**TaViPaCo - Description of the protocol**

*This file is part of the TaViPaCo (Tangible Virtual Patch Cords)*

*The TaViPaCo can be obtained at* [*http://stefanofasciani.com/tavipaco.html*](http://stefanofasciani.com/tavipaco.html) *TaViPaCo Copyright (C) 2018 Stefano Fasciani, University of Wollongong in Dubai*

*Inquiries: stefanofasciani@stefanofasciani.com*

*The TaViPaCo is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.*

*The TaViPaCo is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Less General Public License for more details.*

*You should have received a copy of the GNU Lesser General Public License along with TaViPaCo. If not, see <http://www.gnu.org/licenses/>.*

*If you use the TaViPaCo or any part of it in any system or publication, please acknowledge its authors by adding a reference to this pubblication:*

*S. Fasciani, H. Rahman, 2018, "Tangible Virtual Patch Cords" in proceedings ofInternational Computer Music Conference 2018, Daegu, Korea.*

**The communication medium**

The Host unit communicates with the client over serial uart.

**Enable/Disable**

During start up, the hardware is initialized but it enters a disable mode and waits for a command from the client to start the scanning and reporting process. A single byte command sent from the client can enable or disable the host. The hardware switch can also disable the host. The hardware enable switch takes precedence over the software command, which means, the serial command will not be able to enable the host if the hardware switch is in a disable state.

**Stored Status values**

The host needs to keep track of changes in the source-sink connectivity, Potentiometer values and the mode switch. These values when scanned are stored within the host unit as data variables in arrays. When the host is enabled, it continuously scans for changes in the hardware values and compares them with the data stored in the corresponding arrays. If there is a change/deviation in the values, the array is update with the new value and a report is sent to the client about the new updated value.

**Reportable Events**

The reports are sent from the host unit to the client on the trigger of either of the three events, i.e. Changes in the Source-Sink Connection/Disconnection, changes in the Potentiometer values or changes in the mode switches. These statuses of these values are stored locally in variable arrays within the host. The host continuously samples the hardware values and compares to that stored in the array. If the value does not match, it indicates a change and a report is sent back to the client notifying of the change and the array value is also update with the new value sampled.

*Event: Changes in Scource-Sink Connection Status*

The microcontroller Atmega2560 has its GPIO pins that that can be set to a state of floating when it must be used as an input pin. If the pin is to be set as a source pin, a single bit of a dedicated register can enable a high impedance buffer of the pin allowing it to be driven high or low. The design of the host supports multiple source and sink pins and can detect the connection and disconnection of one to one as well as one to many connections. This is done by a process where only one source is set to a logic high leaving all the other source pins floating. To detect the connection between a source pin and a sink pin, the selected source pin must be the only source pin set high. The sink pin, even if connected to multiple source pins, will remain unaffected by the other pins which are set to a float state. The selected source pin if driven high, will also drive the sink pin high, which will indicate that a connection exists between the selected source and sink pin, whereas if the sink pin reads logic low will indicate a disconnection. The same process is repeated to find the connection and disconnection between any selected source pin and sink pin. The host iterates through all the source and sink pins to detect their connections statuses. If there is a change in their status, the new status is updated in the variable array and a report is sent to the client informing the changes in the connection status.

*Event: Change in potentiometer values*

The potentiometers values are sampled by the ADC peripheral of the microcontroller of the host which supports a 10-bit resolution. This means the potentiometers when rotated from its minimum position until its maximum position can be read as values corresponding between 0 to 1023. However, the potentiometer is prone to some degree of noise which may cause insignificant fluctuation in the values read by about 1 unit, which will generate an unnecessary reporting to the client. To avoid this, the host accepts a tolerance factor of 2 units. If the sampled value of the potentiometer deviates more that 2 units, only then the host will update its value in the array and send a report to the client.

*Event: Change in switch values*

The only switch shoes status change needs to be reported to the client is the mode switch. The status of the pin connected to the switch is initially stored in a variable and the hardware constantly reads the pin to compare it to the last stored value in the variable. If the switch status is changed, the host reads a different value than that of the variable indicating a change in the switch status. A report is sent to the client and the value of the variable is updated with the new status of the switch.

**Enable/Disable command from client**

The client can enable or disable the host using a single byte command. If the value of this byte is equal to 0, the system is disabled. Any non-zero value sent can enable the system and continue the scanning and reporting process. This is the only command that is sent from the client to the host.

**Host to client Report**

There are three types of reports sent to the client. The report can be that of a change in the mode switch sending a Swtich report, Changes in the source -sink connection status sends an IOMatrix report and changes in the potentiometer value sends a PotVal report. The number of bytes sent varies for each type of report varies. The Mode change report consists of only a single byte only informing the client of the arithmetic operation to be performed between source and the sink. The IOMatrix report packs 3 information’s, i.e. the Source Pin’s Id, the sink pins’ id and the connection status. The PotVal report holds 2 information. i.e. the id of the potentiometer which fits in the first byte of the report and the 10-bit potentiometer value split in another 2 bytes, sending a total of 3 bytes for the entire report.

**New Packet detection**

Some reports are single byte and some reports can be multiple bytes. The client needs to be aware of the incoming of a new packet. The 7th bit of the first is always set high to notify the client of the start of a new report. Bit [5:6] indicates the report type. When the client detects a new packet, it checks the report type and then waits for remaining bytes to be received.

**Incomplete report handling of multiple byte report**

If the client receives the first byte of a multiple byte report, it waits further bytes to complete the reception of the report on its end. However, if there is a missed packet, and the host transmits a new packet, the client will detect it and cancel the ongoing reception of the previous report.

**Report format on changes in Mode**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| **Byte 0** | **1** | **0** | **0** | **-** | **-** | **-** | **-** | **Mode** |

**Mode:** The mode value informs the client of the arithmetic operation it has to perform (0 – addition, 1- multiplication)

**Report format on changes in IOMatrix**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| **Byte 0** | **1** | **1** | **0** | **-** | **-** | **-** | **-** | **Conn** |
| **Byte 1** | **0** | **-** | **-** | **SrcId** | | | | |
| **Byte 2** | **0** | **-** | **-** | **SinkId** | | | | |

**Conn:** the value 1 indicates the source and sink are connected, whereas 0 indicates them being disconnected.   
**ScrId**: The id of the source pin   
**SinkId**: The id of the sink pin

**Report format on changes in PotVal**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| **Byte 0** | **1** | **1** | **1** | **potId** | | | | |
| **Byte 1** | **0** | **-** | **-** | **-** | **-** | **AnalogHVal** | | |
| **Byte 2** | **0** | **AnalogLVal** | | | | | | |

**potId:** The Id of the potentiometer **AnalogHVal:** Byte [10:8] of the Potentiometer value  
**AnalogLVal:** Byte [7:0] of the Potentiometer value