINGER_MIDI_WIN32_1.0.10.zip for 32-bit operating systems (Windows XP, Vista, 7 and 8(?)).
  2. BEHRINGER_MIDI_X64_1.0.10.zip for Windows 7 and 8(?)'s 64-bit version.

  (However, does this mean there is no version for *64-bit XP and Vista*?)

  Drawback:

  The resulting device names of the USB-based MIDI ports are not as transparent as those of Behringer's BCF2000/BCR2000 driver 1.2.1.3 (or even 1.1.1.0 and 1.1.1.1). E.g. for a BCR2000 in operating mode U-4 you get 'BCR2000 port 1' and 'BCR2000 port 2' instead of version 1.2.1.3's 'BCR2000[1]' and 'BCR2000[1]-B'.

BC Manager works with all the drivers mentioned above. However, I recommend Behringer's BCF2000/BCR2000 USB driver 1.2.1.3 for any 32-bit version of Windows; to install this driver under the 32-bit versions of Windows Vista or 7, you must set bcr2000-driver-setup.exe's compatibility mode to 'Windows XP (Service Pack 2)'. For the 64-bit versions of XP, Vista, 7 and 8(?), try Behringer's generic USB MIDI driver version 1.0.10.

5. **Using multiple BCFs/BCRs via USB:**

It is very common to run into problems when you try to use two or more BCFs or BCRs via USB simultaneously.

Perhaps it helps to distinguish two aspects of a working USB MIDI driver:

1. The installed USB driver itself, i.e. its dll file(s), etc.
2. The USB MIDI devices defined (for the driver) in the Windows registry.

In my experience, (re-)installing a USB driver often occurs only for a particular hardware device (e.g. BCF2000/BCR2000), thus affecting *only* that device's settings (including its port names).

So for instance, if you first install driver P for hardware devices X and Y, *then* driver Q for device X, device Y may still have its 'old' settings/names. In other words, you may have to install driver Q *twice* in this case; not for the driver itself (once should be enough for that), but to update the settings/names of both individual hardware devices X and Y.

It may help to look at the device settings in Windows' Device Manager.
E.g. under Windows XP, select Control Panel ⇨ System ⇨ Hardware ⇨ Device Manager.
In Device Manager, select View ⇨ 'Devices by connection', then look for the USB devices under 'ACPI Uniprocessor PC' ⇨ 'Microsoft ACPI-Compliant System' ⇨ 'PCI bus' (at least that's where they are in my case): in this way you can see exactly which USB ports use which drivers.

5. Installation of BC Manager

To install BC Manager on your computer, proceed as follows:

1. Download BCManSetup*n.n.n*.zip (where *n.n.n* stands for the actual version number) to your computer from the BC Manager page of the Mountain Utilities web site.

2. Unzip BCManSetup*n.n.n*.zip to any folder on your computer. This creates a single file: BCManSetup*n.n.n*.exe.

3. Run BCManSetup*n.n.n*.exe, and follow its instructions. BC Manager will be installed on your hard disk.
   Note: the installation includes an uninstaller which can be run from the Windows Start Menu via Programs ➜ Mountain Utilities ➜ BC Manager, or via Settings ➜ Control Panel ➜ 'Add or Remove Programs' (XP) or 'Programs and Features' (Vista and later).

Note that when you install a new version of BC Manager, you do *not* have to uninstall any previously installed version first: the old version will be replaced with the new version automatically.

After installation, you can start BC Manager itself (i.e. BCMan.exe). It is highly advisable to switch on your BCFs and BCRs *before* starting BC Manager. In fact, this is *crucial* for *USB* connections (though not for standard MIDI connections), because BC Manager can only work with USB ports that are active the very moment when BC Manager is being started. No matter what connection you're using, the advantage of having your BCs switched on at the moment when you start BC Manager, is that BC Manager should then be able to immediately autodetect all your BCs.

If you have never run BC Manager (in any version) from the installed operating system before, the program notifies you that it can't find your configuration. This is normal: when you quit BC Manager, it stores its setup (including the MIDI setup, B-Control definitions and window layout) in a file called BCMan.ini, which simply isn't present the first time you start BC Manager. BC Manager also notifies you if BCMan.ini does exist but belongs to a previous version; all existing settings are retained.

A useful tip: if at any stage you have messed up your settings, you can execute 'Restart with default setup' from the main window's File pull-down menu: it is then as if BC Manager is running for the very first time again.[1]

[1] Alternatively, you can delete BCMan.ini manually while BC Manager isn't running. BCMan.ini is stored under your home folder, under Windows XP typically in C:\Documents and Settings*Username*\Application Data\Mountain Utilities\BC Manager, under Vista and later typically in C:\Users*Username*\AppData\Local\Mountain Utilities\BC Manager. However, by default XP hides the Application Data folder and Vista and later hide the AppData folder; to be able to navigate to such a folder in Windows Explorer, you have to turn on 'Show hidden files and folders' via Tools ➜ 'Folder options'; alternatively you can simply move to the folder from a command prompt window via the 'cd' command.

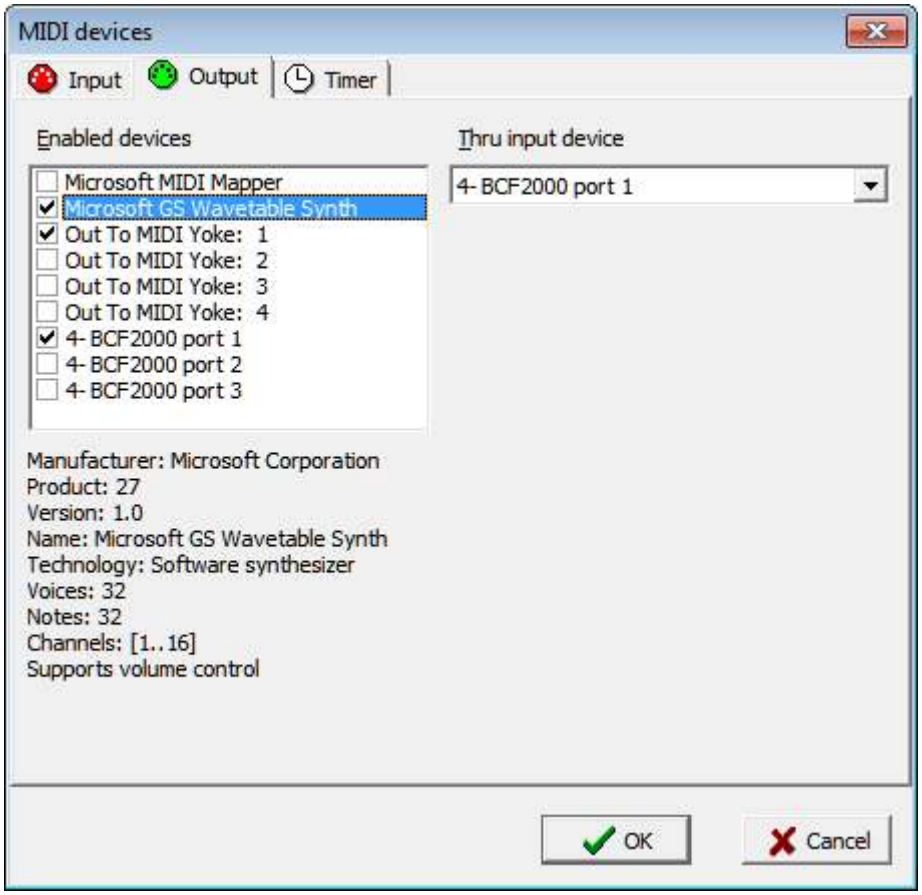
6. MIDI setup

When BC Manager is started for the first time, it automatically sets up data structures for all BCs that are connected to the computer via USB or standard MIDI, provided that these BCs have been switched on *before* BC Manager is started.

If BC Manager doesn't detect any such BC, it sets up data structures for one BCF2000 and one BCR2000. You can still edit these structures, but you can't communicate with any actual BCs until you establish a valid connection.

You can also tell BC Manager manually where to find a particular BC. This involves the following steps:

1. You must ensure that the pertinent MIDI I/O ports are *enabled* in BC Manager:
   From the main window's Options pull-down menu, 'MIDI devices' opens a dialog box in which you can select the MIDI I/O devices to which BC Manager connects:

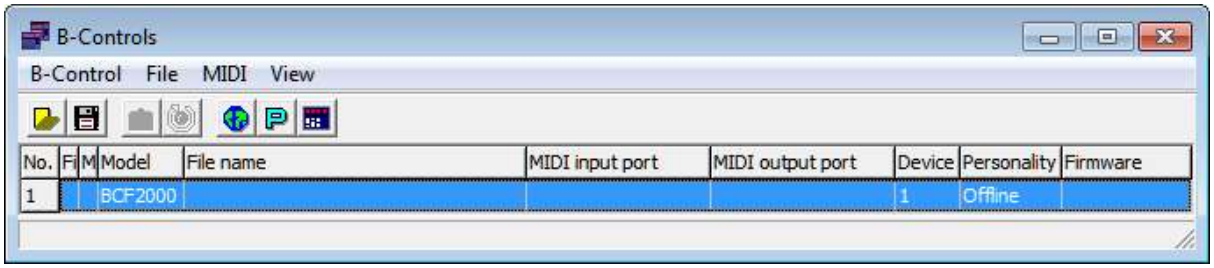


   Selecting the proper I/O devices is relevant in several ways:
   - If you want BC Manager to communicate with a BC, the MIDI input and output ports connected to this BC need to be 'enabled' here.
   - More in general, *all* BC Manager's MIDI utilities (including those not directly related to BCs) can only exchange data via MIDI devices that are 'enabled'.
   - If you want to run other programs using MIDI devices simultaneously, it may be a good idea to keep as many MIDI devices disabled as you can in BC Manager, in order to avoid

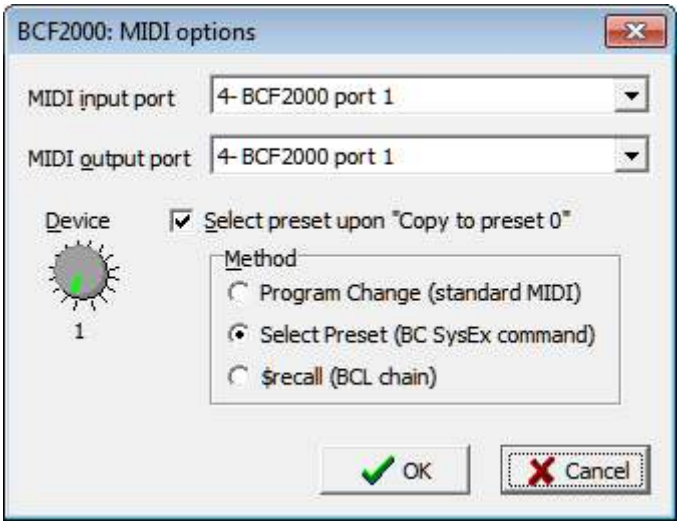
access conflicts.

Tip: In the MIDI devices dialog box you can also set up a ‘soft MIDI thru’ link, by connecting an enabled MIDI input to an enabled MIDI output device. Any MIDI message received by the MIDI input device is then immediately passed on to the MIDI output device. In fact, you can set up as many links as there are devices; the only restriction is that each device can have only one connection.

2. You must set the BC’s MIDI I/O ports and its Device ID to the correct values:
   a. In the list of B-Controls in the B-Controls window, select (highlight) the appropriate B-Control. (If necessary, you can add one via B-Control ➜ New BCF2000/BCR2000.) Obviously, if only one B-Control is defined, it’s automatically selected:



   b. Open the ‘MIDI options’ dialog box via the MIDI pull-down menu, and select the B-Control’s MIDI I/O ports and Device ID (i.e. the number you can see on the BC itself when you press the ‘VAL 2’ button in ‘EG’ mode, entered via EDIT + STORE):

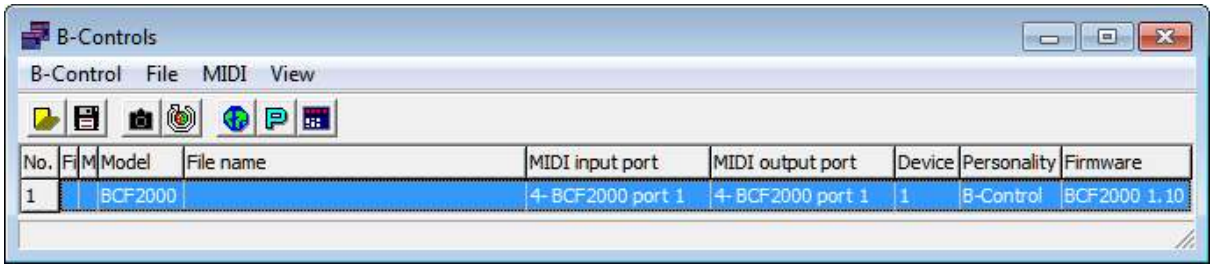


Provided ‘Startup detection’ in the ‘B-Control detection options’ dialog box (accessed via the main window’s Options pull-down menu) is *not* ‘None’, BC Manager automatically performs a ‘refresh connection status’ operation when you click OK. If ‘Startup detection’ in the ‘B-Control detection options’ *is* ‘None’, you must execute ‘Refresh connection status’ manually from the B-Controls window’s MIDI pull-down menu.

In either case, the ‘refresh connection status’ operation first shows the ‘receive firmware version’ dialog box, stating the firmware version returned by the BC; normally this is ‘BCF2000 1.10’ or ‘BCR2000 1.10’. Secondly, for a BCF the mode (‘personality’) is shown in the ‘receive personality’ dialog box. (For BC Manager, standard ‘B-Control’

mode is most appropriate; in any of the emulation modes BC Manager can't do very much.)

After this, the B-Controls window should look like this:



For a BCF, the detected mode is shown in the 'Personality' column. For a BCR (which has no emulation modes) this column states 'B-Control' anyway, provided that the firmware version has been retrieved.

MIDI pipes

Of particular interest is the enabling/disabling of 'MIDI pipes'. 'MIDI pipe' is my term for what is commonly known as a 'virtual MIDI device'. This is a piece of Windows driver software that manifests itself as a virtual MIDI output port plus a virtual MIDI input port: the driver passes any MIDI data sent to the output port to the input port. Hence, if a computer program sends data to the virtual MIDI output port, this data can be picked up at the corresponding input port by any other computer program. Thus, a MIDI pipe allows inter-program MIDI communication. If a MIDI pipe driver is 'multi-client', we can connect more than one program (up to a particular maximum) to the same virtual output or input port.

To my knowledge, the following free MIDI pipe drivers are available (please contact me if you know any others):

- **Hubi's Loopback Device**: 4 multi-client pipes, but for Windows 9*x* only.
- **Sony/Sonic Foundry Virtual MIDI Router**: 4 single-client pipes.
- **Hurchalla Maple**: 12 single-client pipes.
- **LoopBe1**: only 1 multi-client pipe, so not very useful. (No, you can't install more than one copy!)
- **LoopBe30**: 30 multi-client pipes, but the trial version only works for a brief period, and the full version is not free.
- **MIDI Yoke** (http://www.midiox.com/): the NT (/2000/XP/Vista/7/8(?)) version allows up to 16 multi-client pipes, so understandably this has been the most popular MIDI pipe driver for 32-bit Windows versions.

  Problems:

  1. MIDI Yoke's NT version, even though it is 32-bit, *can* be installed under 64-bit operating systems, but (reportedly) the pipes are only accessible to 32-bit DAWs, not to 64-bit DAWs.
  2. To work with MIDI Yoke correctly under Windows versions with UAC (User Access Control) you must apply a manual tweak:

The MIDI Yoke installer tries to create MIDI Yoke's configuration file (MYOKENT.INI) in C:\Windows, but the operating system doesn't allow this and actually creates it in C:\Users*Username*\AppData\Local\VirtualStore\Windows. On the other hand, the MIDI Yoke configuration applet under Control Panel *does* have write access to C:\Windows and will create a second copy of MYOKENT.INI there when you change the settings. However, the latter file will never be never seen by the *driver* (because the operating system keeps redirecting it to the copy in ...\VirtualStore\Windows); in other words, the driver 'won't listen to you.'

To fix this, you must manually remove MYOKENT.INI from C:\Users*Username*\AppData\Local\VirtualStore\Windows or move it to C:\Windows, using administrator rights.

3. As discussed below, the NT version of MIDI Yoke may slow down the termination of BC Manager.

- **loopMIDI** (http://www.tobias-erichsen.de/): this allows you to create and destroy any number of MIDI pipes on the fly. This is probably the best choice, in particular for 64-bit Windows versions. The only drawback is that loopMIDI devices don't persist after a reboot.

If configured improperly, MIDI pipes can easily cause problematic MIDI signal paths. There are several dangers:

Duplication:
If there is first a *split* in the signal path, and then a *merge*, two or more copies of the same MIDI message arrive at the end of the signal path (i.e. the target MIDI device). This is time-consuming in all cases, but — even worse — it's *disastrous* in the case of communication to or from a BC via BCL, as performed by BC Manager:

A BCL chain (a sequence of related MIDI SysEx messages, e.g. a preset dump) must contain strictly incremental MIDI message indexes. Therefore duplicate reception of the same MIDI message within the chain causes the BC (and indeed BC Manager) to *reject* any further MIDI messages belonging to the same BCL chain. Upon reception of a MIDI message containing an invalid index, the BC returns a BCL reply message mentioning 'Error 22' to BC Manager, and the BC's display shows '`Er22`', so if this happens, you have to correct the MIDI signal path.

Feedback:
In general, feedback involves the return of a sent MIDI message to the same MIDI hardware device or computer program that sent the message. Obviously this needlessly slows down operation, although it isn't necessarily disastrous. However, there may also be more sinister effects. For instance, feedback may interfere with BC Manager's communication with the BC, in particular the 'Detect B-Controls' and 'Send firmware' operations.

At the very least you should normally avoid enabling both the output port and the input port of the same MIDI pipe in the same program (e.g. BC Manager), because by definition anything you send to a MIDI pipe's *output* port (the pipe's starting point) is returned at the corresponding MIDI pipe's *input* port (the pipe's end point). So for instance, if you enable both 'Out To MIDI Yoke: 1' and 'In From MIDI Yoke: 1' in a program, then any MIDI data the program sends to 'Out To MIDI Yoke: 1' is returned to the program at 'In From MIDI Yoke: 1'. This type of feedback is usually undesired, except perhaps for monitoring purposes.

The most severe type of feedback occurs when the sender/recipient actually *re-sends* the returned MIDI message: this leads to an infinite loop, which may well grind the sender/recipient (and indeed the whole computer) to a virtual standstill.

In BC Manager this horror can happen if you enable the MIDI Thru feature in the MIDI devices dialog box for an input-output pair already exhibiting feedback. For instance, if you activate MIDI

Thru from ‘In From MIDI Yoke: 1’ to ‘Out To MIDI Yoke: 1’ in BC Manager’s MIDI devices dialog box, then any MIDI data sent to ‘Out To MIDI Yoke: 1’ not only comes back to BC Manager at ‘In From MIDI Yoke: 1’ (via MIDI Yoke’s pipe 1), but is then automatically *re-sent* from ‘In From MIDI Yoke: 1’ to ‘Out To MIDI Yoke: 1’ via the MIDI Thru feature, in principle ad infinitum, although MIDI Yoke does perform some checks that spot and kill the infinite loop — but still...

Close delay:
This problem only occurs with MIDI Yoke NT 1.75 (but *not* with MIDI Yoke *for Windows 95/98/Me*): closing any MIDI Yoke NT *input* port causes a delay of 1 second. (Certain earlier versions even 3 seconds.)

Concerning BC Manager this is mainly relevant during program exit. In principle BC Manager terminates almost instantaneously upon exit, but if all the input ports of MIDI Yoke NT 1.75 are enabled, termination of BC Manager takes some 16 seconds longer than normal! Therefore you should disable as many MIDI Yoke NT input ports as possible in BC Manager’s MIDI devices dialog box, i.e. any MIDI Yoke NT input ports that BC Manager itself doesn’t use. (Note that you can still use any MIDI ports disabled in BC Manager in *other* MIDI programs!)

To help you avoid some of the serious problems discussed above, BC Manager takes the following steps:

- On *first* startup, if BC Manager detects any of the MIDI pipes listed above, it asks you if you want to enable the I/O devices of these pipes. It’s best to answer *No* (to avoid feedback loops, and to avoid MIDI Yoke NT’s close delays during BC Manager’s exit procedure), unless some other program (e.g. MIDI-OX) is routing a BCF2000 or BCR2000 through a MIDI pipe.

- On *every* startup, BC Manager optionally warns you if any MIDI Yoke NT input ports are enabled and thereby cause extra delays during termination of BC Manager. You can enable/disable this warning on the Input tab of the MIDI devices dialog box.

7. Tutorial: working with presets

Probably the best way of manipulating presets via BC Manager is as follows:

1. First make sure that all memory presets are synchronized between the BC and BC Manager.
   Unless one or more memory presets on the BC contain data you don't have anywhere else, the best direction of synchronization is *from* BC Manager *to* the BC. This is because the BC is actually the worst place to store preset data: (1) due to bugs in its firmware, the BC makes certain mistakes when sending preset definitions; (2) presets on the BC can't store some of BC Manager's peripheral preset data (i.e. info fields).

2. Use BC Manager's 'Copy to preset 0' operation in the preset list window (by pressing the spacebar or executing the menu item on the Edit pull-down menu). This achieves two things:
   a. The selected memory preset is copied to the temporary preset in BC Manager.
   b. BC Manager sends a 'select memory preset *N*' command to the BC, provided you have enabled 'Select preset upon "Copy to preset 0"' in the 'MIDI options' dialog box (accessed via the MIDI pull-down menu of the B-Controls window). Note that this operation does not send any preset *data* to the BC, so this only works correctly if the memory preset on the BC has already been synchronized to the corresponding memory preset in BC Manager.

3. In BC Manager, edit the temporary preset ('preset 0') to your liking. Using preset 0 for editing has the additional advantage that you can send individual button/encoder/fader changes to the BC: you can see immediately how the edited element works on the BC itself, and determine which MIDI output it generates via BC Manager's 'MIDI input messages' window.

4. To save the changes you have made within the temporary preset to a memory preset, proceed as follows:
   a. Copy the edited temporary preset to a memory preset in BC Manager.
   b. Send that memory preset to the BC. This achieves the following:
      1. The memory preset data from BC Manager overwrites the temporary preset on the BC.
      2. The memory preset data from BC Manager is copied from the BC's temporary preset to the memory preset on the BC.
      3. The BC's temporary preset is restored to BC Manager's temporary preset data. (In this case this should be superfluous, but of course this is not automatically the case.)

Note that the above editing procedure avoids pressing the BC's PRESET buttons. This is a good thing, because that would only cause loss of synchronization between the BC and BC Manager, hence potential confusion.

To summarize the general philosophy behind the above editing procedure: it's best to do as much editing as possible in BC Manager itself. You merely use the BC's *temporary* preset to verify your changes, and the BC's *memory* presets only as 'dumb receivers' (presumably for later, 'stand-alone' use with your controlled MIDI hardware or software device).

8. The main window

BC Manager's main window only consists of a pull-down menu and a toolbar:

The toolbar merely contains several buttons duplicating menu items.

The menu provides the following operations:

File ➜ Convert ➜ Syx to txt/bc2:
This allows you to convert a syx file containing BCL messages to a txt or bc2 file. (You could then edit this txt or bc2 file in an external editor.)

Notes:

- It doesn't matter whether you select txt or bc2 as the output file extension: the resulting file is always in ASCII text file format.
- This is a completely 'dumb' operation: BC Manager does not check the syntactic validity of the BCL messages in any way: it just strips away the SysEx messages in which the BCL messages are embedded.
- This operation does not affect the B-Control data structures being maintained in BC Manager in any way.

File ➜ Convert ➜ Txt/bc2 to syx:
The reverse of the 'Syx to txt/bc2' operation. The same considerations apply.

Notes:

- The input file can be in ASCII, ANSI or UTF-8 format, but any non-ASCII characters are converted to dots. (Actually, bc2 files will always be ASCII/ANSI, not UTF-8, since Royce Craven's BCn2000 Script Editor (the 'native' application for bc2 files) only works with ASCII/ANSI characters, not with Unicode characters.)
- The input file's number of lines cannot exceed 16384; you should open a txt/bc2 file exceeding this limit in the B-Controls window, then save it as a syx file.

File ➜ Restart:
Terminates BC Manager and starts a new instance of it.

This operation asks you whether you want to save/send any changed data (just as the normal exit procedure (see below) does). Normally you should do so: the new instance of BC Manager is completely separate from the old one, so there is no transfer of B-Control data maintained in RAM; however, if you save any changed B-Control data to file, the new instance of BC Manager then automatically reloads this data.

Restart is particularly useful after a BC's operating mode has changed from/to a U-mode, because any *old* USB-based MIDI devices have then become invalid, and any *new* ones are unavailable until BC Manager has restarted.

File ➜ Restart with default setup:
Functions like Restart (see above), with two differences:

- Since this operation is somewhat 'momentous', a dialog box requires you to confirm that

you indeed want to do this.

- BC Manager's setup file BCMan.ini is deleted before the restart, so that all BC Manager's setup values (such as MIDI I/O port settings and window positions) are restored to their defaults. Consequently, BC Manager's new instance behaves as if you've never run BC Manager before.

This operation might be useful if some setup problem has developed that you find yourself unable to fix quickly otherwise. However, the restoration of BC Manager's default setup also has its drawbacks: for instance, you must set up BC Manager's data contexts for all your B-Controls from scratch; this concerns for instance MIDI I/O port assignments and syx/txt/bc2 file associations. Note, though, that the deletion of BCMan.ini does *not* affect your B-Controls *themselves*, nor the *contents* of any associated syx/txt/bc2 files: it's just that the B-Controls must be *linked* again to the proper MIDI I/O ports and files.

File ➜ Exit:

Terminates BC Manager. If there is any unsaved/unsent B-Control data, you are given the opportunity to save/send them before termination; if you answer 'Cancel' to any question, BC Manager is *not* terminated.

Tip: the associated hotkey (Alt+X) works from almost *any* location in the program, not just from the main window.

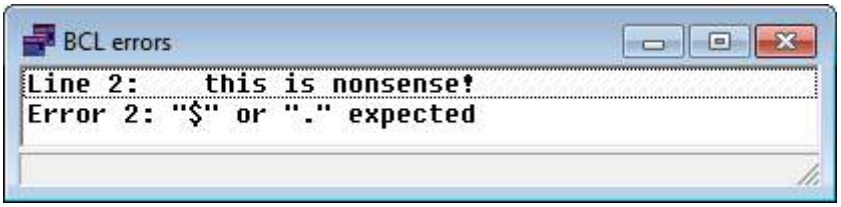
Of course you can also terminate BC Manager by clicking on the ✕ icon on the main window's title bar: the same questions are asked. Pressing Alt+F4 also works, but (unlike Alt+X) only from the main window.

View ➜ B-Controls:

Opens the B-Controls window. This is the gateway to the main functionality of BC Manager. See §10 for more information.

View ➜ BCL errors:

Opens the BCL error window. This window displays any errors that occurred in the most recent BCL script received by BC Manager's BCL interpreter:



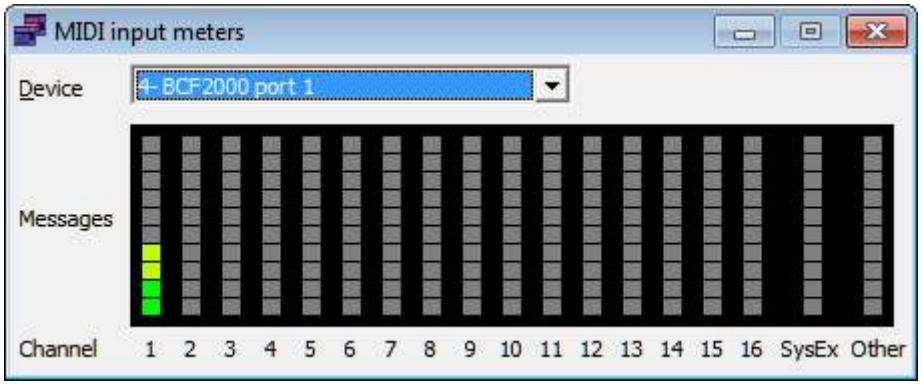
The error messages very closely match the error messages which a BCF or BCR returns when it receives a BCL script.

Note that BCL scripts can be received in three ways:

- You open a BCL file (via File ➜ Open in the B-Controls window).
- You receive a BCL script from a BC via MIDI (e.g. via MIDI ➜ Receive all data in the B-Controls window).
- You execute a BCL script from a BCL editor window (via Run ➜ Execute).

View ➜ MIDI ➜ Input meters:

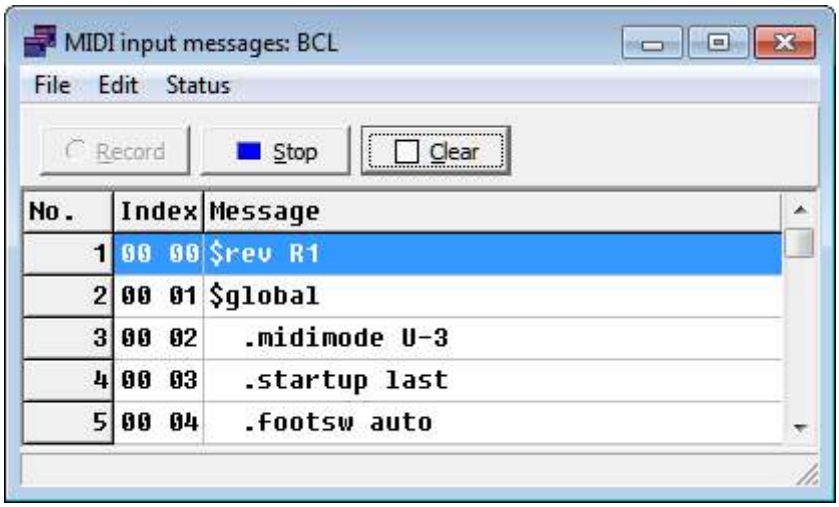
Opens a window showing the messages received recently from the MIDI input devices, via (logarithmical) LEDs per MIDI channel:



This window can be useful for troubleshooting your MIDI connections.
'DISABLED ' in front of the selected MIDI input device indicates that the device is disabled, so no MIDI messages can currently be received from that device. (You can enable devices in the MIDI devices dialog box.)

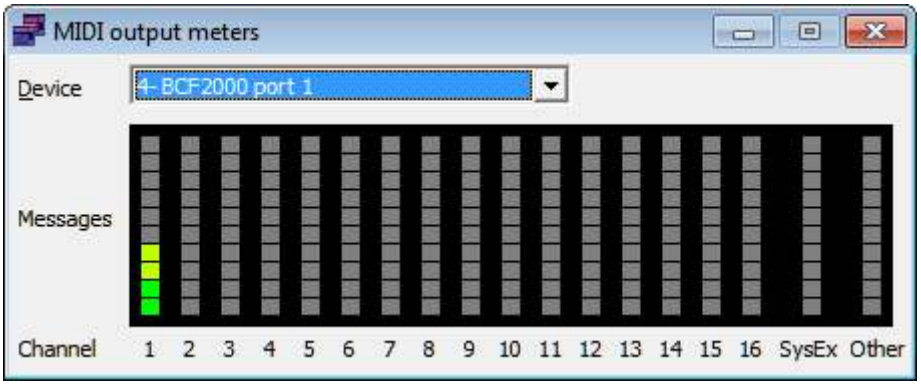
View ➜ MIDI ➜ Input messages (general):
Opens the MIDI input messages window. Here you can record and view messages from the MIDI input devices. See §22 for more information.

View ➜ MIDI ➜ Input messages (BCL):
Opens a window showing a list of the BCL messages received from any BCF or BCR connected to any enabled MIDI input device:



View ➜ MIDI ➜ Mackie monitors ➜ 1/2/3/4:
Opens the indicated Mackie monitor. See §23 for more information.

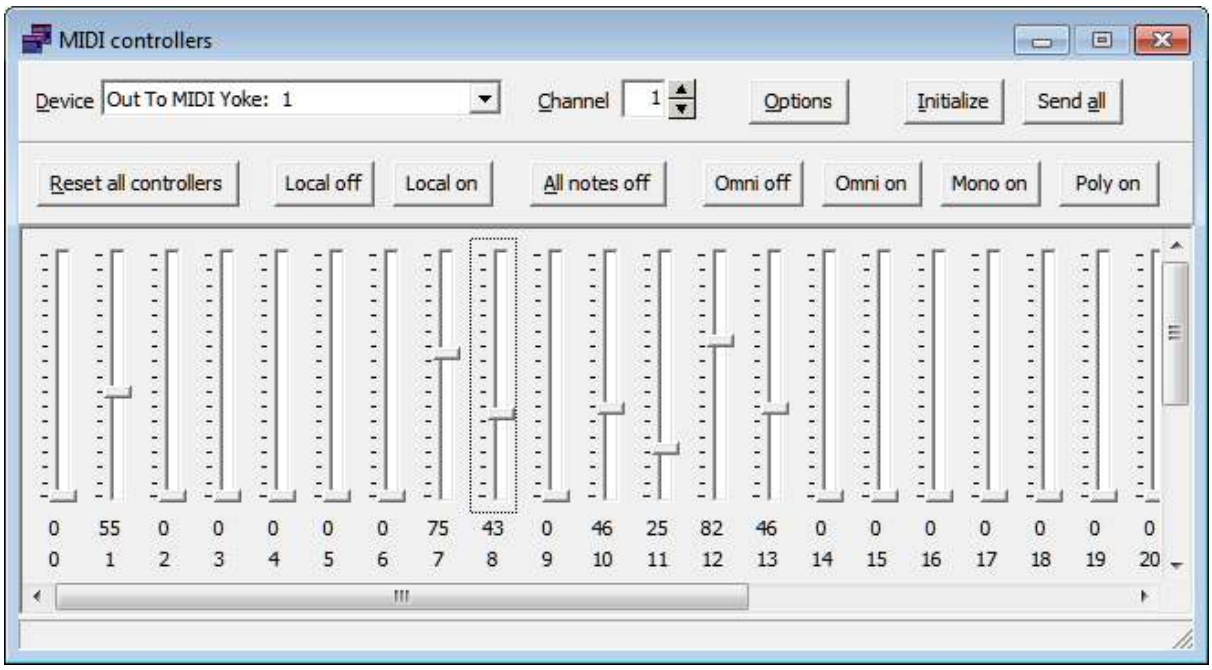
View ➜ MIDI ➜ Output meters:
Opens a window showing the messages sent recently to the MIDI output devices, via (logarithmical) LEDs per MIDI channel:



This window can be useful for troubleshooting your MIDI connections.
'DISABLED ' in front of the selected MIDI output device indicates that the device is disabled, so no MIDI messages can currently be sent to that device. (You can enable devices in the MIDI devices dialog box.)

View ➜ MIDI ➜ Controllers:

Opens a window in which you can send MIDI Control Change messages to any MIDI output device:

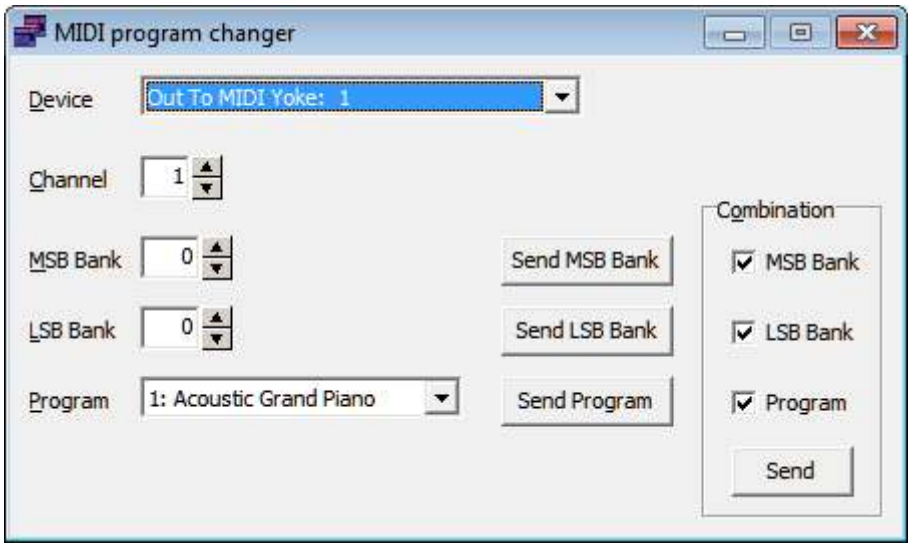


View ➜ MIDI ➜ Keyboard:

Opens a window containing a virtual MIDI keyboard. See §24 for more information.

View ➜ Program changer:

Opens a window from which you can send MIDI Program Change messages prefixed by Bank Select MSB/LSB messages, in any combination:



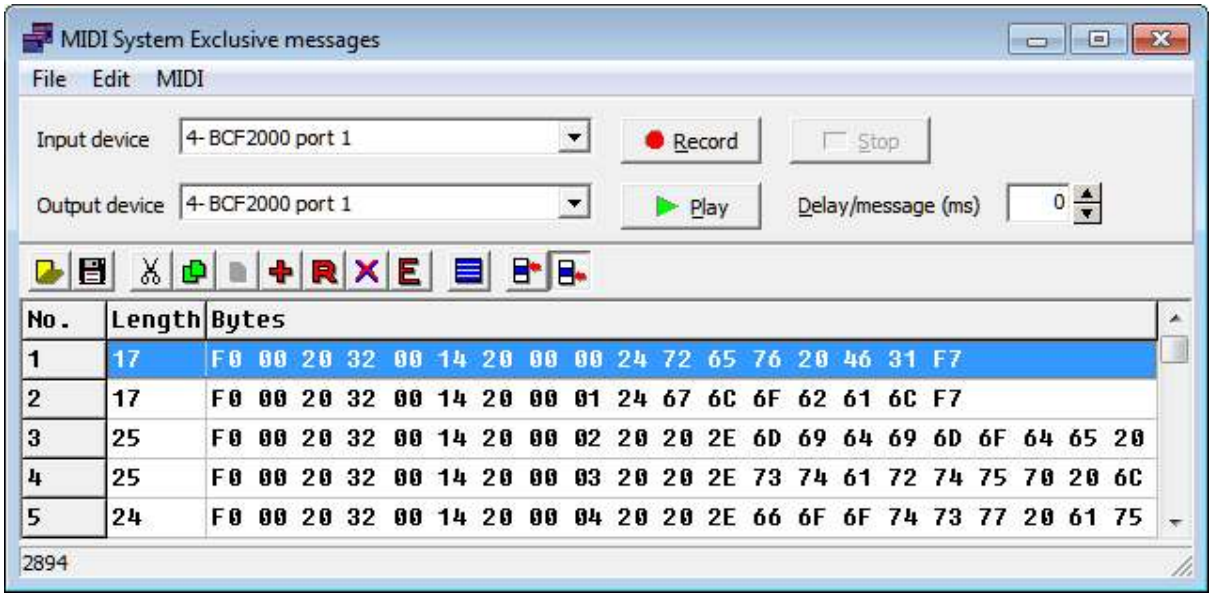
This window is very useful for testing devices that seem unwilling to change programs.

View ➜ MIDI ➜ System messages:

Opens a window from which you can send MIDI System messages. See §25 for more information.

View ➜ MIDI ➜ System Exclusive messages:

Opens a window in which you can perform many actions related to MIDI System Exclusive (SysEx) messages:

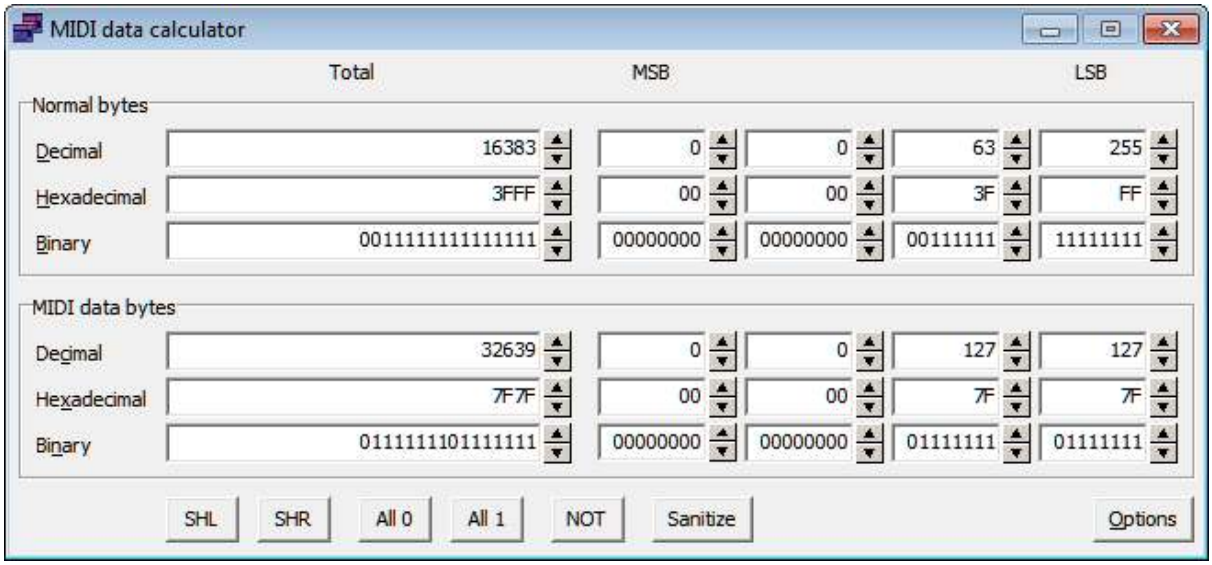


In this window you can create and edit SysEx messages, but also insert SysEx messages from syx files, save messages to syx files, and record and play SysEx messages.

Furthermore, you can extract the SysEx messages from a mid (standard MIDI) file and save them to syx files. One syx file is created for each MIDI track (in the mid file) that contains one or more SysEx messages.

View ➜ MIDI ➜ Data calculator:

Opens a window in which you can convert a 4-byte sequence of 'normal' bytes to their MIDI data counterparts or vice versa. (The MIDI protocol demands that the highest bit ('bit 7') of each MIDI data byte is zero; consequently, if a sequence of two or more MIDI data bytes constitute one number, all bit 7s are 'scrapped' and all more significant bits are shifted to the right.)



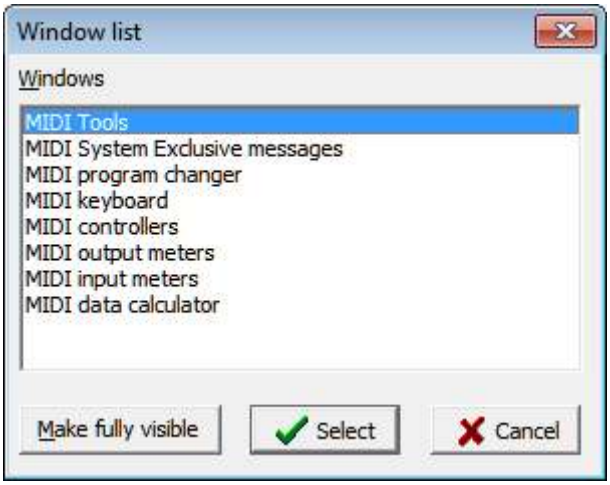
Both the normal bytes and the MIDI data bytes are displayed in decimal, hexadecimal and binary form simultaneously. You can also perform simple operations like SHL and SHR. The Sanitize button clears bit 7 of all four MIDI data bytes. Via the Options button you can customize the window's display.

View ➜ Stay on top:

When this menu item is checked, BC Manager's main window stays on top of any other windows, either belonging to BC Manager or other applications (except of course those that have this stay-on-top property too).

View ➜ Window list:

Opens a dialog box that allows you to quickly navigate to any open window:



Note that the hotkey (Alt+0) for opening this dialog box works from almost *any* location in the program, not just the main window.

By pressing 'Make fully visible' you can move a window that is partially or wholly outside the current monitor(s) into full view.

Options ➜ MIDI devices:

Opens a dialog box in which you can configure the MIDI devices that BC Manager monitors. See §6 for more information.

Options ➜ B-Control detection:

Opens a dialog box in which you can set several options concerning BC Manager's B-Control detection operations. See §9 for more information.

Options ➜ BCL output:

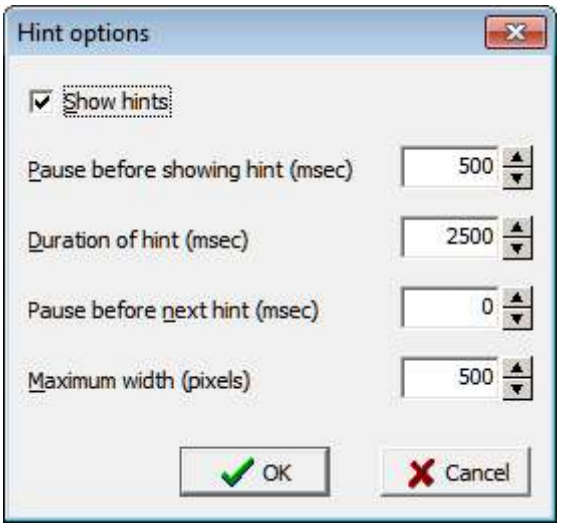
Opens a dialog box in which you can set a number of features of the BCL scripts that BC Manager outputs. See §19 for more information.

Options ➜ Mackie monitors ➜ 1/2/3/4:

Opens a dialog box in which you can set the selected Mackie monitor's MIDI input port and several display options. See §23 for more information.

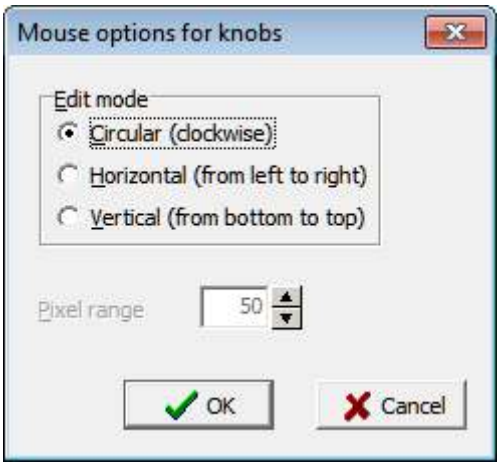
Options ➜ Hints:

Opens a dialog box in which you can set options affecting the hints that are displayed when you move the mouse cursor over buttons etc.:



Options ➜ Mouse:

Opens a dialog box in which you can set the way in which the mouse turns the parameter knobs:



Options ➜ Printer:

Shows the standard Windows dialog box in which you can set your printer's properties. These settings affect all printing operations performed from BC Manager, with the following exceptions: in the layout windows, the 'Print window image' and 'Print strips' operations do *not* adhere to the orientation setting made here, but always use Landscape.

Help ➜ Keyboard and mouse actions for knobs:

Opens a table containing the computer keyboard and mouse actions that you can apply to parameter knobs:

Keyboard and mouse actions for knobs

Key(s)/mouse click	Action
Alt+Enter or Alt+Left click (actual knob)	Open a dialog box in which you can type a new value
Left arrow	Decrease the value by 1
Right arrow	Increase the value by 1
Ctrl+Left arrow	Decrease the value by a 'big' amount (often 10)
Ctrl+Right arrow	Increase the value by a 'big' amount (often 10)
Home	Select the lowest value
End	Select the highest value
Left click (caption/value)	Select the knob under the mouse
Left click (actual knob)	Set the value as indicated by the mouse position
Right click	Change the value by a 'big' amount (often 10) toward the mouse

This is basically the same table as the one in §26 in this manual.

Help ➜ BC Manager Manual:

Opens this manual in the external application associated with the file extension 'pdf'.

Help ➜ B-Control MIDI Implementation:

Opens BC MIDI Implementation.pdf (*BCMI*) in your pdf viewer.

Help ➜ About BC Manager:

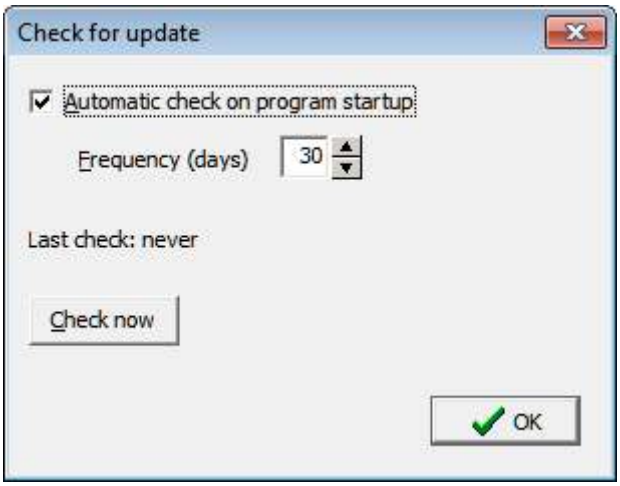
Opens a dialog box containing information on BC Manager, such as its version number and memory usage:

Help ➜ Mountain Utilities web site:

Provided you're connected to the internet, your web browser opens the Mountain Utilities web site, where you can find up-to-date information about BC Manager and other matters related to the BCF2000 and BCR2000.

Help ➜ Check for update:

Opens a dialog box in which you can set the frequency at which BC Manager automatically searches for updates at the Mountain Utilities web site:

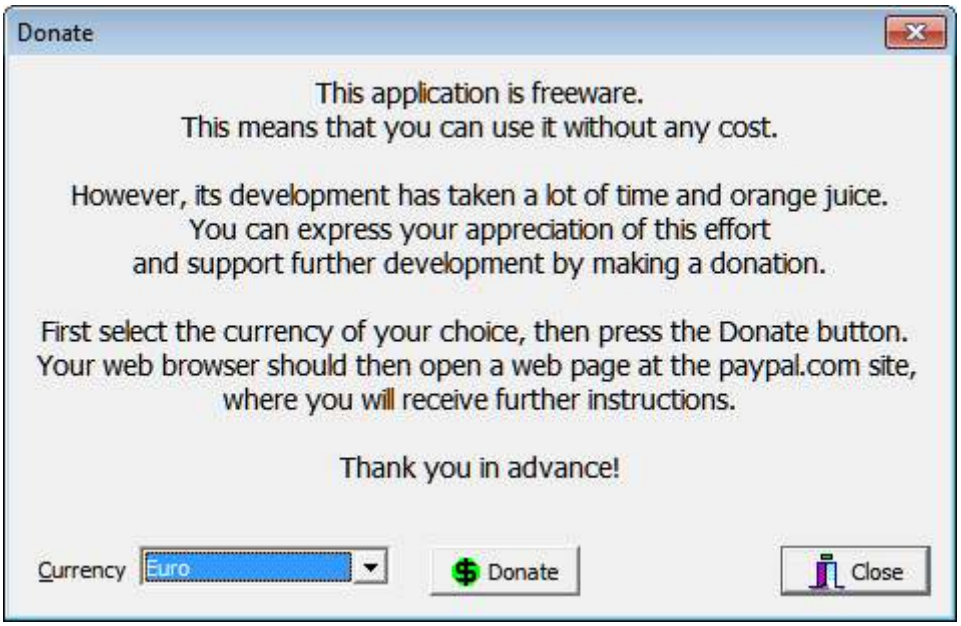


When an update is available, the program asks you whether you wish to open the program's web page at the Mountain Utilities site. You can also check for updates manually, by pressing the 'Check now' button.

Note: If your firewall catches the program's connection attempt and asks you whether you want to allow this, you can safely say yes: no information identifying you or your computer will be sent to the Mountain Utilities web site.

Help ➜ Donate:

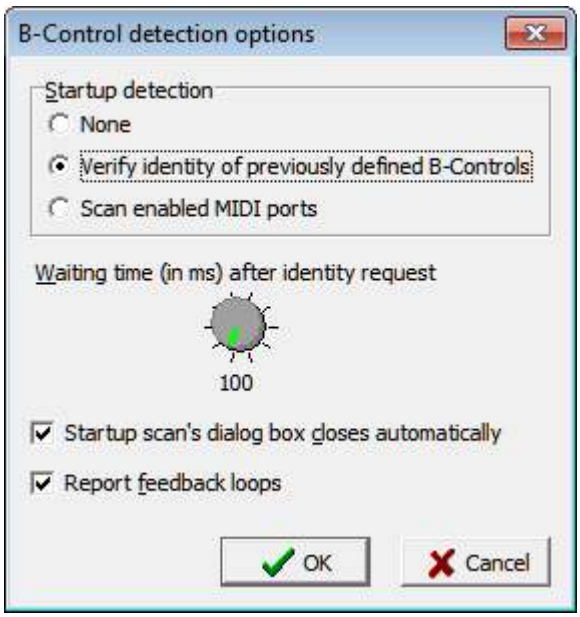
Opens a dialog box in which you can express your appreciation of BC Manager and support its further development by making a donation:



If you press the 'Donate' button, your web browser should open a web page at the paypal.com site, where you will receive further instructions.

9. The B-Control detection options dialog box

The B-Control detection options dialog box is accessed via the main window's Options pull-down menu (➞ B-Detection).



The following options are available:

- Startup detection:
  - None:
    On startup BC Manager does not try to find any previously defined or new B-Controls connected to your computer. This option provides the fastest startup, but you must manually execute the 'Refresh connection status' operation before you can exchange data with any B-Control.
  - Verify identity of previously defined B-Controls:
    On startup BC Manager automatically verifies the presence of all previously defined B-Controls via 'Refresh connection status' operations. This is the default setting.
  - Scan enabled MIDI ports:
    On startup, BC Manager scans all *enabled* MIDI I/O ports: any previously defined B-Controls are reconfirmed, and contexts are established for any newly discovered B-Controls. Unless you regularly add and remove B-Controls, you probably don't want to use this setting, because BC Manager's startup takes a bit longer.

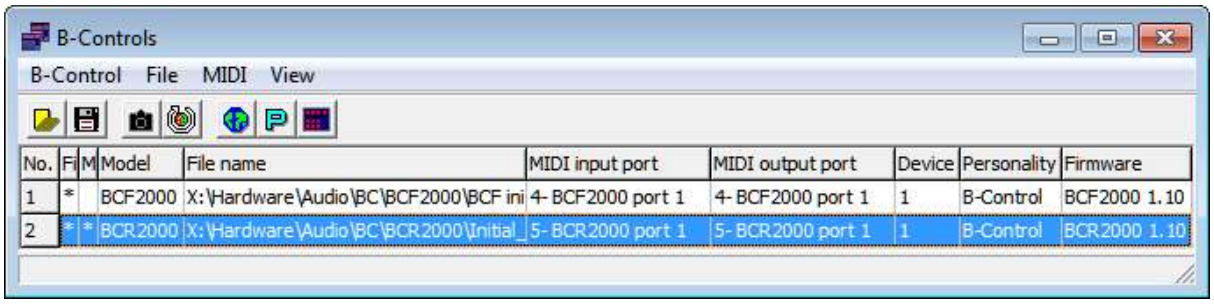
- Waiting time (in ms) after identity request:
  This setting determines how long BC Manager waits for a reply after sending a (firmware) 'Identity Request', as performed during startup and the 'Detect B-Controls' operation (accessible via the B-Controls window's MIDI pull-down menu). You can set this waiting time from 100 to 1000 milliseconds. The default of 100 ms should work fine unless you have a particularly slow MIDI connection; you should only raise the value if a B-Control responds too slowly to an identity request.

- Startup scan's dialog box closes automatically:

If this checkbox is checked, BC Manager's startup scan dialog box closes automatically when the scan is complete. If this checkbox is unchecked, you must manually close this dialog box.

- Report feedback loops:
  This setting determines whether the detection operations inform you of any detected feedback loops from the computer's MIDI output ports to the computer's MIDI input ports. So if these reports annoy you, you can simply suppress them. However, usually it is better to remove the feedback loops themselves, since they can degrade performance.

10. The B-Controls window



The B-Controls window is the gateway to the main functionality of BC Manager. It displays a table showing the B-Controls that BC Manager currently maintains in the computer's RAM. These representations in RAM can be loaded from and saved to files, and received from and sent to actual BCFs and BCRs.

The best way to approach this setup is to use the *file* as the 'master' and the BCF or BCR as the 'slave'. This is because **it is advisable *not* to *receive* the setup data from a BCF or BCR**, for several reasons:

1. MIDI communication with the BCF and BCR via BCL SysEx messages is extremely slow. (This is simply because BCL messages consist of long *text strings*, rather than compact binary data.)
2. The BCF and BCR don't store comments, hence can't transmit them.
3. The BCF and BCR transmit certain aspects of their internal data *incorrectly*. This especially concerns custom output ('`.tx`') lines. See *BCMI* for details.

The columns have the following meaning:

- No.:
  This is simply the number of the B-Control by which BC Manager identifies the B-Control in window captions etc. (This number has no relationship to Device IDs, MIDI channels, etc.) The maximum number of B-Controls you can define is 16.
  Note: BC Manager doesn't prevent you from having different B-Controls use the same file and/or BCF2000/BCR2000 (via identical MIDI I/O ports and Device numbers), but it is advisable not to do so to avoid confusion and data mismatches.

- File:
  An asterisk in this column indicates that the B-Control's data maintained by BC Manager has changed with respect to its associated file, either via manual editing or by receiving data from the BCF or BCR connected via MIDI. The asterisk disappears after you save the file again.
  Note: When you open one of the standard syx files provided by Behringer, the asterisk immediately appears, and when you then try to exit BC Manager, you get the question whether you want to save the file. This behavior is not a bug in BC Manager, but a consequence of the fact that Behringer's syx files do not contain any reference to unused presets. By contrast, when BC Manager saves a syx, txt or bc2 file, *all* presets get stored, even the empty ones, and when *such* a file is reopened, the asterisk does *not* appear.

- MIDI:
  An asterisk in this column indicates that the B-Control's data maintained by BC Manager has changed with respect to the associated BCF or BCR connected via MIDI, either via manual

editing or by opening a file. The asterisk disappears after you send the data maintained by BC Manager to the BCF or BCR.

- File:
  The name of the file associated with the B-Control. This is a syx, txt or bc2 file containing the full BCL definition of the B-Control.

- MIDI input port/MIDI output port:
  The names of the MIDI input and output ports to which the BCF or BCR is connected.
  Special cases:
  - 'ABSENT ' in front of a name means that the device doesn't currently exist. Characteristically this happens to USB-based ports linked to a BCF/R that has been switched off. If you don't need the BCF/R in your current session, you may simply leave the situation as it is. When you switch the BCF/BCR back on and restart BC Manager, the device is immediately available again.
  - 'DISABLED ' in front of a name means that the device has been disabled via the MIDI devices dialog box. You can simply re-enable the device there.

- Device:
  The Device ID that BC Manager uses in its MIDI SysEx communication with the BCF or BCR. For this communication to work, you must make sure that this number corresponds with the Device ID set on the BCF or BCR.

- Personality:
  The current status of the connected BC, as established via the latest 'Refresh connection status' operation. The following values occur:

  - **Offline**:
    The BC is not connected (or at least could not be detected), so no meaningful MIDI data communication is possible. All you can do in this case is resolve the problem and retry 'Refresh connection status'.
    The cause of the problem can be any of the following:

    - The MIDI input and/or output ports are undefined or disabled. Refer to the MIDI devices dialog box, accessible from BC Manager's main window.

    - The Device ID of the BC itself and the Device ID specified in BC Manager don't match.
      Obviously, to remedy this situation, you should change one or both of these Device IDs so that they do match, and then try 'Receive identity' again. Beware: if the BC is in a USB operating mode (U-1/2/3/4), any change to its Device ID causes the USB-based MIDI I/O ports to change their identities, and these new MIDI I/O ports are unavailable to BC Manager until you have quit and restarted BC Manager, and manually enabled them via the MIDI devices dialog box. On the other hand, if the BC is in a standard MIDI operating mode (S-1/2/3/4), you can change the BC's Device ID without having to quit and restart BC Manager.

    - The BC is in Global Setup edit mode, the display showing 'EG'. You should press EXIT on the BC and retry 'Refresh connection status'.

    - The BC is in element edit mode, the display showing 'b *nn*', 'E *nn*' or (BCF only) 'Fd

n'. You should press EXIT on the BC and retry 'Refresh connection status'.

- **Bootloader**:
  The BC has been started up in bootloader mode. BC Manager's functionality is limited.
  The BC enters bootloader mode on startup if there is a problem with its firmware or if you are holding the STORE and LEARN buttons when you press the POWER button. In the former case the BC's display shows '`noOS`', in the latter '`LOAd`'. Note that in bootloader mode the BC cannot communicate via USB-based MIDI connections, only via *standard* MIDI connections (and only via Output A at that). See *BCMI* for further details on bootloader mode.

- **Unknown**:
  The BCF's *firmware version* has been retrieved successfully, but somehow the BCF's *emulation mode* could not be established.

- **B-Control**:
  The BC is in standard B-Control mode. For BC Manager this is the desired value: you can access all the BC's data structures, including global setup, the temporary preset and the memory presets.

- **MC C/So**:
  The BCF is in 'MC C' or 'MCSo' emulation mode. BC Manager's functionality is limited.
  (Note: the fact that no distinction is made between 'MC C' and 'MCSo' is due to a limitation of the BCF's emulation mode identification protocol.)

- **LC**:
  The BCF is in 'LC' emulation mode. BC Manager's functionality is limited.

- **bhuI**:
  The BCF is in 'bhuI' emulation mode. BC Manager's functionality is limited.
  (Note: so far I haven't found a way to 'actively' establish that the BCF is in bhuI emulation mode. Instead, BC Manager simply assumes that the BCF is in bhuI mode when none of the active checks for the *other* modes have led to a positive result. Hence, it takes a while for BC Manager to 'discover' bhuI mode.)

In all cases except 'Offline' and 'B-Control', you can only communicate with the BC via 'Refresh connection status' and 'Send firmware'.

- Firmware:
  The actual firmware identity string returned by the B-Control during the 'Refresh connection status' operation. Normally this should be something like 'BCF2000 1.10' or 'BCR2000 1.10'. Exceptionally, this can be 'BCF2000 BOOTLOADER 1.0' or 'BCR2000 BOOTLOADER 1.0', indicating that the B-Control is in bootloader mode.

The menu provides the following operations:

B-Controls ➜ New BCF2000/BCR2000:
Adds a new B-Control of the specified type to the list of B-Controls. (You are then responsible for assigning a file and/or MIDI ports to this new B-Control.)

B-Controls ➜ Import Reason map:

Imports a Propellerhead Reason 'remote map' file into one or two new B-Controls. (This works for Reason 3.0 to (at least) 7.0.1.) Each Reason device defined in the map file is emulated as a separate memory preset.

The number of new B-Controls depends on the number of Reason devices defined in the file. A Reason 3 map file defines 27 device, which fit in a single B-Control, and the same goes for a Reason 4 map file: 29 devices. However, the map files of Reason 5 and later define more than 32 devices, which require multiple new B-Controls.

See §21 for more details.

B-Controls ➜ Close selected:

Closes the selected (i.e. highlighted) B-Control. This merely removes BC Manager's internally maintained representation of the B-Control, including the *references* to the file and the MIDI I/O ports; it does not *delete* the file itself.

File ➜ Open:

Opens a dialog box in which you can select a syx, txt or bc2 file whose data you want to load into the selected B-Control (as maintained by BC Manager). By default you can see only the existing syx files, but you can see the existing txt and bc2 files if you change the selected item of the 'Files of type' list box.

Note that saving and sending verification dialog boxes show up when appropriate and that the file name associated with the selected B-Control is updated to the name of the opened file. By contrast, these things do *not* happen in case of the Import operation (see below). (The idea is that an *opened* file contains *all* the data for a B-Control, although this is by no means obligatory.)

File ➜ Import:

Works exactly like the Open operation (see above), except that no saving or sending verification dialog boxes show up and the file name associated with the selected B-Control is *not* updated to the name of the opened file. Thus, Import is ideal for assembling a B-Control setup from several different source files.

Note: if you import a file containing a memory preset (i.e. a file containing a preset definition followed by a **`$store`** statement), the B-Control's *temporary* preset is always changed too. This is simply a consequence of how BCL works: any preset definition alters the *temporary* preset, and can only be copied to a *memory* preset by a subsequent **`$store`** statement — of course by that time the *temporary* preset has already been changed.

File ➜ Save:

Saves the selected B-Control's data (as maintained by BC Manager) to the syx, txt or bc2 file currently associated with the B-Control.

File ➜ Save As:

Opens a dialog box in which you can select a syx, txt or bc2 file to which you want to save the selected B-Control's data (as maintained by BC Manager). Syx is the default destination, but you can save to a txt or bc2 file by changing the selected item of the 'Save as type' list box.

BC Manager can reopen any of these file types, but there are several notable differences that determine which file type is best for you:

The difference between txt and bc2 is that BC Manager saves txt files in UTF-8 format and bc2 in ANSI format. Thus, such a txt file retains any Unicode characters in comments and element names, but a bc2 file doesn't. And of course the txt file extension is the simpler choice if you want to edit the files in a standard text editor such as Notepad (note that Notepad supports UTF-8). On the other hand, you might consider bc2 if you want to work with Royce Craven's

BCn2000 Script Editor or if you want to avoid confusion with other, non-BCL txt files in the same folder.

The advantages of txt and bc2 over syx are:

- A txt or bc2 file is always slightly smaller than the corresponding syx file.
- You can edit txt and bc2 files via any external text editor. (However, other BC-managing products might only be able to work with syx files.)
- When comments and element names are saved to a syx file, all non-ASCII characters (ANSI and Unicode) are converted to dots: this is necessary to comply with the syx file format. On the other hand, ANSI characters *are* saved to txt and bc2 files, and txt files even retain the Unicode characters.
- Each individual data section (global setup, preset specs, button, encoder, fader) in a syx file can 'only' contain up to 16384 BCL messages (lines), so syx files can't be used for *extremely* big data sections (i.e. with huge amounts of multi-line comments or 'custom output' MIDI messages).

All in all it's totally up to you which format you choose, but personally I'd say: go for txt or possibly bc2 if you have no specific reason to use syx.

File ➜ 1-10 *filename*:

The names of the (max. 10) most recently accessed file names are shown: if you select a file, that file is reopened.

File ➜ Detach:

Detaches the syx/txt/bc2 file from the selected B-Control. (Note that the file is not *deleted*.)

This operation is mainly useful when you wish to transfer a file from one B-Control to another: whereas BC Manager does allow you to associate a file with more than one B-Control simultaneously, it is advisable to detach the file from the 'old' B-Control to avoid confusion.

MIDI ➜ Receive all data:

Sends a request to the connected BCF or BCR for sending all its data to BC Manager. **You should avoid this operation as much as possible, because of the reasons outlined at the start of this section.**

MIDI ➜ Receive snapshot:

Sends a request to the connected BCF or BCR for its current control value settings. Any data sent back by the BC does *not* alter BC Manager's representation of the selected B-Control, but only shows up in the MIDI input message window, which is accessible from BC Manager's main window via View ➜ MIDI ➜ Input messages (general). So basically this operation is only useful for testing purposes.

MIDI ➜ Send all data:

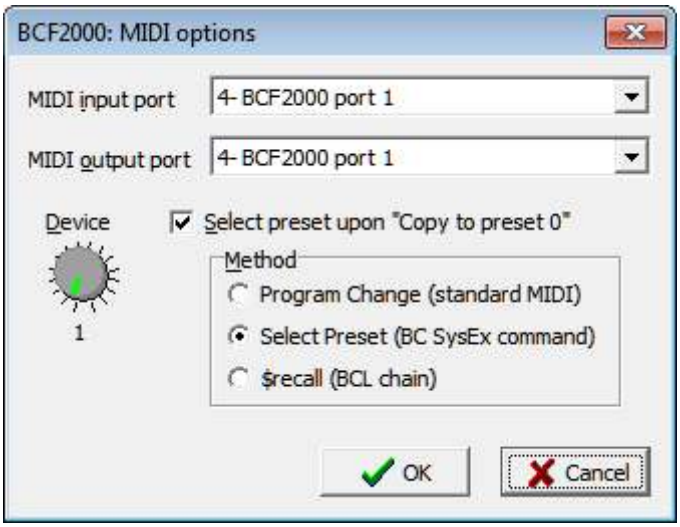
Sends all data (as maintained by BC Manager) to the connected BCF or BCR.

MIDI ➜ Send modified data:

Sends only the *modified* data (as maintained by BC Manager) to the connected BCF or BCR. Compared to the 'Send all data' operation, this usually works much quicker, since e.g. presets that are assumed to be unchanged on the BC are not sent.

MIDI ➜ Options:

Opens a dialog box in which you can set several options related to the MIDI communication with the connected BCF or BCR:



See §6 for a discussion of the MIDI input/output port and Device settings.

If 'Select preset upon "Copy to preset 0"' is checked, the 'Copy to preset 0' operation (in the presets window) attempts to synchronize the connected BCF or BCR by selecting the same memory preset for *its* 'preset 0' (or 'temporary preset'), exactly as occurs when you press the PRESET ◂/▸ buttons on the device itself. Obviously this synchronization attempt only really succeeds if the memory preset in the BC is identical to its counterpart in BC Manager!

You can also select the *method* by which the 'Copy to preset 0' operation tries to synchronize the actual BC:

- Program Change (standard MIDI):
  This sends a standard MIDI Program Change message. This is the fastest method, since this only sends two MIDI bytes. However, it is unsafe in that any *other* devices at the same MIDI port (such as other BCFs and BCRs) are likely to respond in unwanted ways. Another restriction is that the Receive channel (as set in the global setup dialog box) must not be 'Off' and must match the value selected on the connected BCF or BCR.

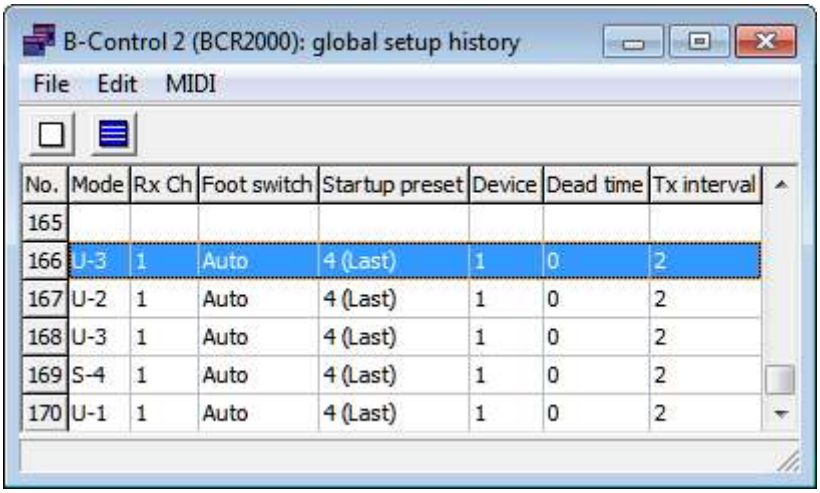
- Select Preset (BC SysEx command):
  This sends a single B-Control-specific 'Select Preset' message (this is actually a MIDI System Exclusive message). Only the intended BCF or BCR should respond (provided its Device ID is correct), so this should be very safe, and is therefore the default and preferred method.

- $recall (BCL chain):
  This sends a BCL (B-Control Language) chain containing a **`$recall`** statement (see *BCMI* for details). The chain consists of 2 or 3 MIDI System Exclusive messages (depending on BC Manager's BCL output options), so this is the slowest of the three methods. The advantage is that you get error feedback,[2] which the other two methods don't provide.

[2] Apart from the BCL Reply messages which the BC returns in response to the BCL chain, the BC also returns any LEARN output bytes if the preset's Request parameter is on. Unfortunately, this LEARN output occurs *before* the BC completes its response messages to BC Manager's BCL chain. BC Manager deals with this by simply ignoring any interfering MIDI messages (SysEx or other), i.e. any messages that are not BCL Reply messages; however, it does mention any interfering *SysEx* messages in its error listbox.

MIDI ➜ Maintenance ➜ Global setup history:

Opens a window in which you can view and edit the BCF/BCR's global setup history:



No.	Mode	Rx Ch	Foot switch	Startup preset	Device	Dead time	Tx interval
165							
166	U-3	1	Auto	4 (Last)	1	0	2
167	U-2	1	Auto	4 (Last)	1	0	2
168	U-3	1	Auto	4 (Last)	1	0	2
169	S-4	1	Auto	4 (Last)	1	0	2
170	U-1	1	Auto	4 (Last)	1	0	2

The BCF internally maintains a global setup history area containing 73 setups; the BCR's history area contains 170 setups. The BCF and BCR fill these areas from bottom to top; so the *active* global setup is always the first one from the start that isn't undefined. Once the area overflows, the BCF/BCR clears it completely and starts filling it from the bottom again.

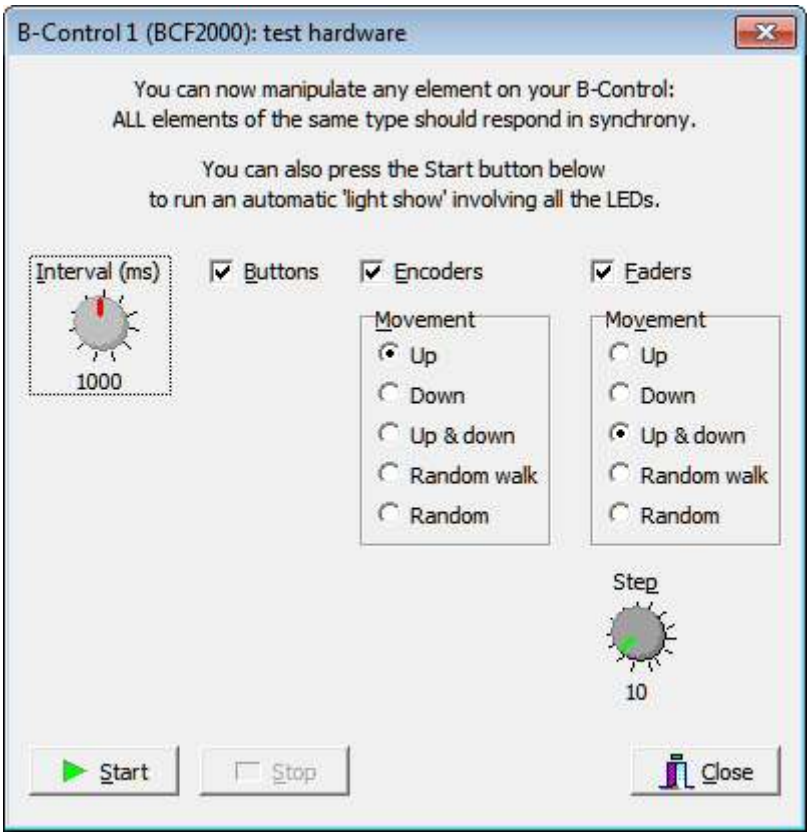
In the global setup history window, you have to run MIDI ➜ Receive to retrieve the global setup history from the BCF/BCR.

Normally there is no need to edit the BCF/BCR's global setup history: you can change the BC's global setup from the 'standard' global setups window, accessible from the View pull-down menu (see below). However, it's nice that you can see which setups you have used in the past; and if you really want to, you can restore a previous setup by removing all the more recent ones and restarting the BCF/BCR.

MIDI ➜ Maintenance ➜ Test hardware:

When you execute this operation, the following sequence of events takes place:

1. BC Manager uploads a special preset to the BC.
2. The 'test hardware' dialog box opens:



As long as this dialog box is open, you can manually press/move any button, encoder, fader, foot controller or foot switch: all other elements of the same type should respond in synchrony.

Additionally, from this dialog box you can press the Start button, which starts a kind of 'light show': BC Manager starts sends sending Control Change messages at regular intervals to the BC, which should cause the values of all buttons, encoders and faders to change in synchrony automatically. You can use the various knobs and checkboxes in the dialog box to customize the properties of the light show.

3. After you have closed the dialog box, BC Manager restores the temporary preset that was active *before* the special preset. (Note: this is the 'preset 0' that *BC Manager* knows about, which *could* be different from the previous temporary preset on the *BC*, namely if before the hardware test there was a mismatch between the temporary preset on the BC and the one in BC Manager.)

MIDI ➜ Maintenance ➜ Send firmware:

Opens a dialog box in which you can select a syx file containing BCF2000 or BCR2000 firmware data. The data in this file is then sent to the selected BCF or BCR to upgrade (or downgrade!) its firmware.

You might want to (in fact: probably *should*) execute this operation if your current firmware version is lower than 1.10. (The current firmware version is briefly shown in the BC's display after you switch it on. In BC Manager it appears in the Firmware column of the B-Controls window and in the 'receive firmware version' dialog box of the 'Refresh connection status' operation.)

Beware: you should make sure that the BC's Operating Mode is *not* S-1 or S-2 when you execute this operation (because these modes use merging, which messes up the

operation). Any U-mode should normally work fine (though U-1 is the safest of these), but the preferred mode is S-4 (with BC Manager receiving the BC's confirmation messages via MIDI OUT A, *not* MIDI OUT B/THRU). S-3 is also possible, provided you use MIDI OUT B/THRU.

MIDI ➜ Refresh connection status:

Sends a request to the connected BCF or BCR for its firmware identity. If the BC is indeed connected, it should respond with the name of the BC plus its firmware version number, so e.g. 'BCF2000 1.10' or 'BCR2000 1.10'. In the case of a BCF, 'Refresh connection status' additionally tries to establish its emulation mode.

The 'Refresh connection status' operation is extremely useful for testing whether the MIDI connection is working correctly bidirectionally. In fact, the establishment of a BC's 'personality' and 'firmware' is a prerequisite for any 'meaningful' MIDI data exchanges between BC Manager and that BC: most MIDI communication operations are disabled if the BC's personality isn't 'B-Control'.

The results of 'Refresh connection status' end up in the Personality and Firmware columns of the B-Controls window. For further details, see the discussions of these columns above.

MIDI ➜ Detect B-Controls:

BC Manager scans for any B-Controls connected via the *enabled* MIDI I/O ports. This works exactly the same as the scan performed on startup if 'Startup detection' has been set to 'Scan enabled MIDI ports'; see the discussion of the 'B-Control detection options' dialog box, accessible via BC Manager's main window.

View ➜ Global setups:

Opens the global setups window. See §11 for more information.

View ➜ Presets:

Opens the presets window for the selected B-Control. (Note: double-clicking on a B-Control's row also achieves this.) See §12 for more information.

View ➜ Layout:

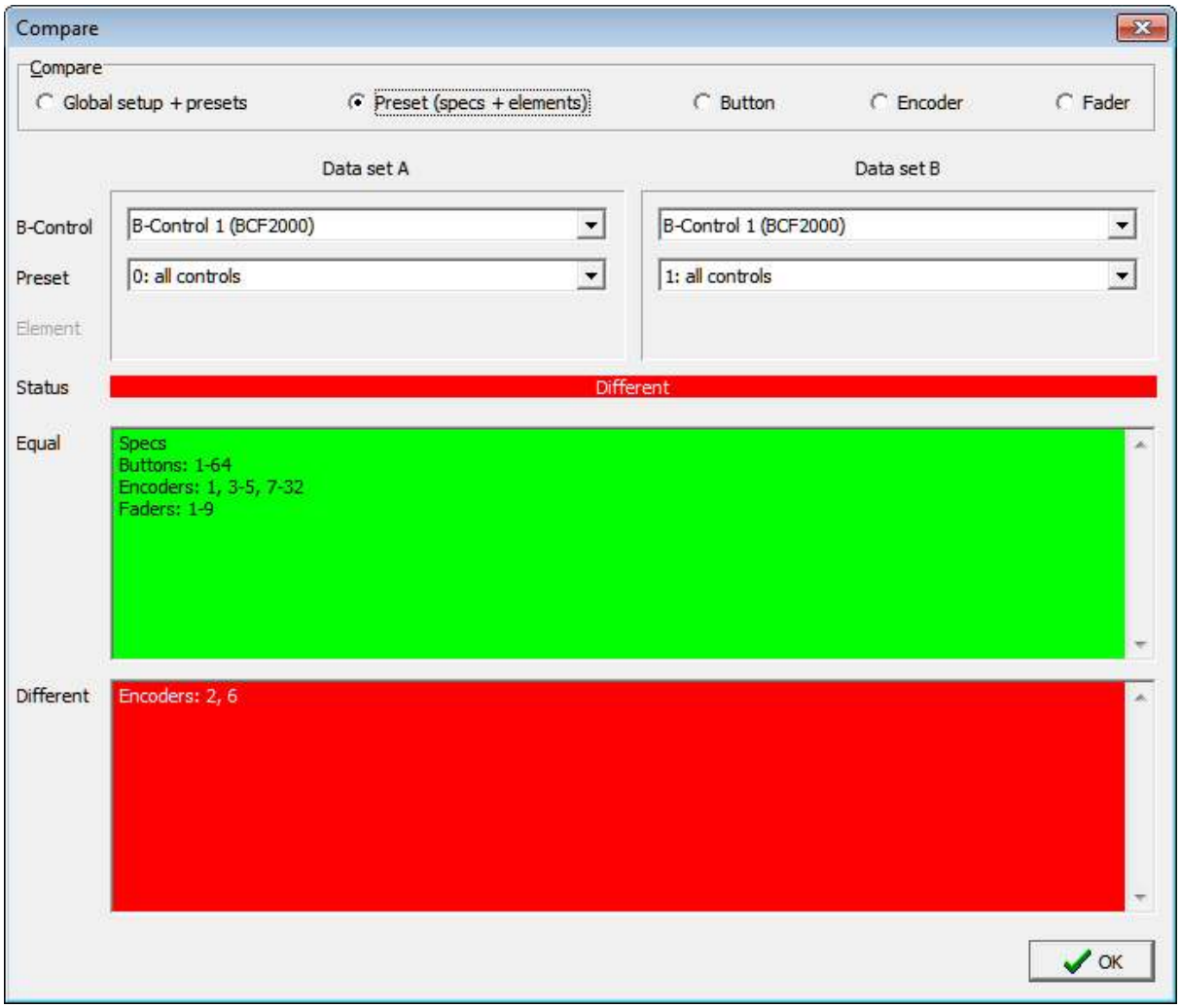
Opens the layout window for the selected B-Control. See §14 for more information.

View ➜ BCL editor:

Opens the BCL editor window for the selected B-Control. See §20 for more information.

View ➜ Compare:

Opens a dialog box in which you can compare presets and elements:



View ➜ Monitor:

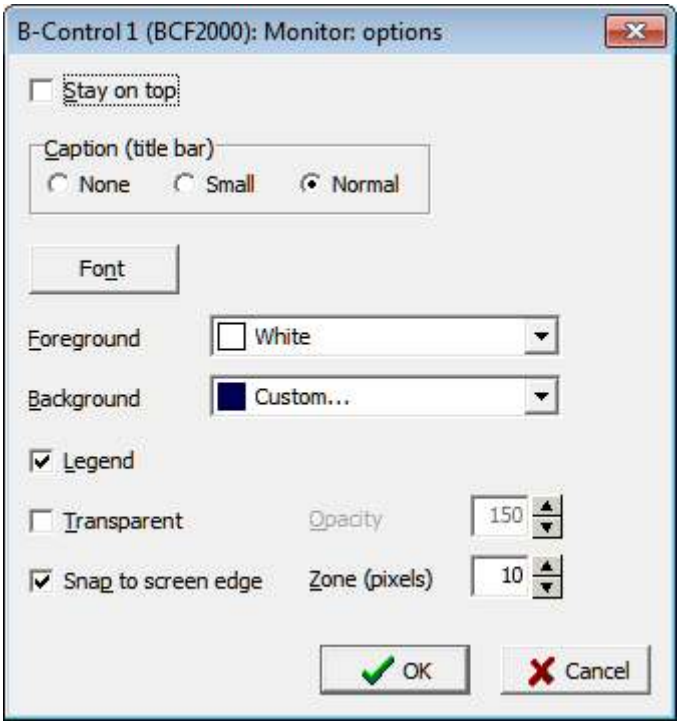
Only available for a BCF2000 in any emulation mode except bhuI. Opens a window that shows the display messages that the BCF outputs. (This works quite like Behringer's BCFview utility.)

Note the difference with BC Manager's *generic* Mackie monitor facility (available from the main window via View ➜ MIDI ➜ Mackie monitors): a generic Mackie monitor intercepts the Mackie display messages *from* a MIDI application *to* a BCF (in whatever mode) or to a BCR, but *here* we capture messages *from* a BCF (in emulation mode).

A subtle point is that a BCF in emulation mode actually *routes back* any Mackie display message it receives (from the MIDI application) *as a BCF display message* (the two formats differ slightly). So you don't need to use a *generic* Mackie monitor window for a BCF in emulation mode: the BCF outputs *both* the converted Mackie messages and its own messages.

View ➜ Monitor options:

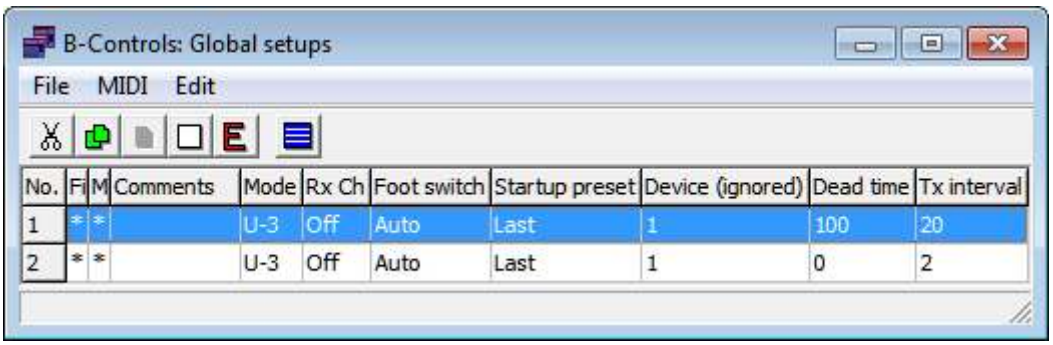
Opens a dialog box in which you can set the BCF2000's monitor display options:



This works practically the same as the options dialog box for the *generic* Mackie monitor windows; see §8 under Options ➜ Mackie monitors.

11. The global setups window

The global setups window displays a table containing the global setup values of the currently maintained B-Controls:



No.	Fi	M	Comments	Mode	Rx Ch	Foot switch	Startup preset	Device (ignored)	Dead time	Tx interval
1	*	*		U-3	Off	Auto	Last	1	100	20
2	*	*		U-3	Off	Auto	Last	1	0	2

The columns have the following meaning:

- No.:
  This is the number of the B-Control, exactly corresponding with the B-Controls displayed in the B-Controls window.

- File/MIDI:
  Asterisks in these columns indicate that the representation of the global setup (as maintained by BC Manager) has changed with respect to its associated file or the BC connected via MIDI. See §10 for further explanation.

- Comments:
  A text that is editable via the Info tab of the global setup dialog box. This text can be of any length.

- Other columns:
  These display the actual global setup parameters. See *BCMI* for details.
  Note: as of BC Manager 2.3.3, the Device IDs shown in the 'Device (ignored)' column are completely ignored, i.e. they are never sent to the BC or written to a syx/txt/bc2 file.

The menu provides the following operations:

File ➜ Export table:
Opens a dialog box in which you can export the table of global setup parameters to a specially formatted 'txt' file. You could use this as an input file to create a table in an external word processor.

File ➜ Export window image:
Opens a dialog box in which you can export an image of the global setups window to a bitmap ('bmp') file.

File ➜ Print window image:
Prints an image of the global setups window.

MIDI ➜ Receive:

Sends a request to the connected BCF or BCR for sending its global setup to BC Manager.

MIDI ➜ Send:

Sends the selected B-Control's global setup to the connected BCF or BCR.

Edit ➜ Cut:

Copies the selected global setup(s) to the global setup clipboard and initializes the selected global setup(s).

Edit ➜ Copy:

Copies the selected global setup(s) to the global setup clipboard.

Edit ➜ Paste:

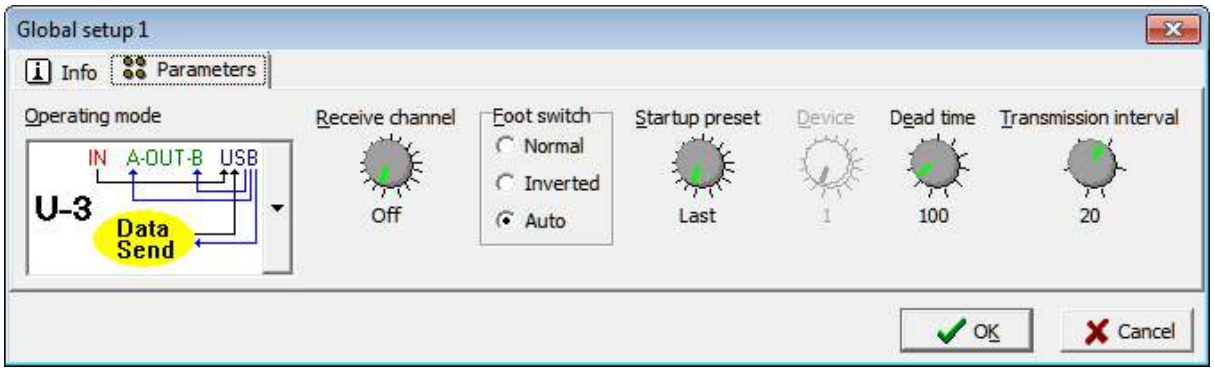
Pastes the global setup clipboard to the selected global setup(s).

Edit ➜ Initialize:

Initializes the selected global setup(s).

Edit ➜ Edit:

Opens a dialog box in which you can edit the parameters of the selected global setup. If you have selected more than one global setup, a *sequence* of dialog boxes will be offered: one for each global setup.



You should be very careful with the 'Operating mode' parameter: when a BCF/BCR receives this parameter via MIDI, it updates its connection setup, which can invalidate the connection with BC Manager when you restart the BCF/BCR! The other parameters can be changed with impunity.

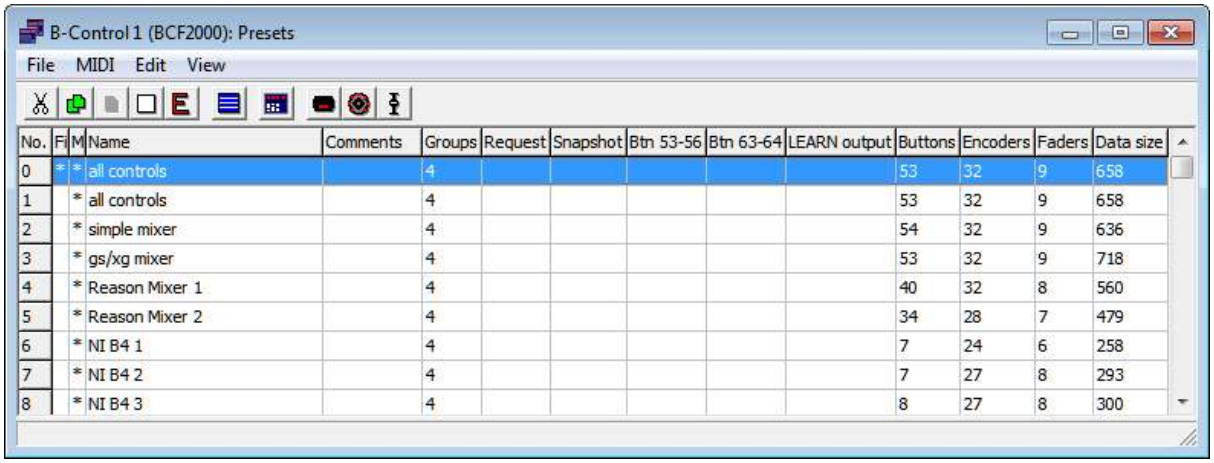
Note: the 'Device' parameter shows the device ID *received* from a BC or *read* from an old-style or third-party syx/txt/bc2 file; however, to avoid mid-stream communication loss, the value displayed here is never *sent* (or *saved*, for that matter), and accordingly you can't edit it. BC Manager only uses the 'Device' value defined in the 'MIDI options' dialog box (accessible from the B-Controls window via MIDI ➜ Options) for all its device ID 'fingerprints' in communication with the BC and in syx files.

Edit ➜ Select all:

Selects all global setups.

12. The presets window

A presets window displays a table containing the main features of a B-Control's presets. A presets window is also the gateway to miscellaneous windows detailing a preset's buttons, encoders and faders. (See *BCMI* for the details of all preset and element parameters.)



No.	F	M	Name	Comments	Groups	Request	Snapshot	Btn 53-56	Btn 63-64	LEARN output	Buttons	Encoders	Faders	Data size
0	*	*	all controls		4						53	32	9	658
1		*	all controls		4						53	32	9	658
2		*	simple mixer		4						54	32	9	636
3		*	gs/xg mixer		4						53	32	9	718
4		*	Reason Mixer 1		4						40	32	8	560
5		*	Reason Mixer 2		4						34	28	7	479
6		*	NI B4 1		4						7	24	6	258
7		*	NI B4 2		4						7	27	8	293
8		*	NI B4 3		4						8	27	8	300

The columns have the following meaning:

- No.:
  This is the preset number. 0 refers to the temporary preset, 1-32 to the memory presets.
  Note: the BCF and BCR use the temporary preset as a gateway to the memory presets. That is: it is only possible to change a memory preset on a BC via MIDI by sending the memory preset's data as the *temporary* preset and then sending a command that orders the BC to store its temporary preset as a particular memory preset. Obviously, as a result of this operation the original temporary preset data on the BC has been lost. Thus, it is rather tricky to keep the temporary preset synchronized. BC Manager does try to achieve this by resending the original temporary preset after sending one or more memory presets. However, this ploy only succeeds if the original temporary preset data on the actual BCF or BCR matches the temporary preset data maintained by BC Manager. This will not necessarily be the case; e.g. when you manually switch to another preset on the BC via the Preset ←/→ buttons, the BC's temporary preset data gets set to the data of the memory preset concerned. The bottom line: don't assume that the temporary preset's data as maintained by BC Manager automatically matches the temporary preset on the actual BC.

- File/MIDI:
  Asterisks in these columns indicate that BC Manager's representation of the preset has changed with respect to its associated file or the BC connected via MIDI. This concerns the preset's general settings, but also its elements: buttons, encoders and faders. See §10 for further explanation.

- Name:
  The preset's name, editable via the Info tab of the preset dialog box.

- Comments:
  A text that is editable via the Info tab of the preset dialog box.

- Groups/Request/Snapshot/Btn 53-56/Btn 64-64:
  Parameters that are editable via the Parameters tab of the preset dialog box.

- LEARN output:
  The lines of bytes defined via the 'LEARN output' tab of the preset dialog box.

- Buttons/Encoders/Faders:
  The number of currently active buttons/encoders/faders. This counts all the elements that produce standard or custom output.

- Data size:
  Memory presets (unlike the *temporary* preset) are restricted in complexity. Each button, encoder or fader defining standard or custom output takes up space, and so does the preset's LEARN output definition. On the BCF 4956 bytes are available for each memory preset, on the BCR 4344 bytes.

  This column shows the current number of bytes occupied by the preset. Once this number exceeds the available space, an asterisk is included. (However, you aren't barred from sending such a preset to the BCF/BCR; this may change in a future version of BC Manager.)

The menu provides the following operations:

File ➜ Export selected preset(s):
Opens a dialog box in which you can export the selected preset(s) to a syx, txt or bc2 file.

Note that if the selected range includes preset 0 (the 'temporary' preset), this preset is always put at the *end* of the file: this is necessary because of BCL's **`$store`** protocol.

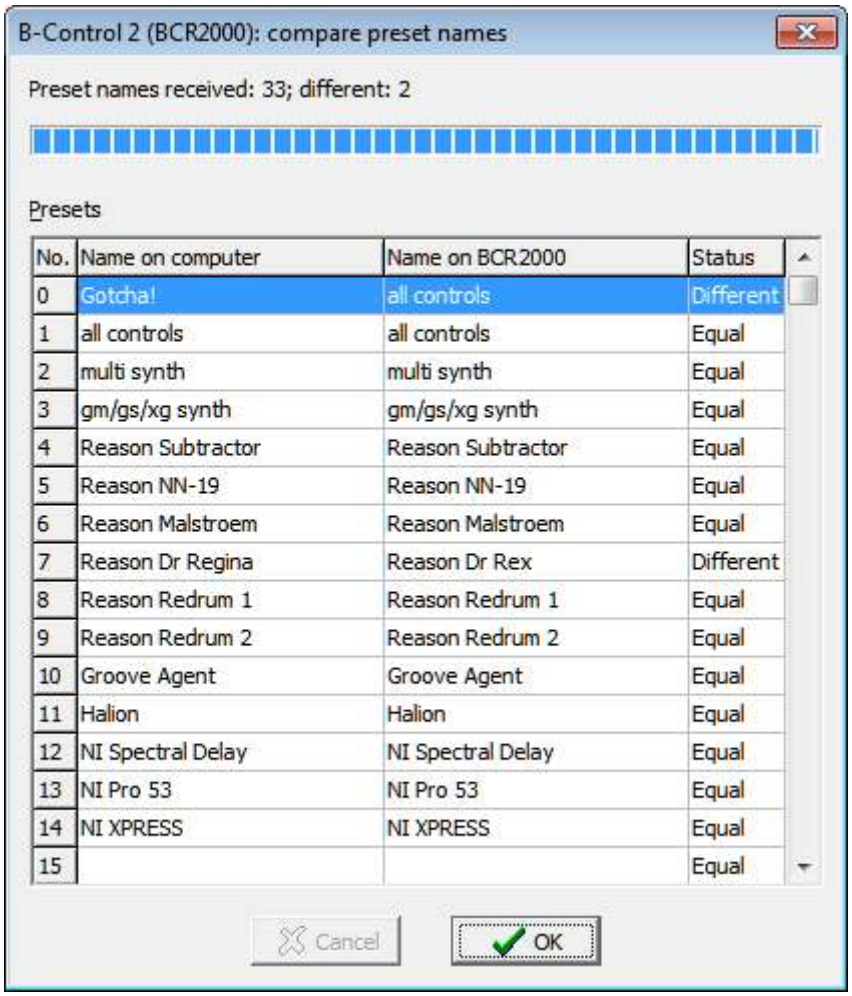
File ➜ Export table:
Opens a dialog box in which you can export the table of preset parameters to a specially formatted 'txt' file. You could use this as an input file to create a table in an external word processor.

File ➜ Export window image:
Opens a dialog box in which you can export an image of the presets window to a bitmap ('bmp') file.

File ➜ Print window image:
Prints an image of the presets window.

MIDI ➜ Receive:
Sends a request to the connected BCF or BCR for sending the selected preset(s) to BC Manager.

MIDI ➜ Compare preset names:
Opens a dialog box in which each preset name maintained by BC Manager is compared with its counterpart on the connected BCF or BCR. A table shows the names as maintained by BC Manager and those found on the connected BCF or BCR, and how the counterparts compare:



Note that the preset names (let alone complete presets) do not get *synchronized*: it's your own responsibility to subsequently achieve this (in whatever direction). The only *lasting* effect from the comparison concerns the MIDI column in the presets window: for your convenience, presets with different names receive an asterisk, indicating (as usual) a mismatch between BC Manager and the connected BC.

Beware: even when two counterpart names are reported equal, their presets as a whole may still differ! This is why BC Manager doesn't give as much prominence to the preset names as Behringer's own BC-EDIT program: as explained elsewhere in this manual, you are advised to use the syx/txt/bc2 files as masters, and your actual BCF2000s and BCR2000s as slaves. In any case, if any pair of preset names *differs*, you should definitely synchronize the presets concerned in the direction that is appropriate at the time; however, in line with BC Manager's philosophy, the direction should usually be *from* the computer *to* the BCF or BCR.

MIDI ➜ Send:

Sends the selected presets to the connected BCF or BCR.

Edit ➜ Cut:

Copies the selected preset(s) to the preset clipboard and initializes the selected preset(s).

Edit ➜ Copy:

Copies the selected preset(s) to the preset clipboard.

Edit ➜ Paste:
Pastes the preset clipboard to the selected preset(s).

Edit ➜ Initialize:
Initializes the selected preset(s).

Edit ➜ Edit:
Opens a dialog box in which you can edit the general settings of the selected preset. If you have selected more than one preset, a *sequence* of dialog boxes will be offered: one for each preset.

Edit ➜ Set ➜ Show value (No/Yes)/Default (No/Yes/Value)/Channel/MMC Device:
Changes the selected parameter of *all* elements (buttons, encoders and faders) in the selected preset(s) to one particular value in one go. The 'No' and 'Yes' cases apply the changes without further ado, in the other cases a dialog box opens in which you can set the parameter value.

Note that in case of Channel or MMC Device you won't *see* any changes in the element windows for elements which currently don't define a standard output message that uses the parameter concerned. Also note that the individual element windows (q.v) offer many other Set operations.

Edit ➜ Select all:
Selects all presets.

Edit ➜ Copy to preset 0:
Copies the selected memory preset to preset 0, the 'temporary preset'. (Obviously this doesn't work when preset 0 *itself* is selected.)

Note that this may also update the temporary preset on the connected BCF or BCR, depending on the settings in the MIDI options dialog box (accessed via the B-Controls window).

View ➜ Layout:
Opens the layout window. See §14 for more information.

View ➜ Buttons/Encoders/Faders:
Opens the Buttons, Encoders or Faders window. See §15 for more information.

13. The preset dialog box

The preset dialog box is accessed via the presets window. It allows you to edit a preset's general settings (as opposed to the individual buttons, encoders and faders).

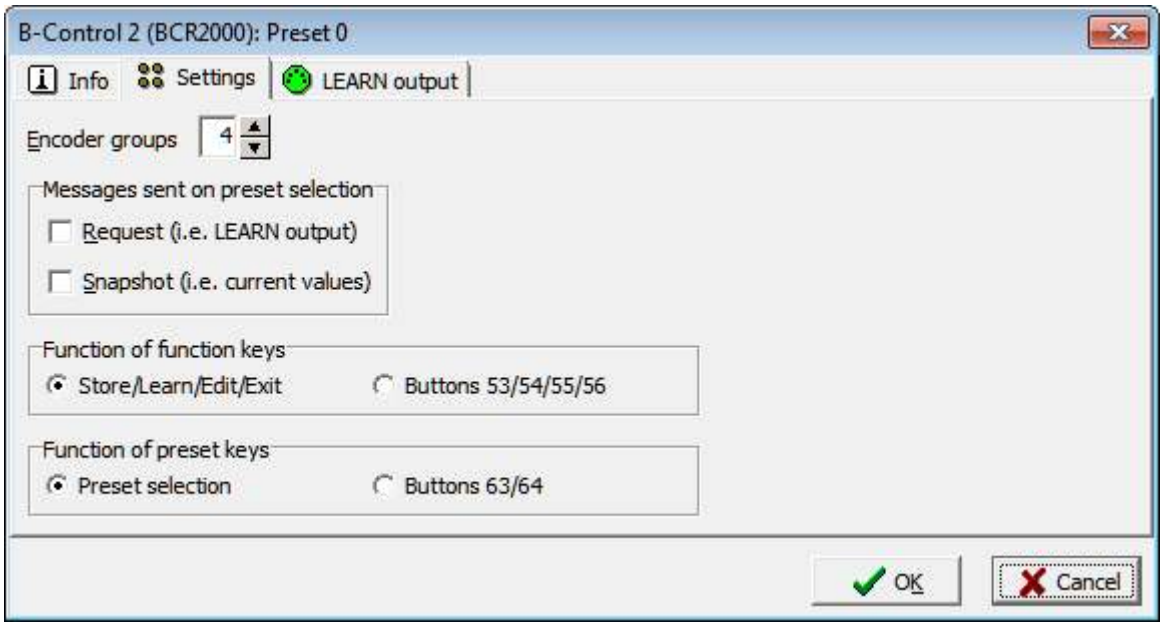
There are three tabs:

1. Info:

- Name:
  The preset's name.
  The BCF and BCR actually maintain this name, which is why the maximum length is 24 characters.
  Another restriction is that the inclusion of one or more apostrophes (') in the name is impossible in certain specific cases, because the BCF and BCR interpret preset names in a buggy way. What is and is not allowed is hard to explain in simple terms, but BC Manager makes the OK button of the dialog box inaccessible whenever the name is invalid; when this happens, simply remove the apostrophe(s) and the OK button becomes accessible again.

- Comments:
  A completely free text that can be of any length. Note that the BCF and BCR themselves don't store this text, so it can only be maintained via the *file* associated with the B-Control.

2. Settings:



- Encoder groups:
  A number from 1 to 4 indicating how many groups of button and encoder definitions associated with the 8 push-encoders are accessible on the BC. E.g. if this number is 1, only group 1 (i.e. buttons and encoders 1-8) is accessible, if it is 4, all 4 groups (i.e. buttons and encoders 1-32) are accessible.
  Note: irrespective of this value you can always edit any push-encoder definition in BC

Manager.

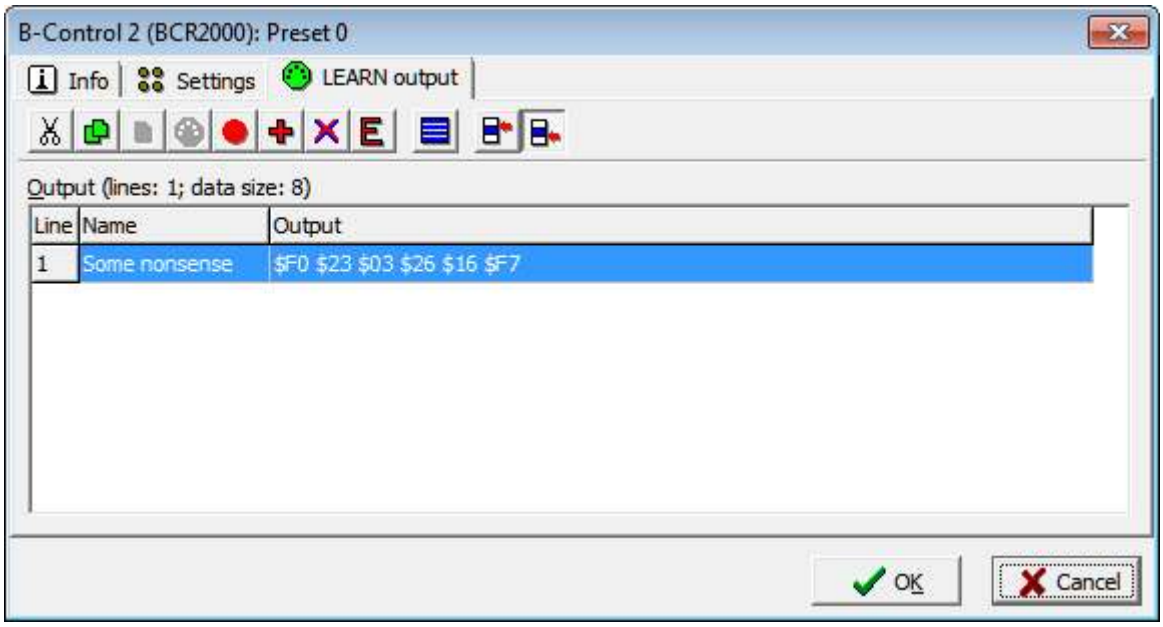
- Request (i.e. LEARN output):
  Determines whether the BC outputs the 'LEARN output' bytes upon preset selection.

- Snapshot (i.e. current values):
  Determines whether the BC outputs its current control values (i.e. the positions of its buttons, encoders and faders) upon preset selection. (If 'LEARN output' bytes are *also* sent upon preset selection, the 'LEARN output' bytes are sent *first*.)

- Function of function keys:
  Determines whether the four function keys function as STORE, LEARN, EDIT and EXIT or as the 'ordinary', assignable buttons 53-56. Note that the latter is automatically the case if the *preset keys* (see below) function as buttons 63-64.

- Function of preset keys:
  Determines whether the two preset keys decrease and increase the number of the currently selected preset, or function as the 'ordinary', assignable buttons 63 and 64. Note that if the latter is the case, the *function keys* (see above) always function as buttons 53-56: you then can't use them as STORE, LEARN, EDIT or EXIT. (Just to be clear on this point: this is not some sadistic feature of BC Manager; it's simply a restriction imposed by the BCF and BCR themselves.)

3. LEARN output:
Here you can define a sequence of 'LEARN output' lines, each containing a sequence of bytes:[3]



The BC outputs the bytes in these lines via MIDI whenever the user presses EDIT + LEARN, provided that the EDIT and LEARN keys are *not* used as buttons 55 and 54 respectively – cf. the 'Function of function keys' parameter (see above). The BC also outputs these bytes upon preset

[3] Each LEARN output line generates its own '**`.tx`**' BCL statement: this is possible because the BCF and BCR accept multiple '**`.tx`**' statements. See *BCMI* for more information.

selection, provided that the Request parameter (see above) is *on*.

See §17 for further details.

14. The layout windows

The layout window is accessed via the presets window. It provides a physically more or less accurate top view of the actual BCF2000 or BCR2000, and shows the selected preset's element labels:

The layout window's editing facilities are somewhat limited: it is primarily intended as an overview; the presets window and the element windows offer more advanced editing operations.

Color coding:
If a button or encoder is dysfunctional on the BCF/BCR due to the preset settings, its background in the layout window is pink. (Note that you can still *edit* these buttons and encoders.)

The following relationships hold between preset settings and element functionality:

- The number of encoder groups (1-4) determines whether the encoders and buttons in GROUP 2, GROUP 3 and GROUP 4 are enabled, and also whether the four keys on the ENCODER GROUPS panel function as encoder group selectors or as ordinary buttons 57-60. Note that there is an inverse relationship: the more encoder groups are enabled, the *fewer* keys on the ENCODER GROUPS panel function as normal buttons, because these have to be used for encoder group selection.
- The 'Function of function keys' setting determines whether the STORE, LEARN, EDIT and EXIT keys function as special function keys or as ordinary buttons 53-56. (Note that the

'Function of preset keys' setting affects 'Function of function keys'.)

- The 'Function of preset keys' setting determines whether the two PRESET keys function as preset selectors or as ordinary buttons 63 and 64.

Navigation:
Note that you can 'tab' from one group of labels to another by means of Tab (forward) and Shift+Tab (backward). You can also move directly to a particular group by pressing the character underlined in the group's caption (you don't even have to keep the Ctrl key pressed). Within a group you can use the arrow keys (and even Home, End, PgUp and PgDn).

Menu:
The menu provides the following operations:

File ➜ Export window image:
Exports an image of the layout window to a bitmap ('bmp') file of your choice.

This operation offers the same option of converting the window's blue and pink backgrounds to white as 'Print window image' does (see below), although in this case the default is *No*.

File ➜ Print window image:
Prints an image of the layout window.

Note that this operation *always* uses *Landscape* orientation for the printed page, irrespective of the setting made in BC Manager's main window via Options ➜ Printer.

You are asked whether you want to print the window's blue and pink backgrounds as white to save ink:

- If you answer *No*, the window is printed in full color, exactly as it is showing on the computer screen. (If you've got a black & white printer, the result is of course in gray scale.)
- If you answer *Yes* (the default), a few other optimizations are made too: the preset number LEDs and a few other non-essential things are left out. All in all you should save quite some ink, no matter what type of printer you have. (The status bar, however, is still printed as it is showing on the computer screen, i.e. in full color or gray scale.)

File ➜ Print strips:
Prints a page containing strips containing the preset's element labels.

Note that this operation *always* uses *Landscape* orientation for the printed page, irrespective of the setting made in BC Manager's main window via Options ➜ Printer.

The sizes of the printed strips are such that the strips should fit on your BCF or BCR if you cut them out. Obviously you don't want to glue them to the device, so maybe the best thing to do is to sellotape small vertical holding strips to the BCF or BCR, then simply slide the strips printed here underneath these holding strips.

Edit ➜ Cut:
Copies the selected (highlighted) element to the pertinent element clipboard and initializes the selected element.

Tip: you can *move* an element definition to another element directly via mouse-dragging: this usually works faster than cutting and pasting. See the bottom of this section.

Edit ➜ Copy:
Copies the selected (highlighted) element to the pertinent element clipboard.

Edit ➜ Paste:

Pastes the first element on the pertinent element clipboard to the selected (highlighted) element.

Note that 'Paste' always takes data from the 'native' clipboard for the currently selected element. So for instance if you are currently on a button, you can thus only paste a button from the button clipboard. By contrast, 'Paste button/encoder/fader' (see below) always concerns 'foreign' clipboards.

Edit ➜ Paste button/encoder/fader:

Pastes the first element on the indicated 'foreign' element clipboard to the selected (highlighted) element. So for instance if you are currently on a button, you can thus paste an encoder from the encoder clipboard or a fader from the fader clipboard. Note that parameters in the target element not contained in the source element on the clipboard are set to default values.

Edit ➜ Initialize:

Initializes the selected (highlighted) element.

Edit ➜ Edit:

Opens a dialog box in which you can edit the selected (highlighted) element.

Edit ➜ Preset:

Opens a dialog box in which you can edit the general settings of the selected preset.

View ➜ Next preset:

Selects the next preset. Equivalent to pressing the upward button next to the LEDs showing the preset number.

View ➜ Previous preset:

Selects the previous preset. Equivalent to pressing the downward button next to the LEDs showing the preset number.

View ➜ Element labels:

This setting determines the nature of the element labels, i.e. what is displayed in the element boxes. There are three choices:

- Name, else output:
  This is the default. The element's name is displayed, except if the name is empty, in which case a summary of the element's standard and/or custom output is displayed (usually this includes the Channel, Type and main parameter value).
- Name:
  The element's name is displayed. If the name is empty, nothing is displayed.
- Output:
  A summary of the element's standard and/or custom output is displayed. If no standard or custom output is defined, nothing is displayed.

Note: regardless of this setting, the status bar at the bottom of the window always shows the selected element's name and output summary.

View ➜ Presets:

Opens the presets window for the B-Control to which the layout window belongs. See §12 for more information.

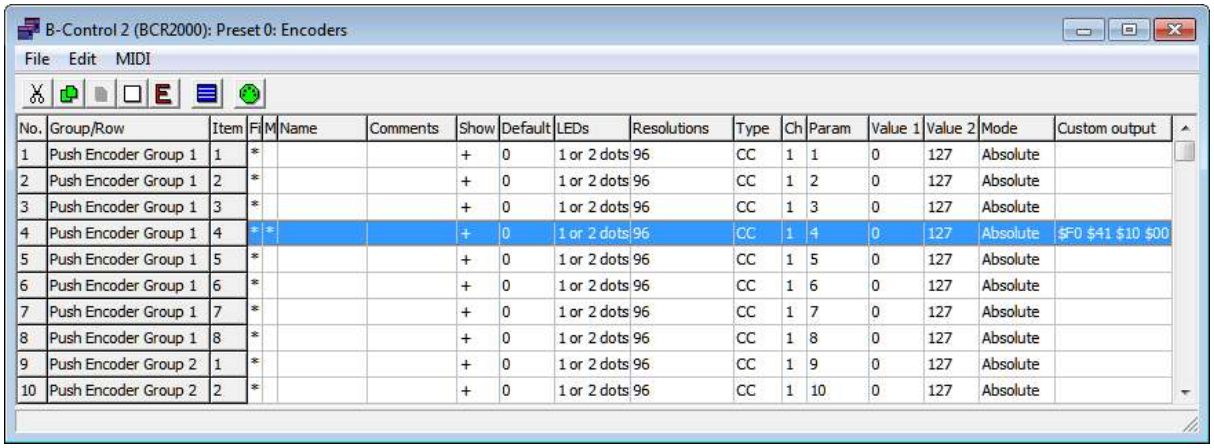
View ➜ Buttons/Encoders/Faders:

Opens the Buttons, Encoders or Faders window of the B-Control to which the layout window

belongs. See §15 for more information.

15. The element windows

An element window displays a table containing the main features of a preset's buttons, encoders or faders:



No.	Group/Row	Item	Fi	M	Name	Comments	Show	Default	LEDs	Resolutions	Type	Ch	Param	Value 1	Value 2	Mode	Custom output
1	Push Encoder Group 1	1	*				+	0	1 or 2 dots	96	CC	1	1	0	127	Absolute	
2	Push Encoder Group 1	2	*				+	0	1 or 2 dots	96	CC	1	2	0	127	Absolute	
3	Push Encoder Group 1	3	*				+	0	1 or 2 dots	96	CC	1	3	0	127	Absolute	
4	Push Encoder Group 1	4	*	*			+	0	1 or 2 dots	96	CC	1	4	0	127	Absolute	$F0 $41 $10 $00
5	Push Encoder Group 1	5	*				+	0	1 or 2 dots	96	CC	1	5	0	127	Absolute	
6	Push Encoder Group 1	6	*				+	0	1 or 2 dots	96	CC	1	6	0	127	Absolute	
7	Push Encoder Group 1	7	*				+	0	1 or 2 dots	96	CC	1	7	0	127	Absolute	
8	Push Encoder Group 1	8	*				+	0	1 or 2 dots	96	CC	1	8	0	127	Absolute	
9	Push Encoder Group 2	1	*				+	0	1 or 2 dots	96	CC	1	9	0	127	Absolute	
10	Push Encoder Group 2	2	*				+	0	1 or 2 dots	96	CC	1	10	0	127	Absolute	

The table columns have the following meaning:

- No.:
  The element number. For buttons this number runs from 1 to 64, for BCF encoders from 1 to 32, for BCR encoders from 1 to 56, and for faders (BCF only) from 1 to 9 (where fader 9 is actually the Foot Controller).

- Group/Row:
  The general location on the BC.

- Item:
  The specific location within the general location on the BC. Usually a column number.

- File/MIDI:
  Asterisks in these columns indicate that the representation of the element (as maintained by BC Manager) has changed with respect to its associated file or the BC connected via MIDI. (See §10 for further explanation.)

- Name:
  The element's name, editable via the Info tab of the element's dialog box.

- Comments:
  A text that is editable via the Info tab of the element's dialog box.

- *Other columns*:
  These are all specific element parameters, editable via the element's dialog box.

The menu provides the following operations:

File ➜ Export table:
Opens a dialog box in which you can export the table of the element parameters to a specially

formatted ‘txt’ file. You could use this as an input file to create a table in an external word processor.

File ➜ Export window image:
Opens a dialog box in which you can export an image of the element window to a bitmap (‘bmp’) file.

File ➜ Print window image:
Prints an image of the element window.

Edit ➜ Cut:
Copies the selected element(s) to the pertinent element clipboard and initializes the selected element(s).

Edit ➜ Copy:
Copies the selected element(s) to the pertinent element clipboard.

Edit ➜ Paste:
Pastes the pertinent element clipboard to the selected element(s).

Note that ‘Paste’ always takes data from the ‘native’ clipboard. So for instance in a button window, you can thus only paste buttons from the button clipboard. By contrast, ‘Paste button(s)/encoder(s)/fader(s)’ (see below) always concerns ‘foreign’ clipboards.

Edit ➜ Paste button(s)/encoder(s)/fader(s):
Pastes the indicated ‘foreign’ element clipboard to the selected element(s). So for instance in a button window, you can thus paste encoders from the encoder clipboard or faders from the fader clipboard. Note that parameters in the target elements not contained in the source elements on the clipboard are set to default values.

Edit ➜ Initialize:
Initializes the selected element(s).

Edit ➜ Create simple ➜ Program Change/Control Change:
These operations allow you to quickly set up an element for a ‘standard’ Program Change and Control Change message. You are asked to define the MIDI channel and the program or controller number, and BC Manager automatically sets up the *whole* element according to what BC Manager considers the appropriate settings.

These operations also work very nicely if you have selected more than one element: you can then also define increments for both channel and program/controller number per element.

Note: the difference between ‘Create simple’ and ‘Set’ (see below) is that ‘Create simple’ sets up *whole* elements, whereas ‘Set’ only affects the *selected parameter* of the selected elements.

Edit ➜ Edit:
Opens a dialog box in which you can edit the parameters of the selected element. If you have selected more than one element, a *sequence* of dialog boxes will be offered: one for each element. See §16 for more information on element dialog boxes.

Edit ➜ Set ➜ *Various parameters*:
Changes the selected parameter of *all* the selected elements to one particular value in one go. **The trick is to *first* select more than one element, *then* execute the Set operation.**

The ‘No’ and ‘Yes’ cases apply the changes without further ado, in the other cases a dialog box opens in which you can set the parameter value. If you have selected more than one element, some of these dialog boxes also contain a ‘Step per row’ knob: this determines by which amount the selected parameter increases or decreases for each next element.

Note that in many cases you won’t *see* any changes for elements which currently don’t define a standard output message that uses the parameter concerned. Also note that the presets window (q.v.) offers even wider-ranging Set operations for several parameters.

Edit ➜ Select all:

Selects all buttons, encoders or faders.

MIDI ➜ Send:

Sends the selected element(s) to the connected BCF or BCR. This operation can only be executed for preset 0 (the temporary preset); this restriction is simply a consequence of the BCL protocol: it is impossible to send individual elements of *memory* presets to a BC.

16. The element dialog boxes

An element dialog box is accessed via the corresponding element window. It allows you to edit all the settings of a single button, encoder or fader. **For in-depth discussion of these settings, see *BCMI*.**

An element dialog box contains four tabs:

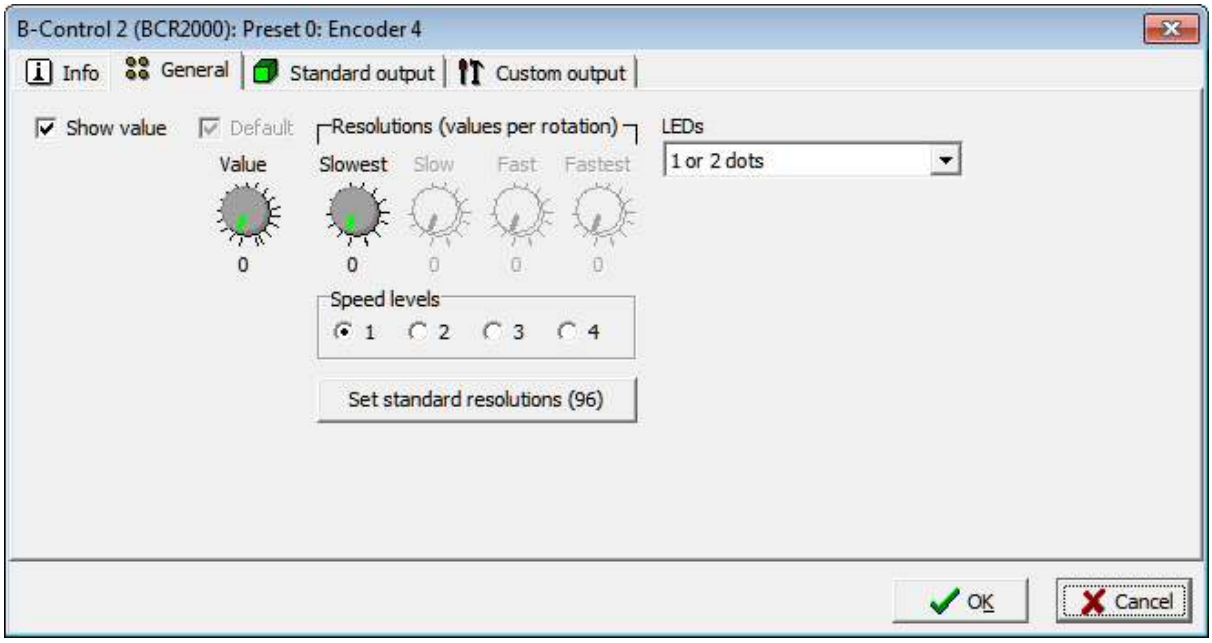
1. Info:

- Name:
  The element's name, which can be of any length.

- Comments:
  A completely free text that can be of any length.

Beware: the BCF and BCR themselves don't store Name or Comments, so these fields can only be maintained via the *file* associated with the B-Control.

2. General:



- Show value:
  Determines whether moving the element results in new values being shown in the BC's display.

- Default and Value:
  Respectively determine whether the element has a default value and what that value is. (See *BCMI* for discussion of these settings.)

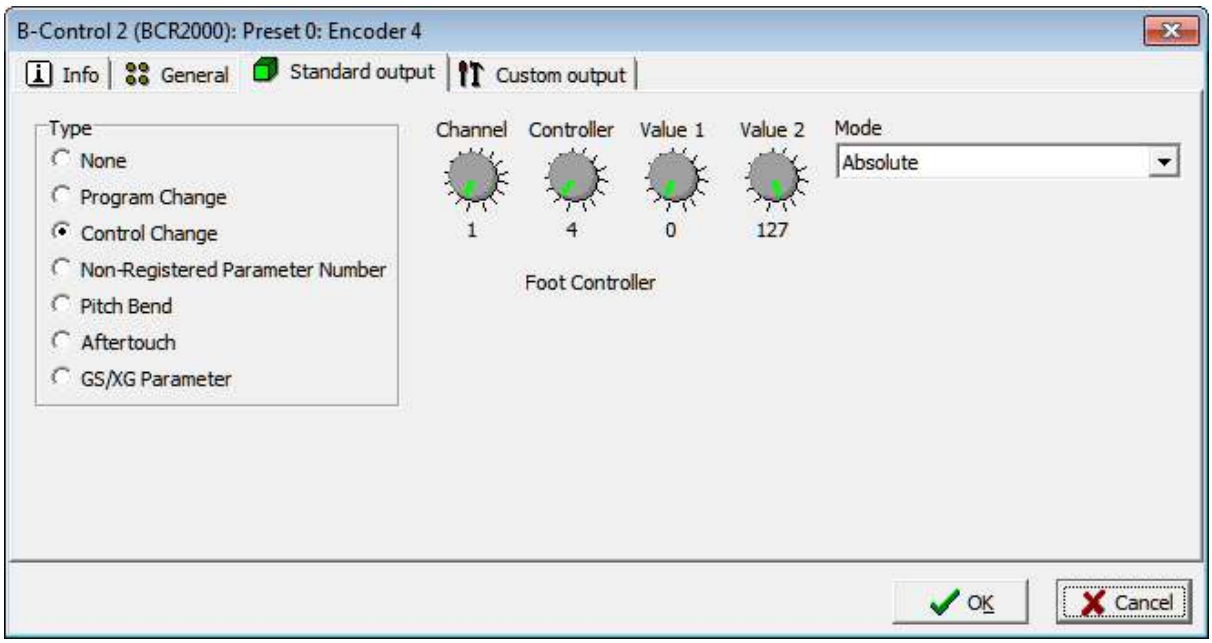
- Resolutions [encoder only]:
  Determines the encoder's responsiveness. You can specify from one to four values, each applying to a particular rotational speed level.

- LEDs [encoder only]:
  Determines the way the encoder's LEDs reflect the encoder's current value. (Note: in BCL this setting is called '**`.mode`**'.)

- Sync [fader only]:
  Determines the way the fader physically moves after value changes sent to the BCF via MIDI. (Note: in BCL this setting is represented by the '**`.motor`**' and '**`.override`**' settings.)

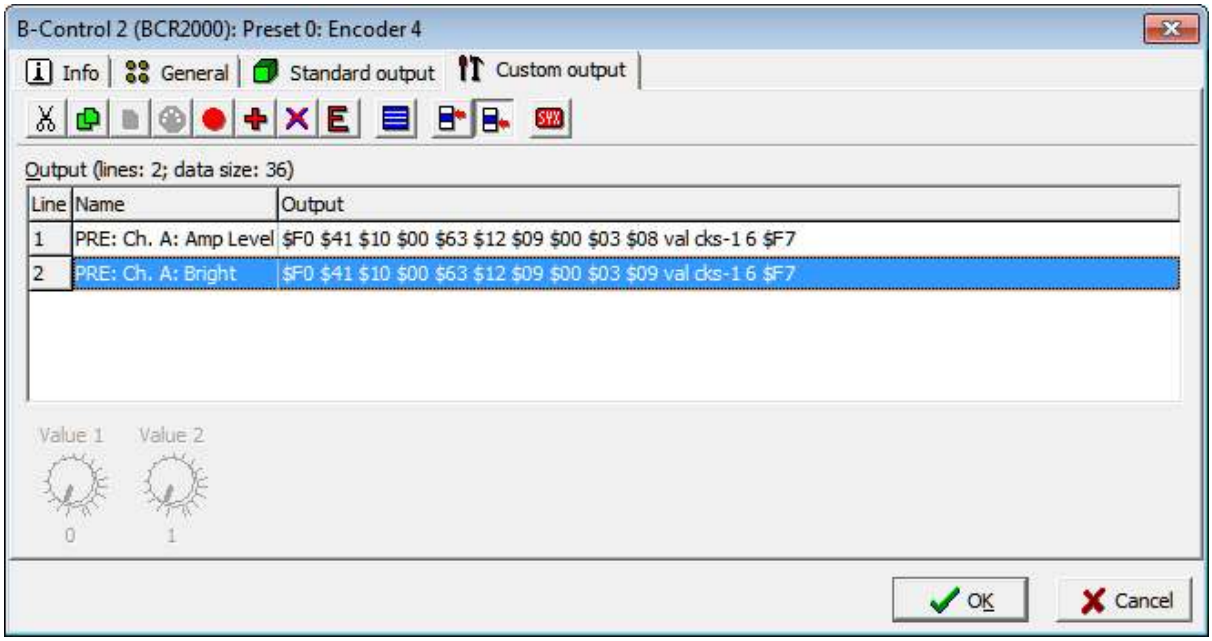
- Override by button [fader only]:
  Determines the button you can keep pressed to temporarily disable the fader's motor.

3. Standard output:



This tab contains all parameters related to a 'standard' MIDI output message that the element can output, such as Program Change and Control Change. These are the MIDI messages listed in Behringer's manual for the BCF2000 and BCR2000. (Note: in BCL these messages fall under the '**`.easypar`**' category.)

4. Custom output:



Here you can assign a sequence of 'custom output' lines to the button, encoder or fader. Each custom output line contains a sequence of bytes, possibly also certain special identifiers. (Note: in BCL these custom output lines fall under the '**`.tx`**' category.)

These custom output lines can be edited in exactly the same way as the LEARN output messages that can be assigned to a *preset*. See §17 for details.

Note that it is possible to assign *both* a standard output message *and* one or more custom output messages; in this case the element outputs the *standard* message first, then the custom messages.

The Value 1 and Value 2 parameters shown on the custom output tab allow you to set the minimum and maximum values of the element. Note that Value 1 and Value 2 only apply if the *standard* MIDI output message's Type parameter has been set to 'None': if any actual standard message has been assigned, *that* message's minimum and maximum values prevail.

The same goes for the Mode parameter on the custom output tab for buttons: it is superseded by any Mode parameter setting on the standard output tab.

17. Editing LEARN and custom output

A preset's 'LEARN output' and an element's 'custom output' can be edited in nearly identical ways. The only difference is that LEARN output messages can only consist of fixed byte values, whereas custom output messages may also contain special identifiers, such as references to the element's value.

In principle the *total* amount of LEARN or custom output bytes that you can define is 125, but this number decreases by 2 for each additional LEARN or custom output *line* you use (in other words: the definition of the line itself takes up space too), and the special identifiers that can be used in custom output also decrease this amount in specific ways (see *BCMI*).

Thus, it might seem best to simply put all the data in a single line, but this is not the case: **due to a bug in the BCF and BCR's firmware, the BCF and BCR generate additional, spurious MIDI output whenever a LEARN or custom output line contains a System Exclusive message ($F0 ... $F7) followed by any other byte**. So to avoid spurious output, any MIDI *System Exclusive* message should be in line-final position. And for clarity and editability you might as well put *all* MIDI messages (not just the System Exclusive ones) on separate lines (unless you're running out of space, of course).

BC Manager also allows you to define a name for each LEARN or custom output line. This name cannot be stored in the BCF or BCR itself, but only in the syx, txt or bc2 file, so it is merely for your own reference.

You can add and edit LEARN and custom output lines via the toolbuttons at the top of the LEARN and custom output tabs in the preset and element dialog boxes, respectively. Alternatively you can use keystrokes (but only if the *output table* has input focus!); these keystrokes are also specified in these pop-up hints, and are repeated here for convenience:

Description	Key(s)/mouse click
Cut selected line(s) to clipboard	Shift+Del
Copy selected line(s) to clipboard	Ctrl+Ins
Paste clipboard	Shift+Ins
Paste MIDI message clipboard	Ctrl+Shift+Ins
Record MIDI messages	Alt+R
Insert new line	Ins
Delete selected line(s)	Ctrl+Del
Edit properties of selected line(s)	Alt+Enter or double-click
Select all lines	Ctrl+A
Paste/insert before current line	Alt+B
Paste/insert after current line	Alt+A
Insert SysEx parameter [custom output only]	Alt+S

Note that ‘Paste/insert before current line’ and ‘Paste/insert after current line’ are mutually exclusive *settings* (rather than data-changing actions): they affect the position of subsequent paste and insert operations, relative to the current (highlighted) line.

‘Paste MIDI message clipboard’ inserts a single line containing the bytes currently stored in the ‘MIDI byte clipboard’. You can put MIDI messages on this clipboard from the ‘MIDI input messages’ window and the ‘MIDI System Exclusive messages’ window.

‘Record MIDI messages’ directly records MIDI messages from *any* enabled MIDI input device (so not just from the one linked to the B-Control).

‘Insert new line’ and ‘Edit properties...’ show a dialog box in which you can edit an individual LEARN or custom output line, including the name and the sequence of individual output items. Individual output items can be edited via a *further* dialog box.

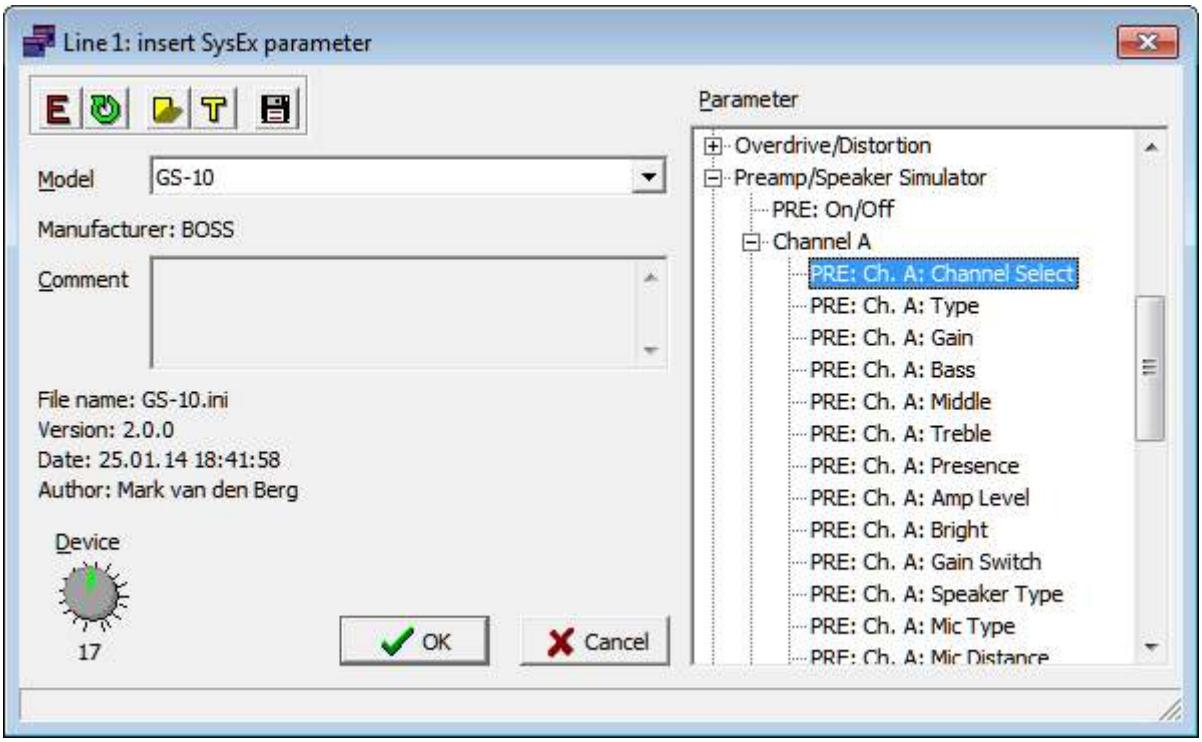
A special case is the ‘Insert SysEx parameter’ button on the ‘Custom output’ tab of the element dialog boxes. Pressing this button opens a dialog box in which you can use BC Manager’s SysEx preprocessor, providing as much ‘user comfort’ as possible. See §18 for further discussion.

The LEARN/custom output tab informs you about various problems in your current LEARN/custom output definition:

Type	Message
Hint	Orphan End-Of-Exclusive ($F7)
Warning	Non-standard termination of SysEx message
	Byte after SysEx message causes spurious output
	Checksum in non-SysEx message
	Multiple ntimes (only the last one works)
Error	Data byte expected
	Status byte expected
	Invalid status byte
	Checksum starting index too high
	Incomplete MIDI message
	Invalid ntimes loop

18. The Insert SysEx parameter dialog box

The ‘Insert SysEx parameter’ dialog box (accessible via the ‘Custom output’ tab of the element dialog boxes) provides a simple way of creating SysEx messages for certain parameters of certain MIDI devices:



Basically you only have to select the MIDI device from the Model listbox and the parameter from the Parameter listbox.

You must also ensure that the value for ‘Device’ is correct: this should be the value for System Exclusive communication selected on the *receiving* MIDI device. (It has *nothing* to do with any Device ID setting associated with the *controlling* BCF or BCR.)

If you then press ‘OK’, BC Manager’s SysEx preprocessor inserts the appropriate MIDI System Exclusive message as a custom output line in the BC element’s definition.

Support for each particular model is provided via a special SysEx model definition file. Thus, BC Manager’s SysEx preprocessor takes away most of the trouble of defining SysEx messages: the person writing the SysEx model definition file only has to provide a list of parameter addresses, ranges and names, and the end users can simply assign these parameters to their BC elements *by name*, having to know nothing about any technical details.

The BC Manager package already includes several SysEx model definition files, and you are encouraged to write definition files for other models and upload these to the file section of the Mountain Utilities web site: http://mountainutilities.eu/userfiles/b-control.

Each model appearing in the ‘Insert SysEx parameter’ dialog box has an associated SysEx model definition file with the extension ‘.ini’ in your Models folder. Under Windows XP this folder typically resides under C:\Documents and Settings\ *Username*\Application Data\Mountain Utilities\BC Manager, under Vista and later typically under C:\Users*Username*\AppData\Local\Mountain Utilities\BC Manager.

Tip: the subfolder Templates of the folder to which BC Manager was installed (typically, on a 32-bit system C:\Program Files\Mountain Utilities\BC Manager, on a 64-bit system C:\Program Files (x86)\Mountain Utilities\BC Manager) contains the files BOSS.ini, Roland.ini, Waldorf.ini and Yamaha.ini; New.ini is for any other manufacturer. You can use these templates as starting points for your own SysEx model definition files.

The 'Insert SysEx parameter' dialog box contains several buttons which help you create or edit SysEx model definition files:

- Edit definition file:
  Opens the SysEx model definition file for the selected model in the external editor associated with '.ini' files (e.g. Notepad).

- Reload definition files:
  Reloads all model definition files into BC Manager, so that the Model and Parameter boxes reflect any changes. You must do this when you have changed one or more definition files behind BC Manager's back, i.e. while it has been running. Upon reloading, BC Manager will inform you if there is any syntactic error in one of these files.

- Open definition folder:
  Opens the folder containing the model files in Windows Explorer.

- Open template folder:
  Opens the folder containing the template files in Windows Explorer.

- Create report:
  Saves a CSV file containing a table of all the parameters in the selected definition file, including each parameter's actual BCL line that it generates (for the Device value selected in the Device knob).

  You should be able to import the CSV file in any 'good' spreadsheet application. It may help to know that the file's field separator is a comma and its field delimiter the doublequote character ("). Beware: you should *not* use Quattro Pro X5 to open these files, because it interprets and reformats any 'bare' hexadecimal as a decimal number if it can (in particular this may ruin the 'Min H' and 'Max H' columns); OpenOffice Calc 3.3 works fine though; I don't know about Excel.

SyxEx model definition file format

A SysEx model definition file must be in ASCII/ANSI or UTF-8 format. (Tip: in Notepad you can save to a UTF-8 file via File ➜ Save As ➜ Encoding ➜ UTF-8.)

A SysEx model definition file must consist of a header, followed by any number of parameter definition lines (optionally within subsections). After the header, empty lines are allowed everywhere, and so are comment lines, i.e. lines starting with a semicolon (;).

1. Header

The header must start with the following line:
[MODEL]

After that there must be a sequence of *Name=Value* lines, in the exact order as shown in the table below. (*n* stands for a normal decimal number, *h* for a bare hexadecimal in the range 00-7F.)

Name	Value	Description
ProgramName	BC Manager	Identifies the application
ProgramVersion	1.0.0 *or* 2.3.0 *or* 2.4.0 *or* 2.4.2 *or* 2.5.0	The minimal BC Manager version required to parse this definition file.
ManufacturerName	*character string*	The name of the model's manufacturer; can be anything
ManufacturerID	*h [h ...]*	One or more bytes, specifying the Manufacturer ID to be included in the SysEx messages.
ModelName	*character string*	The name of the model; can be anything
ModelID	*[h [h ...]]*	The sequence of 'model' bytes
DevicePosition	AfterModel *or* BeforeModel *or* WithCommand	Specifies where the Device ID is output. WithCommand is for the Yamaha protocol: the Device ID forms the low nibble of the command byte.
MinDevice	*h*	The Device ID's minimum. Must be a single hexadecimal digit if DevicePosition is WithCommand.
MaxDevice	*h*	The Device ID's maximum. Must be a single hexadecimal digit if DevicePosition is WithCommand.
DefaultDevice	*h*	The Device ID's default. Must be a single hexadecimal digit if DevicePosition is WithCommand.
Command	*[h [h ...]]*	The sequence of command bytes. Must be a single hexadecimal digit if DevicePos is WithCommand.
AddressLength	*n*	Range: 1-8

ChecksumMethod	*[n]*	*n*'s range is 1-3, corresponding with BCL's '**`cks-1`**', '**`cks-2`**' and '**`cks-3`**'. If this is left blank, *no* checksum is used.
ChecksumStart	*[n]*	The index of the first byte included in the checksum; the starting F0 has index 0. If ChecksumMethod is left blank, this must be left blank too.
FileVersion	*n.n.n*	This definition file's version
FileAuthor	*character string*	The name of this definition file's author (could be you...!)
Comment	*character string*	Displayed in the 'Insert SysEx parameter ' dialog box

Notes:

- All string literals in the header are case-insensitive.
- Empty lines are not allowed in the header.
- BC Manager 2.3.0 and higher can still read SysEx model definition files in the oldest format (with 'ProgramVersion=1.0.0'), but to avoid confusion it is recommended that you upgrade ProgramVersion to 2.3.0 or higher.
- In BC Manager 2.3.0-2.3.2, ManufacturerID could only be a single byte. Multiple bytes are only supported by 2.4.0 and higher. Accordingly, if ManufacturerID consists of multiple bytes, ProgramVersion must be set to 2.4.0 or higher.
- The ModelID field can be left blank: this is appropriate for certain old Yamaha synths that don't use a fixed Model ID as the fourth byte of a SysEx message, but a variable 'group' number, which is more easily represented as the most significant byte of the *address*; see for example the definition file for the TX81Z (included in the BC Manager package). On the other hand, for Roland/BOSS and Waldorf devices ModelID should *never* be blank (although BC Manager doesn't warn you if it is).
- The Command field can specify any byte sequence (including an empty one). Since this sequence is output immediately before each parameter's address byte(s), you can 'cheat' concerning the command and parameter address bytes: If different parameters in the definition file need different command bytes, you obviously can't define a common command byte; instead you should leave Command blank and start each individual parameter address with the desired command byte. On the other hand, if all parameter addresses start with the same byte(s), you could add this byte or these bytes to Command.
- Before BC Manager 2.4.2, AddressLength's range was 1-4, so definition files using 5-8 bytes must set ProgramVersion to 2.4.2 or higher.
- All that BC Manager does with FileVersion, FileAuthor and Comment is display them in the 'Insert SysEx parameter' dialog box. So in theory FileVersion can be in any format you like, but the suggested format (*n.n.n*) is recommended to ensure future compatibility. It's a good idea to increase the FileVersion value whenever you make a new version of the file.
- Parameter subsections (discussed below) were introduced in BC Manager 2.5.0, so any SysEx model definition file using these must set ProgramVersion to 2.5.0.

2. The parameter section

The parameter section follows the header and consists of any number of parameter definitions, optionally within subsections:

a. Parameter definition

Each parameter definition is a line having the following format:

```
ParameterAddress | MinValue - MaxValue | Data ;ParameterName
```

Note that the verticules, the hyphen and the semicolon are obligatory literals here.

ParameterAddress:
A sequence of bare hexadecimal bytes in the range of 00-7F (separated by spaces) indicating the parameter's address. The number of bytes in this sequence must be equal to the AddressLength value specified in the file header.

MinValue:
One or two bare hexadecimal bytes in the range of 00-7F indicating the parameter's minimum value. If two bytes (separated by a space) are specified, these bytes constitute an MSB-first 14-bit value, i.e. with bits 0-6 of the first byte being bits 7-13 of the resulting value, and bits 0-6 of the second byte being bits 0-6 of the resulting value.

When the user inserts the custom output line, BC Manager copies *MinValue* to the Min knob on the 'Custom output' tab of the element dialog boxes.

MaxValue:
One or two bare hexadecimal bytes in the range of 00-7F indicating the parameter's maximum value. Similar semantics as *MinValue*.

Data:
A string of characters in BCL custom output (**`.tx`**) format. BC Manager inserts this string in the generated custom output line at the position where the parameter value is expected, so normally this string is just val (for a 1-byte parameter) or val7.13 val0.6 (for a 2-byte parameter), but in fact you can put anything here that leads to a valid custom output line. E.g. in the file XG.ini (included in the BC Manager package), *Data* is val12.13 val8.11 val4.7 val0.3 for a particular parameter. You're not even restricted to 'val'-type identifiers: fixed byte values and any other special identifiers are allowed too; see *BCMI* section 14.6. So in theory you can do weird things here, but remember that whatever you enter here will be embedded within the custom output line *as is*. In particular, you should be aware that fixed *hexadecimal* bytes must be prefixed by '**$**' and bytes *without* '**$**' are interpreted as ordinary *decimal* numbers; this contrasts sharply with *ParameterAddress*, *MinValue* and *MaxValue*, which must be bare hexadecimals.

ParameterName:
The name of the parameter as it will appear in the 'Parameter' box of the 'Insert SysEx parameter' dialog box.

b. Subsection

- A subsection starts with a line consisting of an opening brace ('{') optionally followed by the subsection's name (which will be displayed in the tree in the `Insert SysEx parameter' dialog box). Spaces and tabs in front of { or between { and the start of the name are ignored.
- A subsection may contain any number of subsections and parameter definitions. (So you can nest subsections.)
- A subsection ends with a line consisting of a closing brace ('}'). Spaces and tabs in front of } and any characters after } are ignored. (So you can't close two nested subsections by '}}' on the same line: each closing brace must occur on a separate line.)

For how subsections work in practice, see D-5.ini and GS-10.ini in the BC Manager package.

Examples

On the following pages some typical SysEx formats of individual manufacturers are described. Please refer to the headers of the respective template files in the Templates subfolder of BC Manager's installation folder.

Roland/BOSS

The format of Roland/BOSS data messages looks like this:

Byte(s)	Meaning/comments
F0	Start of exclusive message
41	Manufacturer: Roland/BOSS
Device	Often 00-1F with default 10
Model	1 or more bytes
12	Command: DT1 (Send data one-way)
Address	Length depends on model
Data	Parameter value(s)
Checksum	Applies the **`cks-1`** method to Address and Data
F7	End of exclusive message

In a definition file for this format, the header's MinDevice, MaxDevice and DefaultDevice must be in the range of 00-1F. Common ranges for specific Roland/BOSS models are 00-0F, 10-1F and 00-1F; DefaultDevice is usually 00 or 10. Note that Roland/BOSS user manuals usually refer to Device IDs in decimal form with 1 added to the internal value; so for instance if you see the Device ID range described as '1-16' or '17-32', you know this means 00-0F or 10-1F respectively.

ChecksumMethod must be 1. ChecksumStart must be 4 (for F0, 41, Device and 12) plus the number of model bytes.

So for example, for the BOSS GS-10 (a multi-effect guitar processor) the pertinent section of the definition file's header looks like this (taken from GS-10.ini, included in the BC Manager package):

```
ManufacturerName=BOSS
ManufacturerID=41
ModelName=GS-10
ModelID=00 63
DevicePosition=BeforeModel
MinDevice=00
MaxDevice=1F
DefaultDevice=10
Command=12
AddressLength=4
ChecksumMethod=1
ChecksumStart=6
```

Note that ChecksumStart is 6, because ModelID is two bytes long.

Now consider the following parameter definition from the same file:

```
01 03 00 00 | 00 - 0F | val ;LCD Contrast
```

This defines a 4-byte parameter address for a single-byte parameter in the range of 00-0F by the name of ‘LCD Contrast’. When a user selects this parameter via the ‘Insert SysEx parameter’ dialog box, BC Manager’s preprocessor could generate the following custom output line:

```
$F0 $41 $00 $00 $63 $12 $01 $03 $00 $00 val cks-1 6 $F7
```

Thus, BC Manager automatically uses the correct Manufacturer (**`$41`** = Roland/BOSS), Device (the first **`$00`**), Model (**`$00 $63`** = GS-10), Command (**`$12`** = DT1), parameter address (**`$01 $03 $00 $00`**) and checksum specifier (**`cks-1 6`**). Note that Device is chosen by the end user in the ‘Insert SysEx parameter’ dialog box.

Waldorf

The format of Waldorf data messages looks something like this:

Byte(s)	Meaning/comments
F0	Start of exclusive message
3	Manufacturer: Waldorf
Model	1 byte
Device	00-7F with default 00; 7F means *any* device
Command	1 byte; model-dependant
Address	Length depends on model
Data	Parameter value(s)
Checksum	Optional (depending on model and command); if included, it applies the **`cks-2`** method (included bytes depend on model and command)
F7	End of exclusive message

Waldorf's SysEx format expects the Model ID before the Device ID (unlike Roland/BOSS and Yamaha), so DevicePosition must be AfterModel.

So for example, for the Waldorf Q (a synthesizer) the pertinent section of the definition file's header looks like this (taken from Waldorf_Q.ini, included in the BC Manager package):

```
ManufacturerName=Waldorf
ManufacturerID=3E
ModelName=Q
ModelID=0F
DevicePosition=AfterModel
MinDevice=00
MaxDevice=7F
DefaultDevice=00
Command=20
AddressLength=3
ChecksumMethod=
ChecksumStart=
```

Now consider the following parameter definition from the same file:

```
00 00 0D | 00 - 7F | val ;Sub Vol OSC1
```

Based on this definition, BC Manager's preprocessor could generate the following custom output line:

`$F0 $3E $0F $00 $20 $00 $00 $0D val $F7`

Here **`$3E`** is the Waldorf Manufacturer ID. **`$0F`** is the Model ID for the Waldorf Q. The first **`$00`** is

the user-selected Device. **$20** is the Q's 'Sound Parameter Change' command. **$00 $00 $0D** is the address. There is no checksum in this case (as indicated by the fact that the value of ChecksumStart is left blank in the definition file's header).

Yamaha

The format of Yamaha data messages looks something like this:

Byte(s)	Meaning/comments
F0	Start of exclusive message
43	Manufacturer: Yamaha
1*x*	1 = Command: Individual Parameter Change *x* = Device
Model	Optional; I'm not sure if this can be more than 1 byte
Address	Length depends on model
Data	Parameter value(s)
F7	End of exclusive message

In the definition file, DevicePosition must be WithCommand.
MinDevice, MaxDevice and DefaultDevice must be in the range of 0-F; normally you would set MinDevice to 0, MaxDevice to F and DefaultDevice to 0.
Command must be 1.
Since no checksum is used, ChecksumMethod and ChecksumStart must be left blank.

So for example, for XG synthesizers the pertinent section of the definition file's header looks like this (taken from XG.ini, included in the BC Manager package):

```
ManufacturerName=Yamaha
ManufacturerID=43
ModelName=XG
ModelID=4C
DevicePosition=WithCommand
MinDevice=0
MaxDevice=F
DefaultDevice=0
Command=1
AddressLength=3
ChecksumMethod=
ChecksumStart=
```

Now consider the following parameter definition from the same file:

```
00 00 00 | 00 00 - 0F 7F | val12.13 val8.11 val4.7 val0.3 ;Master
Tune
```

Based on this definition, BC Manager's preprocessor could generate the following custom output line:

```
$F0 $43 $10 $4C $00 $00 $00 val12.13 val8.11 val4.7 val0.3 $F7
```

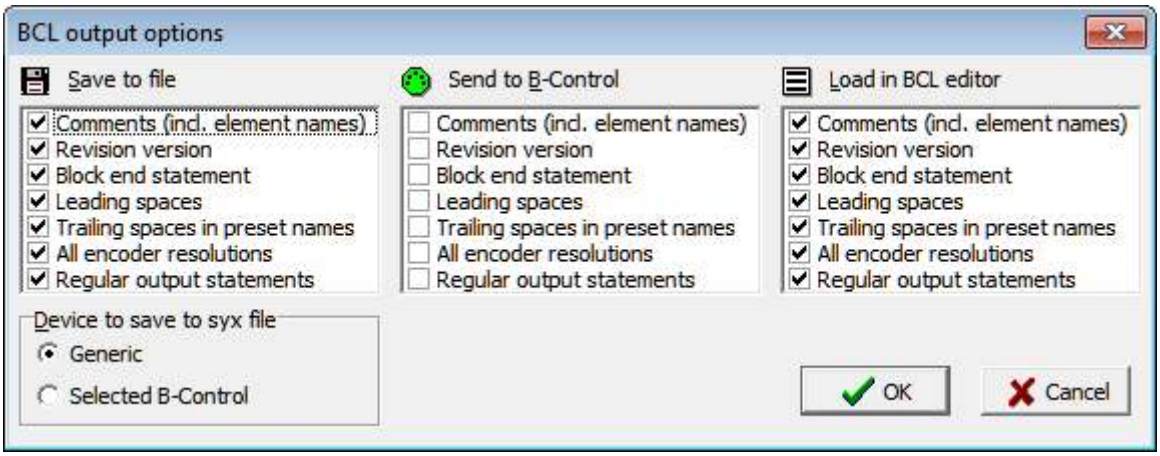
Here **$43** is the Manufacturer ID for Yamaha. The high nibble of the next byte (`$1`) stands for

‘Individual Parameter Change’. The low nibble is the Device (`$0` in this case). **`$4C`** is the XG Model ID, and then follow the 3-byte address for Master Tune (**`$00 $00 $00`**) and the four data specifiers.

19. The BCL output options dialog box

The BCL output options dialog box is accessed via the main window's Options pull-down menu (➜ BCL output).

Leave the settings in this dialog box alone unless you know what you're doing.



There are three output targets, each having its own set of features:

- Files:
  These settings apply when you save a B-Control's settings (as maintained by BC Manager) to a syx, txt or bc2 file, i.e. via File ➜ Save/Save As in the B-Controls window.

- Actual B-Controls:
  These settings apply when you send a B-Control's settings to a BCF or BCR connected via MIDI.

- The BCL editor:
  These settings apply when you execute a Load operation from the BCL editor window.

For each output target, seven settings are defined. In each case, the *checked* state represents the 'standard' BCL protocol. *Unchecking* an item leads to shorter BCL messages, hence to more compact files and higher MIDI transmission speed. (See *BCMI* for detailed information on these optimizations.)

By default, all items are *unchecked* for sending BCL messages to a BC via MIDI. Most importantly, the BCF and BCR don't store (let alone return) comments anyway, so there is no reason to send comments *to* the BCF or BCR. The other options also increase MIDI transmission speed at no conceptual cost.

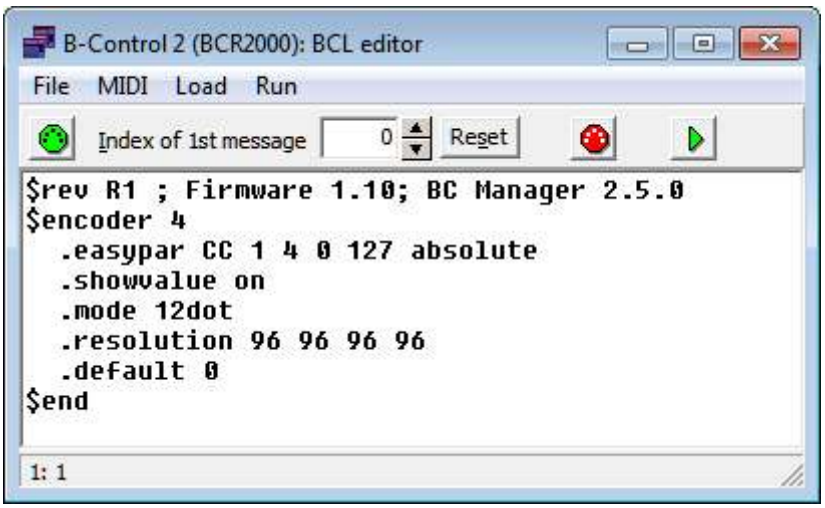
On the other hand, by default all items are *checked* for file saving and loading in a BCL editor. **The 'Comments' option for 'Save to file' is the only option that you normally *must* leave *checked*: otherwise any comments and labels you have defined for the global setup and for presets and elements will be lost in the saved file.** All other options can be unchecked at will; they are merely checked by default because on modern computers BCL script file sizes are hardly relevant, and the standard BCL format provides greater compatibility with other BCL script readers such as BC-EDIT and the Birdie editor (although I haven't actually tested to which extent these programs could read 'non-standard' BCL).

20. The BCL editor windows

Note: This section is only worth reading if you intend to write or edit BCL scripts. If you don't, simply skip it.

Each B-Control in the list has its own BCL editor window. These windows can be opened from the B-Controls window's View pull-down menu.

The BCL editor windows are intended as testing and debugging tools, not for ordinary usage: BC Manager allows you to do virtually anything you would ever want via its 'normal', graphical windows and dialog boxes. Working with the a BCL editor window requires in-depth knowledge of BCL, the language underlying most data communication with the BCF and BCR. See *BCMI* for details.



A BCL editor window provides a basic editor for BCL texts. A BCL text can be opened from and saved to a file, sent to the connected BCF or BCR, and loaded from and entered into BC Manager's internal representation of the B-Control's data.

The menu provides the following operations:

File ➜ Open:
Loads a syx, txt or bc2 file into the editor. Beware: any existing text in the editor is cleared!

File ➜ Save:
Saves the editor's text to a syx, txt or bc2 file.

MIDI ➜ Send:
Sends the editor's text to the connected BCF or BCR.

MIDI ➜ Receive global setup/preset 0:
Sends a request to the connected BCF or BCR for the data in its global setup or preset 0 (the 'temporary' preset), respectively. **Any data returned by the BC does *not* appear in the BCL editor window itself, but in the 'MIDI input messages: BCL' window** (accessible from the main window), provided that this window's Record button has been pressed. Beware: any received data is *also* entered in BC Manager's representation of the B-Control, as shown in the

global setups window and the windows associated with preset 0, respectively.

The idea of these operations is simply to check how a BC returns data previously sent *from* the BCL editor window, rather than to *edit* any data returned by the BC in the BCL editor window.

Load ➜ Empty block:
Clears any existing text in the editor and loads a basic block consisting of an appropriate **`$rev`** statement, an empty line and a **`$end`** statement. Useful for starting small tasks from scratch.

Load ➜ Global setup:
Loads BC Manager's representation of the B-Control's global setup into the editor as a BCL text block.

Load ➜ Preset 0 ➜ Selected buttons/encoders/faders:
Loads BC Manager's representation of a range of buttons, encoders or faders of preset 0 (the 'temporary' preset) into the editor as a BCL block. The loaded range is determined by the current selection in the pertinent element window. To avoid confusion, this operation is only allowed if preset 0 has actually been selected in the presets window.

Load ➜ Selected presets:
Loads BC Manager's representation of a range of presets into the editor as a BCL block. The loaded range is determined by the current selection in the presets window.

Load ➜ All:
Loads BC Manager's complete representation of the B-Control into the editor as a BCL block.

Run ➜ Execute:
Executes the editor's text: BC Manager's representation (and its display) of the B-Control's data is updated according to this text. BC Manager's BCL interpreter reports any syntax errors in virtually the same way as an actual BCF or BCR.

The toolbar contains two items not present in the menu:

- Index of 1st message:
  Determines the index of the first BCL message for the 'Send' operation. This value should normally be left at 0. To be changed only for *very* advanced testing.

- Reset:
  Resets the 'Index of 1st message' to 0.

21. Reason maps

Propellerhead Reason (in any case versions 3.0-7.0.1) 'natively' supports the BCF2000 and BCR2000. This works as follows:

Reason supplies two remote map files, one for the BCF and one for the BCR: BCF2000.remotemap and BCR2000.remotemap respectively. (Under Windows XP these files are typically located in C:\Documents and Settings\All Users\Application Data\Propellerhead Software\Remote\Maps\Behringer, under Vista and later in C:\Users\All Users\Propellerhead Software\Remote\Maps\Behringer.)

Reason uses these remote map files in conjunction with the file Behringer.remotecodec (typically in C:\Program Files\Propellerhead\Reason\Codecs) for its native support of the BCF and BCR: this support mode is triggered if you select the BCF2000 or BCR2000 via Edit ➜ Preferences ➜ Keyboards and Control Surfaces ➜ Add.

In this native support mode, Reason takes full control of the BC:

On initialization of its link with the BC, Reason sets the BC's global Transmission interval to 2 and Deadtime to 400; furthermore, it assigns fixed Control Change messages to *all* the elements (buttons, encoders, faders) of the temporary preset, and prevents you from switching to any memory preset for as long as Reason has control of the BC.

After this initialization, whenever you assign ('lock') the BC to a particular Reason device (mixer, synth, etc.), Reason merely changes its internal mapping from the fixed Control Change numbers on the BC to the controls of the Reason device: it is precisely these mappings that are contained in the remote map files. (Actually, Reason may also update the LED modes of the BC encoders, but apart from that it doesn't change its initial setup of the BC's temporary preset.)

BC Manager provides access to Reason's remote map files via the 'Import Reason map' operation, which is accessible via the B-Controls pull-down menu of the B-Controls window. This operation converts the mappings defined in the remote map file to memory presets, one for each Reason device, in as many new B-Control contexts as needed. (Reason 5 and later define more than 32 devices, so these wouldn't fit in a single B-Control context.)

However, since Reason's native support mode takes full control of the BC's temporary preset (and without reference to memory presets), it would be pointless to edit these memory presets generated by BC Manager: it's impossible to use them under Reason in any way. So these memory presets are only useful for documentation: for instance, you can visualize and print the mappings via the layout window.

The memory presets generated by BC Manager also allow you to see which BC elements are currently unassigned: this is useful because in Reason itself you can *add* mappings from BC elements to Reason device controls via Options ➜ Remote Override Edit Mode.

Note that many Reason control names are quite long, so you will probably want to widen the layout window so that these names show up in full.

The 'Import Reason map' operation loads the Reason device and control names from the remote map file:

- The name of each Reason device is copied to both the name field and the comments field of a memory preset. (The reason for this duplication is that a name field can only contain up to 24 characters, which results in quite a few truncated device names; the comments fields receive the *full* device names.)
- For each Reason device, the names of the mapped controls are copied to the corresponding element name fields of the memory preset.

'Import Reason map' also sets up the actual data structures in exactly the same way as Reason. That

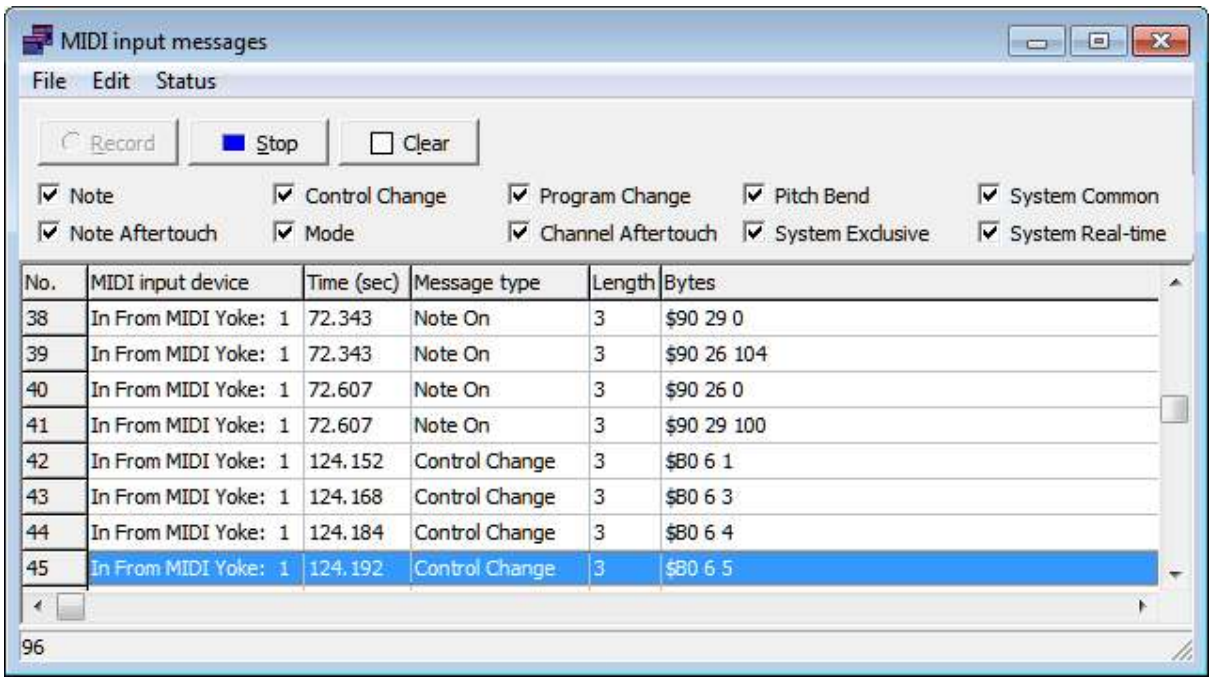
is: BC Manager changes the values of the global Transmission interval and Deadtime settings, and it assigns Control Change messages to *all* preset elements, even the ones that aren't currently linked to any Reason control.

By default a layout window showing a Reason map displays only the Reason control names and 'low-level' Control Change summaries for those elements that are currently mapped. However, you can make the window also show the unmapped elements by selecting View ➜ Element labels ➜ 'Name, else output' from the window's menu.

Note: Reason's native support for the BCF2000 and BCR2000 has *nothing* to do with the mappings defined in Reason's MIDI Implementation Charts.pdf. The Control Change mappings specified in that file are *totally* different: they only apply to Reason's '<Other> MIDI Control Surface' devices and External Control Bus inputs.

22. The MIDI input messages window

The MIDI input messages window is accessed via the main window: either via the View pull-down menu (➜ MIDI ➜ Input messages) or by clicking on the related toolbutton:



The MIDI input messages window allows you to record and view MIDI messages sent to BC Manager from any of the currently *enabled* MIDI input devices, as defined in the MIDI devices dialog box (opened via the main window's Options pull-down menu). Thus, this window is very useful for advanced troubleshooting of a B-Control's output messages. You can also copy recorded messages to the 'MIDI message clipboard' for inclusion in the LEARN and custom output definitions of BC presets and elements, and you can save recorded messages to files (in various formats).

All recorded MIDI messages are displayed in the table at the bottom of the window, one per row. The following columns exist:

- No.:
  The sequential number (index) of the message in the table.
  This number is for reference only. It has no further meaning: when you remove a message, the numbers of all subsequent messages simply decrease by one.

- MIDI input device:
  The MIDI input device from which the message was received.

- Time (sec):
  The time at which the message was received, counted from the moment BC Manager was started.

- Message type:
  The type of the message: Control Change, System Exclusive, etc.

- Length:
  The number of bytes in the message.

- Bytes:
  The bytes of the message. The formatting (hexadecimal, decimal etc.) can be set via the options dialog box, accessed via the Edit pull-down menu.

The menu provides the following operations:

File ➜ Save MIDI file:
Saves the selected (highlighted) MIDI messages to a standard MIDI file ('SMF'). You can load this file in a sequencer program etc.

Two versions of this operation are available via a submenu:

1. 'Times relative to first-saved message':
   The original recording times are maintained, but for convenience a displacement is applied: all messages are saved with their times 'normalized' to the *first* message saved; so the first message saved itself always appears at time 0.

2. 'All times zero':
   All messages are saved with their times set to zero.

Technical notes:

- The MIDI file is in 'format 0', i.e. a single track.

- For convenience, the name of the program ('BC Manager') plus its version number is included as the track name. (It completely depends on the receiving program whether you can see this in any way.)

- The file includes a tempo specification of 120 BPM.
  Beware: It seems that when you import a MIDI file into an *existing* Sonar 7 project, Sonar ignores this file tempo of 120 BPM and wrongly interprets the message times according to the existing project's tempo. In the case of a file saved via 'Times relative to first-saved message', this may lead to unwanted stretching, so it's best to only import such a file into a Sonar project having a tempo of 120 BPM. (I haven't tested later Sonar versions yet.)

- MIDI 'running status' is automatically applied, i.e. where possible the status bytes of channel messages are removed.

File ➜ Save binary file:
Saves the bytes of the selected (highlighted) MIDI messages to a binary file. Note that the recording times are *not* saved: you should save to a standard MIDI file for that (see 'Save MIDI file' above).

You can select 'bin', 'syx' or any other extension for the output file, but your choice does not affect the *content* of the output file in any way.

Beware: a *syx* output file is only valid (i.e. usable in a standard way by other programs) if it *only* contains *System Exclusive* MIDI messages. And since BC Manager specifically allows you to create a syx file containing only the recorded System Exclusive messages (see 'Save System

Exclusive messages' below), the 'save binary file' operation is primarily intended to facilitate further processing by some specialistic computer program expecting a 'flat' sequence of MIDI messages. (Typically this is a program you write yourself!) Note that you can also save MIDI message bytes to a *text* file (see below), which may or may not be easier for further processing.

File ➜ Save text:
Saves the selected (highlighted) MIDI messages to a text file: the bytes of each message are output on a separate line. The bytes are written in the formats defined in the options dialog box (cf. Edit ➜ Options), so exactly as they are currently being displayed in the Bytes column of the window's table.

You could process the output file in an external text editor, then convert them to a binary file: see 'Convert text file(s) to binary file(s)' below.

File ➜ Save System Exclusive messages:
Saves any selected (highlighted) MIDI *System Exclusive* messages to a syx file.

File ➜ Convert text file(s) to binary file(s):
Converts a text file containing lines of hexadecimal bytes (*without* '$' prefixes) to a binary file. You can select 'bin', 'syx' or any other extension for the output file, but your choice does not affect the *content* of the output file in any way.

This is a somewhat obscure utility that could be applied to a text file created by a 'Save text' operation (see above), possibly edited afterwards via a normal text editor (such as Notepad). Specifically, you can thus convert a text file containing only MIDI System Exclusive messages to a legal syx file.

Edit ➜ Copy to MIDI message clipboard:
Copies the selected (highlighted) MIDI messages to the MIDI messages clipboard. Note that MIDI 'running status' is automatically applied, i.e. where possible the status bytes of channel messages are removed.

Edit ➜ Delete:
Removes the selected (highlighted) recorded MIDI messages.

Edit ➜ Clear:
Removes all recorded MIDI messages.

Edit ➜ Select all:
Selects all recorded MIDI messages.

Edit ➜ Options:
Opens a dialog box in which you can set various options related to the MIDI input messages window:

- Buffer size:
  Sets the number of MIDI messages that can be recorded. The default is 65536; this is also the maximum. Note that lowering this setting removes any existing recorded messages beyond the new buffer size.

- Buffer overflow protocol:
  Determines what happens if the buffer is full (as determined by the 'buffer size' setting) when a MIDI message comes in:

- Clear:
  The whole table is cleared, and the incoming message is entered at number 1. This is the default setting.

- Shift:
  The existing message at number 1 is removed from the table, all other messages shift back one position, and the incoming message is added at the bottom.
  Beware: this setting can be very time-consuming.

- Freeze:
  The incoming message is ignored. However, the recording process itself isn't stopped automatically, so when you manually remove one or more recorded messages (e.g. via the Clear button), new messages will be recorded again.

- Stop:
  Recording stops automatically.

- Scroll to new message:
  Determines whether the message table automatically scrolls to any incoming MIDI message.
  'On' is the default, but may result in 'frantic' scrolling when MIDI input is heavy, which may also starve other parts of the program. For instance, the MIDI input and output meter windows may become unable to update their gauges at the required frequency, so that not all incoming messages are displayed. So if you want a quieter display, switch scrolling off.
  Note that the number of recorded MIDI messages is always shown on the status bar at the bottom of the window: this allows you to establish that messages are being recorded even when you have disabled scrolling.

- Byte formats:
  Determines the ways in which MIDI message bytes are formatted: this affects both the window's Bytes column and the 'Save text' operation.
  Separate settings are available for 'status' and 'data' bytes in both System Exclusive and non-System Exclusive messages. A byte in a MIDI message is a status byte if it is in the range of $80-$FF (128-255), and a data byte if it is in the range of $00-$7F (0-127).

Status ➜ Record:
Starts the recording process.

Status ➜ Stop:
Stops the recording process.

The panel below the menu contains the following items:

Record/Stop/Clear buttons:
These buttons duplicate the corresponding menu items.

Note/Note aftertouch/etc.:
These checkboxes determine which incoming MIDI messages are recorded. Checked means 'yes'.

23. The Mackie monitors

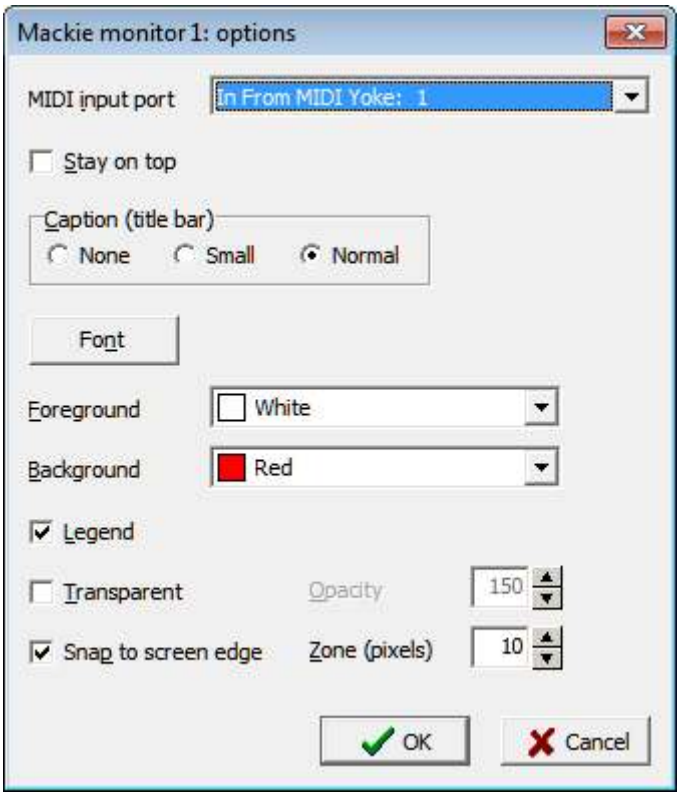
Certain MIDI applications (e.g. Reason) send display messages to a MIDI output port where they suspect a Mackie Control. (Note: In my experience MIDI applications only send these Mackie display messages in response to parameter changes sent *from* the (pseudo-)Mackie Control *to* the MIDI application, i.e. when you turn a knob on the Mackie Control. At least I've noticed that (curiously enough) Reason does *not* send a display message when you move a fader or knob in Reason itself.)

You can intercept these Mackie display messages and show them in one of four Mackie monitor windows, accessible from the main window's View pull-down menu (➜ MIDI ➜ Mackie monitors ➜ 1/2/3/4):

Typically you would use one of these Mackie monitors for a BCF or BCR set up as a Mackie Control via a *preset*. For a BCF in an actual Mackie *emulation mode* (where presets don't exist) it is better to intercept the display messages sent *by the BCF*: see the View ➜ Monitor command in the B-Controls window.

You set up things as follows: you send the MIDI application's output to a MIDI pipe (virtual MIDI device), you capture this pipe's output in a Mackie monitor in BC Manager, and you pass on the MIDI pipe's output to the BCF/BCR's input by means of BC Manager's MIDI Thru facility (set via the main window: Options ➜ MIDI devices).

You set the Mackie monitor's MIDI input port and customize its display via a dialog box that you can access via the main window's Options pull-down menu (➜ Mackie monitors ➜ 1/2/3/4) or by right-clicking on the monitor window itself:



This dialog box offers three options for ‘Caption (title bar)’:

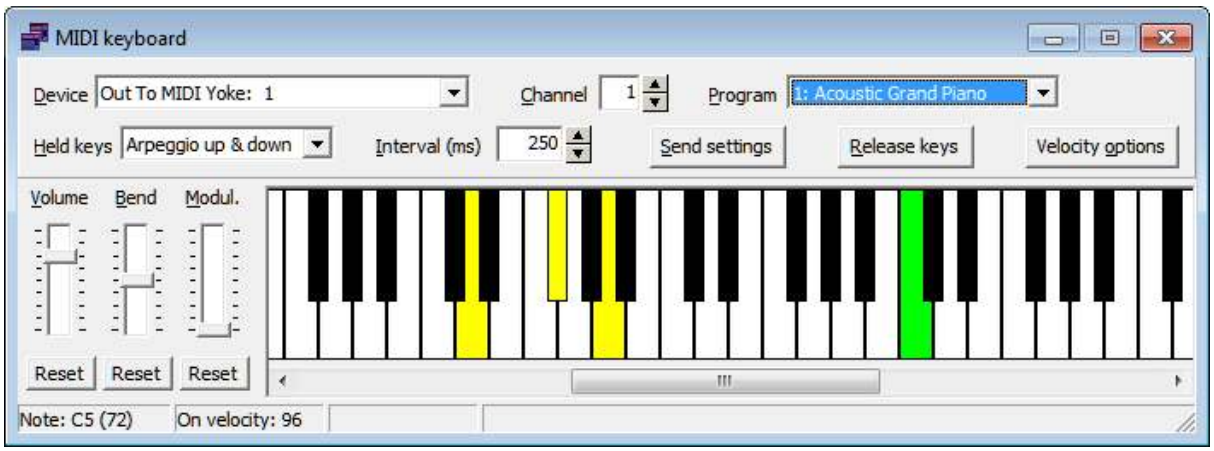
- None:
  The monitor window has no border and no title bar. So you can’t move the window around the screen via the title bar, but you can do so by keeping the left mouse button down anywhere in the window itself.
  If ‘Stay on top’ is checked, the Windows taskbar shows the monitor window on a separate button, but you can’t use that button to minimize (i.e. temporarily hide) the window.

- Small:
  The monitor window is a ‘tool window’: it has a border and a small title bar *without* a minimize icon.
  If ‘Stay on top’ is checked, the Windows taskbar does *not* show the monitor window on a separate button.
  So there is no way to minimize (i.e. temporarily hide) the window.

- Normal (this is the default setting):
  The monitor window has a border and a normal title bar (including a minimize icon).
  If ‘Stay on top’ is checked, the Windows taskbar shows the monitor window on a separate button: if you click that button, the monitor window toggles between being shown and being minimized, independently of the application’s main window.

24. The MIDI keyboard

The MIDI keyboard is accessed via the main window's View pull-down menu (➜ MIDI ➜ Keyboard).

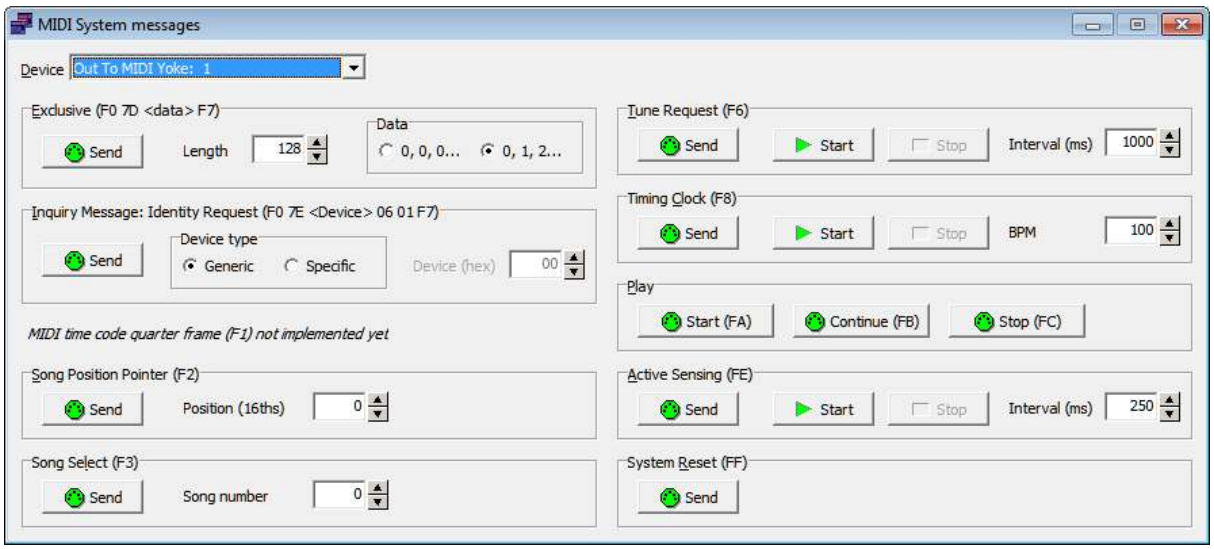


From the MIDI keyboard you can send note messages (and several related other messages) to any MIDI output device (e.g. a synthesizer):

- Pressing the 'Send settings' button sends the selected program, volume, pitch bend and modulation to the selected MIDI output device.
- To play an *individual* note, *left*-click on a key: the key turns green for as long as you keep the mouse-button pressed.
- To *hold* a note, *right*-click on a key: the key turns yellow, until you click it again (or clear all held keys).
- The 'Held keys' drop-down box determines how the held keys are interpreted: they can be played as a chord (i.e. simultaneously) or as arpeggios.
- 'Interval (ms)' determines the chord or arpeggio time interval.
- Pressing the 'Release keys' button clears all currently held keys.
- If you hold the Shift key while left-clicking or right-clicking, all currently held notes are cleared before the newly selected note is played or held. (In other words: it's as if you've pressed the 'Release keys' button.)
- Note names displayed on the status bar follow the Roland octave numbering protocol.
- Beware: the chord/arpeggio timing is not very accurate, because the 'simple' Windows timer is used (rather than the multi-media timer, which is more exact but also puts a larger strain on the computer). In particular, performing actions like opening a listbox may stall the timer. In other words: don't try using the chord/arpeggio modes for actual music production!

25. The MIDI System messages window

The MIDI System messages window is accessed via the main window's View pull-down menu (➜ MIDI ➜ System messages).



From this window you can send MIDI System Exclusive (SysEx), System Common and System Real-time messages – presumably mostly for testing purposes.

Some remarks:

System Exclusive:

- Each SysEx message generated here contains 7Dh as Manufacturer, which indicates non-commercial, generic use. Thus, these messages are normally ignored by all receiving MIDI devices.

  The purpose of these messages is to test SysEx signal flow, in particular the behavior of devices (drivers, sound cards, devices with 'MIDI Merge' facilities, etc.) through which these messages pass. For instance, some devices handle SysEx messages beyond certain lengths incorrectly. (E.g. 'our own' BCF2000 and BCR2000 make garbage of any SysEx message longer than 1019 bytes sent from/to their MIDI IN/OUT sockets via their USB connections.)
- The 'Length' field indicates the *total* length of the SysEx message, i.e. F0 7D <*data bytes*> F7.
- The 'Data' field determines whether the data bytes are all 0, or increment by 1 from 0 to 127 (then start from 0 again).
- To generate your own SysEx messages, use the 'MIDI System Exclusive messages' window instead.

Beware: the timing of the sequences of Tune Request, Timing Clock and Active Sensing messages (generated via the Start buttons) is not very accurate, because the 'simple' Windows timer is used. (Cf. a remark in §24.)

26. Using the computer keyboard and mouse

BC Manager's user interface uses mostly standard Windows controls (buttons, checkboxes, pull-down boxes etc.). This means that it may sometimes be easier to use the keyboard instead of the mouse for particular operations.

Of particular interest are the keystrokes and mouse-clicks that you can apply to parameter knobs:

Key(s)/mouse click	Action
Alt+Enter *or* Alt+Left click (actual knob)	Open a dialog box in which you can type a new value (this only works for purely numerical knobs, i.e. knobs without items like 'Off')
Left arrow	Decrease the value by 1
Right arrow	Increase the value by 1
Ctrl+Left arrow	Decrease the value by a 'big' amount (often 10)
Ctrl+Right arrow	Increase the value by a 'big' amount (often 10)
Home	Select the lowest value
End	Select the highest value
Left click (caption/value)	Select the knob under the mouse
Left click (actual knob)	Set the value as indicated by the mouse position
Right click	Change the value by a 'big' amount (often 10) toward the mouse

And here are a few standard Windows keystrokes worth mentioning:

Control	Key(s)	Action
Any	Tab	Select the next control
	Shift+Tab	Select the previous control
Checkbox	Space	Toggle the setting on/off
Pull-down box	Left/Up arrow	Select the previous item
	Right/Down arrow	Select the next item
	Home	Select the first item
	End	Select the last item
	Alt+Up/Down arrow	Open/close the pull-down list

27. Item dragging

You can directly copy or move any preset, button, encoder or fader definition (or *range* of definitions) from one position to another, as follows:

1. If your intended source item or range of items is in a preset or element window, first select (highlight) this item or these items as you normally do (via mouse or keyboard).

2. Move the mouse to the source item(s). (If the source item is in a layout window, this does *not* have to be the currently selected item.)

3. If you want to *copy* one or more elements rather than *move* them, press a Shift or Ctrl key and *keep* it pressed.

4. Press the right mouse button and *keep* it pressed. (If the source item is in a layout window, the source item immediately becomes the currently selected item as well.)

5. If you have pressed a Shift or Ctrl key under step 3 above (for copying), you may release this key now (although this is not required).

6. Move the mouse to the (first) target item. The target must be of the same type (preset, button, encoder or fader) as the source item, but it can be in *any* B-Control's preset, element or layout window. The mouse cursor is a 'No' symbol when it is on an *invalid* target, and an arrow plus rectangle whenever it is on a *valid* target.

7. Release the right mouse.

8. The 'old' settings of the source item(s) have now been moved to the target item(s), and the source item(s) has/have become empty (i.e. initialized).

Note that these dragging operations completely bypass the button, encoder and fader clipboards, so they do not affect any items already there.

28. Known problems

MIDI Thru:

BC Manager's MIDI Thru feature only passes on *short* MIDI messages, i.e. any message *except* SysEx (System Exclusive).

This is because BC Manager achieves its MIDI Thru feature by simply calling the midiConnect function in Windows' MMSystem library: basically Windows handles all Thru traffic behind BC Manager's back, but unfortunately midiConnect doesn't pass on SysEx messages. (Incidentally, the Huskervu utility's MIDI Thru feature doesn't pass on SysEx messages either, so it probably uses midiConnect as well!)

I may try to find a work-around for this in a future version of BC Manager. In the meantime you should use MIDI-OX if you need to pass on SysEx messages via a MIDI Thru connection. (Apparently MIDI-OX doesn't use midiConnect, but handles all MIDI Thru traffic manually, which might actually be marginally slower than midiConnect, for non-SysEx messages that is...)

USB MIDI ports:

You should not connect or disconnect a BC via its USB cable while BC Manager is running, since this may lead to nasty error messages; instead, you must exit and restart BC Manager manually. I'm working on a fix, but I don't know if and when this will be made available.

Window widths:

If the screen dimensions are too small, big windows of *fixed* size can get cut off. Normally you're safe with a screen of 1024 × 768 pixels, but you can run into problems when you decrease the screen size of a virtual machine running BC Manager.

Alternative DPI (dots per inch) settings:

Nearly all screen elements scale correctly under alternative DPI settings. However, the bitmaps used in the pull-down menus *don't* scale, which leads to rather cramped-looking pull-down menus at high enlargement DPI settings, because the heights of the menu items follow these bitmaps instead of the menu items' *names* (which *do* scale).

Presets that are too complex:

Currently BC Manager doesn't prevent you from sending a preset to the BC that is too complex for the BC to store in full.

29. Wish list

- BC Manager versions for Linux and Mac OS X.
- Direct access to the presets on the BCF and BCR, bypassing BCL.
- BCL syntax highlighting.
- Find and replace text in names and comments.
- An undo feature.

There are quite a few other features that I would like to incorporate in BC Manager, but haven't got round to mentioning here. If you have any suggestions for improvement, please post these in the B-Control forum at the Mountain Utilities web site: http://mountainutilities.eu/forums/b-control.

30. Frequently asked questions

The subsections on the following pages contain some of the most relevant questions concerning BC Manager that users have come up with. Some actual questions appear here almost *verbatim*, others have been generalized, and a few were invented by yours truly specially for this section... Thanks to everyone for asking these questions!

General

[1] ***Why did you write BC Manager?***
Mainly because I am mad...

But seriously: after I had bought my BCR2000, I soon got very disappointed with Behringer's own BC-Edit (see the next question), and to some extent also with the Birdie editor (because of its limited capabilities and its complicated layout). Whereas I liked Royce Craven's BCL script editor (although back then it didn't have all its current capabilities), I wanted to write a GUI editor that doesn't require the user to know BCL with all its sordid details and quirks.

[2] ***What are BC Manager's advantages over Behringer's own BC-Edit?***
1. The transfer of data between BC-Edit and BCFs/BCRs doesn't work with the latest versions of JRE (Java Runtime Environment). (At least not under Windows; I have no idea about JRE under other operating systems.)
2. BC-Edit doesn't support a lot of the advanced tricks of which the BCF and BCR are capable, such as 'LEARN output' and 'custom output'. BC Manager provides almost complete support.
3. BC Manager has far more advanced editing facilities, and an extensive array of MIDI tools that can be used for testing.

[3] ***Why is BC Manager freeware?***
Back in 2007, when I announced that I had started writing BC Manager, someone in the BC2000 Yahoo group suggested that he'd even be interested if I made it shareware. However, I have never really considered this, for several reasons, all amounting to the feeling that it would be more trouble than benefit: setting up a method for payment, building in a registration method, trying to prevent piracy. Perhaps most importantly: I would simply not feel comfortable demanding money from people I consider, in some abstract way, 'my friends'. Relationships between me and the other members in the BC2000 Yahoo group would have become subtly 'different'. Besides, there were quite a few people who had (in their own ways) contributed to the program: Michael Kukat, who (as far as I know) wrote the first report on BCL, and Royce Craven with all his excellent work on BCL and his BCL script editor. And I feared that people might get more reluctant to contribute to the program in the future (pointing out bugs or making suggestions for new features etc.), for in a sense they would then become some kind of unpaid employees. Maybe these fears were a bit exaggerated, but I would find this all a bit unpleasant — so that's why BC Manager has always been free.

[4] ***Is there any chance of a Linux or Mac OS X version of BC Manager?***
Yes, but don't hold your breath. So for now, if you wish to use BC Manager under Linux or OS X, try running it from within a virtual machine manager. Concerning Linux, BC Manager has been reported to work perfectly under Wine, and it should also run under any other VM manager. On the Mac, BC Manager has been reported to work under Parallels; again, other VM managers should work as well.

[5] ***Do you have a PayPal address so I can shoot you a few bucks in appreciation of all your efforts?***
Yes. There are two ways:
- You can donate via BC Manager itself from the main window's Help menu via Donate: this opens a dialog box in which you can select your currency, and you are directed to the PayPal web site.
- Alternatively, go to the Donate page at the Mountain Utilities web site. This will also direct you to the PayPal web site.

[6] ***Who or where should I send questions to?***
http://mountainutilities.eu/forums/b-control

Editing

[7] ***In the B-Controls window, how can I remove the BCF2000? I only have a BCR2000 and would like to keep things clean.***

In the table, select (highlight) the line containing the undesired device, then execute Close from the B-Control pull-down menu. The undesired device should disappear (and not return when you restart BC Manager).

[8] ***What is 'preset 0' and what is its relevance?***

Preset 0 is also known as the 'temporary preset'. This preset exists only implicitly on the BCF and BCR: whenever a memory preset (1-32) gets selected, that memory preset gets copied to the temporary preset area. In BC Manager preset 0 is *explicit*: you can work with it *almost* as any other memory preset. I say 'almost' because it has some peculiarities:

Any manual preset editing on the BC itself affects only the temporary preset. That's why the STORE operation exists on the BC: STORE copies the temporary preset to a memory preset; and of course the temporary preset is lost when you switch to another memory preset or switch off the BC.

Furthermore, any uploading of presets to the BC (e.g. by BC Manager) goes *via* the temporary preset on the BC. What happens when you upload a *memory* preset (say: preset 6) to the BC, is this: First the BC's temporary preset gets overwritten with the preset data being uploaded. Then a '$store 6' command (appended to the data stream sent to the BC) tells the BC to copy the new temporary preset data (i.e. the preset data that has just been uploaded) to memory preset 6.

[9] ***What do the Request checkbox and the 'LEARN output' tab in BC Manager's Preset dialog box refer to?***

If the current preset on the BC has its Request parameter set to *on*, the BC automatically sends the preset's 'LEARN output' whenever you select that preset. If Request is *off*, the BC only sends this 'LEARN output' when you press EDIT + LEARN on the BC itself.

On the 'LEARN output' tab you can define a sequence of MIDI messages for each individual preset. You're completely free as to the nature of these MIDI messages (although the total length is restricted to something like 125 bytes).

Probably Behringer's original idea was that this LEARN data sent from the BC should be a request for settings getting sent back *from* the other device *to* the BC: hence the name 'LEARN', i.e. learned *by* the BC. However, that doesn't have to happen: you can of course have the BC send *anything*, not necessarily triggering a MIDI response from the other device back to the BC. One obvious idea is to have the BC simply send a MIDI Program Change message indicating that the BC has indeed selected *that* preset. So e.g. for preset 1 you could define '$C0 $00', for preset 2 '$C0 $01', etc. Another application is the initialization of a receiving synthesizer or so.

[10] ***What is a snapshot? And why would I use it?***

A snapshot is a facility built into the BC: it's a sequence of MIDI messages sent by the BC, specifying all the current button, encoder and (for the BCF) fader positions. (Typically these are mostly MIDI Control Change messages, but this depends on the preset definition.) So making a snapshot amounts to physically touching all the controls on the BC simultaneously, triggering the MIDI messages associated with these controls.

A snapshot should not be confused with the preset's *definition*, which is a sequence of complicated MIDI System Exclusive messages and specifies *how* the preset works, but does *not* include the current, actual positions of the buttons, encoders and faders.

A snapshot is very useful for saving a particular favorite setup of (for instance) a synthesizer: any computer program capable of receiving MIDI data (Cubase, Sonar, etc.) may capture and save the snapshot (e.g. as a standard MIDI file), and then you can resend the snapshot to the synthesizer at any

later time, so that the synthesizer is set exactly to your favorite setup.

There are several ways to trigger a snapshot:

- On the BC itself: hold the EDIT button, then press the '◂ PRESET' button.
- From BC Manager: execute 'Receive snapshot' from the B-Controls window.
- A bit more advanced: From BC Manager, record a snapshot from the BC and put it in a button definition. Then send that button definition back to the BC, so that whenever you press that button on the BC, the whole snapshot is output. In this way you could define a preset that is a collection of synthesizer setups!

It is also possible to have the BC send a preset's snapshot whenever you *select* that preset (e.g. via pressing the '◂ PRESET' and 'PRESET ▸' buttons): thus you ensure immediate synchronization between the preset's current values and e.g. a synthesizer. In BC Manager you can switch this feature on and off via the Snapshot checkbox on the Settings tab in the Preset dialog box.

[11] ***In the element (button/encoder/fader) tables, is there a way to change a particular parameter (e.g. the MIDI channel) of the selected elements? It seems that even when you select several rows, you can only change the parameters for one element at a time.***

The 'E' button or the Edit menu item indeed simply edits each selected encoder *in turn* and *as a whole*. So that is *not* what you should use in this case.

Instead, to edit just a single parameter of a range of elements, use the Set submenu under the Edit pull-down menu: there you can select the parameter of your choice (e.g. 'Channel') that you want to change in one go for all the selected elements. A dialog box appears in which you can select the new parameter value. What's more: the 'Step per row' knob allows you to define an increment: e.g. if 'Step per row' is 1, the *first* element gets set to the value you have indicated, but the second becomes 1 higher than the first element, and so on. *(See the next question for a concrete example of this procedure.)*

Beware: afterwards you may not actually see the parameters change in the element table if the Type parameter is 'None'. But you can set Type in a bulk operation as well, so that can be fixed easily...

[12] ***Wouldn't it be nice if the paste operation could increase the previously copied element's CC/NRPN number by 1 for every successive element? You could then set up one element, then create the others almost instantaneously.***

BC Manager currently doesn't allow this particular 'complex paste' operation. However, you can achieve basically the same result as follows:

1. Select a range of elements (e.g. encoders 1-10).
2. From the Edit menu, execute Set ➜ Type and choose (for instance) NRPN.
3. From the Edit menu, execute Set ➜ NRPN ➜ NRPN, setting NRPN to the first desired value and 'Step per row' to +1.
4. Set any other parameters (e.g. Value 1 and 2) in the same way.

Alternatively:

1. Set up encoder 1 completely (via the Encoder dialog box).
2. Copy encoder 1.
3. Select encoders 2-10. (Or possibly 1-10, since you're going to set the first selected encoder's NRPN value anyway in step 5 below; you then don't even have to set up encoder 1's NRPN value in step 1 above.)
4. Paste (this pastes encoder 1 to all the selected encoders).
5. Execute Set ➜ NRPN ➜ NRPN, again setting 'Step per row' to +1.

[13] ***The BC allows me to make a button send incremental Program Change messages. I can set this on the BC itself by entering the button's edit mode and setting push encoder 6 to 'InC'. However, I can't do this in BC Manager; why not?***

This feature was introduced in the BC's firmware version 1.10. However, due to a bug in this firmware, the BC doesn't include the 'InC' setting in its preset dumps, nor can it be set via MIDI. Consequently, BC Manager (or any other BC editor) cannot work with this setting, and destroys it whenever it sends the button's settings to the BC.

[14] ***I use BC Manager to make tweaks on my BC. Is there a way to make a toggle-off button skip through the available options? For instance, for an effect parameter with only 3 or 4 options I would like to make the button jump in skips, so that continuous presses reveal values of 0, 64, 127, 0, 64, 127, 0, 64, etc. Is this possible?***

In BC Manager, go to the 'Standard output' tab of the button dialog box. There, for a button with 'Type' set to Control Change, you can set the 'Mode' parameter to 'Increment', and then the 'Increment' parameter to the desired jump value. Obviously, a limitation of the increment parameter is that the step size is fixed. Please consult the sections called 'Button increment mode' and 'Increment mode' in *BCMI* for in-depth discussion of the increment feature.

[15] ***Behringer's BC-Edit program doesn't allow me to assign note messages to an encoder. Is this possible in BC Manager?***

No in that it's simply a feature of the BCF and BCR themselves that their 'standard output' protocol only allows the assignment of note messages to buttons, not to encoders or faders.

Yes in that BC Manager allows you to assign *any* messages (including note messages) to buttons, encoders and faders under the 'custom output' protocol. However, this is a bit more work, since you have to construct the MIDI byte sequence manually:

A MIDI Note On message has the format '$9*c n v_on*', where *c* (a hexadecimal digit from 0 to F) is the MIDI channel (1-16) minus 1, *n* (0-127) is the note number, and *v_on* (1-127) is the (onset) velocity. So e.g. '$90 64 127' (or the equivalent '$90 $40 $7F') constitutes a Note On message on MIDI channel 1 for note 64 with maximum velocity.

For an encoder you can also include the knob's value (position) in the output message, by using the keyword 'val'. You can use this keyword for either the note number or the velocity, so for instance: '$90 val 127' or '$90 64 val' — it depends on your application which of these options is appropriate. Note that you must define the range of 'val' via the *Value1* and *Value2* parameters of the encoder's custom output definition.

You also have to decide whether you need to add the corresponding Note *Off* message ('$8*c n v_off*') to the custom output definition (or possibly '$9*c n* 0', of course). This depends on the requirements of the receiving device.

[16] ***I have played with the* Default *value setting with no difference and I am a bit confused by the value range of 0-16383.***

Default is a generic parameter, i.e. it applies to *any* 'standard' or 'custom' output message definition, so its wide range of 0-16383 is necessary to support even those output message types that use 0-16383. In fact, as far as the 'standard' output definitions are concerned, this wide range is only used by a few encoder and fader parameters (i.e. CC and NRPN in the 14-bit modes), not by any button parameters. So you should only ever set *Default* to the range currently indicated for *Value 1* and *Value 2* on the 'Standard output' tab, i.e. usually 0-127.

[17] ***When I set the encoder's LEDs to a 'fancy' setting like 'Bar' or 'Spread' on my BCR2000, only the eight push encoders are affected. Is this normal?***

Yes. Although the BCR2000 allows you to *program* the 'fancy' LED settings for *any* encoder, these fancy settings only *work* for the eight push encoders, not for the twenty-four 'simple' encoders. (As Royce Craven has pointed out, this limitation is probably built into the BCR2000 because the LEDs' power consumption would be too big if they could all be lit simultaneously.)

So this is why BC Manager's encoder dialog box says 'Warning: this setting only works for push

encoders' if you select a fancy setting for a simple encoder, and why the LEDs column in the encoder list window says 'INVALID ...'.

[18] ***Is it possible to enter values with the numeric buttons on the keyboard instead of using the mouse?***

When you're on a knob, press Alt+Enter. See §26 for further applicable keyboard shortcuts. (Note: Kip's BC Manager video erroneously talks about *Ctrl*+Enter: that doesn't work.)

[19] ***I'm curious why BC Manager uses knobs in some of the dialog boxes. Some of the ranges kind of threw me off, but then again I'm not a big fan of dialing with a mouse in general. It just doesn't feel natural or intuitive to me to drag up to have a knob turn right...***

In my experience, knobs on computer screens are almost exclusively seen in music software: apparently the idea is that musicians are used to working with knobs on actual musical hardware. Personally I wasn't a mouse user at first either (this may be a generation issue — I started using computers in the 1980s!), and accordingly had my doubts about knobs: instead, my programs used to have 'spin edit boxes', i.e. text edit boxes containing up and down arrows. However, around 2003 I finally decided to develop a rotational knob: that took me about a full month, but I was very satisfied with the result. So since then I have used both knobs and spin edit boxes in my programs. It's always a dilemma which to use where: there are arguments in favor and against both:

E.g. BC Manager uses a spin edit box for the number of encoder groups (1-4), because the range is small and would look a bit weird on a knob.

On the other hand, knobs have the advantage that you can set any 'remote' value almost immediately. This is especially useful if the range is large, e.g. the typical MIDI range of 0-127. This would be much harder to do with a spin edit box (well, you could *type* the value, of course).

When you say that some of the ranges threw you off, I suppose you mean those ranges that go up to 16383. I agree that these are a bit difficult to manipulate (since we typically only want to set a value in the 0-127 subrange). However, I can think of several reasons (some more convincing than others) for keeping things like they are now:

1. The fact is that these ranges *are* that large, and at least the user now *sees* that potential immediately.
2. You can always change a knob's value directly by means of the arrow keys (cf. §26).
3. For most knobs it is possible to open a dialog box containing a spin edit box by pressing Alt+Enter; so in that sense the user has that alternative available anyway whenever mouse operation is difficult.
4. Via the main window's 'Mouse options' dialog box, you can set the direction in which you have to move the mouse to affect the knob value, and (if you select 'horizontal' or 'vertical') the resolution at which this happens. Also note that he mouse becomes more accurate when you widen its circle of movement.
5. There is perhaps an issue of consistency (for what it's worth): why use spin edit boxes for certain parameters and knobs for others (in the same dialog box)? Things might start looking a bit messy.

So all in all I think the current setup is reasonable, combining the 'best of both worlds'. Still, I must admit that I do share your basic instinct, so if and when I get a clever idea, I may change things after all.

(One thing that might work would be to narrow the range of the '0-16383' knobs to '0-127' unless the wide range is actually valid (e.g. if Mode has been set to a 14-bit value) and/or if the user specifically *requests* the wide range via an additional '14-bit range' checkbox or so. But again, in a sense that would *also* complicate things, so I don't know... It's just one of those headache-causing user-interface dilemmas...)

[20] ***How do I 'search and replace' a particular byte in custom output messages? I need to do this***

for a series of almost identical SysEx messages, only differing in the byte(s) specifying the parameter addresses of my synth.

Select the elements you wish to change, then from the element window's Edit pull-down menu, execute Set ➜ Custom output ➜ Item, and fill in the appropriate values in the dialog box that appears.

[21] ***Can you implement an undo feature in BC Manager? I have had to restart BC Manager several times, when I cut a line, then paste, but do something in the wrong order. This is the case, say when I am revising a layout, moving things around to better suit me. I get stuck, I copy something, then paste, or paste the last thing before I cut... I really need an undo feature. Otherwise I have to stop, close, not save, start over since last save, and I don't save after every single move, although I could.***

From a programmer's point of view, an undo feature (in particular an application-wide one) is absolutely one of the most horribly difficult and time-consuming things to implement: I don't think I've ever seen a professional word processor or programming environment whose undo feature was *not* buggy in some way... The problem is this: the more editing operations a program has, the more difficult it is to keep track of all the interacting effects of these operations, hence the more difficult it is to reverse all these operations. So I don't think that it is on the cards in the near future, but I have added it to the wish list...

Instead of closing, can't you just reopen the current file? One thing that might also help in this respect: you can open as many 'B-Controls' as you wish; note that you don't have to link each of these to a *real* BCF/BCR (or even to a file). So instead of saving your intermediate presets (or even buttons etc.) to file, you could copy them to another 'B-Control' now and then.

[22] ***Is there a place in BC Manager where the underlying BCL script is visible/editable?***

You can open a BCL editor window (one for each B-Control) from the View pull-down menu of the B-Controls window.

A note to clear up a potential incomprehensibility you may run into: when you execute 'Receive global setup' or 'Receive preset 0' from the BCL editor window's MIDI pull-down menu, you can only see the B-Control's response in the (*global*) 'MIDI input messages: BCL' window. You can open this window via the main window's View ➜ MIDI submenu; you must also press the Record button in that window before any response is actually recorded/displayed.

[23] ***I loaded selected presets into the BCL editor. When I select e.g. presets 11-16, they all load, but so does preset 0, which is tacked on the end. Was that your intention?***

Yes:

When sending presets to the BC, BC Manager first sends (to use your example) memory presets 11-16, then preset 0 (the temporary preset). The latter is necessary for restoring preset 0 to the current values maintained by BC Manager, because preset 0 on the BC is used as a gateway for presets 11-16. So it seems logical to copy this behavior in the BCL editor window — which is why preset 0 is appended. The idea is that the user can edit the BCL text, then send it directly to the BC with the same 'automatic' restoration facility of preset 0.

However, I admit that in other situations (such as yours) it might be easier to leave out preset 0 at the end. I could include an on/off-switch in the BCL editor window's menu, or simply add another menu item that leaves out preset 0.

Note that if you simply want to export a *single* preset at a time, you can simply use preset 0 itself for this — then you obviously don't get an extra, appended preset.

However, you can also export presets directly from the preset list window — in that case you *don't* get the extra preset 0!

Files

[24] ***How do I import one or more individual presets (stored in a syx/txt/bc2 file) in the Presets window?***

You have to go to the B-Controls window: from its File pull-down menu you can execute either Open or Import. Neither of these operations initializes the whole data area before loading the data from the input file. So the input file can contain any BCL data you like: presets or even individual buttons etc. The only difference between Open and Import is that the Open operation renames the 'current file name' (as shown in the B-Controls table) to your imported file.

[25] ***I have the Presets window open and I load my presets using the Import operation. They automatically go into preset 0, and I then copy and paste them into their appropriate slots. Is it normal for the syx file to automatically go into preset 0? If so, great! If not, what should I do to fix that, and is my 'workaround' all right?***

By nature any preset definition *as such* is a definition for the *temporary* preset. The syx file may or may not contain a $store 1-32 command after the preset definition as such. So if a preset file doesn't contain a $store command at the end, BC Manager can only load that preset into preset 0.

Syx files saved via BC Manager's 'Export selected preset(s)' operation (from the Presets window via the File pull-down menu) *do* append the $store command, but other preset files people have made may not, in which case you indeed have to copy them to the intended locations manually.

In accordance with the BCF and BCR themselves, BC Manager's Import operation (from the B-Controls window) always first loads the preset into preset 0, then if the file contains an appended $store *N* command, BC Manager copies that preset 0 to memory preset *N*.

By the way: if you wish to find out whether a preset syx file contains a $store command, you could convert the syx file to a txt file (in BC Manager's main window via File ➜ Convert), then open the txt file in any text editor (e.g. Notepad), and look for '$store' at the bottom of the file.

[26] ***How should I save my presets to make them available to other people?***

In the Presets window, select (highlight) the preset(s) you want to save, and choose 'Export selected preset(s)' from the File pull-down menu. In the dialog box you can then select the output file type (syx, txt or bc2) via the 'Save as type' box. For widest compatibility (e.g. with Mac computers) you should save your presets to syx files.

An important decision you have to make is whether the syx file contains the temporary preset (i.e. 'preset 0') and/or one or more memory presets.

The saved file will contain all the presets' respective '$store' commands. However, if preset 0 (the temporary preset) is among the selected presets, preset 0 will be saved *last*, and will *not* be followed by a $store command. This protocol is similar to what happens when you send a bunch of presets to an actual BC: the BC uses the temporary preset as a gateway to any memory presets: formally *any* preset definition concerns the *temporary* preset, and is only *copied* to a memory preset the moment a $store command is encountered.

An obvious advantage of saving *memory* presets is that a single syx file can contain up to 32 memory presets, whereas there can be only one *temporary* preset in a single syx file. Since files containing memory presets are no problem to users of BC Manager, a syx file containing memory presets is probably easier to use than a collection of files each containing only one preset. If the user wants, he/she can create an additional 'B-Control', open the file containing the memory presets into that B-Control, and then copy and paste the presets to the desired locations in the B-Control linked to the actual BC.

However, for greatest flexibility, you should save all your presets to separate syx files, each containing the temporary preset (i.e. a preset definition without a $store *N* command). If you export *memory* presets (e.g. 11 to 16) to a single syx file, the output file contains $store (11/12 etc.)

statements, which is potentially dangerous to an unsuspecting user, because uploading such a syx file to a BC immediately overwrites those memory presets on the BC.

[27] ***I have looked at a txt output file created by BC Manager. There is a lot of stuff at the end that will set a lot of presets to a blank preset. For instance:***

```
$preset
  .name '                          '
  .snapshot off
  .request off
  .egroups 4
  .fkeys on
  .lock off
  .init
$store 24
$preset
  .name '                          '
  .snapshot off
  .request off
  .egroups 4
  .fkeys on
  .lock off
  .init
$store 25
```

... Why does BC Manager do this?

The File ➜ Save operation from the B-Controls pull-down menu saves complete sets of memory presets (1-32) to a file, to avoid any confusion that might occur when you *open* a preset file that leaves out certain presets: you could then end up with a very confusing mix of presets that you were already editing and the newly loaded ones.

See the description of the File asterisk column in §10 for an explanation of this point in relation to Behringer's own syx preset files.

If you want to mix presets from different files manually, you can simply create one B-Control for each file, then copy and paste.

MIDI communication

[28] ***Since BC Manager is largely useless when the BCF is in an emulation mode, how do I find out what is controlling what in emulation modes?***

First of all, you should be prepared for 'a few' quirks, caused by the way the BCF behaves in its emulation modes or by the expectations made by the hardware or software at the other end of the MIDI connection.

For finding out which MIDI messages a particular knob on the BCF sends in any emulation mode, you can always use a 'MIDI pipe' (e.g. MIDI Yoke) to insert a MIDI monitoring program like BC Manager or MIDI-OX. (If you wish to use BC Manager in this way, you have to set up a 'MIDI Thru' connection via BC Manager's MIDI devices dialog box.)

In general, a little perseverance is called for here: making a controller like the BCF and a receiving device understand each other is often a struggle, even if you have worked with MIDI and indeed the BCF/BCR for a long time.

[29] ***Why won't BC Manager 1.5.1 receive data from my BCF even though it recognizes it and can send data?***

BC Manager is primarily meant for working with the BCF in standard 'B-Control' mode. When a BCF is indeed in B-Control mode, BC Manager should in principle be able to communicate with it fully. However, when a BCF is in any emulation mode, communication is quite limited; for instance, no presets are available.

All BC Manager versions prior to 2.0.0 simply expected the BCF to be in B-Control mode, so confusingly you could still send presets to the BCF in an emulation mode — but of course the BCF wouldn't actually accept these presets. As of version 2.0.0, BC Manager autodetects the BCF's mode (as reported in the Personality column of the B-Controls window) and automatically disables any operations that don't apply to emulation modes; for instance, you can't even send presets if the BCF is in an emulation mode.

[30] ***Why can't BC Manager's 'Detect B-Controls' detect my BC?***

The 'Detect B-Controls' operation itself doesn't enable any MIDI input or output devices, so *before* executing 'Detect B-Controls' you must *manually* enable your newly connected BC's MIDI input and output devices in the MIDI devices dialog box. (The only exception to this is when you start up BC Manager for the very first time: in this case all MIDI I/O ports are automatically enabled.)

Note that if you have switched on a BC connected via USB when the current instance of BC Manager was already running, you must first restart BC Manager before the BC's I/O devices even appear in the MIDI devices dialog box.

Furthermore, the BC must not be in Global Setup ('EG') or Edit mode.

[31] ***I want my BCF/BCR to control my synthesizer via the BCF/BCR's standard MIDI input/output sockets. When I'm programming the BCF/BCR via BC Manager, I use a U-mode, but then I can't simultaneously access my synthesizer for testing. So I keep switching between U-mode (for programming) and S-mode (for testing), which causes BC Manager to report a lost connection, so I have to restart BC Manager all the time. Isn't there an easier way?***

First of all, this 'connection loss' is not BC Manager's fault: it's simply a feature of Windows: as far as I know, a program's MIDI I/O port configuration cannot change while the program is running. In my experience any MIDI software package for Windows suffers from this problem.

But to answer the question: you *don't* have to switch between U and S operating modes for programming and testing. Instead, you can route the BC's signals to your synth via BC Manager:

1. On the BCF/BCR, select U-3, or possibly U-2. (U-3 supports both MIDI OUT A and B; U-2 only MIDI OUT A.)

2. Connect your synth (bidirectionally) to the BCF/BCR's MIDI input socket and to the MIDI output socket of your choice (cf. the U-mode chosen above).

3. Start BC Manager. From the main window's menu, open Options ➜ MIDI devices:
   (a) On the Input tab, select (highlight) the BC's input control device; then enable it (via the checkbox) and set its 'Thru output device' to the BC's MIDI output socket of your choice.
   Also select and enable the BC's MIDI input socket, and set its 'Thru output device' to the BC's output control device.
   (b) On the Output tab, enable the BC's output control device and the MIDI output socket of your choice. (Their 'Thru input device' settings should already have been taken care of by point (a) above.)
   Note: in the above procedure, I haven't specified MIDI device *names*, since these vary, depending on the installed USB MIDI driver. But e.g. if you're using a BCR2000 and the Behringer driver vs. 1.2.1.3, you should typically connect 'BCR2000[1]' ↔ 'BCR2000[1]-A'.

Now the BCF/BCR should be able to talk to BC Manager and the synth simultaneously. You can test the connection from the BCF/BCR to the synth by checking whether BC Manager's MIDI input and output meters respond when you generate MIDI output on the BC or the synth: in the input and output meter windows, the 'LEDs' of the connected input and output devices should light up simultaneously.

One warning though: you cannot pass MIDI *System Exclusive* messages from your BC to your synthesizer this way. This is a limitation of BC Manager's MIDI Thru facility. See §28 for explanation.

[32] ***How do I upload presets (for instance Behringer's original factory presets) to my BC? I could have sworn I was doing it correctly by uploading, pressing the STORE button, using the preset arrows to find the bank of choice, and then pressing STORE again, but it appears to not have worked.***

It's easier to do this from BC Manager's Presets window: once you have put a preset in its place (e.g. by opening its syx file), you select (highlight) the preset, then go to the MIDI pull-down menu and execute Send. In principle it should be unnecessary to press any buttons on the BC itself.

If you wish to restore the BCF or BCR's original presets: they're in the files Factory_Presets_BCF.zip and Factory_Presets_BCR.zip respectively, available from the Behringer web site.

[33] ***Do I have to press any buttons on my BC to receive data from BC Manager?***

No, you don't have to do anything on your BC, except that you have to make sure that your BC is not in Global Setup mode (the display showing 'EG'; in this case it doesn't accept *anything*) or in Edit mode for a button, encoder or fader. And 'of course' a BCF must be in standard 'B-Control' mode, not in an emulation mode. However, in all these cases BC Manager warns you that things aren't working when you send a preset or element definition to the BC.

[34] ***When I try to send a preset to my BC, BC Manager's dialog box stops in the middle. Why?***

Several things can go wrong. For instance, the very connection may be lost (this happens more often with USB connections than standard MIDI ones). Whatever the source of the problem, one thing to realize is that BC Manager's 'send preset' routine uses a handshaking protocol: BC Manager sends a message, then waits for the BC to confirm reception of that message before sending the next message. (But of course this doesn't tell you what exactly is wrong.)

One thing that you can do after a failed transmission is to execute 'Refresh connection status' (either via the button or the item in the MIDI menu). If this doesn't work either (as reported in the Personality and Firmware columns), the problem is indeed a sudden loss of connection, due to switching off the BC, entering Global Setup or Edit mode, etc. On the other hand, if 'Refresh connection status' reports nothing unusual, the problem may be more specifically connected with the data you're trying to send. You should also watch out for feedback loops and duplications in the MIDI paths to and from the BC: these may mess up the MIDI SysEx message 'chains' of which preset and element dumps consist.

[35] ***I tried to send a preset to my BC via BC Manager, but BC Manager hung in the middle of the transfer process, so then I sent the syx file via MIDI-OX. What could have caused the transfer to fail in BC Manager but succeed in MIDI-OX?***

When BC Manager sends a preset to the BC, it uses a bidirectional, handshake protocol. That is: it sends one SysEx message, then waits for the BC to reply with a SysEx acknowledgment message before sending the next message. So this may be the reason for your problem with uploading your data from BC Manager: maybe somehow the BC didn't return the correct acknowledgment message, or BC Manager received an unexpected message back via a feedback loop, upon which it decided to stop transmitting.

By contrast, MIDI-OX simply sends all the SysEx messages to the BC with fixed delays between them, without waiting for any acknowledgments from the BC. MIDI-OX's method has two drawbacks:

1. If the BC doesn't actually accept the data (for whatever reason), MIDI-OX doesn't inform you, let alone halt the transfer.
2. Perhaps surprisingly, in practice MIDI-OX's method is usually *slower* than BC Manager's, because of MIDI-OX's fixed delays between the messages (by default 60 msec, I think): BC Manager continues immediately upon reception of an acknowledgment from the BC.

Note: BC Manager version 2.0.0 and later allow you to use one-way, generic SysEx transfers as well (via the 'MIDI System Exclusive messages' window), but it is inadvisable to use this method for BC SysEx data such as presets — unless you don't have a bidirectional connection, perhaps.

[36] ***Does BC Manager overwrite all 32 presets each time it sends a preset to the BCF/BCR?***

No. When you *send* one or more individual presets to the BCF/BCR, only *those* presets are sent (plus the 'temporary' preset, but that's a different matter). Only when you execute 'Send all data' does BC Manager send all 32 (possibly 'empty') presets to the BCF/BCR.

By contrast, BC Manager's Save operation (in the B-Controls window) does save all 32 presets to any syx/txt/bc2 file it creates (even if some of these presets are 'empty'). However, the 'Export selected preset(s)' operation (in the Presets window) saves only the preset(s) you have selected.

[37] ***How can I save a particular parameter setup made on my BC for my synthesizer? (I.e. the current values (positions) of the various buttons, encoders and faders in a preset.)***

Execute the BC's snapshot function, by pressing EDIT + '◂ PRESET' on the BC, or pressing the 'Snapshot' button in BC Manager. This sends all the current positions of the BC's elements (at least those defining Control Change messages etc.) to BC Manager.

BC Manager can capture such a sequence of CC messages in various ways, and you can then do various things with this sequence. For instance:

1. You can save it as a MIDI file. (Go to the 'MIDI input messages' window for this.)
2. Even better: you can put such a sequence of CC messages in the definition of one BC button, so that you only have to press this button to send all those settings to your synthesizer. In this way you can create whole banks of favorite parameter setups. You could put them under free buttons in the same preset as the one containing the individual parameters, or dedicate separate presets to them.

To do this via BC Manager, in the button dialog box (of a free BC button), go to the 'Custom output' tab and press the 'Record MIDI messages' button (i.e. the round red button on the toolbar). This opens a dialog box saying 'Record MIDI messages'. Then press the 'Snapshot' button, wait for the BC to send the data, and press OK. The CC messages are then assigned to the BC button of your choice — all you have to do is send this button definition to the BC.

[38] ***I pressed Record under the 'LEARN output' tab and executed Snapshot. This came back with 'Messages recorded: 114' and 'Bytes recorded: 229'. After this I click OK, but then I cannot exit the Preset's 'LEARN output' tab. It gives me the option to cancel but not to accept the new line it created from the snapshot. What am I doing wrong here?***

The BC has a limit of 125 bytes for the LEARN output sequence. However, as a courtesy BC Manager allows you to *record* up to *999* bytes: you can edit all these bytes (copy, paste etc.), but you cannot 'save' them as the actual LEARN output sequence, which is why OK is disabled until the number drops to 125 or lower.

[39] ***Sometimes BC Manager produces a sequence of error messages saying things like 'Error: Error preparing MIDI input header. There is no driver installed on your system.' What does this indicate?***

This error message indicates that the USB connection (as maintained by Windows) between the BC and the computer has been lost, i.e. behind BC Manager's back while it has been running. (The reason why you sometimes get the same message repeated many times is that these messages can be triggered by each of BC Manager's 32 MIDI input buffers it tries to set up.)

There are several potential causes for this:

1. You have switched off a BC connected via USB manually while running BC Manager. Any attempt at communication between BC Manager and this BC will then produce this type of error. Beware: once you have switched a USB-connected BC off while BC Manager is running, it doesn't work to just switch the BC back on: you must also exit and restart BC Manager.

2. A sudden connection loss (e.g. in the middle of a preset dump) *might* be due to a problem in Windows, but this isn't very likely. (At least I think I have never actually experienced this.)

3. Your BC's USB controller (actually a part of the BC's CPU) is defective.

   This is not very likely, but definitely possible: about six people in the BC2000 Yahoo group reported this problem over a period of about three years (around 2007-2010). In fact I myself was one of these people: my first BCR kept shutting down the connection in mid-operation. (Mind you: this happened regularly while I was writing *BCMI* and developing BC Manager!) It took me almost a year to conclude that it really was the USB controller on the BCR itself. Then I finally went back to the shop; they soon gave me a new BCR — luckily it was still within the 2-year warranty period. My new BCR has never given me any problems — I didn't even have to reinstall any Windows USB drivers or whatever.

   One way to find out if you really have a faulty USB controller is to try and send the same data via a *standard* MIDI connection: even my BCR with the defective USB controller never had any problems in the S-modes. So switch off both the BC and the computer, disconnect the USB cable between them, and connect the BC to the computer via e.g. the computer sound card's MIDI ports. Put the BC in an appropriate S-mode (i.e. S-3 or S-4) and in BC Manager send your presets to the BC via that connection. If this *does* work, you may indeed have a faulty USB controller, hence need a new BC; but if this doesn't work either, you should probably check your preset settings and the BC's Global Setup parameters.

[40] ***I have tried dumping several presets to my BC using BC Manager. Everything I transfer***

appears to go smoothly, but when I turn the encoders they all have a minimum of 0 and a maximum of 1. I have experimented with changing the default value in the scripts. This seems to work but turning the associated encoder will only change its value by 1 or 2. Any suggestions?

Are you by any chance sending *memory* presets to the BC? Basically the problem with that is that the BC's actual behavior is determined by the *temporary* preset (a.k.a. 'preset 0' in BC Manager). For more explanation, see under 'Editing' in this FAQ section.

[41] ***When I transfer a preset from BC Manager, it doesn't matter what MIDI channel I have set, it always ends up being channel 16 after transferring to the BC. Why is this so? I know that I can change the channel manually on the BC, but that kind of defeats the purpose of BC Manager.***

See the previous question.

[42] ***When I send a preset from BC Manager to my BC, the BC doesn't appear to accept the new values, and some of the button and encoder lights go on, then off again. My preset settings are only set correctly when I press the left PRESET button followed by the right PRESET button. What's going on here?***

When BC Manager sends memory preset *N* to the BC, the following sequence of events take place:

1. BC Manager sends memory preset *N* as such to the BC's temporary preset.
2. BC Manager sends a 'store *N*' command to the BC, ordering the BC to copy its temporary preset (containing your memory preset data) to memory preset *N*.
3. BC Manager sends the temporary preset *as it occurs in BC Manager* to the BC. This aims to restore the temporary preset you were (supposedly!) working on, undoing the effect of step 1 above (which overwrote the temporary preset on the BC with your memory preset data). So if the current temporary preset defined in BC Manager is 'empty', all the lights indeed go off.

After this procedure you can indeed select your memory preset by pressing the BC's two PRESET buttons in sequence, since pressing a PRESET button copies a memory preset's data to the temporary preset.

This reflects how the BC works: the display innocently shows 'P-..' all the time, but in fact the temporary preset is the only preset that you can edit on the BC itself and the only one that ever outputs MIDI data when you move the buttons, encoders and faders. The memory presets can only be accessed by copying them to the temporary preset. And of course, when you select a memory preset, the temporary preset (including any changes you have made) is lost.

The distinction between temporary and memory presets is a common one on synths etc., but always liable to give rise to confusion of the did-I-save-my-changes type. That BC Manager allows you to (seemingly) bypass the temporary preset by sending memory presets directly, is probably a good thing, but I admit that it can be confusing.

But of course the BC *does* respond automatically when you send it the temporary preset ('preset 0') from BC Manager (you can even send individual buttons/encoders of the temporary preset, which is great while you're building up your presets). But then of course if you like these temporary settings, you ultimately have to copy them to a memory preset (either via the STORE button on the BC itself or by means of BC Manager). However, this method is still not ideal; see §7 for the *intended* way of manipulating presets via BC Manager.

[43] ***If the BC requires a preset change to actually select a memory preset you have just sent, wouldn't it be easier if BC Manager sent a MIDI Program Change message after the SysEx send was finished?***

There would be several problems with this:

Curiously, the BC has no (SysEx) command for finding out *which* memory preset is currently selected on the BC. So BC Manager would have to guess the desired memory preset number from the

number of the memory preset that you have just sent to the BC. However, BC Manager's 'Send preset' operation allows you to send a *range* of presets, so how would BC Manager know which of those presets you want to select on the BC (i.e. copy to the BC's temporary preset)?

Another problem: how would BC Manager know that you don't care about any pre-existing data in the temporary preset? (This data would get lost after a MIDI Program Change message selecting a memory preset on the BC.) So in a sense this would involve a totally different scheme of working with the temporary preset: 'implicitly' rather than 'explicitly'. The advantage of the BC Manager's current, explicit scheme is (as mentioned above) that you can work directly with the temporary preset (and its individual elements) as a separate entity.

[44] ***Say I record an arbitrary MIDI message (under 125 bytes) from a device (not necessarily a BCF/BCR) to the MIDI clipboard in BC Manager. Can I then go to a particular element, edit the custom output field and paste the recorded MIDI bytes via Ctrl+Shift+Ins?***

Yes.

Printing

[45] ***Is there a way to save the print strips output to a file, e.g. in prn or bmp format?***
Creating graphical file formats is a very complicated business concerning scaling, fonts etc.

I have no idea how to create prn files. And the main problem with bmp is that the resulting file gets pretty huge if it has to represent a high-resolution (e.g. 600 dpi) image, even in black & white. Lower resolutions, e.g. 150 dpi, provide far less satisfying results: one of the main strengths of the 'print strips' operation is exactly the high font resolution; by contrast, the 'print window image' operation yields a pretty awful resolution in printing terms.

I have experimented with Windows metafiles (wmf/emf), but haven't been able to get things right for those formats. They should be able to provide a very compact representation of the drawn objects (i.e. lines + text, in case of BC element strips), but it has turned out to be very hard to get them right (in terms of scaling etc.), in relationship with the image and word processors that can read metafiles. And besides, a metafile doesn't constitute a one-to-one representation of what you get when you print to the printer directly, so it's a bit dangerous in that sense.

[46] ***If I am working on a computer without a printer or one with several attached, the 'print strips' operation sends the file to whatever the default printer is, but is there a way to have the print strips actually bring up the printer dialog?***
You can deviate from the default printer via BC Manager's main menu: Options ➜ Printer. However, anything you set in that dialog box only holds for the current run of BC Manager: once you restart BC Manager, you get the defaults again.

[47] ***In what format does the 'print strips' operation output the print data?***
Frankly I have no idea how it works. BC Manager just defines a printer object (as defined by a linked programming library BC Manager uses), then it draws a number of lines and characters on that printer's 'canvas'. So I suppose it uses the native printing language of the actual printer being used.