Bliss-Box API Documentation

Enums and eeprom:

Controller enum		Controller cont.	
NO_DETECTION	0		
COL	1	SNES	27
gx4000	2	NES_GUN	28
SATURN_DIGITAL	3	V_BOY	29
A7800	4	NES_ARKANOID	30
VEC	5	WII_CLASSIC	31
A5200	6	WII_MPLUS	32
HPD	7	CDI	33
SATURN_ANALOG	8	SAC	34
GC	9	DC_TWIN	35
ATMARK	10	NES_POWERPAD	36
JAG	11	THREE_DO_ANALOG	37
DRV_CNT	12	GRAVIS_EX	38
WII_NUNCHUK	13	MSSW	39
INTELI	14	HAMMERHEAD	40
DC_ASCI	15	PADDLES	41
DC_PAD	16	BALLY	42
NES	17	ATARI_KEYPAD	43
GC_WHEEL	18		44
N64	19	SPEEK	45
GEN_3	20	PC_GAMEPAD	46
GEN_6	21	SNESS_NTT	47
SMS	22	COL_FLASH_BACK	48
TG16	23	NEO	49
CD32	24	A5200_TB	50
THREE_DO	25	PSX_NEGCON	51
PC_FX	26	PSX_JOGCON	127
		FC_NES	52
		FC_ARKANOID	53
		TG16_6BUTTON	54
		WII_DRUM	55
		ARCADE	56
		SUPERGUN	57
		SATURN_GUN	58
		SMS_GUN	59
		DC_GUN	60
		PADDLES_GEMINI	61
		DC_PAD_RF	62
		FC_POWERPAD	63
	255	ATARI_TB	64
		PSX_DIGITAL	65
Searching mode		TOWNS	
		WII_GUITAR	67
		WII_DJHERO	68
		WII_TABLET	69
			70
		FC_NTT	71
		FAIRC	72
		DC_FP	73
		DC_MARACA	74
		PSX_GUITAR	75
			76
		A7800FB	77
		XE1_AP	78
		COL_WHEEL	79
		PSX_FS	115
		PSX_DS	119
		PSX_DS_GS	121
		PSX_DS2	83

EEPROM DATA 2.0

```
Reserved        x x x x x x x x x x x x x x x x (0 - 15)
1                |major version
          |  |Player number
                |  |  |minor version
                |  |  |  |
                |  |  |  |  |option udlr
                |  |  |  |  |  |analog to dpad
                |  |  |  |  |  |  |Disable autopause
                |  |  |  |  |  |  |  |Disable Hotswap
                |  |  |  |  |  |  |  |  |
                |  |  |  |  |  |  |  |  |
                |  |  |  |  |  |  |  |  |          |--|--|--|--pressure
               x x x x x x x x x x x x x x x x (16-31)
2 mapper data x x x x x x x x x x x x x x x x (32 - 47)
3 mapper data x x x x x x x x x x x x x x x x (48 - 63)
```

EEPROM DATA 3.0

```
1                |major version
                 |  |Player number
                 |  |  |minor version
                 |  |  |  |forced ID /
                 |  |  |  |  |option udlr
                 |  |  |  |  |  |Disable all combo options
                 |  |  |  |  |  |  |Disable autopause
                 |  |  |  |  |  |  |  |Disable Hotswap
                |  |  |  |  |  |  |  |  |Hotkey
                |  |  |  |  |  |  |  |  |  |TURBO
                |  |  |  |  |  |  |  |  |  |  | Product
                |  |  |  |  |  |  |  |  |  |  |
                |  |  |  |  |  |  |  |  |  |  |       |--|--|--|--pressure
               x x x x x x x x x x x x x x x x (0 - 15)
               x x x x x x x x x x x x x x x x (16 - 31)
2 mapper data x x x x x x x x x x x x x x x x (32 - 47)
first byte indicates save.
```

Product Enums - Each product (needs its own USB ID) has an ID

1 4-play
2 Gamer-Pro
3 A.I.R.
4 Bridge
5 BlisSTer
6 Gamer-Pro Advaned

—----------

PAYLOAD_USE for generic transfers ENUM
PAYLOAD_USE_NATIVE 1

			127	**PAYLOAD_USE_RANGE 2**
	…… …… ……	**USB ID** **Vendor 16D0** **ID 0D04** **ID 0D05** **ID 0D06** **ID 0D07**	**Year** **2018** **2018** **2015** **2019**	**Notes** **ID for 0x04 can also be set to 0d05,0d06, and 0d07, this was reserved for player ID. This only applies to Gamer Pro/GP (jr), GPA, 4-Play, and BlisSTer**
Modes bit mask **red items are read only** **Bit 0: autopause state** **Bit 1: atl map bit 0** **Bit 2: atl map bit 1** **Bit 3: Hotswap Disabled** **Bit 4: UDLR mode** **Bit 5: Analog to D-pad Or Disable all combos(3.0)** **Bit 6: Autopause Disabled** **Bit 7: d-pad only mode** **Red items are read only**		**1:0 paused:unpaused** **- atl map bit 0** **- atl map bit 1** **1:0 on:off** **1:0 on:off** **1:0 on:off** **1:0 on:off** **1:0 on:off** **When setting bits, they are absolute. If you set just one bit all others will go off. Set multiple bits for multiple options. For example to set apd and UDLR use 0x50. (0x10 + 0x40 = 0x50)**		Autopause disabled: **Stops the controller from pressing start when unplugged.** **Alt map - there are 4, 0: default, mapping 1, mapping 2 mapping 3 Available on on 4.78 and greater.** Hotswap Disabled: **will not detect controllers when the controller is unplugged, a reset is needed. This is not a desirable features and can cause undesired effects.** UDLR**: this mode makes any analog controller's d-pad respond to analogs instead.** Analog to D-pad: **converts analog data to d-pad buttons for d-pad only controllers (i.e snes) . *NOTE*: *The Bliss-Box will send d-pad only controllers (i.e snes) u,d,l,r buttons to analog by default.*. This is only available on 2.0. 3.0 has an all combo disable feature that turns of the 3.0 cobo buttons options.** **D-pad only mode: This mode is a 1 until the Bliss-Box see analog movement. It can be used to detect d-pad only gamepads.So it is read only**

NOTE: The Bliss-Box uses V-USB and therefore the highest speed is USB 1.1 high speed (not full speed). The v-usb is also not a hardware USB, so most of the code is spent appeasing the host. This works ok for v-USB but causes a few issues for sending data from host to client if the device needs to do anything time sensitive. Many of these controllers need just that. So it is designed to put the controller first and USB control transfers second. It simply will not work the other way around. All the host software needs to know, is that a control transfer can and will fail from time to time. Fortunately all USB API's handle errors. If you get an error, simply try again. Make sure to build in a time out so you do not get stuck in a loop if a real error condition is present.

Get commands	Report ID	Returned Data	info
Get adapter info	0x11	Byte 0: controller type Byte 1: modes bit mask Byte 2: minor version Byte 3: player ID (subtract 3 from result) Byte 4: major version	This get command will return all data for the adapter. Controller, modes.
Get Controller Payload	0x10	Data Byte 0 player ID Byte 1 button row 1 Byte 2 button row 2 Byte 3 button row 3 Byte 4 X main stick Byte 5 Y main stick Byte 6 Z AXIS 1 Byte 7 X Second stick Byte 8 Y Second stick Byte 9 Z AXIS 2 Byte 10 Slider Byte 11 Dial Byte 12 HAT	... This is used for latency improvements. A USB Control transfer is much faster than an interrupt transfer thus this data will come to the software faster but it is not guaranteed ever poll. So there can be an occasional 8ms delay. The normal USB poll for Bliss-Box is every 16ms guaranteed. So this is still going to be the fastest bet but it is not built in to DX so you need to use the API.
Get Pressure PSX	0x15	Byte 0: player ID Byte 1-12: pressure data.	R,L,U,D,/\,0,x,[],L1,R1,L2,R2
Get native response	0x16	Byte 0: player ID Byte 1: **PAYLOAD_USE** Byte 2: Size Byte 3-Size: data	Reply from controller would be limited to 255 currently.

Get EEprom	0x17	Byte 1-49: pressure data. FYI: Player ID is in EEprom.	Returns EEprom as explained above.
Get LCD Data	0x18	Byte 0: player ID Byte 1: -193 LCD data (192)	Gets LCD data stored in the eePROM.
Set Commands Report ID always 0x12	**Command code**	**Data Sent**	**Info**
Set modes/options	0x01	Byte 0: modes bit mask Byte 1: 1=Save 2=restore	If using second byte set Byte 0 to 0xff. Second byte restores defaults or saves to eeprom.
Set turbo speed	0x03	Speed of turbo 100is default rane is 1-254	
Set range selection	0x06	Bit mask 7 bits last is alway off.	1 n64 2 vectre 3 gameport 4-6 not in use
Set Rumble Motor	0x04/0x05	Use 0x04 for const and 0x05 for sine/square 3 bytes can be used. Byte 0, byte 1, byte 2 examples: 0 (this stops all rumble) 1, amount (starts rumble with amount) 2, amount, duration (2 bytes littleEndian) (starts rumble with amount and duration)	If the first byte is a 1 and no amount is given it defaults to 0xff (infinite) If the first byte is 2 Duration defaults to 0xFF and amount to 0xFF if not specified. NOTE: Not all controllers have an amount (n64, gc).and PSX can not produce low value (< 100 or so). Amount ranges from 1-255

Set Player	0x08	Byte 0: player number Byte 1: also resets port	Players can be 1,2,3, or 4 if byte 1 is set, device will reset
Set reset	0x10	none	Resets the Device
Set 4 Psx Pressure Button to extra analogs.	0x20	4 bytes Mapps to HID: 1 Slider 2 Dial 3 z Axis 4 z Rotation	Default: (8) [] -> slider (7) X -> dial (5) L2 -> z Axis (6) R2 -> z Rotation Available are: 1 d pad - R 2 d pad - L 3 d pad - U 4 d pad - D 5 /\ 6 0 7 X 8 [] 9 L1 10 R1 11 L2 12 R2
Set Forced Controller	0x40	Controller ID from **Controller enum**	Must be > 0 (HARMFULL)
Set Hotkey	0x80	Set the pressed button(s) to be the hot key	Must hold down 1 button the the first row.
Send commands	**Command code**	**parameters**	

Send LCD_IMG Report ID 0x14	0x24	None, just raw data.	**Dreamcast only**
Send button mappings Report ID 0x13	0x02	Byte 0: Future use. Byte 1-24: numerical numbers Byte:25 : 1 indicates a save 0 is a temp mapping	Each byte holds the default mapping for the corresponding HID assignment. TIP: The command line tool will also understand a button mapping for ps3 and 360
Generic send (uses send command 0x12) Send Native Data Uses generic transfer type 1 (PAYLOAD_USE_ NATIVE) Send range Data Uses generic transfer type 2 (PAYLOAD_USE_ RANGE) This is just a way to send an arbitrary amount of data. Can be used for anything. There was no reason to make a new RID for it. So it uses the set RID 0x12	0x25 - denotes generic send	A header is sent first, each of the following packets are additional data. Data has a max of 255. Each packet is 5 bytes of data except the first. First packet contains the header. 0:Rid=0x12 1:command=0x25 2: header=0 3:size HI byte | (currently second byte 4:size LO byte is | unsupported) 5:use type (PAYLOAD_USE_NATIVE) 6,7(data 0,1) If size is > 2 another packet must be sent Second line additional data. 0:Rid=0x12 1:command=0x25 2: position, starts on 2 then goes up by 5. use 0xff (255) for the last transfer packet. i.e. (3,8,13...255) 3-7 for remaining data.	Example 0x12,0x25,0,0,7,1,d1,d2 0x12,0x25,1,d3,d4,d5,d6,d7 Uses 1 native send data to controller natively **Supported controllers are n64,gc,psx,DC** See: "Get native response" for fetch data using use 1. 2 n64 range (256 bytes) Sets the new stick range. 3 per map (GPA only) Sets up to 16 ID's for custom mapping. (16 * 8) 128 bytes

C# example code can be seen in this sourceforge project
https://sourceforge.net/projects/bliss-box-api/
See BBAPI.cs send command functions

Command line usage - Use the **-help** option You can find the Command line Interface (CLI) from the link above.