AZ Controller User Guide

Table of Contents

Introduction

AZ Controller is a Control Surface Integration Platform for Cakewalk's SONAR Digital Audio Workstation (DAW).

It is designed to communicate using user definable logic between your Hardware MIDI Controller and SONAR.

With AZ Controller, any device which communicates via MIDI can be integrated with SONAR as deep as SONAR supports.

All DAW software vendors provide in-built functionality to accept commands from midi control surfaces, whether they are specifically designed for controlling DAW parameters or as assignable controls on a MIDI keyboard. With SONAR this is called Active Controller Technology or ACT. Many control surfaces are arbitrarily configurable in what messages they send. But the part "in between" is either configurable but simple (MIDI assign, ACT) or complex but fixed by proprietary communication protocols.

AZ Controller supports arbitrary complexity and it is fully configurable. It is not bound to any particular device. This opens the way to get "perfect" integration for each user without asking Cakewalk or the hardware producer to consider your required functions be added to their product roadmap.

By comparison, Cakewalk's supplied Generic Surface plug-in can also directly map what you touch to some parameter. The supplied ACT MIDI Controller plug-in has built-in logic (Banks, Context, etc.) but this logic is fixed and assumes you have "normal" set of controls (8 rotors, 8 sliders, 8 pads or normal buttons and 1 software shift button). There are other control surface pre-sets available which are designed for (and limited to) specific devices only.

AZ Controller supports arbitrary logic. It can be configured in the same simple way as the Generic Surface through the ACT MIDI Controller, or as complex as proprietary control surfaces and beyond.

Features

Below is a list of the Major features supplied with the AZ controller;

- Flexible (programmable) logic with support for any number of modifiers, banks, etc. and combinations of them as well as time based reaction (double clicks etc.)
- Support for any type and number of controls, including simple knobs, faders, buttons, (touch sensitive) pads (with colour indication), endless encoders (with ring indication), motorized faders and touch strips
- Supports touch functionality (automation overwrite) either in direct (when indicated by the hardware) or emulated (using extra buttons) modes
- Supports message feedback to controllers, including motors, rings, LEDs and displays

- Fixed sized built-in display with user defined information (from fixed text up to parameter values)
- Numerous methods for target strip/parameter selection, including "by name"
- "direct", "catch", "instant control" with response curves and "step" (endless encoders imitation) with configurable resolution for knobs and faders
- Support for different modes of the same physical control
- Visual ACT mapping
- Transparent (direct) ACT operations
- ACT "speed dial" and "save/recall" for quick parameter control
- FX, Synth and Filters (ProChannel) parameters direct control (without ACT)
- Keyboard shortcuts, including keypress sequences
- MIDI messages sent to controller, including dynamic construction of MIDI/SysEx messages
- Jog functionality
- Real time monitoring for input events and executed actions
- Parameter range control
- Reusable "Macros"/functions
- TTS (Text To Speech) support
- 32/64bit, installer/uninstaller
- Supports SONAR up to 2015
- One of the most cryptic user interfaces you have ever seen 😉

Other Features

- All simple/compound MIDI messages are recognized, bulk SysEx can be used as input for buttons.
- Where Am I (WAI) control
- Unlimited number of user definable software states, "value map" mode for selection
- Strip selection by absolute id, current, WAI and by name
- Value change modes: direct, catch, instant, toggle, predefined values, step, endless, ribbon
- All SONAR commands, including legacy commands
- All strip parameters (including Sends) except surround

- Context control
- Automation controls
- Ability to block complete MIDI channels, SysEx and some combinations, per MIDI channel, per device
- Targeted movement between markers
- Parameter range, to control specific part of the whole range or for use with hardware with unusual range settings
- User defined text can be displayed in Control Bar
- Strip level monitoring
- Tunable "Jitter fix" for motorized faders

If you are unfamiliar with logical control configuration (or experience playing Logical Quest games) and you do not want days in an attempt to understand how to change the volume of the first track with the first slider on your surface, please read at least Quick start guide below.

Quick Start Guide

To Use **AZ Controller** you should;

1. Have Cakewalk SONAR installed
2. Connect your Hardware Controller and configure it as MIDI Input device in SONAR
3. Follow the AZ Controller Installation instructions
4. Start SONAR, make a new project (do NOT open your important work now!) and add several audio tracks (more than 8 are recommended to explore the potential of the software)
5. Open SONAR "Preferences", MIDI/Control Surfaces page and add AZ Controller using "Add" button (it looks like a star), select your controller as the "Input device"
6. Check that ACT and WAI check boxes are checked for the "AZ controller - 1". Close "Preferences" dialog
7. Open AZ Controller property page (from menu "Utilities/AZ Controller - 1" or from "ACT Module" Doc)
8. Enter "Quick start" into "Presets:" name, and press save button
9. Turn/move/press controls on your controller and watch "Last MIDI Event" (just under "Preset", on top of the property window)
10. Do you see something is changing there? It should start with "Ch.:" (MIDI channel number). If nothing is changing there, something went wrong or your Controller is in unsupported mode (Mackie). In order to fix that, check that your controller works with "Cakewalk Generic Surface" plug-in. If your controller is working fine with it, but not with my plug-in, please ask me for help in the forum. But if it does not work with "Generic Surface", please use the official forum and/or Cakewalk support.

If the last step was successful, you can proceed with the "simplest possible" configuration:

1. Click on "Options" tab (within AZ Controller Properties window)
2. In "Hardware Controls", after "<New control>", enter "Rotor 1" and press "Save" (right after "Unknown").
3. You should see "Rotor 1" in the "Hardware Controls" first drop box
4. Click on "Hardware" tab
5. You should see "Rotor 1" under "Hardware control"
6. Move a rotor (or slider) on your Hardware Controller, which produces changes in "Last MIDI Event" and the statement shown is not "Unsupported"
7. Click "Attach" and then "Assign MIDI"
8. Under "Logical control" you should now see "(Ch.: ....) Rotor 1 :"

9. Click on "Logic" tab. You should see the same name under "Logical control"
10. In "Action configuration", press "New" button 2 times
11. Select the first "Undefined" Action in the list
12. Under "Action configuration" change "Undefined" to "Strip"
13. Select the second "Undefined" in the Action list
14. Under "Action configuration" change "Undefined" to "Value"
15. Turn your rotor. You should see that the volume of the first track is changing!
16. Save your first AZ Controller preset

Please contact the forum at http://www.azslow.com if you have problems understanding the steps above.

Although other controller software may be simpler to set up the real power of the AZ Controller is the possibility to create complex configurations. You can keep using the Quick Start preset or dive into the next section which explains the rich functionality at your fingertips. Hopefully you will not be disappointed.

The User Interface

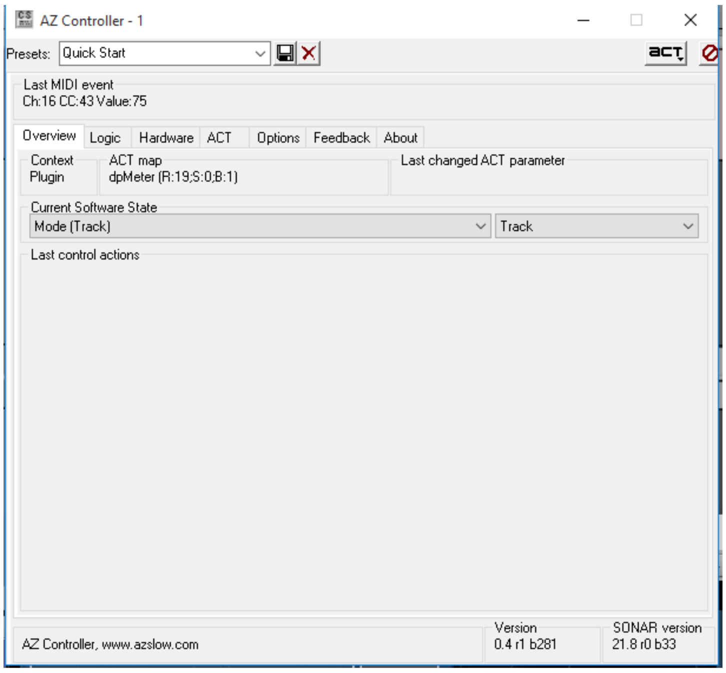
The User Interface Explained.

The User Interface (UI) for AZ Controller is implemented inside the SONAR Control Surface Property Page. It can be called from "Utilities" menu or from ACT Module section of the function ribbon. For more information how to open the window, work with presets and "ACT Learn" please read the SONAR documentation.

The UI was built to make deep configuration possible. It may not win any awards for style but the goal of the plug-in is to offer features not available in the standard SONAR interface and to bridge the gap between SONAR and controllers made by vendors with no native SONAR support.
It is strongly recommended to close the window when you are not changing/checking the configuration. While internal engine is speed optimized, the graphical part is not. The risk of crashing SONAR is increased when the window is open.

The window is divided into "Last MIDI event" indicator, several pages (Tabs), configuration section and pure informational footer. If you need to report any problem with this plug-in, do not forget to mention what you can see in the "Version" and "SONAR version" areas in the lower right corner.

The Overview Tab



Last MIDI event
Ch:16 CC:43 Value:75

Context
Plugin

ACT map
dpMeter (R:19;S:0;B:1)

Last changed ACT parameter

Current Software State
Mode (Track)
Track

Last control actions

AZ Controller, www.azslow.com

Version
0.4 r1 b281

SONAR version
21.8 r0 b33

This page shows what it going on with the plug-in.

On top you can see:

- The current SONAR Context
- Current ACT Map name with the number of detected controls of each kind: (R)otors, (S)liders and Switches(B)
- Last changed ACT parameter (from SONAR direction, so not only what is changed with Controller). The control id, internal value (is always between 0 and 1), the name of changed parameter and the latest value are shown

In the next section you can monitor states of user defined State Sets. You can also manually change the current State (as with changes from performed Actions, the Current State is not saved into preset).

The "Last control actions" section, lists successfully executed Actions from one Action List. If the Action List has no such Actions, the information is not removed. At the moment, selecting an already selected parameter (ACT or Strip) is not counted as successful action. That is why only the Value set action is shown when you continuously modify the same parameter.

Last MIDI Event

When the plug-in receives a MIDI event, it is immediately displayed. Only the very last event is shown. If you press a key on your keyboard and do not keep it pressed, the "Note OFF" event is shown. "Note" (assuming ON) is actually also displayed, but you should keep the key pressed to see it.

The current version of the plug-in recognizes all possible MIDI messages. Known messages are displayed with abbreviation (N(Note), Ch(Channel), CC(Control Change), PC(Program Change), etc.). Most System Exclusive (SysEx) messages are shown in Hex form.

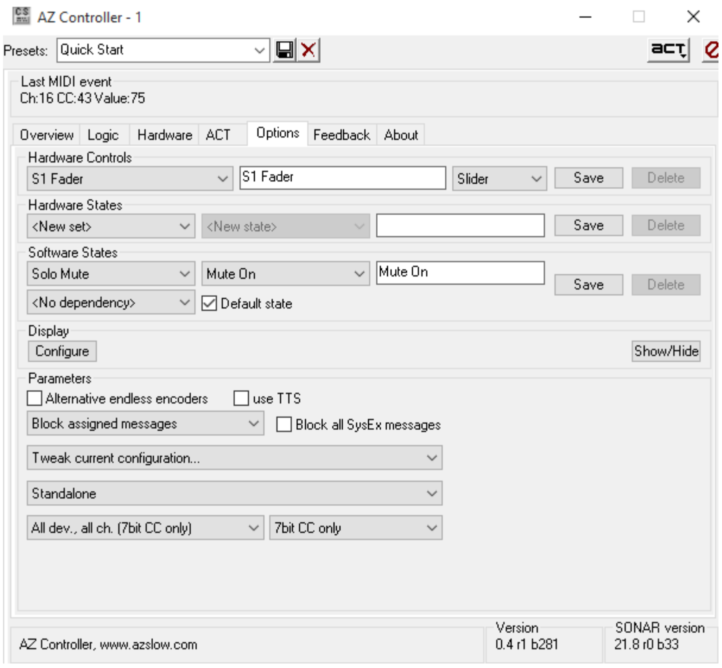
If "OFF" is not displayed, the event is processed with "Note:ON" software state set (explained later).

In some situations, single MIDI messages are not shown as an Event (RPN/NRPN, see Parameters in the Options Tab section).

If the event is assigned to a Logical Control, the information about this Logical Control is also displayed. If the Logical Control is attached to a Hardware Control Context, the Context is shown. The internal Logical Control ID is shown otherwise.

MIDI messages assigned to a "System" Control are still processed normally, but they are not shown as "Last event". If a periodic unwanted MIDI message appears, you can assign it to a "System" Hardware Control (see below). You can at any time, change the control type from "System" to something else to view the event and its associated actions.

The Options Tab


AZ Controller - 1

Presets: Quick Start

Last MIDI event
Ch:16 CC:43 Value:75

Hardware Controls: S1 Fader | S1 Fader | Slider

Hardware States: <New set> | <New state>

Software States: Solo Mute | Mute On | Mute On
<No dependency> | ☑ Default state

Parameters
☐ Alternative endless encoders ☐ use TTS
Block assigned messages ☐ Block all SysEx messages
Tweak current configuration...
Standalone
All dev., all ch. (7bit CC only) | 7bit CC only

AZ Controller, www.azslow.com
Version 0.4 r1 b281
SONAR version 21.8 r0 b33

Unless you are planning to use the plug-in for ACT mapping only (see corresponding section), you should start to configure the AZ Controller here.

All user definable names are entered in this Tab. Several common rules (limitations) apply:

- All named objects are recognized in system by internal (hidden) IDs, not by the text you are entering. The system will not warn you, so for example if you call 5 different rotors "Rotor", they still are recognized as different. It is best to ensure unique names are used to avoid confusion later. It is sometimes perfectly ok to use equal names, but only when the meaning is clear. For example, "Shift" can be "On" and "Off". Also "Alt" can be "On" and "Off". It is recommended to use descriptive, distinguishable but short names. That can save you time and nerves.
- All names are kept in lists. The first name entered will be the last in the list. If you desire to see names in a specific order, please plan in advance. There is (currently) no possibility to change the order. So, for example, if you want to see "Prog1","Prog2" and "Prog3", create them in reverse order as "Prog3", "Prog2" and "Prog1". The last entered Software State is special, see the explanation later.

- Unlike the rest on the interface, entering text and selecting items in combos will not immediately change the configuration. Each section has a separate "Save" button which affect only this section.
- You can delete an object when there are no more references to it. If the object is in use the "Delete" button is greyed out. There is no indication where it is used, but all possible places are predictable. You can modify an object name at any time. It will be shown under the new name once you save the change.
- To create new Control/Set/State just select "<New control/set/state>" in the corresponding combo. Enter the new name in the blank box and press "Save".
- When you create a new Set, a new State "Default" is automatically created in it because any Set must have at least one State.
- You can edit Set name by selecting "<Edit set>" in the State Combo.

Hardware controls

For each Hardware Control which generates a recognizable MIDI Event you should register exactly one "Hardware control" in this plug-in. Even in cases where a control generates more than one MIDI Event depending on the state (see Hardware States). The only reasonable exceptions are normal keys, in cases where your controller is combined with a keyboard as well as any modulators. You can check which controls produce recognizable events using "Last MIDI event" monitoring.

There is yet another important exception when you want create two Hardware Controls for one physical control: in cases where the control is touch sensitive (that is not the same as velocity/pressure sensitive). You can recognize such controls by precisely looking what "Last event" displays.

Once you have touched such a control without moving it, you see one event type (note or CC) but as soon as you start moving/turning it, you see another message type. For example for a slider that sends a message when touched it make sense to define in addition to the standard "Slider1" with a control type "Slider" another such as "Slider1_touch" control with "System" type and assign the first (touch) MIDI event to it. The most practical application for such control will be "Touch 'Slider1'" Action (see the 'Touch' Action description).

If your hardware controller allows that, it is strongly recommended to check that no other controls under any conditions generate exactly the same MIDI Event. If the MIDI channel is different it is ok. If a control generates the same MIDI Event in more than one state that is also not a problem. If you do not follow this rule, "Instant" mode will not work at all and "Catch" mode is not going to work properly.

The correct controller type (Pad/Rotor/Slider) should be set. This type is used when the same events for different types have different meaning.

The “System” type has special interpretation. All MIDI assigned to Logical Controls for Hardware Control with the system type will not be shown in the "Last MIDI event". Action execution progress for these Controls will not appear in the "Overview" tab. That is useful to know in case your controller generates some periodic events on its own ("ping" or some "time code" not blocked by SONAR), this also prevents you seeing (and assigning) normal messages. The whole "protocol handshake" can be "hidden" there as well.

Example: with the MPK Mini you would define 8 pads: "B1"..."B8" and 8 rotors: "R1"...."R8". You should NOT define hardware Banks/mode/prog switches and normal keys.

Hardware States

If there is a Hardware Control which triggers no MIDI Event but changes what other controls send, it should be defined as "Hardware State Set". And all its possible positions should be defined as States in this Set.

There is no reason to define controls which have no influence on what the plug-in receives. For example, arpeggiators, power buttons and such.

Example: for a MPK Mini you would define "Prog" with states of: "Prog1"..."Prog4", "Pad bank" with states of: "Bank1" and "Bank2" and "Pad mode” with states of: "Note","CC","PC".

Software States

There are several system defined sets, which you cannot modify (and so they are not visible in the Options Tab). But they can be used as an Action Condition (see later) the same way as User defined states.

The available system defined software states are;

- **Note:** ON, OFF - indicates either the last event was "Note OFF" or "CC OFF" (the last one is triggered in case associated hardware control is a Pad and CC value was 0) . In all other cases it is "ON". All action conditions assume a pre-defined condition of "Note: ON" which can be changed to "OFF", but cannot be removed

- **MIDI Track:** Yes, No - either currently selected parameter comes from MIDI Track. In case the parameter is not valid or from ACT, the State will be "No"
- **Last Action:** OK, Failed - either previous executed Action was successful. Note that the status correspond to the very last executed Action (so, conditions for that action were ok) and it is overwritten every time. The exception is "Monitor" Actions in the logic list: since they are not really executed, they do not change the status. For example, you can check either "Catch" has already caught the value or not yet. Also you can implement "Is final" on Failure by inserting an "Undefined" Action after the operation in question, with "Is final" set and a "Last Action:Failed" condition.
- **Selection:** Valid, Invalid - indicates that the last parameter selection (ACT or Strip) was successful
- **Transport:** Rec,Play,Stop - current SONAR transport state + additional transport related sets: Loop, Fast forward/Rewind, Scrub, Auto punch
- **Context:** Track,Console,Plugin - current SONAR context
- **Container:** None, UserBin, ProChannel. The plug-in from which the UI container is currently in focus. Note, SoftSynth also has a UserBin type
- **ACT Map:** No Mapping, names of all VSTs/PC modules seen by the plug-in so far
- **Value:** 0...127 - mapped to states last MIDI value

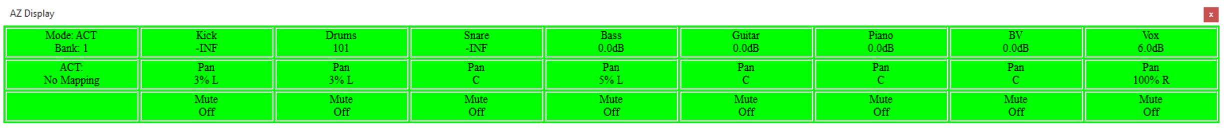
User defined State Sets are not touched and not used by the system. The first State in the list will be Default State. But you can change that (every time you prepend state, it will become default for historical reasons). It is set as current every time SONAR is started or the preset is reloaded. *Note:* when you have just defined a Software State Set, the first defined State (so, the last one once you have entered) becomes the current. When you save and reload the preset, the first state in the list will be the current. So it is recommended to change the state immediately after definition of the whole set to the first (on the Overview Tab). You can Add (prepend) or Append States. You can also append 10 states at once, they will get positional numbers as names prefixed by specified in the text field string.

It is not mandatory to define any User State Set. But defining and using them uncover the real power of logic programming in this plug-in. It is up to your creativity.

Just to give you an idea of what is possible. To implement "ACT MIDI Controller" built in logic, you need "Shift":"On","Off"; "ACT":"On","Off"; "Strip": "Track", "Bus", "Master"; "Button bank": "1".."4", "Rotor bank": "1".."4" and "Slider bank": "1".."4" (see "ACT MIDI Explained").

Advanced: you can declare a user Set as dependent on another user Set. In this case there will be separate "current" State for each State in that other Set. For example, you have "Resolution" with states "Coarse" and "Fine". But you want to remember resolution for each "Mode" ("Mix", "ACT") separately. You can define "Resolution_Mix" and "Resolution_ACT" as two separate Sets. Using such Sets without dependencies would require duplicated actions (like "Mode:Mix - Resolution_Mix:Coarse - Value MIDI", "Mode:ACT - Resolution_ACT:Fine - Value MIDI"). You could say that "Resolution" depends on "Mode" instead. In whatever Action you use "Resolution", then (even in Set State Actions), the current State of "Resolution" corresponding to the current state of "Mode" will be used. State monitors on the dependent Set are also called when the "master" current State is changed, even in cases that do not produce the change in dependent Set (in this example, if "Mode" is changed, State Monitor for "Resolution" will be called even in cases where current states in both modes are the same).

Display


AZ Display

Mode: ACT Bank: 1	Kick -INF	Drums 101	Snare -INF	Bass 0.0dB	Guitar 0.0dB	Piano 0.0dB	BV 0.0dB	Vox 6.0dB
ACT: No Mapping	Pan 3% L	Pan 3% L	Pan C	Pan 5% L	Pan C	Pan C	Pan C	Pan 100% R
	Mute Off	Mute Off	Mute Off	Mute Off	Mute Off	Mute Off	Mute Off	Mute Off

Independent from the Property Page, the "always on top" not re-sizable window can be used to show some information. The information is organized as a matrix of cells. The size of this matrix, the number of characters per cell, the alignment within cells, the font and background color can be customized. Cells can be visually grouped (for example, to use one cell for parameter name and another for the value).

The display can be activated/deactivated either by the button there or by "Display" set of Function Action.

The content of each cell is set separately. See "Display" Action documentation.

Parameters

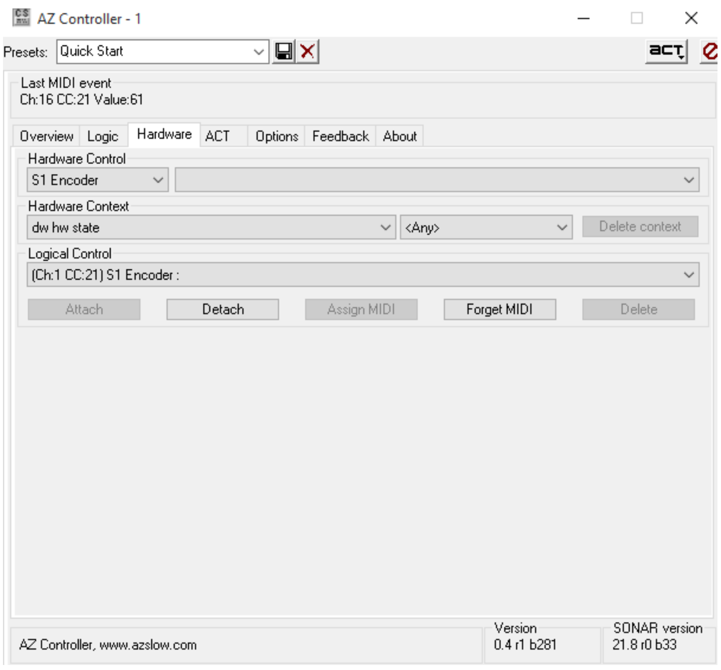
Parameters determine the general behaviour of the plug-in. It is important to set them correctly. The default should work fine for most cases, do not modify anything there until you understand what are you doing.

- Alternative direction for Endless encoders. In short, if required set Reverse flag in Set Value for endless encoders to get correct direction, you can set the flag globally here (and unset it in Set Value). Since some Actions which use Endless encoders (Set State) have no such flag, that is the only way to make them work correctly

- Use TTS (Text To Speech) if possible. At the moment, only SAPI 5 is supported. Use with the "Say" function.
- Ignore MIDI input. Useful in cases where the instance of the plug-in with this flag set is used for sending MIDI commands to external hardware/loop devices. CS Input cannot be left unset in SONAR, so you are forced to set something even if it is unused. If the same device is used by other instance already, it makes sense to set "Block all channels" and "Block all SysEx"
- A different blocking strategy can be used in the next parameter. By default, the plug-in will block only all assigned events with some action specified. If your controller has no keys and you are not going to use "MIDI learn" functionality in SONAR (for the Matrix view for example), it is safe to set "Block all channel messages". That is also good option during the assignment procedure, since in this case messages will not unexpectedly change something in VSTs. Other options are either to block one particular MIDI channel, or all channels except one. The last case is interesting for combined keyboard devices, as you can allow keys on one channel go to the track and block everything else (on other channels). But note that once these options are used, assigned messages will not be blocked explicitly (SONAR limitation). For example, you block channel 1, but your slider is on channel 2. Your slider could "leak" into the track/VST settings.

  Tip: if you use ACT MIDI and Generic CS plug-ins simultaneously for one controller, you can block the controller by adding AZC (as the last!) and setting "Block all..." (and maybe also with "Block all SysEx"), without configuring anything else in AZC.
- Unlike for other messages, SONAR has no fine blocking for SysEx messages. You either ask to block all of them or they all go through. Note that there are some messages which are neither SysEx nor Channel. Current knowledge suggests, they are not currently being sent to plug-ins (but AZC will recognize them in case that is changed in SONAR).
- To simplify reconfiguration it is possible to:
  - Clear all current assignments. Useful for example in cases where you completely changed the mapping on your controller.
  - Reset the configuration. Useful in case you have not created an "Empty" preset.
- Cooperation mode. See separate chapter (toward the end of this manual)
- MIDI interpretation mode (see the explanation at the end if this manual)
- Window (Plug-in Property Page) height

The Hardware Tab



Here we teach the plug-in to understand what is going on with our Hardware Controller.

For each Hardware Control already defined in the Options Tab we should define one or more Hardware Contexts. It is a combination of States from Hardware State Sets in which the Control triggers particular MIDI events.

Each distinct MIDI Event (only channel and type, value is not considered) you are going to use, should have associated Logical Control. While not recommended, several contexts can generate the same MIDI Event. That is why "Context" and "Logical control" are separated.

Using the following procedure you can repeat to set up all your controls in all states, but for test purposes it is recommended to start just with several key controls. For example the MPK mini can have up to 4x2x3x8=192 pad contexts and 8x4=32 rotor contexts. It can take a while to define them all.

- Put the controller into a known to you hardware state (in respect to program,bank,mode,etc)

- Touch a Control, let say Rotor 1. You should see its MIDI code in the "Last MIDI event" monitor
- Select the right name from the "Hardware Control" combo, "Rotor 1", "R1" as you named it before
- Select "<New context>" on the right of the control name (if that is the first context you are going to define, it will already be selected)
- Select "<New control>" in the "Logical control" section
- Touch your Hardware Control again. Has something changed? If yes, this MIDI code is already in use. This special case is explained later.
- Now you should see "Attach" and "Assign MIDI" buttons enabled. Press them both in any sequence. "Attach" will connect Logical Control to the Context and Assign will connect MIDI Event to the Logical Control. Context and Control are created, and so you no longer see "<New context>" and you see something like "(Ch:...) Rotor 1" in Logical Control combo
- If you know that the code this Hardware Control generates depends on a specific Hardware State, select it in the Hardware Context section (first select the Set and then the State). Repeat that for all dependencies. If you want to undo, just select "<Any>" in the corresponding set. If some hardware mode does not change the code the control generates (for example: for the MPK mini, "Pad mode" and "Pad bank" does not change Rotors operation), just keep "<Any>" for such set. The Context name (and a part of the Logical Control name) contains the list of selected Hardware States. For example: "B5 : 'CC' 'Pad bank 2' 'Prog1'", for Pad number 5, within Program 1, Bank2 and CC mode
- Proceed with next Context/Control

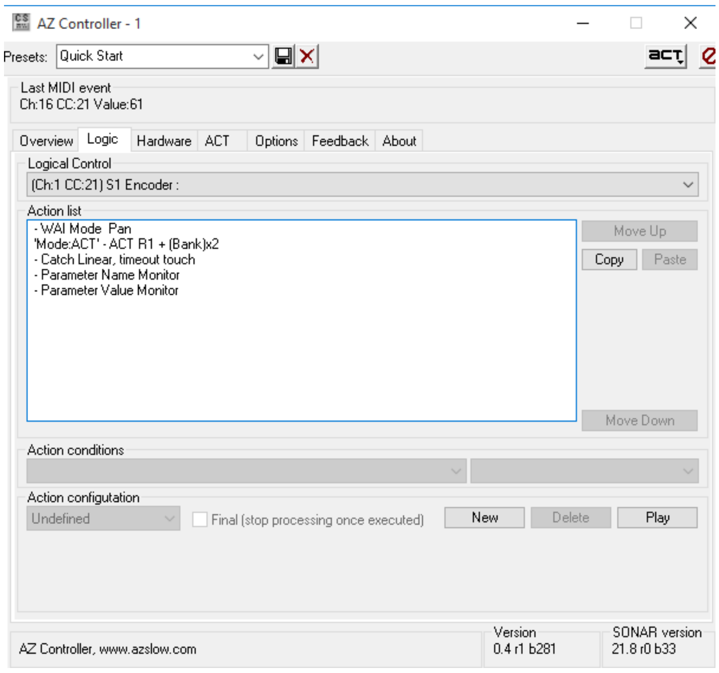
If your Controller does not have any mode switches, so each control can generate one and only one MIDI Event, you just require one Context per control.

In cases where the MIDI Event is already in use; if it is used only in another context of the same Hardware control, there is no problem. Just select "<New context>", find the already defined "Logical control" and press "Attach". Define the context as usual. But in cases where that code is used by another physical control, while you can follow the instructions above, you are looking for trouble. While this is not so bad for buttons, for rotors and sliders you no longer can use instant or catch mode mapping. Not to say the indication in the "Last MIDI event" will be wrong, since the plug-in has no chance to find out which of these controls was already engaged.

Once you have defined Hardware Contexts and Logical Controls for some of your Controls, It is recommended to recheck there are no mistakes. Touch your controls and check that "Last MIDI event" displays your control/context correctly.

There can be some Logical Controls without any MIDI assigned. They can be used to organize control independent monitors (State monitors, Timers) or even some "program" sequence (initial bidirectional communication with some advanced controllers).

The Logic Tab



Ch:16 CC:21 Value:61

(Ch:1 CC:21) S1 Encoder :

- WAI Mode Pan
'Mode:ACT' - ACT R1 + (Bank)x2
- Catch Linear, timeout touch
- Parameter Name Monitor
- Parameter Value Monitor

Undefined

AZ Controller, www.azslow.com

0.4 r1 b281

21.8 r0 b33

Now, when AZ Controller knows what our Hardware Controller is doing, it is time for the most interesting part of the configuration. We can define what the plug-in should do.

While you can select Logical Control in question directly, is it easier just to touch the corresponding Hardware Control. The correct Logical Control is then selected automatically.

If you plan to use some controls as playable MIDI input into SONAR, for example in Matrix or playing drums, do not define any **Action** for them. As soon as you add some action, even "Undefined", the corresponding MIDI code is blocked. SONAR can no longer receive the code (till you clean up the action list).

Pads for example may perform different functions dependent on the midi channel so assignments can be configured on one channel for example Ch.16 but no action defined for the other MIDI channel which is used for triggering samples (usually Ch. 10).

It is possible to change the position of Actions, so unlike state definition you are no longer asked to think "in advance".

If you do not have any Actions in the list, the only enabled button is "New". It adds new "Undefined" action into the list. Every time you press "New" yet another action is added. You can select any action to "Delete", "Move" or modify it. **Warning:** All operations have instant effect, especially dangerous are Monitor actions list for Timer monitors since they are executed without explicit action from your side. It is recommended to temporarily switch the speed of such monitors to "Once" during any action list modifications.

If no Action is currently selected, selecting Action "New" will put the new Action as the first into the list. Otherwise the new Action is inserted right after any currently selected Action.

It is possible to select several sequential actions using Shift+Arrows or mouse with Shift or Control (usual Windows list operations). Move, Delete, Copy and Play will work with the whole selected block. Note that still only one Action is focused, even in cases where you have more Actions selected. Action options, New and Paste operations referencing this focused Action.

"Copy" is a special case. If some Actions are selected, the reference to them is remembered. If no actions are selected, the whole list of Actions is remembered. Note that only the reference is saved. If you change the action after "Copy", the reference will point to the modified Action and not to the original one.

"Paste" inserts marked by "Copy" Actions right after currently focused Action or on top of the list. You can paste Action(s) into the same or another Logical Control. But you cannot paste inside a copied block, so to replicate several Actions inside one list, select the block from top to bottom (so focused action is the last one in selection). Otherwise the last selected in the block action will be the first and paste will not work.

When the Action List is in focus, Ctrl+C, Ctrl+V, Ins and Del keyboard keys will work as shortcuts for the above mentioned operations.

"Play". When one Action is selected, it is executed unconditionally. If there is no selection or several actions are explicitly selected the list is executed with conditions, as it will be when corresponding hardware control is touched. Previous "value" (from whatever control it was) is used and "Note" is always "On". To test the response on some particular input, use MIDI with "loop" option (see corresponding explanation later). When using "Play" in the Monitor list, Monitor conditions (in the corresponding Logical List) are always checked.

For each Action you can define Conditions which must be met. Conditions work on the principle "Current state of Set X is Y", for example "Current state of Transport is Rec". The Action is executed only in cases where all such conditions are True. So, that is the equivalent to "Execute the action if: Transport is Rec AND Context is Console AND Selection is Valid AND etc.". If at least one condition is False, the action is not executed (see the intro into programming on the website). Please note that in cases where an action is to change the State explicitly, that will affect condition checks in any following actions. If an action changes some Sonar State implicitly (for example, you execute "Fast Forward" command), the corresponding State Set is not updated immediately and the following actions in the same list will still see the old State.

You can mark an action as "Final" (check box near the action type). If this action is executed (so the conditions check was passed) AND Action was "successful" (that is action type dependent) AND the "Final" flag is set, the action list processing is stopped. Any following actions are not executed.

The list of possible Actions is long and subject to continuous extension (if you have any special requests, post them in the "Wishes" section of the website):

- **Undefined.** That "Action" does nothing, but is always successful. Add this if you want block MIDI code from some control without defining something else (to avoid unwanted influence on your work in SONAR, some MIDI codes can do almost or completely invisible changes in VST plugin setting). So, it is not altogether useless.
- **Set state.** Change current State in some User defined State Set. "Loop" means switch to the first after the last or to the last before the first. Special is "Map MIDI value". That can make your Rotor function as a software states switch. For example you may find that easier than switching banks using buttons. It is also possible to change the state from MIDI Value, "Next" or "Previous" will be selected based on controller nature (Step, Endless, Ribbon). In Monitors (not during Monitor "dry run" of Logic Actions nor during normal Logic execution) "<Set to monitored>" will set real state from the current temporarily state. "Map parameter value" takes current value of selected parameter and maps the same way as MIDI. Note that if it is used after Value action, it will take it into account. When "Set engine state" flag is set, both temporary and the real states are changed in Monitors. But when "Set engine state" flag is set in the Control Action, the action will not modify temporary states during Monitor "dry run". That way it is possible

to detect "real engaged" controls in monitoring. Please be careful with that option, since it is easy to create a "busy loop" by changing the state in the State Monitor for itself.

- **ACT.** Select ACT parameter. You directly specify which one (see ACT topic). Note, this will just select the parameter, not change it. In addition to the direct number, it is possible to specify the "Shift" Software State set. The shift will be the position of current State within this Set. For example, if the Set "ParShift" has state "0","1","2","3" (in the order as seen in the Overview Tab) and the current state is "2", there will be additional shift 2. The state name plays no role, so if you rename "2" to "OneTwoThree", the shift is still 2 as the position of the state (counted from zero!). You can also specify "multiplier" other then 1. State Set shift plus multiplier are useful for configuring "banks". Just define a "Bank" state set, let say with state "1","2","3". Then you can configure controls as "ACT+R1+'Bank'x8", "ACT+R2+'Bank'x8"...etc. till "ACT+R8+'Bank'x8". To switch bank, you then put "Set State 'Bank' <Next>" action for some button. Please note, that "Bank" 2 really defines shift 1 (counted from zero, so "1" is 0, "2" is 1) and the result is R1+1*8=R9 for the first encoder
- **Strip.** Select Strip and Strip Parameter. You have plenty of possibilities to find the correct strip, with offset and skipping MIDI tracks when desired. Special case is selection by name. For that, define some State Set (for example "Track") and the state with the name in question (for example "VOX"). The plug-in will search for the strip with such a name and will take it as a reference (the subject for shift and skipping MIDI). If the track and parameter are found, "Selection" is set to "Valid" and to "Invalid" otherwise. You can use it in the next Action condition, for example to select a different parameter. If the correct strip is already selected by previous Strip Action, you can change the parameter only by choosing "<No>" as the Strip type. Strip type can be set directly or using the position of current State in some Set. Normally that is going to be "Strip" Software State Set with States: "Track"/"Bus"/"Master" (exactly in that order to have desired effect). "Current" parameter is a read only indication that the strip has the focus (1.0) or not (0.0). "Rude solo" (read only) when set to 1.0 indicates that at least one track is soloed (bus solo does not affect it). "Rude mute" (read/write) is to mute/unmute all tracks, monitored with value 1.0 in case some unseen track is muted. "Rude rec." indicates global Record Arm status. When used with Automation Write monitor, it will indicate global Automation Write status. As with ACT, that action only selects the parameter.
- **Send.** Use "Strip" Action to select the strip first. Leave "Volume" parameter there. The Send is specified by number, as seen in the strip. The number can be calculated as position of the current State in the specified Software State Set. As with other "Select" actions, it does not set the value. Use a "Value" Action afterwards to do this.

- **Value.** Set selected parameter value. It assumes ACT, Strip or Synth parameter selection is done before. All parameters in SONAR are mapped to the range 0.0 to 1.0 (so 0.23, 0.333, etc.), independent from the final parameter (button, switch with 4 positions, volume, etc.). Simple MIDI controls send values from 0 up to 127, or no values at all. The following options to select desired value are provided (where a controller value is used, the Logical Control should be associated with exactly one Hardware Control):
  - **Direct.** Last registered value from Hardware Control is mapped directly or reversed.
  - **Catch.** Hardware Control should first pass current value for parameter. It works as Direct from that point on.
  - **Instant.** Hardware Control starts to change parameter immediately. The change will be proportional to the difference between Hardware Position and real Parameter Position. If the position matches, it starts to function as Direct.
  - **Toggle.** Change parameter value from 0 to 1 and from 1 to 0. Input value is not used, useful for buttons (toggle on/off).
  - **0, 1, or other Absolute value.** Just set specified value (0.5 is exactly "middle" point, for example pan center). Interleave for Buses accept 0.5 for auto detection.
  - **Step.** Special mapping for Rotors and Sliders to imitate endless encoders. When the control is Right/Top from the center, it will increase the value when it is turned right and do nothing when it goes left. When the control is Left/Bottom from the center, it will decrease the value when it is turned left and do nothing when it goes right.
  - **Inc/Dec.** Increment or Decrement parameter value by one "Step".
  - **Endless.** The only option for Endless encoders. Please note that "Acceleration" is a hardware control property, not a software feature. Try different combinations of "Accelerated" and "Reversed" to get correct working result, starting with both unset.
  - **Ribbon.** Apply relative change in value to parameter. Introduced with ribbon strips in mind, to support sliding (should be used with touch events to ignore the first position after touch since it can be arbitrary)

  *Step, Increment and Decrement have a Step Size parameter. MIDI means the step is 1/127, the same as one "tick" in Direct Linear mode. Fine has step size "0.001", close to 10 times more precise then Direct value. You can also specify the tick in percent from the whole range. It is a good idea to either have one dedicated rotor with "Step Fine" or two buttons with "Inc/Dec Fine" to fine tune all parameters. Since the last parameter changed is remembered by plug-in, you set approximate value by other controls and then use these dedicated controls for precision. "Native" step size tries to use the Step size specified by SONAR. Unfortunately, SONAR will either report 0.0025 or nothing at all, so it is useless at the moment.

Direct, Catch and Instant modes use mapping curves, Convex has more precision at the middle (good for the Volume for example) and Concave is opposite to Convex (I have not found good example for it). I tend to think "Instant Convex" works better then "Instant Linear" for me.

Tip: Reverse flag can make a normal rotor/slider act as a cross-fader. The last option is Touch related. The default is "Auto". "Touch" should be used in cooperation with "Touch" Action (see below), and is "Auto" in case the control is not yet touched. Check SONAR documentation for "Timeout" meaning (I have not used it and I do not know what it means).

- **WAI.** SONAR has information as to which strips (for each strip type) are currently controlled by particular plug-in instance. You can see that by color bars near corresponding strips (if there was no Control Surface configured when the project is loaded, you can only see these bars after project reload or SONAR restart). You can configure AZ Controller to operate with arbitrary strips, so that is up to you what to show as WAI. If "width only" flag is set, only the number of strips is set and the current first WAI strip is kept unchanged. If this flag is not set, "Strip" Action should be used prior to WAI Action to select the first strip in range. The strip can be aligned so you can set WAI to fixed positions only (for example, with align set to 4, WAI Action with strips 1 to 4 still set WAI to track 1). Only one Strip Type WAI can be changed by one WAI Action, the strip should exist. But you can change or keep the width for each strip type in one Action. To set "default" width, use "Initial" flag. All WAI Actions with this flag set act as "Set width only" during preset loading and executed unconditionally. If there are several "initial" actions, all of them are applied in the loading sequence (which is in a somehow arbitrary order).
- **FX.** Select FX parameter. Use "Strip" Action to select the strip first. Leave the "Volume" parameter there. The strip in question should have some effects with controllable parameters in the FX Bin (not PC). The first option is the name of FX. All FX used by any track should be listed there. You can also select "<any>". The next parameter is "Shift". If no particular FX is specified, the "Shift" is the FX number in the chain (zero based). In case of a specified FX, "Shift" should be skipped. The current state position of some Software State Set can be used instead of an absolute number. For example, you have "EQ", "Comp","DDD" and again "EQ". To control the second "EQ" you can specify either "<any> 3" or "EQ 1". The third parameter is the parameter number within FX. If you have selected a particular FX and it is currently used somewhere else in the project, its

parameter names are displayed after the number. The order of parameters is the same as in the "Add automation..." list. Parameter "Shift" can be configured the same way as for ACT Action. As other "Select" actions, it does not set the value. Use a "Value" Action afterwards.

- **Command.** Execute a SONAR Command. SONAR expose a list of commands (more than 400!) and the plug-in can ask to execute these. You can put Menu commands in different combinations, just for convenience. All commands have no parameters. Please search in the SONAR documentation what each command does, or you can just try them to see the effect (some of these commands do not produce the expected result, but that is out of the control of the plug-in). At the end of the list, there are all the capital commands with "*" at the beginning. These are not reported as SONAR commands, hardcoded into API. Most of these commands are obsolete and not functional, but some of them are still working "undocumented" features. It would be nice to do a clean up here and give meaningful names to still working commands

- **Functions.** A collection of actions which do not require additional parameters. They are separated from Commands for clarity, Commands are provided by SONAR while Functions are programmed inside the plug-in.
  - **ACT Lock.** Fix the context. See SONAR documentation for details (ACT supports only one context in time, so you can not have two different ACT for two different surfaces).
  - **ACT Learn.** Turn on/off ACT Learn mode. See separate ACT topic in the documentation section and Control Surfaces/ACT board.
  - **Select strip.** The strip should be already specified by the Strip Action. This function makes chosen strip current (selected). This does not change focus between strip types. Tracks and Buses both have the "current" strip, but only one from them is highlighted in SONAR. You still can move the other one, but that will be "invisible" till you change the focus using a mouse (or using "Move input focus..." Command Action from plug-in).
  - **Automations.** Read/Arm automations and Snapshot function. The parameter should be preselected with Strip, FX, Send, ACT or Rack Actions. To turn the automation on for all parameters for the strip, set "Select" to "All" in the previous Strip Action. Synth parameters are not affected by "all parameters" functions.
  - **User Interface.** Show/hide AZC User interface (the Property Page).
  - **Context.** Switch ACT context. "Focus" changes current context, "Open" changes the context and opens the plug-in window, "Close" changes the context and closes the plug-in window (toggling the window is not supported by API). "Next" and "Previous" cycle forward/backward along all FX in the strips. To use "pure"

functions, select the FX with Strip/FX or ACT Actions. "ACT R1" will obviously select the current context ("R1" is safer than "B1" since the parameter should exists in the SONAR ACT mapping). With Strip/FX Actions you can select a particular FX for ACT (with "Focus") and open its window (with "Open"). Examples:

  - **Close** current plug-in and open the next one: "ACT R1", "Close", "Open next"
  - **Open** Channel tools for the first Track in WAI (not closing the current FX): "Strip Track <First in WAI> +0 Volume", "FX 'Channel Tools'", "Function Context Open"

- **Transport.** Direct control for transport related functions. Also move current time to In/Out time of Loop and Punch as well as setting these times from the current time. The result can be different from SONAR commands with the same name. Two major differences that have so far been found: Scrub command does not allow scrubbing with jogger, while Scrub function does; Rewind and Fast forward as commands are not working correctly when the operation was stopped by the time line (the beginning of the project for Rewind). GoTo Data Cursor works in Clip Select/Edit mode only (there is no Data Cursor otherwise).
- **Display.** Show/hide internal Display window.
- **Focus container.** Move focus to one of the strip "containers". Interoperability with the "Focus" function is yet to be understood. In the CW code, the VS700 plug-in only makes use of that method to switch between "User bin" and "ProChannel"
- **All strips operations.** (Dis)arm all tracks, (Un)mute all tracks/buses, (Un)solo all tracks/buses.
- **Update status of plug-in** (which is shown in Control Surface Module on SONAR Control Bar) from current Text
- **ACT last change Scan.** Detect which ACT parameter was changed since the last call. For normal operations this should be put into a timer
- **ACT last change Select.** Select detected by Scan parameter for future processing (Value for example)
- **Focus strip type.** Move focus to specified strip type view.
- **Jitter.** Some endless encoders have a kind of "jitter". They periodically send opposite value even in case you continuously turn the encoder in one direction. This Function, inserted with "Final" flags set before Value Action (it will normally "Fail", till it thinks the value from encoder should be ignored), checks that current value is opposite from the last one, so the first such change is ignored

  - **Select.** "Select clip in centered track" (in track view) nearest to the time line. "Select clip" if project has no folders (does not work otherwise, Sonar limitation) selects the clip on currently focused track
  - **Say.** Use TTS engine (should be enabled in the Options Tab and supported by Windows) to speak current text (see Text Action)
  - **Find FX.** Starting from currently selected Strip (with Strip Action), this function search for the first existing FX till the end of strip type. If found, it is selected (for future Focus/Open/etc. functions), otherwise current selection will be invalid (means FX not found)
- **Keyboard.** Trigger Shortcuts. The first three "keys" are special, they can be used in menus (as specified by API). All other keys can be "pressed" with "Shift", "Alt" and/or "Ctrl" engaged. These modified keys can be also pressed alone (sometimes useful with "Alt"). Key send can be repeated several times (useful for predefined selection in presets). Also current Text (set by Text Action) can be translated into a key sequence.
- **Rack.** Select Synth and the Synth parameter in the Synth Rack. The same rules as for FX selection apply. The selected Synth parameter can be used as any other selected parameter, for subsequent Value, Automation and Context change actions. Parameter "Shift" can be configured the same way as for ACT Action.
- **Filter.** An attempt to directly control Filters (used in ProChannel modules in X). Theoretically the procedure is the same as for FX and Synth. But focus/UI will work only with (to be understood) cooperation with "Focus container". Function and parameter control is extremely buggy. EQ and Compressor parameters do almost work 100%. To control other modules, the strip must be highlighted. That is a SONAR (up to X3e) limitation. Only SONAR provided modules are recognized, (tested on X2 and X3, X1 is untested). Till SONAR is fixed in that respect, this Action should be used with care.
- **MIDI.** Send short midi events back to the controller. "Initial" flag can send the specified command on preset load (unconditionally). This action can be used to switch LEDs, highlight rings near endless encoders and potentially move motorized sliders (untested). All possible midi messages are supported. To simplify configuration, there is"<Use Ctrl MIDI>". The message assigned to the related control will be used in this case (this works for Monitors as well). "Value" as the value to send, will take last value received for normal action or current monitored parameter for Monitor action. Value "0" is sent as "Note OFF" when the message type is "Note". In case it is not desired (so it should send "Note ON Velocity 0"), select "Send 0 as ON". If this instance is the Master in Cooperation mode, it is also possible to send the message to "Slave" instances. See Cooperation mode chapter. When the "Loop" flag is set, the message is looped back to the input of the plug-in instead of devices. That can be used for assigning messages you can not generate any other way (when constructing presets for a device you do not have at the moment),

testing control actions and organize time delayed reactions (like double clicks, see tutorial on the web site)

- **Monitor.** This is not an Action by itself. It establishes the Monitoring Point. Either periodically (for all Monitor types except State Monitor) or on the State Change (for State Monitor), the whole list of Action up to the Monitor Action in question is executed in "Dry run" mode. That creates environment which is identical to what is going to happen in case the Control is activated by an assigned MIDI message (except for control value dependent lists and State changes with "Set engine state" flag set). Note, that all changes during such an execution (state switches, values, etc.) will not be applied in reality. But they are visible in the Monitor Feedback actions. "Note" condition for the Monitoring Action is set in that environment prior the execution, so it is possible to monitor "Note OFF" as well. Once the actions are executed and only in cases where the Monitor Action was "executed" as well (if its Conditions are valid), the decision is taken either of the Feedback Actions for this Monitor should be executed, depending on the Monitor type:
    - *Timer* Monitors are executed unconditionally.
    - For *Parameter Name* Monitor, the system checks whether the Parameter selected at the Monitor Action point is changed. The parameter selection works the same way as for the Value Action
    - *Parameter Value* Monitor is executed in cases where the value of selected Parameter is changed
    - *State Monitor* is activated only in cases where the specified state was changed
    - *Command feedback* is activated in cases where the last executed "static" Action (command, function, etc.) is different.
    - *Level Monitor* is for watching the strip level value. It is then available for "Set State <Map MIDI value>" Monitor actions or other actions which use current value. Note that some Strip parameter should be activated before monitor
    - *Parameter Automation Read* Monitor is executed in cases where Automation Read is changed for the parameter. The value is set to 0/127 (0./1.) when it is disabled/enabled
    - *Parameter Automation Write* Monitor is executed in cases where Automation Write is changed for the parameter. The value is set to 0/127 (0./1.) when it is disabled/enabled

Monitor checks can be "reset" by a special action (see later). So a monitor (or all of them) can be forced to be re-executed.

If the "double trigger" flag is set, Monitor is executed second time even in cases where there was no change. This is useful only in cases where your controller periodically "skips" MIDI command on input.

Each Monitor has a Priority. During one monitoring Cycle, it is guaranteed that Monitors with lower number in priority will be evaluated first. The order inside one priority is random. In cases where Monitor reset/triggers some other Monitor, it can be triggered within the same cycle. With priority specified, you can be sure that is going to happen. For example let say State Monitor priority 0 triggers two other Monitors with Priority 1 and 2.

Then these Monitors will be triggered exactly in that order, within one monitoring cycle. There are rare cases when it is required, normally you can leave the default priority (0 for State Monitors and 1 for all other).

Monitor has a "minimal re-trigger rate" in cycles (~1/13 of a second). It can be that your hardware is not able to accept display (or value) update rates at maximum speed (13 times per second). But you still want to see the first change fast (perceptible update speed is way better then). In that case you can set how fast your device is able to accept data (6 cycles for 2 times per second) while keeping your monitor speed "ultra". The first change will be spotted and displayed as fast as possible, but the next change will be transferred only 1/2 of a second after the first one.

- **SysEx/MIDI.** Send SysEx message or arbitrary MIDI message(s) back to the controller (or Slave controller in Cooperation mode). It can be specified "as is" or in pieces. In the last case, the message construction should be started with "Begin" and ended with "End". Parts can be bulk, from Software Set or from Text (as ASCII). Current state position can be appended as one byte. Also the label of the current state can be appended. In this case, it should follow the same rules as the message: 2 hex digits per byte, bytes can be space separated. In both cases, some numbers can be added to the position or the last byte in label respectively. Only one SysEx message can be constructed in "one go" but in MIDI mode several messages can be specified (auto detected)
- **Jog.** Move time line. Mapping is the same as in "Value" Action. MBT and SMPTE units are supported, with multiplier.
- **Touch.** It is possible to inform SONAR that some parameter is "touched". That is very useful for recording automations with touch sensitive motorized faders. If there is no "touch" (the default is "Auto"), SONAR will only record changes. In cases where some automation already exists, it will change the parameter (and move the fader!) once you are no longer changing the value. So, touch is the way to say "I am controlling the

parameter now, forget what was written before till I am done". Normally, the action is assigned to separate touch MIDI message from the same physical controller ("Touch" on Note ON and "Release" on Note OFF), but in case your control is not touch sensitive, you can assigned that action to some other button. You do not specify the parameter to touch directly, instead you specify which Logical Control you touch. The plug-in will deduce which parameter that control is going to change the same way as for Monitoring, but looking for the first working "Value" Action.

In cases where you forget to release something (some logical bug in you configuration), there is a "Release all" option (release everything touched before). Please note that more then one control can be simultaneously touched and each Touch requires corresponding Release. Also do not forget to set "Manual touch" in the corresponding "Value" Action. Touch will have no effect otherwise.

- **Text.** Text Action prepares the text for future operations (display).
  - Fixed text is just a constant
  - Parameter name will form the string from current selected (or monitored) parameter. "Origin" is the topmost origin of the parameter (strip, synth, ACT context), "Container" is the second order origin (FX inside strip).
  - Parameter value is the current controller value in the Logical Control Action list and retrieved from SONAR current value for the monitored parameter in the Monitor Action List
  - Current time in SONAR (as MBT or SMPTE)
  - Current state of some Software State Set
  - Fixed text specified by HEX numbers. Some hardware displays support special characters which are hard to enter otherwise
  - Current state of some Software State Set, the State name is interpreted as HEX numbers.

  The formed text string is constrained to fitting into specified number of characters. This is achieved by filling with spaces, by trimming, or by "smart" abbreviating. If filling is required, you can specify alignment. After formatting, the result can replace current Text or it is appended/prepended to the current text.

- **Product specific send** sends MIDI/SysEx messages to the controller which are specific for the hardware. At the moment, only Mack.. LC (complete), MCU and C4 (partial) sends are supported. For complete description of these sends, get the official Logic Control User Manual (can be found in the Internet). There are plans to write dedicated tutorial as well. Below is short description only:

  - Hello. Should be sent as the first message to set the device id. This id is remembered and is used in all successive operations. This should normally trigger an "EHLO" response from the device, at which point the plug-in should answer with "Connect". Then the device will then reply either with a "Confirm" or "Error" message. In the first case, the device has accepted the invitation and is in "Online" mode. Some devices require a periodic "Ping" from plug-in to stay online.
  - LCD transfer current text (see Text Action) to display
  - Time/Assign display also transfer text, but to 7 segment displays. The information can be transferred either incremental (changes only) or complete (make sense to do this after device reconnection). Either SysEx or MIDI is supported (not every device support both modes).
  - VPot transfer current/monitored parameter value to the specified endless encoder ring
  - Meter will set meter on the device from strip origin of current/monitored parameter. Other sends are used to configure its behaviour
- **Monitor reset** can reset a remembered value for monitor(s). That effectively trigger Parameter monitor the next time it is checked and State monitor immediately. When used with a particular monitor, it is possible to specify when it should be checked next time. It is possible to say that in cases where the Monitor in question is already planned to be executed, the execution time should not be changed. "As usual" for Once monitors disarm them (but remembered condition is still cleaned)
- **Display draw** current text (see Text Action) into specified cell of the internal display. That information is saved even when display is hidden.

  Tip: there is no "Initial" flag for this action, since it needs cooperation with the Text action. One possible way to draw initial text into the display, is to define a Slow Timer Monitor with an "Initialized:No" condition, output the required text in it and set "Initialied:Yes" afterwards (do not forget "Set engine state").

- **GoTo Marker**. Move "current" time (cursor) to specified Marker. You can select the reference marker (from which the shift is counted) and the shift. In addition to direct absolute shift, you can specify Software State Set based shift (as for parameters in ACT Action). In cases where you specify a particular State, Set based shift is not applied and Marker is searched BY NAME of the selected State. In cases where a "From the end" or "From cursor, backward" are used, Set based shift and search are done in reverse direction (absolute shift is applied directly). For example, "From the end" -1 "XXX" "MyMark" (where "MyMark" is the name of some State from "XXX" Set) will search for Marker "MyMark" starting second from the last marker inside the project (not the last! for that uses "+0"!). "From cursor, forward" "-1" is the same as "Previous Marker" Command, with the same rules applied (in case the current cursor is between markers, it will be moved to the current marker).

- **Range.** This action has a dual purpose. In cases where your hardware controller, for some reason produces strange maximum values, you can set "HW Min/Max" parameters. In cases where you want control only a subrange of some parameter, you can set it (and leave HW empty). All values are specified in range from 0 to 16383 which correspond to the minimum and maximum parameter values in sonar (0 to 1.). The reason for such a "strange" number is that it is the maximum representable within 14 bits, the most precise resolution supported by MIDI (in Pitch Bend messages, for example). Example: let say you want limit the range 0..1 to 0,25...0,75. You should then set Low to 0,25*16383=4096 and Hi to 0,75*16383=12227. If this action is inserted BEFORE a Parameter Monitor Action, during MIDI sending in monitor the reverse calculation is done. Let say you have slider which controls specified range (0.25 .. 0.75) and it is motorized. You obviously expect that the position sent back by the plug-in will be the same as you use to set the parameter.

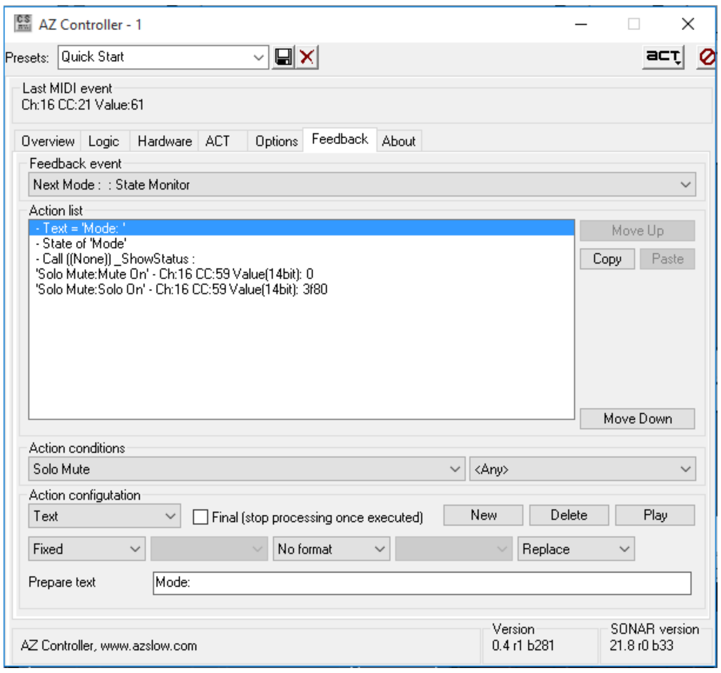
- **Call.** Execute actions from another control. Think about Subroutines in programming or Channel Strip presets in a DAW. This can be used to define some action list in one control and use it from several other controls instead of replicating the same actions. Note that the action is "successful" only when some action with a “Final” flag inside called control was successful, so it is possible to set Final for the call action itself with logical result. Calls can be used from Monitors as well. But the action list is still defined as a "control". Note that in such cases, only actions available in Monitors are going to work, "Logic only" actions (list parameter selection) will not be executed.
- **Text buffer.** There are several global Text buffers which can be used to save/combine/retrieve current Text. For example there are separate Monitors for each channel but you want compose the whole text for Hardware Display. Then each Monitor can save its text into specific part of one Text Buffer and then the whole buffer is transferred to the hardware. Note that "Set" operation sets the current Text from the buffer, while "Replace", "Insert" and "Append" operations update the Text Buffer using current Text.
- **Sync FX.** This will try to synchronize all parameters from one (in case some FX parameter is the peer, set by FX Action) or all FXes of some strip (previously specified by Strip Action) to/from FXes of the strip, specified in this Action. The selection is close to the same as in Strip Action. "Next"/"Previous" means next or previous to the peer strip. "Limit" options specify the maximum number of parameters which are different. If FXes do not match in more parameters, there will be no synchronization attempt. For real

practical use, put this action into Time Monitor and use Strip Name based selections to avoid problems (if you use absolute track numbers, once you insert another track, that action will start synchronize something you were not expecting).

- **Save/Recall current parameter.** Currently selected parameter (the result of Strip/ACT/FX/... Actions) can be saved into one of several slots. You can recall it later. It is also possible to "recall" a currently focused strip. The differences between such recall and Strip <current>: recall set Track/Bus is dependent on focus and recall can be used in Monitors. During recall, it is possible to set some Set into the Strip Type of the parameter (see Strip Action for such Set rules). If there is not sufficient states in this set or no strip parameter is selected (ACT, Synth), the current State in this Set is unchanged. For the meaning of "Set engine state" see Set State Action description.
- **Shuttle.** Turn Fast Forward/ Fast Backwards based on the current State in specified Software State Set. The set should be in the form "<<2", "<<1", "Stop", ">>1", ">>2". If the number of States is Even, there will be no "Stop" triggered by this Action. You can define more states to increase the number of possible "speeds" and change the Step size to adopt the "weight" of each state. Use example with Knob: in the Knob Actions list add "Set State" "From MIDI" <the State set>, "Shuttle" "4" <the State set>. The Knob is a "Shuttle wheel" now. Note: implemented only with FF/RWD Sonar functions, it does not work with Scrub.
- **JitterFix.** In case you have a control (for example Motorized Fader) that periodically sends some values on its own, your parameters can be unexpectedly changed by your surface without your input, which is not nice at all. JitterFix (it has "Success" when the message should be blocked, so use with "Final" flag and before any "Value" or other parameter changing Action) sets up the range within which all changes are ignored. Note that in case you send parameter value back to the control, you should use "<Use Ctrl MIDI>" so the system knows about it. You can change the range at any place, up to 16383 to ignore all values. In case you do not want the input till feedback, send some value, set "Block till set by MIDI" flag, this JitterFix Action will be "successful" till the value is initialized by MIDI Send value.
- **Select.** Sonar has special functionality to select data in the track view. Without a surface, it can be controlled by the Numeric keyboard part. Selection is done by (independent from timeline and currently focused/selected tracks) cursor, which is always vertically centered in the track view upon mode activation. Data can be selected by time or by clip. The selection can be extended. Encoders/knobs can be used (in step, endless and ribbon mode). All relevant parameters can be in parametric mode (using State Sets with two States)
- **Clip.** Selected clip (see previous action) can be modified. Supported are: nudge, crop (beginning/end) and fade (beginning/end)

- **Scroll/Zoom.** To be used with Select/Clip Actions. Scroll/Zoom Track view. Not all combinations of parameters produce a reasonable result, but that is how Sonar reacts to them.

The Feedback Tab



AZ Controller - 1

Presets: Quick Start

Last MIDI event
Ch:16 CC:21 Value:61

Overview | Logic | Hardware | ACT | Options | Feedback | About

Feedback event: Next Mode : : State Monitor

Action list:

```
- Text = 'Mode: '
- State of 'Mode'
- Call ((None)) _ShowStatus :
'Solo Mute:Mute On' - Ch:16 CC:59 Value(14bit): 0
'Solo Mute:Solo On' - Ch:16 CC:59 Value(14bit): 3f80
```

Action conditions: Solo Mute | <Any>

Action configutation: Text | Final (stop processing once executed) | Fixed | No format | Replace

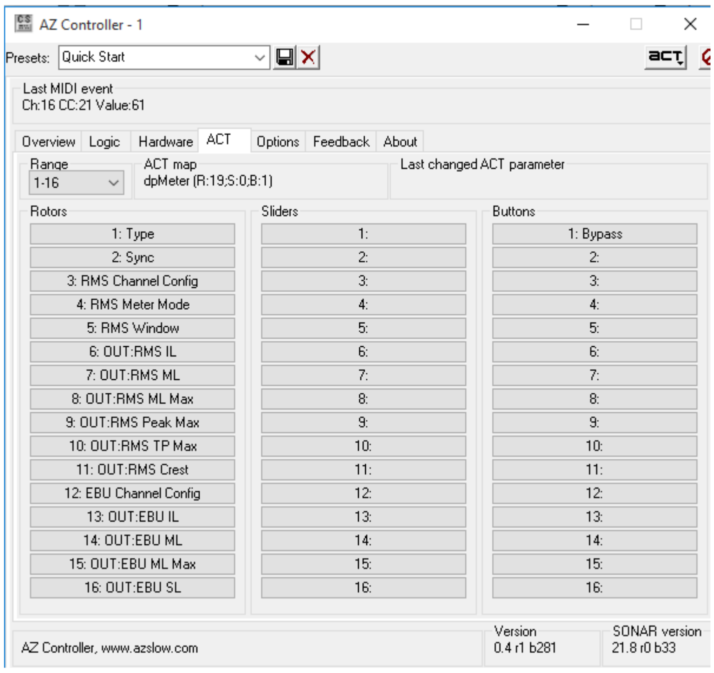
Prepare text: Mode:

AZ Controller, www.azslow.com | Version 0.4 r1 b281 | SONAR version 21.8 r0 b33

For each Monitor Action, a corresponding Feedback Action List can be configured here. Only relevant system states and actions (at the moment MIDI only) are allowed. The interface is the same as for the Logical Control Action list. Please note that depending on the surface, additional actions may be needed for proper indication. For example the MPK midi switch off LED under pad once the pad is depressed. To keep it on (after "Play" command for example), you have to send "On" event on "Button off" in the Logic section (if "Transport" state is "Play"). You can also switch it On/Off here to keep the indication correct when it is changed by other methods.

For Parameter Monitors, current "Value" (can be selected in the MIDI Action) is the value from the monitored Parameter.

The ACT Tab


AZ Controller - 1

Presets: Quick Start

Last MIDI event
Ch:16 CC:21 Value:61

Overview | Logic | Hardware | ACT | Options | Feedback | About

Range: 1-16

ACT map: dpMeter (R:19;S:0;B:1)

Last changed ACT parameter

Rotors	Sliders	Buttons
1: Type	1:	1: Bypass
2: Sync	2:	2:
3: RMS Channel Config	3:	3:
4: RMS Meter Mode	4:	4:
5: RMS Window	5:	5:
6: OUT:RMS IL	6:	6:
7: OUT:RMS ML	7:	7:
8: OUT:RMS ML Max	8:	8:
9: OUT:RMS Peak Max	9:	9:
10: OUT:RMS TP Max	10:	10:
11: OUT:RMS Crest	11:	11:
12: EBU Channel Config	12:	12:
13: OUT:EBU IL	13:	13:
14: OUT:EBU ML	14:	14:
15: OUT:EBU ML Max	15:	15:
16: OUT:EBU SL	16:	16:

AZ Controller, www.azslow.com

Version: 0.4 r1 b281

SONAR version: 21.8 r0 b33

This tab is special. It is not connected with other Tabs and it is not related to your Hardware Controller. This tab shows ACT Mapping for current ACT context. You can browse "pages" with the "Range" combo.

In ACT Learn mode, each parameter is a clickable button. When you press it, the plug-in sends to SONAR the current value of this parameter. This is a convenient way to change your ACT mapping. Since the value is not changed, there is no risk of unintentionally modifying your current settings. And since a Hardware Controller is not required, you can just use it stand alone.

Please read ACT topic on the website for more information.

Cooperation mode

It is possible to control several physical surfaces inside one preset. Two controllers can then work in Cooperation mode. For example, if one of your controllers has buttons and the other encoders, you can "cooperate" them so you can switch what encoders control using buttons. AZ Controller supports one chain of up to 4 devices. One device should be configured as Master, and other as Slaves. Instances of slave devices simply forward MIDI messages to and back from the master.

To switch instance into Master or Slave mode, select the corresponding option in the Options tab. The configuration of Slave instances (except the mode option and events blocking) is then unused. The whole configuration of all devices should be done in the master instance.

Since separate devices can send the same MIDI messages, it is important to distinguish between such messages. Messages from slave instances inside the AZC Property Page are prefixed by "Sx", where "x" is the slave number. Messages from the device connected as the Master instance are not prefixed (shown as in default Stand Alone mode).

<u>Warning:</u> SONAR can be a bit buggy in preset operations for several instances of the same plug-in. When loading/saving a preset in one instance, please hide Property Pages of all other open instances. The preset of the First (in the Control Surface list) instance will be saved otherwise, independent from the window you have pressed the button. Also note that in all instances SONAR will show the same preset name.

So a practical configuration sequence would be:

1. Add required number of instances in the SONAR configuration and attach corresponding devices as Inputs/Outputs
2. For each slave, reset configuration and select "Slave X" in the option Tab. You can save the preset as "Slave X" (manually typing in the preset name field), but that is not mandatory (SONAR always saves current configuration with presets and that "preset" is simple to restore).
3. For the master, manually change the preset name as "Master <whatever>" (you will see "Slave X" in case you saved something for slaves!), just do not forget to do this later.
4. From that moment on, use the Master property page only. Save the Master preset periodically (just in case SONAR corrupts the automatic file, which unfortunately happens in cases of crashes). If you open Slave Property page, you will still see "Master" in the preset name. That is "normal", but do not try to press "save" there.

Incoming MIDI processing

Most MIDI messages are self contained, one message means one event to act on.

But there is one MIDI message type, "Control Change" (or CC, started with Bx byte) which is different. While all CC messages also can be interpreted as self contained with one byte of data (7bit, 0...127), some of them can (and should) be interpreted as a part of a compound message. Note that there is no hint in the message itself that it is either self contained or just a part of something else. Some CC messages can be a part of several (up to 3!) different compound messages.

Another problem is that there are no strict restrictions how compound messages should be delivered. This applies to which order and whether all parts should be transferred every time. MIDI specification explicitly mentions that it is not required in some cases. But hardware producers are free to do what they want.

Terminology:

- MSB - Most Significant Byte. MIDI messages can use up to 2 bytes values, in reality they can use only 7 bit (out of 8) from each byte. That means 2 distinct numbers. Each can be 0..127. The resulting number is <number 1>*128+<number 2>. Where <number 1> is going to be "Most Significant", and so MSB
- LST - Least Significant Byte. It is <number 2> in the previous formula.

To understand how MSB and LSB are working, let say both of them are just a digit from 0 to 9. Putting 2 together, we get 00,01,02...48...97,98,99. The first digit will be more significant than the second. When MSB is received, with a value of let's say 70. LSB has a value of 5. then MSB+LSB = 70+5 = 75.

Any CC can be self contained, but that is the whole list of overlapping CCs. :

- **CC0 - CC31**: can be MSB of 14bit CC compound message, but see later for future use cases
- **CC32 - CC63**: can be LSB of 14bit CC compound message, but see later for future use cases
- **CC96 - CC97**: can be increment/decrement for parameters

- **CC98 - CC99**: can specify the identifier of Non-Registered Parameter Numbers (NRPN) (see later)
- **CC100 - CC101**: can specify the identifier of Registered Parameter Numbers (RPN) (see later)
- **CC6 and CC38**: can set the data for (N)RPNs

As you can see, an extremely hard case is CC6. It can be:

1. Just normal 7bit CC
2. MSB from 14bit CC 6
3. data MSB from (any) RPN
4. data MSB from (any) NRPN

Speaking about the second problem, should we expect CC31 after CC0, even in case we already KNOW they are MSB and LSB for CC 14bit 0?

Lets say MSB and LSB (CC7 and CC39) are 2 digit in the value of Volume in form "-2.8dB". So, CC7=2 means "-2.0dB" and CC39=8 means "-0.8dB"

1. Lets have a look what is going to happened when the fader is always sending both values.
   a. if we react to LSB only, everything is ok. We receive CC7=2 (remember, but not trigger) and then CC39=8, calculate "-2.8dB" and set the volume. Everything as it should be.
   b. if we react to both messages, there is going to be some jumping. We receive CC7=2 and set "-2.0dB", then we receive CC39=8 and set "-2.8dB". That means, continuous moving of this fader is going to produce: "-2.0dB", "-2.4dB", "-2.0dB", "-2.5dB"..."-2.0dB", "-2.9dB". Do you see the problem? It is "jumping" from "-2.9dB" to "-2.0dB" when it is not required. In this case we "keep" LSB instead, we have jumps in reverse direction: "-3.0dB", "-2.0dB", "-2.9dB". Not so many as in the first case, but still.
2. If the fader is NOT sending MSB/LSB until that is required, so it sends CC7=2 and then CC39=1, CC39=2,... CC39=9, CC7=3.
   a. if we react to LSB only, we "skip" some cases when it is not sent.
   b. if we react to both messages, we have the same problem as in 1.b
      So, it looks like there is no "perfect" solution in this case.

With NRPN and RPN the situation is even worse, there are 2 "address" messages before (always the same) CC6 and CC38 with data.

Incoming MIDI processing in AZ Controller

At the moment, AZ Controller has two modes processing assigned messages and three additional modes which make required assignments possible.

The two general modes are:

1. In "flexible" mode, each MIDI message triggers a corresponding event. Like 1.b/2.b in the previous example. And as explained, that is never optimal but somehow works always.
2. In "strict" mode, the plug-in will trigger an event only in cases where corresponding messages are sent in a fixed order and only on the last message in a sequence. For CC 14 bit, the sequence is "MSB then LSB". Fro (N)RPN it is "MSB address, LSB address, MSB data, LSB data".

In general, unless your hardware is extremely tricky, you should use "strict" mode.

The three additional modes to make the assignment possible are:

1. Simple 7 bit. All CC messages are interpreted as self contained. Let say you want assign CC6 as 7bit and CC7+CC39 as 14bit. Without this mode, CC6 will be interpreted as a part of CC6+CC38 14bit CC.
2. Strict, with additional "address" part in data MSB. There is some hardware which use CC 14 bit (so CC6+CC38 for example) to send 7bit data, but MSB of the data is used as an additional identifier. So CC6=2+CC38=9 means that "Volume 2 is -0.9dB" and CC6=3+CC38=7 means that "Volume 3 is -0.7dB" (and not "Volume 6 is -2.9dB" and "Volume 6 is -3.7dB" respectively).
3. The same as previous, but with an identifier in the data LSB

Please note, that once assigned, CC7bit is recognized in ALL modes. 14bit messages (CC/RPN/NRPN) with identifier in data (MSB or LSB) are recognized in all strict 14 bit modes. "Normal" 14bit messages (CC/RPN/NRPN) are recognized in all modes except "simple 7bit".

Which mode is required for my device?

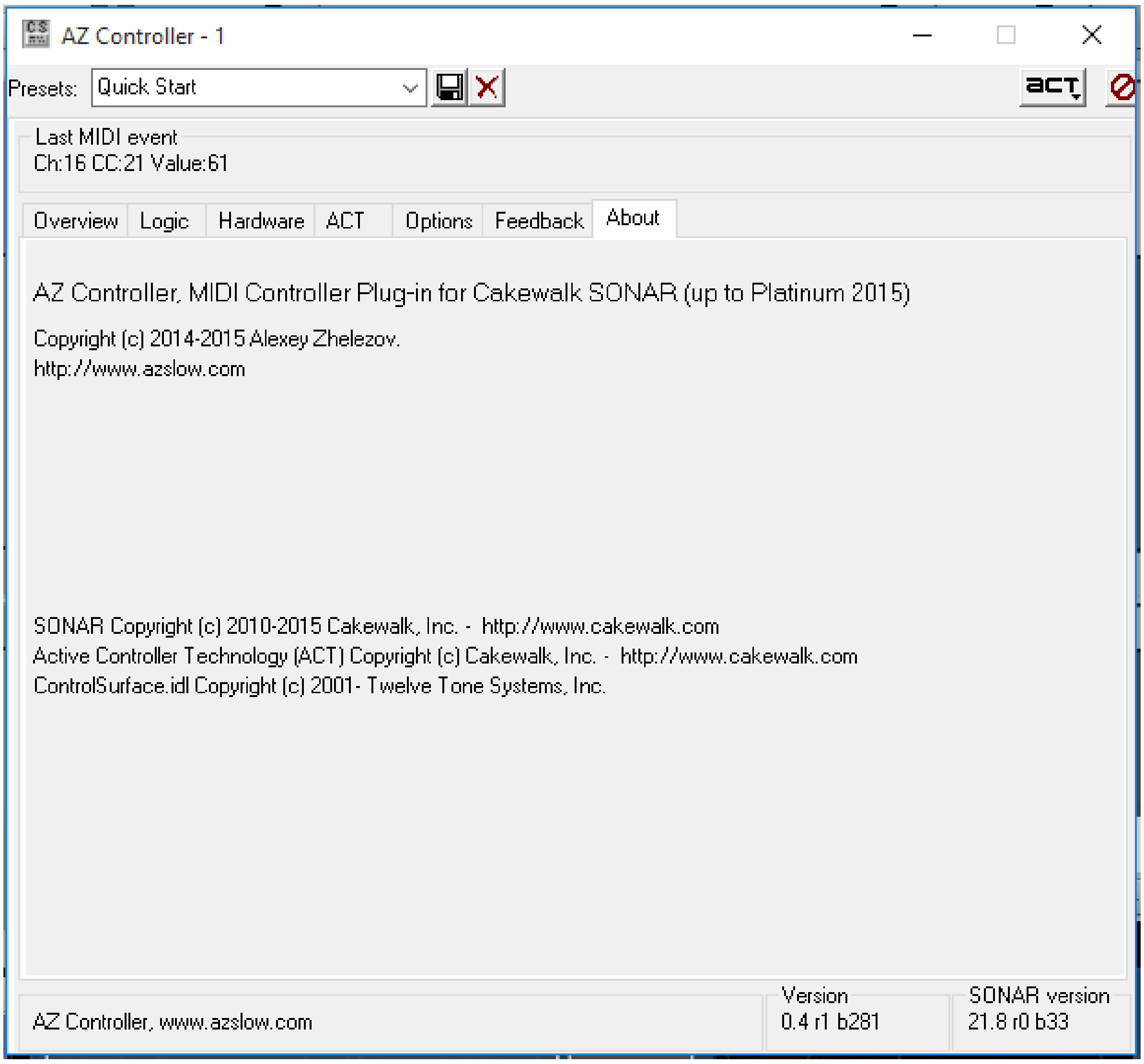
That is device specific. You normally can find that information in the documentation.

Examples:

- Most simple devices as well as some more advanced (MCU) are communicating by 7bit CC only. If your device can be configured to work in such a way, I strongly recommend to do so
- Faderport's fader is sending CC 14bit
- HUI is using CC14 and CC7 bit (even from CC14 range). Also it makes use of "ID in data MSB".
- StudioMix communicates by NRPNs only, fortunately in strict mode
- QU in mixer control mode (not in DAW control mode where it interacts with 7bit CC only!) communicate with NRPNS with "ID in data LSB"

Copyright

All copyright information is displayed in the about tab.
Please do not re-distribute this software for commercial purposes.



Last MIDI event
Ch:16 CC:21 Value:61

AZ Controller, MIDI Controller Plug-in for Cakewalk SONAR (up to Platinum 2015)

Copyright (c) 2014-2015 Alexey Zhelezov.
http://www.azslow.com

SONAR Copyright (c) 2010-2015 Cakewalk, Inc. - http://www.cakewalk.com
Active Controller Technology (ACT) Copyright (c) Cakewalk, Inc. - http://www.cakewalk.com
ControlSurface.idl Copyright (c) 2001- Twelve Tone Systems, Inc.

AZ Controller, www.azslow.com
Version 0.4 r1 b281
SONAR version 21.8 r0 b33