Automatic Audio Switch

Automatic Switching

Mainly the switch will look if there is any signal or not (for optical this means if there is light in the cable or not).

In most cases if there are multiple active sources on the inputs, the left input always has priority over the right input.

However the switch also has the possibility to analyze the signal on the current active input and checks if there is no audio (all bits in the S/PDIF packet zero) or if the packets are marked as invalid (Windows computers typically transmit invalid packets when there is no audio). If there is no valid audio for some seconds (10 seconds per default – may be changed via software) and there is a signal on a lower prioritized input, the switch will mark the current active input as invalid and fall back to the next lower prioritized input. In this case it will stay on the lower prioritized input as long as the signal is lost on this lower prioritized input or until a signal on a higher prioritized input pops up (also when the signal on the input which is marked as invalid is lost and pops up again).

It is also possible to disable the zero and inactive packets detection via software to treat the signal always as valid signal, also when zero or invalid packets are transmitted for some time.

The exact way how you can use the switch depends on how your devices behave, so please take time to read also the use cases to understand the behavior of the switch.

Many devices also have a “keep-alive” option that can be switched on or off – to work well with the switch it is always good to disable the “keep-alive” option, so that the signal is fully stopped when there is no audio.

Typical use case examples

Use Case with RaspberryPi, PlayStation and TV (this is also one of my setups)

Input 1: Raspberry Pi (switches the S/PDIF signal off when there is no audio)

Input 2: PlayStation (switches the S/PDIF signal off when switched off)

Input 4: TV (always transmits a valid audio signal when switched on)

As the TV always transmits a valid signal, it will always switch to the TV when the other two devices are off (a device that always transmits a valid signal only works correctly when plugged into lowest priority).

When the PlayStation is on the switch will switch to it and fall back to the TV when it is off again.

When the Raspberry Pi is on (no matter if the PlayStation is on) the switch will switch to it and fall back to the PlayStation it is on or to the TV if the PlayStation is off.

Use Case with 2 Windows computers (this is also one of my setups)

Input 1: Raspberry Pi (switches the S/PDIF signal off when there is no audio)

Input 3: Windows computer 1 (always transmits a signal, but invalid packets when no audio)

Input 4: Windows computer 2 (always transmits a signal, but invalid packets when no audio)

Let's assume the Raspberry Pi is off, both computers transmit valid audio and computer 1 is currently the active input.

The switch will stay on computer 1 as long as there is a valid signal.

If there is no valid signal for some seconds (10 seconds per default – may be changed via software) the switch will mark the input as invalid and move on to computer 2 (it will not know when the signal of computer 1 gets valid again).

It will then stay on computer 2 as long as there is a valid audio signal here.

When the signal of computer 2 also gets invalid, it will also mark this input as invalid and move on again to computer 1.

It will wait again for some seconds for a valid signal and if there is no valid signal it will move on again to computer 2.

If both computers stop playing audio the switch will toggle between the two inputs with the defined timeout until there is again a valid signal on one of the inputs – it will then stay on this input as long as there is a valid signal here or until a signal on a higher prioritized input pops up.

When the Raspberry Pi finally starts to transmit a signal, it will always override the two toggling computers because of the higher priority and the switch will switch to it. When the Raspberry Pi stops transmitting the switch will fall back again to the two toggling computers.

Use Case with a TV and a Windows computer (this is an example that can't work good)

Input 3: Windows computer (always transmits a signal, but invalid packets when no audio)

Input 4: TV (always transmits a valid audio signal when switched on)

When the TV is on and the computer is off, the switch will switch to the TV.

When the computer is on and transmits audio the switch will switch to the computer.

However, when the audio of the computer stops and the computer transmits invalid packets for some seconds the switch will mark the input as invalid and fall back to the TV.

As the TV always transmits a valid signal it will always stay here and never go back to the computer (only when the signal of the computer fully stops and pops up again – for example on a reboot).

The only thing that could be done here is to disable the zero and invalid packets detection. The switch will then switch to the computer and always stay on the input of the computer as long as there is any signal (no matter if valid or not) and only falls back to the TV again when the computer is fully switched off again.

Status LEDs

Each input of the switch has a 2-color status LED (red/green).

Current active input:	solid
Inactive inputs:	blinking

No signal available:	red
Signal available:	green
Signal available, but invalid or zero:	orange (red+green)

solid red:	current active input without signal
solid green:	current active input with active signal
solid orange:	current active input with signal that is invalid or zero
blinking red:	inactive input without signal
blinking green:	inactive input with signal available
blinking orange:	inactive input where the last signal state was invalid or zero

Alternative LED Mode

The switch also has an alternative LED mode that can be activated with a command over the virtual COM port via USB (or of course with the “Config” button in the client tool).

	Current active input	Inactive input
No signal available	blinking red	LEDs off
Signal available	solid red	solid green
Signal available, but invalid or zero	blinking orange	solid orange

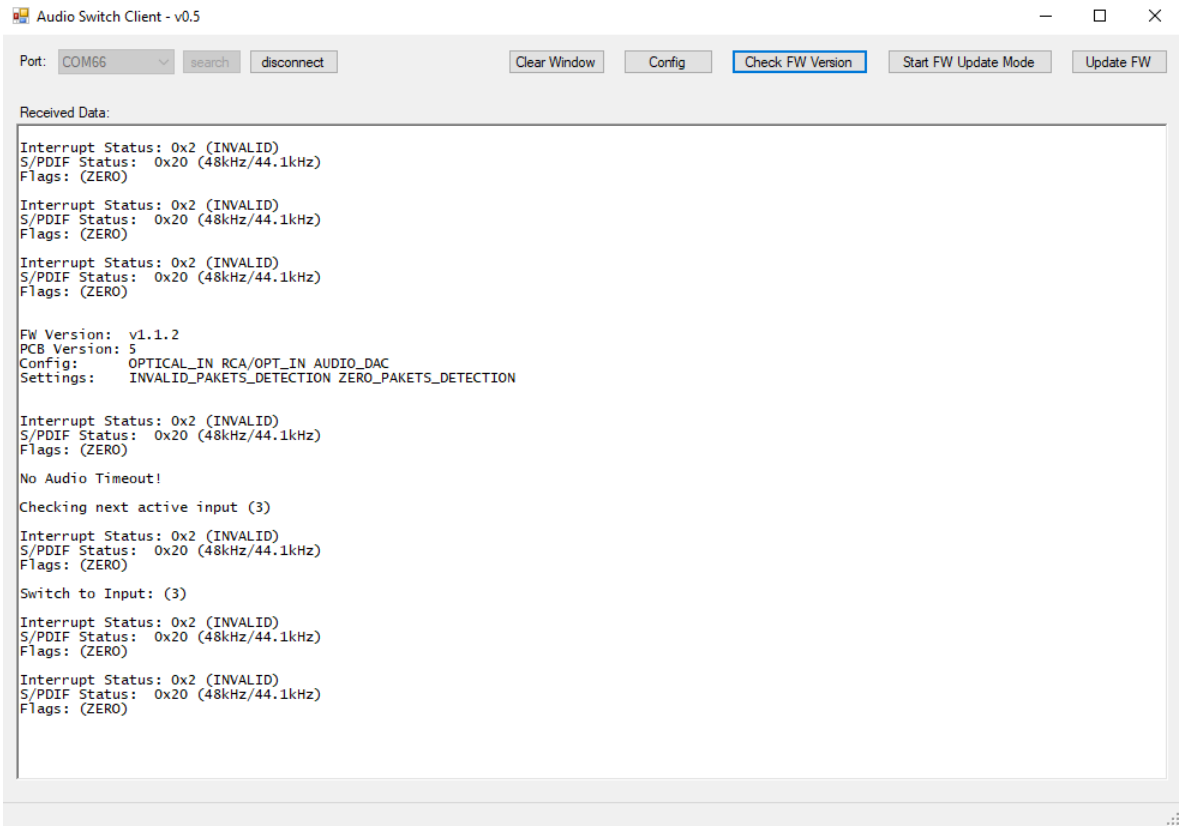
USB Connection

When the switch is plugged to a computer, it will register as virtual COM port.

This is confirmed working on Windows and Linux, but should also be working on Mac (untested).

For Windows a client tool is available (the client tool will ask for admin rights when it is started – unfortunately this is needed to find the correct COM port automatically based on the USB VID/PID).

If Windows doesn't recognize the switch automatically as COM port the download package also contains a .inf driver file that can be selected as driver to point Windows to the correct internal driver to use (the driver itself is part of Windows).



The switch will send status messages every 2 seconds that show the status of the current active input (it is not possible for the audio processor to monitor another input at the same time).

UNLOCK: The S/PDIF receiver cannot lock onto the data stream

INVALID: Received packets are marked as INVALID (no audio)

ZERO: All bits in the received packets are ZERO (no audio)

(Some old hardware versions don't have access to the ZERO flag!)

Commands

It is possible to control some functions or change some settings of the switch by sending commands to it.

All commands start with a '$' and end with a CR or LF ('\r' or '\n').

Command	Description
$FWV\r\n	Check the current firmware version and configuration
$FWU\r\n	Start firmware update mode (Atmel bootloader)
$CLA0\r\n	Disable LEDs on current active input (solid LED)
$CLA1\r\n	Enable LEDs on current active input (solid LED)
$CLI0\r\n	Disable LEDs on inactive inputs (blinking LEDs)
$CLI1\r\n	Enable LEDs on inactive inputs (blinking LEDs)
$CLR0\r\n	Disable LEDs on inactive inputs without signal (only blinking red LEDs)
$CLR1\r\n	Enable LEDs on inactive inputs without signal (only blinking red LEDs)
$CLM0\r\n	Enable Alternative LED Mode
$CLM1\r\n	Disable Alternative LED Mode
$CI0\r\n	Disable invalid packets detection ("I" = big "i" and not a small "L")
$CI1\r\n	Enable invalid packets detection ("I" = big "i" and not a small "L")
$CZ0\r\n	Disable zero packets detection
$CZ1\r\n	Enable zero packets detection
$CT*\r\n	Configure timeout to fall back to a lower prioritized input when invalid or zero packets are received. The byte "*" will be interpreted as number between 51 and 250 which results in a timeout between 1 and 200. For example when "*" is "A" ($CTA\r\n) the number (ASCII) is 66, so the timeout that will be set is 16 seconds.
$I*\r\n	Switch to a specific input ("I" = big "i" and not a small "L") "*" is the number of the input as character. For example when "*" is "2" ($I2\r\n) the switch will switch to input 2. The input numbers are an internal numbering scheme of the switch. For a 4 input switch the inputs are 1-2-3-4 For a 7 input switch the inputs are 1-5-2-6-3-7-4 (when looking onto the switch from the front)

Firmware Update

To avoid unnecessary problems a firmware update is recommended only if you need a special feature of a newer firmware.

Install Atmel Flip

To update the firmware of the switch “Atmel Flip” must be installed (requires also Java).

It is highly recommended to restart the computer after installing “Atmel Flip” (otherwise there are often issues with Java).

This should be done before putting the switch into firmware update mode (otherwise the driver for the bootloader will not be installed automatically).

Bring switch into update mode

Start the client tool and connect to the switch (upper left corner of the client tool).

(The client tool will ask for admin rights when it is started – unfortunately this is needed to find the correct COM port automatically based on the USB VID/PID).

Click “Start FW Update Mode” (upper right corner of the client tool).

The switch will disconnect from the Client tool and start in a special firmware update mode.

Run firmware update

Click “Update FW” in the client tool (upper right corner of the client tool).

(COM port of the switch is not available at this time and the client tool can’t connect!)

Select the firmware file - a console window will pop up and run the update.

When the update is finished, the switch will restart normally.

Sometimes the switch is not recognized after this restart – in this case just unplug it from the computer and plug it in again and the COM port should be available again.

Download Package

The firmware update package contains the following files:

- Installer for Atmel Flip
- .inf Driver File
- Client tool
- Newest firmware file

Download links:

http://tindie.planetinaction.com/AudioSwitchFirmwareUpdatePackageV113.zip