User Guide
UART Protocol, General

BIT-UG-0000_A

This user guide describes the general UART control interface characteristics and operation.

1 Serial Command Message Formats

1.1 Overview

The set of commands and status responses transmitted via the device serial port (UART) is 100% ASCII text-based, and is formatted in such a way that it can be manually entered from and displayed using a terminal program, such as HyperTerminal, RealTerm, or PuTTY.

1.2 Prompts

As is customary with terminal interfaces, the text dialog exists within a framework of prompts and end-of-line characters.

At startup the user is presented with a prompt.

As the user enters characters they are echoed back to the terminal program for display.

When the user presses the <Enter> key at the end of a command or status request, an end-of-line sequence is returned to the terminal and a new line is started.

At this point actions diverge:

– if the user entry was a command, the system then returns a new prompt.
– if the entry was a status request, the system returns a new status line:
(a) starting with a field of spaces, the same width as the prompt
(b) the response data, in text form
(c) end-of-line characters
(d) a new prompt.

To summarize, with user entry contained in curly braces:

```
<prompt> {user_entered_command<CR><LF>}

<prompt> {user_entered_status_request<CR><LF>}

         <status_response><CR><LF>

<prompt>
```

where:

```
<CR> = Carriage return

<LF> = Line feed
```

Pressing a keyboard <Enter> key will insert the <CR><LF> sequence.

The text for <prompt> may be project-dependent.

1.3 General Command and Status Response Format

```
[<ChanChar>-]<Label>[ [<Index> ]<Argument>]
```

`<ChanChar>`

Multi-channel devices use a single ASCII character to identify the channel being addressed. For example, left and right channels are often identified as channels ‘L’ and ‘R’. The general system is always considered to be channel ‘0’, and entering no channel at all implies channel ‘0’. The channel character precedes the Label field, and is separated from it by a hyphen or minus sign.

`<Label>`

Each command or status response is uniquely identified by a short text label. For example, the command to set a display brightness level is typically “BRT”. Labels are kept short, generally under 9 characters. A list of commands is supplied for each product manufactured, and the exact text for the labels will be found in these lists. Labels are case-insensitive, and are customarily displayed in uppercase. The receiving device will typically convert all text to uppercase prior to parsing.

`<Index>`

A message is optionally indexed. Typical uses for indexed messages are accessing internal tables, or accessing the register set in an externally attached I2C device. Indices are sent as unsigned numbers, and may be either decimal or hex, the latter using the standard 0x.. terminology.

`<Argument>`

Most commands and all status responses have an argument value. For the vast majority of messages it is an unsigned integer value. Some arguments however are signed, and some are floating point. Arguments may also be entered as hex (0x..), but this is generally only useful in the unsigned integer case.

1.4 Error Responses

An error response is a simple string describing an error condition.

An error message is typically returned following a command transmission that contains erroneous data, or following an invalid status request. In the latter case the error message is returned in lieu of the status data.

1.5 Command and Status Examples

For these examples the <Label> field is shown as "LABEL".

Command, no data:

```
LABEL
```

Command, Unsigned data:

```
LABEL 22
```

or

```
LABEL 0x16
```

Command, Floating point data:

```
LABEL 2.8
```

Command, Indexed, Signed data:

```
LABEL 100 -22
```

Status Request:

```
LABEL
```

Status Request, Indexed:

```
LABEL 100
```

Status Request, Indexed, Request to return all values sequentially:

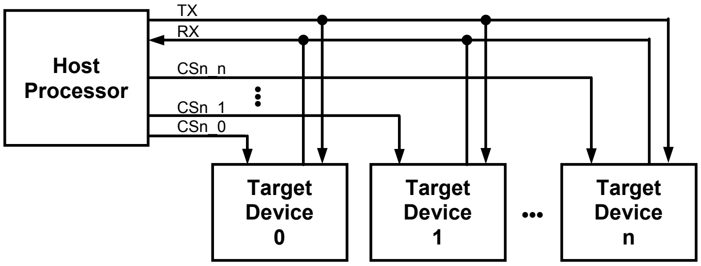
```
LABEL
```

The final example will produce an individual response line for each and every value in the indexed array.

1.6 UART Chip Select (CSn)

The Chip Select (CSn) function, when present, supports multiple target devices on a common UART bus similar to an SPI bus. Asserting a target device's CSn (logic low) activates its UART interface, deasserting its CSn (logic high) deactivates and tri-states the UART interface. No automatic bus-switch arbitration is provided, bus switch timing management to prevent corrupted serial messaging is the responsibility of the user.

Multiple target device UART topology is illustrated below.



Contact

Bild Innovative Technology LLC
2656 W Montrose Ave Suite 203
Chicago, IL 60618
(773) 697-9345
(760) 450-6633
pbeerdeke@bildinnotech.com