soundart

TECHNICAL DOCUMENT

Communications protocol specification for a custom front panel in the chameleon

Revision 2.1 | 01.2006
Protocol version 1.5

www.SoundArt-hot.com

Table of Contents

Chapter

1 Protocol Implementation

1.1 Introduction

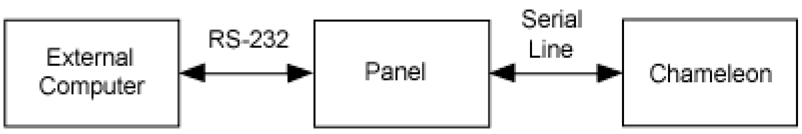
This document describes the communications protocol to implement a compatible control panel for the Chameleon base board. Communication is made between the ColdFire in the Chame

leon side and the front panel, and also from an external computer (using the Chameleon Toolkit) to the custiom panel via RS-232. Custom logic (usually a microcontroller) implemented in the front panel side must to carefully follow this protocol in order to properly exchange information with the Chameleon.

1.2 RS-232 Functionality

When a new custom front panel is connected to the Chameleon base board, the RS-232 functionality to connect the device to an external computer (intended only for allowing application debug messages using the Chameleon Toolkit in the computer side) should be implemented by the custom panel in order to allow the Chameleon hardware to exchange data with the computer. If no RS-232 functionality is desired, all the references to it throughout this document can be ignored.

The custom panel acts as a ‘bridge’ between the Chameleon and the computer, and therefore it has to implement a RS-232 line to connect with the external computer, as shown in the following figure.



Messages sent from from the Chameleon to the external computer are first received by the custom panel logic through the serial line wich connects them. Then the panel will re-sent it to the computer through a RS-232 line. Equally, the messages sent from the computer to the Chameleon are received by the custom panel logic through the RS-232 line, and the panel re-sends these messages to the Chameleon through the serial line, merged with with their own specific communication messages.

1.3 Software Description

The protocol described in this document is compatible with the version 0x12 or above of the boot code installed in the Chameleon. If you are in doubt about this point, please contact Soundart.

The communications protocol between the Chameleon and the panel is based on packets sent and received through a two wire bidirectional asynchronous serial line, at a speed of 57600 baud. Each packet consist of a set of consecutive bytes sent one by one through the serial line. Each byte consist of 8 bits and additionally contains 1 start bit and 1 stop bit, without any parity bit. The communication between the external computer and the panel follows the same protocol, and it is implemented over a RS-232 line with the same baud rate and bit configuration.

All the packets must start with a 2 byte header and a maximum length of **PACKET_SIZE_MAX** (32 bytes including the header).

LEN CMD [··· DATA ···]

The first byte in the header is the total packet length (including the two bytes in the header). The second byte specifies the type of the packet. The remaining data bytes contained in the packet codify different parameters depending on the type of packet, as described below.

An special packet type is **RES_ACK**, which is used to control the communication flow. This control is necessary in the panel since the logic which processes the messages is intended to be the slowest device in the communication, so the Chameleon and the external computer must wait the panel to finish the current command execution before to send another command.

This way, when the Chameleon or the external computer send a packet to the panel, they will wait to receive a **RES_ACK** before to send the next packet, i.e, the panel must send a **RES_ACK** packet for each command it receives.

Commands sent by the Chameleon without meaning for the existing panel should be ignored, but they still require the custom panel to send a **RES_ACK** packet for each command.

1.4 Types of Packets

Packets are divided into several categories: the packets sent from the Chameleon to the panel, which are named with the prefix **CMD** and their command byte is coded with their 6th and 7th bits cleared (i.e. in the range 0x00..0x3F), and the packets sent from the panel to the Chameleon, which are named with the prefix **RES** and their command byte is coded with the 6th bit set and the 7th bit clear (in the range 0x40..0x7F). Commands sent and received by the Chameleon and the external computer through the custom panel have the prefix **SERIAL** and their command byte has its 7th bit set (in the range 0x80..0xFF).

Next subsections explain the possible commands in detail.

```
0x00 - CMD_PANEL_INIT ()
```

Format

0x02 0x00

Description

This packet is sent by the Chameleon to the panel to initialize the data structures and to detect the model and version of the panel. As a response to this command (and independently of the **RES_ACK** that the panel is required to send after each command received), the panel must send the **RES_INFO** packet, to let know the Chameleon which type of panel is connected to the system.

If the panel has potentiometers, the panel must also send a **RES_POT** packet for each of its potentiometers at the receipt of this command, so the Chameleon can initialize the associated values on his side. Likewise, if the panel implements a keypad, it must also send a **RES_KEY** packet to allow the Chameleon to determinate the initial state of the keys (and so for instance to detect some key pressed when the device is switched on).

The **CMD_PANEL_INIT** command can be sent by the Chameleon any time it requires to know the complete panel status.

It is important to note that the Chameleon do not send a **CMD_PANEL_INIT** command at power on or after a system reset. This is due to the fact that the Chameleon is intended to boot up quicker than the panel logic, so it could happen that the first **CMD_PANEL_INIT** was ignored by the panel. For this reason, the panel must send a **RES_INFO** packet (and a **RES_POT** and/or **RES_KEY** packet if it has potentiometers and/or keys) once it is initialized, whithout having received a **CMD_PANEL_INIT** from the Chameleon.

`0x02 - CMD_LED (LED1, LED2, LED3, LED4)`

Format

LEN 0x02 LED1 [LED2 [LED3 [LED4]]]

Description

This packet is sent by the Chameleon to the panel to establish the state of the LEDs present in the panel. The parameter LEDn codifies a different led with each bit. A bit set means that the corresponding led is on, and a bit clear means that the led is off.

LEN indicates the length of the packet and it depends on the numbers of bytes needed to codify all the existing leds. Its minimum value is 3 (8 LEDs) and its maximum is 6 (32 LEDs).

`0x03 - CMD_LCD_CLEAR ()`

Format

0x02 0x03

Description

This packet is sent by the Chameleon to the panel to clear the contents of the LCD display, if there's any.

`0x04 - CMD_LCD_PRINT (POS, TEXT)`

Format

LEN 0x04 POS ··· TEXT ···

Description

This packet is sent by the Chameleon to the panel to display some text in a LCD display, if there's any. LEN is the packet's length and it depends on the length of the text string to be displayed. Its value is LEN=3+trlen(TEXT). The maximum length of TEXT is 30 characters.

POS codifies the position in the LCD where the text is going to be printed. This position is coded into rows and columns. The POS byte is coded as follows:

Bits 0,1,2,3,4 – Column (0..31)

Bits 5,6 – Row (0..3)

`0x05 - CMD_LCD_REDEFINE (CHAR, ROW0...ROW7)`

Format

0x0B 0x05 CHAR ROW0 ROW1 ROW2 ROW3 ROW4 ROW5 ROW6 ROW7

Description

This packet is sent by the Chameleon to redefine the custom pattern of some of the user definable characters in an LCD display, if there is any.

CHAR codifies the character number to be redefined. ROW0…ROW7 are 8 bytes which contain the bitmap of each of the files which made the character. Only the five LSB of each byte are valid, allowing to redefine characters of 8x5 pixels.

`0x10 - CMD_PRIVATE(DATA)`

Format

LEN 0x10 ···DATA···

Description

This packet can be used by the custom panel to receive custom data from the Chameleon. The contents of DATA are filled by the application running in the Chameleon through the use of the panel_out_private() function of the Chameleon Panel API. DATA can't exceed **PACKET_SIZE_MAX** - 2 bytes (30 bytes in total) in length, and it has only meaning for the custom panel and the application running in the Chameleon.

The panel API in the Chameleon will define the panel_out_private() function to be of the form:

```
rtems_boolean panel_out_private(int panel,
rtems_unsigned8 data[30], int length);
```

`0x40 - RES_ACK()`

Format

0x02 0x40

Description

As explained above, this packet is only sent by the panel to control the communications flow. The panel must send a RES_ACK packet after each packet receives, so indicating to the Chameleon that it is ready to receive another packet.

`0x41 - RES_INFO(PANEL, MODEL, SERIAL_NUMBER)`

Format

0x0B 0x41 0x04 PANEL MODEL SERIAL_NUMBER CHECKSUM

Description

This packet is sent by the panel to the Chameleon to inform about the model and version of its OS. PANEL (standard Chameleon #1 panel is 0x04) codifies the different panel types that can exists for different Chameleon versions. This way, the model number will inform unmistakablely about the elements (keys, leds, potentiometers, LCD and incremental encoders) implemented in the panel, allowing the software in the chameleon to execute accordingly to this information depending on the connected panel. MODEL is the model of the Chameleon, 0xA0 is model #1.

SERIAL_NUMBER is the serial number of the Chameleon unit in ASCII, ie. Serial number 123456 would be 0x31 0x32 0x33 0x34 0x35 0x36. The CHECKSUM is calculated from the serial number, it also forms the last two digits of the serial number on the Chameleon unit (MM-SSSSSS-HH, where MM is the Chameleon model, SSSSSS is the serial number and HH is the CHECKSUM).

This packet is sent after a power up or a system reset, once the panel is initialized, and also as a response to the **CMD_PANEL_INIT** command.

`0x42 - RES_POT(POT, VALUE)`

Format

0x04 0x42 POT VALUE

Description

This packet is sent by the panel to the Chameleon when a change in the current value of a potentiometer is detected, after a power up or a system reset and as a response to the **CMD_PANEL_INIT** command (in the two late cases, a **RES_POT** packet must be sent for each potentiometer present in the system).

The POT parameter codifies the index potentiometer whose value has changed.

VALUE indicates the current value of the specified potentiometer. It is a 7 bit unsigned value (ranged into 0x00..0x7F).

`0x43 - RES_KEY(KEY1,KEY2,KEY3,KEY4)`

Format

LEN 0x43 KEY1 [KEY2 [KEY3 [KEY4]]]

Description

This packet is sent by the panel to the Chameleon when a change in the status of any of the present keys in the panel has been detected, after a power up or a system reset and as a response to the **CMD_PANEL_INIT** command.

LEN indicates the length of the packet and it depends on the number of bits needed to codify all the possible keys. The minimum value of LEN is 3 (8 keys) and the maximum 6 (32 keys).

The values of KEYn codify the status of all the keys in the panel, with one bit associated to each key. A bit set means that the corresponding key is pressed and a bity clear means the key is not pressed.

It must be noted that if the **RES_KEY** packet is overriden by the use of a **RES_PRIVATE** message (e.g. to increase the maximum keys available), a **RES_KEY** packet should be sent anyway upon a **CMD_PANEL_INIT** receipt. At least one key of the custom panel should be identified as a “SHIFT” key to allow the Chameleon to boot in Waiting Mode. At least the status of this key (bit 3 in the KEY2 byte) has to be sent in a **RES_KEY** packet after a **CMD_PANEL_INIT**.

0x44 - RES_ENCODER(ENCODER, INCREMENT)

Format

0x04 0x44 ENCODER INCREMENT

Description

This packet is sent by the panel to the Chameleon when a change in the position of some incremental encoder is detected.

ENCODER indicates the index of the encoder whose position has changed. INCREMENT indicates the total value changed since the last **RES_ENCODER** for that encoder was sent. This value is signed in 2 complement. The valid range of increment/decrement in each packet is ranged from -64..+63 (0x40..0x3F).

0x50 - RES_PRIVATE(DATA)

Format

LEN 0x50 ···DATA···

Description

This packet can be used by the custom panel to send custom data to the Chameleon. The contents of data are filled by the custom panel, and the application running in the Chameleon can read it through the use of the panel_in_private() function of the Chameleon Panel API. DATA can't exceed **PACKET_SIZE_MAX** - 2 bytes (30 bytes in total) in length, and it has only meaning for the custom panel and the application running in the Chameleon.

The panel API in the Chameleon will define the panel_in_private() function to be of the form:

```
rtems_boolean panel_in_private(int panel,
rtems_unsigned8 data[30], int *length);
```

`0x80 – SERIAL_LINK(DATA)`

Format

LEN 0x80 ···DATA···

Description

This packet is used in the communications between the Chameleon and the external computer through the custom panel. The panel can receive this message either from its serial line connected to the Chameleon or the RS-232 line connected to the external computer. When the panel receives this packet on one side it must forward it to the other side with the same format. As with any other packet, it must also answer to the sender with a **RES_ACK** once this packet is processed.

The contents of DATA are irrelevant for the panel and only have meaning for the Chameleon and the computer.

`0x81 – SERIAL_LINK_END(DATA)`

Format

LEN 0x81 ···DATA···

Description

This packet is identical to SERIAL_LINK, but it signals the end of a message that can be made of several packets. If a SERIAL message is made of only one packet, this packet will be of type SERIAL_LINK_END.

1.5 Protocol Summary

A summary about how the communication sequence is performed is provided for clarity.

- After a power up, once the panel is initialized it sends to the Chameleon a **RES_INFO** packet, a **RES_POT** for each of its potentiometers (if there's any) and a **RES_KEY** (if there's any key).

- Once the Chameleon knows about the existence of the panel, it will send different commands at run time depending on the requeriments of the current application. For each of these commands, the panel will execute the command (or ignore it if it has no meaning for him) and will send a **RES_ACK** packet for each of them. If the command is of type **CMD_PANEL_INIT**, it will send also a **RES_INFO** packet followed by a RES_POT for each potentiometer present and a **RES_KEY**.

- Every time the panel logic detects a change in the status of a potentiometer, a key or an incremental encoder, it will send the corresponding notification packet to the Chameleon.

- When a system reset is produced, the sequence will restart from the beginning.

1.6 Electrical Specification

RS-232 connection

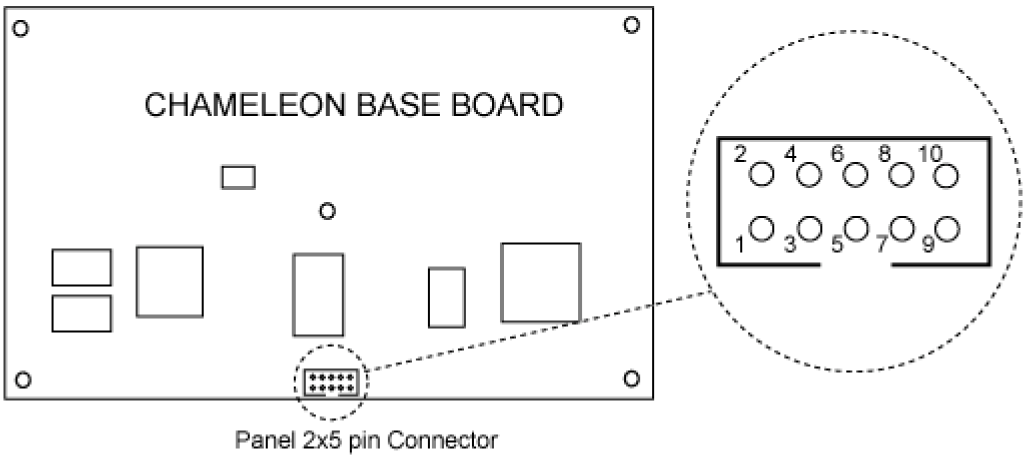
As described above, the custom panel must implement a standard RS-232 line in order to enable the serial communication of the Chameleon and an external computer. The line speed must be set to 57600 baud. This line must be connected to the standard female DB-9 connector placed in the Chameleon rear panel, labeled as 'RS-232 link'. The RS-232 flow control signals are not used and therefore should left unconnected. This connector follows the standard DB-9 pinout for RS-232 without flow control:

- 2: Chameleon TX
- 3: Chameleon RX
- 5: GND

It is recommended to place EMI filters on the TX and RX line, as close as possible to the connector pins.

Chameleon-Panel connection

The Chameleon base board provides a connector to which the custom front panel has to be connected. This connector includes the signals needed to implement the serial protocol and a supply voltage of 5 VDC (500 mA max). The following figure shows the position of this connector in the Chameleon board and its pin-out.



This connector is a standard 2x5 IDC 2,54mm socket. The signals for each pin are the following:

- 1: +5 VDC
- 2: +5 VDC
- 3: GND
- 4: GND
- 5: Chameleon Serial Transmit (Output)
- 6: Chameleon Serial Receive (Input)
- 7: GND
- 8: GND
- 9: Chameleon Reset Ouput
- 10: GND

The Chameleon Transmit and Receive signals are connected to a ColdFire UART port. The logic zero has a 0V level, and the logic one has a +5V level, both referenced to the GND pins of the connector. The custom logic in the panel side must connect the Chameleon Transmit line to its own Receive line, and the Chameleon Receive Line to its Transmit line. The idle state of both lines is a logic one (+5V). For the serial protocol, the Start Bit is a logical zero (0 V) and the Stop Bit is a logical one (+5V). Communication speed is 57600 baud.

The circuitry implemented in the panel can use the 5 VDC power supply pins present at the panel Connector. The only restriction about this is that the circuitry feeded must not exceed a supply current of 500 mA for a proper system operation.

The Chameleon Reset Output signal is an active low reset signal with can be activated by the Chameleon every time that a system reset is required (mainly at the startup and when a new application is downloaded to the device). Custom logic in the panel must reset its circuitry whenever this signal is activated. This pin is usually connected to the RESET input of the microcontroller in the panel. This line is tied to 5V when idle. Once a reset occurs in the system, this pin will be set to 0V for a typical time of 2 ms.