Medication Container Serial Protocol

1 Protocol Specification

1.0 Overview

The protocol described by this document applies directly to a programmable medication container with four storage bins and the ability to store multiple doses per bin. Bins are the highest level entities, followed by doses, medication, and description strings. For protocol conversation, A stands for application commands and D stands for device responses.

1.1 Initiating Communication

In all cases the terminal or PC application must act first, thereby activating a serial interrupt for starting communication with the device. Initiation includes the following conversation:

A: US\0 - Upload Schedule Command
D: OK\0 - Ready to proceed response

Following the initiation, the protocol immediately begins transmitting the first bin descriptor outlined in section 1.2.

1.2 Bin Descriptors

Each bin is set to a specified time and includes some arbitrary number of doses. Three fields, hour, minute, and num_doses, are used to fully qualify a bin followed by the corresponding dose descriptors.

A: B\0 - Bin header
D: OK\0 - Ready to proceed
A: hour# - single char specifying hour in military time (0-23)
A: min# - single char specifying minute (0-59)
D: OK\0 - Ready to proceed
A: #doses - single char specifying number of doses for this bin
D: OK\0 - ready to proceed

A bin descriptor is immediately followed by its corresponding dose descriptors (1.3), numbering exactly that specified by the #doses transmission.

1.3 Dose Descriptors

Doses contain information pertaining to specific medication instance during the device's time frame. Therefore, to avoid redundancy, doses are transmitted as a fix point amount decimal number and a medication descriptor pointer to be specified later.

A: D\0 - Dose header
D: OK\0 - Ready to proceed
A: #amnt - char representing a decimal amount*10
A: med* - medication descriptor pointer
D: OK\0 - Ready to proceed

Dose descriptors are transmitted until the number specified in the bin descriptor is reached. Once complete, a new bin descriptor begins unless the fourth begin has already been described, in which case the medication header (1.4) is transmitted.

1.4 Medication Section Header

The medication section includes an arbitrary number of medication descriptors and therefore requires a count variable for the simplest possible storage. This number is specified before any medication descriptors are transmitted.

A: MH\0 - Medication portion header
D: OK\0 - Ready to proceed
A: #meds - single byte specifying total number of medications

The medication section header is always followed by the number of medication descriptors (1.5) specified in the section header.

1.5 Medication Descriptors

Medication descriptors happen to be the most complex entity necessary for scheduling. The string describing the medication is referenced by index, along with four special medication attributes agreed upon between the device and application prior to compilation. A medication descriptor conversation takes place as so:

A: M\0 - Medication header
D: OK\0 - Ready to proceed
A: name* - single byte: string name pointer
A: color* - single byte: color pointer (preset)
A: shape* - single byte: shape pointer(preset)
A: spec1* - single byte: special pointer(preset)
A: spec2* - single byte: special pointer(preset)
D: OK\0 - Ready to proceed

Once the number of medication descriptors transmitted matches the number specified in the medication section header, the protocol continues on to begin string name transmissions.

1.6 String Names

After all descriptors and headers have been transmitted, the string names list may be sent to the device. Each string is sent sequentially followed by a null terminated. The end of the list is indicated by two null characters immediately following one another.

A: S\0 - string section header
A: name\0 - arbitrary string of chars
... repeat ...
A: \0\0 - end of string transmission
D: OK\0

Following string names transmission the user may choose to verify the newly downloaded schedule or simply disconnect the device.

2 Data Verification

2.0 Overview

An application or terminal user can randomly verify the device data with regard to the planned medication schedule, perhaps stored on the interfacing PC. While verification is highly recommended after each download, it is not required under the protocol specification due to the reliability of the hardware interface. Verification mainly involves a backwards transmission of the schedule from the device to the PC application. The protocol does not provide verification functionality, it merely supplies the data necessary for the check.

2.1 Verification Initialization

The verification process begins with a unique header indicating to the device than a PC application wishes to check its data version against that of the device. This transmission interrupts the device operation and begins transmission immediately.

A: VD\0 - verify data command
D: OK\0 - ready to proceed

The device then initiates a reverse schedule protocol starting with bin descriptors as described in 2.2.

2.2 Reverse Schedule Protocol

Rather than designing an entirely separate protocol for verification, a reverse schedule protocol is used. For example, once verification is initialized, the device begins acting as the application would during the transmission of a bin descriptor. Likewise, the application plays the role of the device while simultaneously verifying each piece of data and noting any discrepancies.

3 Potential Revisions

- remove the “OK\0” ready to proceed command in favor of command echoing
- transmit verification data in a format more suitable to the user application