Weeder Technologies

90-A Beal Pkwy NW, Fort Walton Beach, FL 32548 www.weedtech.com 850-863-5723

AN200

Host Programming Guidelines

Author: Terry J. Weeder

INTRODUCTION

To communicate with and control the stackable data modules from a host, simple ASCII character strings are used. The command sets supported by these modules consist of groups of characters that form data packets. The syntax of these packets are specifically designed for user simplicity as well as leanness to minimizing the load demand on the communications bus. This application note explains some of the details of the communications protocol and examines standard practices which should be observed when setting up software control routines using a PC, laptop, or single-board computer as a host.

DATA PACKET FORMAT

A command string used for control or data transfer between the host and a stackable data module is delivered in the form of a packet made up of standard ASCII characters. To reduce the size and thus minimize data bus congestion, the packets are constructed in a manner which eliminates the need for delimiters separating the different fields of the packet. With the exception of the numerical data field, all fields are confined to a single character.

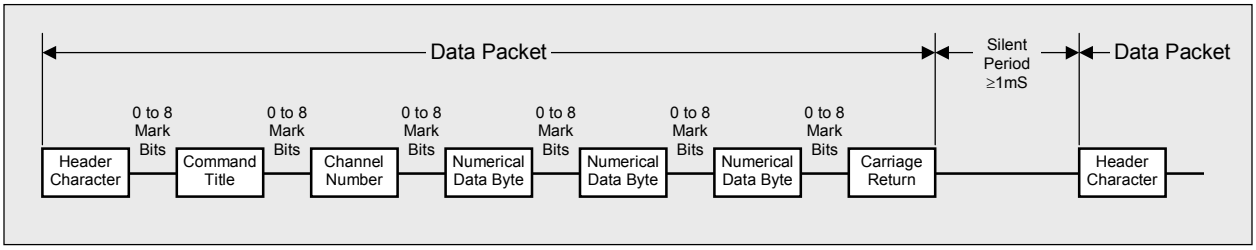
The first field in a packet will always be the header character which denotes the physical address of the module transmitting or being targeted by the host. A packet is always terminated with a carriage return (0D Hex or 13 Dec). The actual length of the packet will vary from command to command depending on the amount of data that must be transferred to perform the desired function. The field following the header will be an upper-case letter indicating the command title. In cases where the command can be directed to more then one channel on a board, an additional field immediately following the command title will contain a character which indicates the channel number associated with the command. Any numerical data will always be transmitted in the last field of the packet and will be in ASCII format.

PACKET TIMING RESTRICTIONS

As shown in figure 1, each character or byte in a data packet should be transmitted sequentially with no more then 8 mark bits (833 µS) separation between individual bytes throughout the entire packet. Furthermore, there should be at least a 1 mS silent period between all data packet transmissions regardless of the direction the packet is traveling on the communications line. These two stipulations are required to fulfill the multi-drop, anti-collision technique used by the communications protocol at the physical layer.

Although the separation between individual bytes in a packet should conform to that mentioned above, in actuality, a data module could successfully receive a packet which

FIGURE 1: DATA PACKET FORMAT



Note: The number of data bytes between the Command Title and the Carriage Return will vary depending on command.

contains a silent period of up to approximately 1.2 seconds. Anything longer however, will cause the internal watchdog timer to expire, in which case the error message will be returned to the host. **CAUTION:** Violating the maximum byte separation within a packet will cause problems if more then one data module is connected in the network. This is due to the fact that the un-addressed modules down the line will treat the break in the data flow as a start of a new packet.

A common mistake is to use a terminal program to transmit commands to a data module one character at a time as they are being typed in from a keyboard. Although this will work if typing very quickly, and as long as only one module is being used in the network, it is best to build a complete packet ahead of time and then transmit it as a character string without any breaks.

NUMERICAL DATA

The stackable data modules transmit and receive numerical data in ASCII format. Hence, the number 65,535 will be transferred one character at a time, most significant digit first (6 5 5 3 5). It is clear that the number of characters will vary depending on the value of the numerical data, as will the length of the field used to deliver it. Fortunately, most software platforms provide input and output function calls which automatically convert data to and from this format so the user does not have to be concerned with the fundamentals. The stackable data modules use a similar form of input/output conversion functions built in to their communications routines. If need be, when sending data from the host, the user can pad the most significant digits with 0's to force all numerical data to a specific character length, but this is not necessary and most often not practiced.

The command sets include many operations which require some form of numerical data to be passed back to the host. Because this data will always lie within the last field before the carriage return, the host can capture the data with a standard INPUT statement using the carriage return as the field delimiter. For instance, the host should read each of the proceeding single character fields (header, command title, channel number, etc.) one byte at a time, and then grab the numerical data with a multi-byte input statement as if reading it from a sequential file.

Some single-board computers do not incorporate the higher level communications functions common with programs running on a PC or laptop. In these cases, the multi-byte numerical data must be processed on a byte level. To do this, simply read each character individually and perform the following arithmetic:

- Get first digit and place in register "A".
- If another digit received, multiply register "A" by 10, then add the new digit to it.
- Repeat the last step until receiving a carriage return. Register "A" will contain the result.

Keep in mind that these digits are transmitted in ASCII format. To convert to decimal, strip off the upper nibble of each byte before adding it to register "A".

RECEPTION CONFIRMATION

There are two basic command types, "active" and "passive". An active command requests some form of real time data from the module it is talking to, or instructs a module to begin some form of conversion or operation. This could be a voltage reading, frequency measurement, ramp execution, timer run-off, etc. In these cases, the host will be waiting for a result or indication that the process has been completed.

A passive command, on the other hand, does not request data from a module, it simply sets a configuration variable or adjusts an operating parameter of some sort to be used later on in a process. To verify reception of a passive command, a data module will echo the command string back to the host after loading it into its memory. The host can use this response to confirm delivery of the command.

In some applications it may not be necessary to utilize this reception confirmation, however the user must be aware of the presence of this return data. **IMPORTANT:** Each time a passive command is sent to a data module, the return confirmation will be loaded into the communications buffer at the host end. This data will remain in the buffer and be retrieved the very next time the host reads the COM port even if the host is trying to capture data from a subsequently issued active command. This will cause confusion to the most adept programmer if not expected or considered.

If the reception confirmation feature is not needed in a particular application, simply read

and dump the contents of the COM port buffer after every passive command that is issued by the host. This will assure that the buffer remains empty and ready for live data input from one of the modules. In fact, it is good practice to always follow each output statement with an input statement when transmitting a command from the host.

CONTINUOUS DATA ACQUISITION

When operating one or more data modules in a continuous data acquisition mode, it is very often desirable to have the host transmit and receive data packets at the fastest rate possible. In these circumstances, great care must be given to assure the absence of any data packet timing violations. For example, consider the following programming loop.

- Host sends a command to read something from board "A".
- Host waits for board "A" to respond then sends a command to read from board "B".
- Host waits for board "B" to respond and then goes back to the first step.

The operation above will be in direct violation of the timing restrictions. Why? Examine the second step in the loop. As soon as board "A" responds, the host immediately sends a data packet to board "B". Because there is no timing gap between the packet transmitted by board "A" and the packet transmitted by the host, board "B" assumes that this is a continuation of the previous packet and simply ignores it.

Note however, that if the host is sending successive commands to the same board there will not be a problem even though the timing restrictions would still be in violation. This is because the data module transmitting a response has a clear indication of when its own data packet ends, and thus when a new one from the host begins.

To assure that all timing restrictions are met in a multi-board continuous data acquisition network, have the host wait at least 1 mS after receiving a data packet from a module, before beginning its own transmission. The flowchart shown in figure 2 demonstrates the correct method for sending commands from a host in a multi-board configuration and taking advantage of the fastest rate permissible by the communications protocol.

FIGURE 2: MULTI-BOARD ADDRESSING

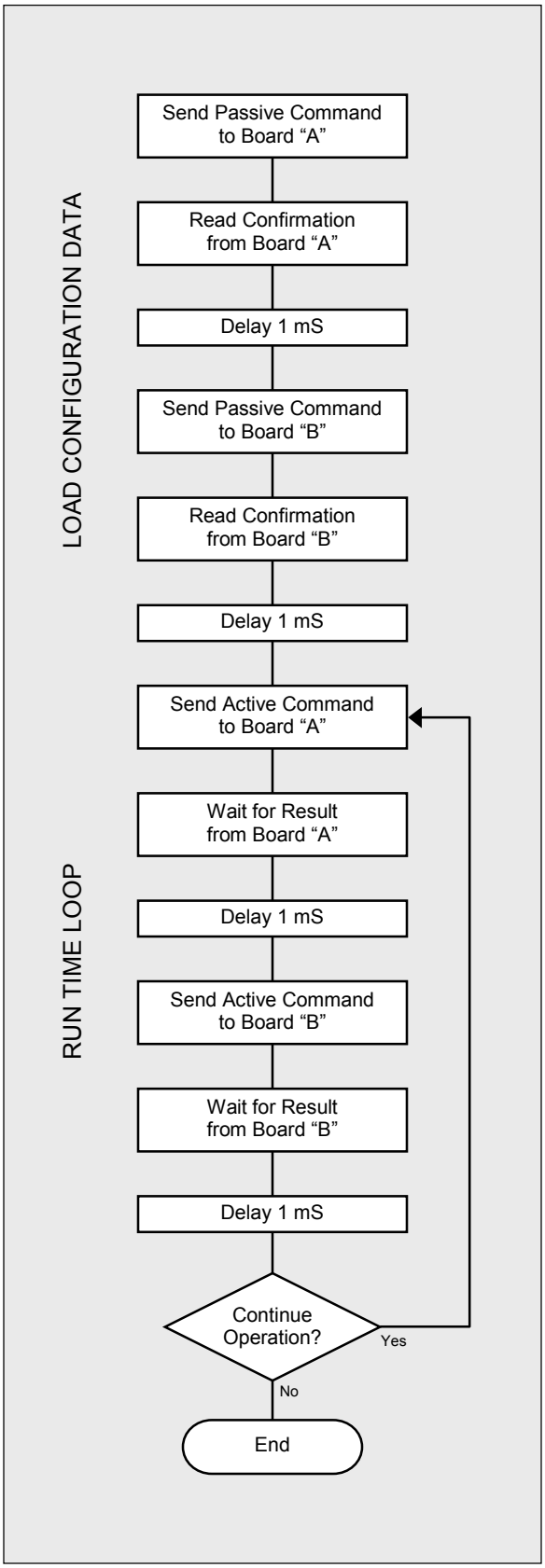
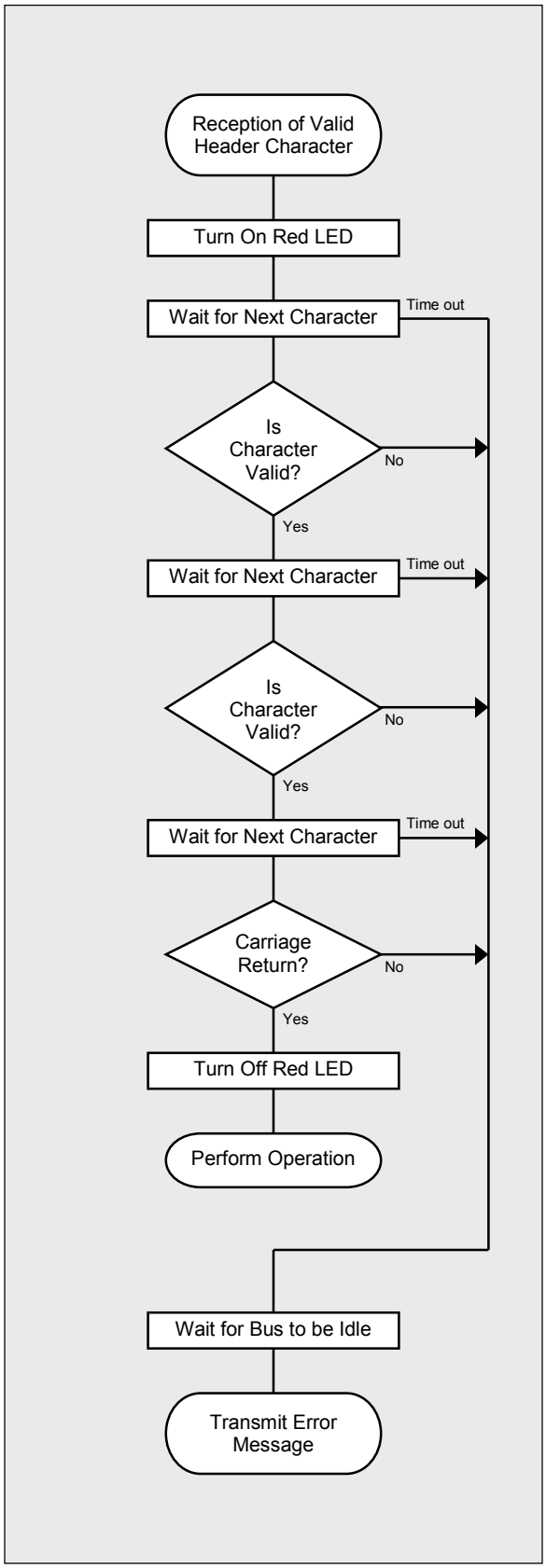


FIGURE 3: RECEIVE SUBROUTINE



COMMUNICATIONS DIAGNOSTICS

There are a number of key features common to the stackable data modules which provide clues for communications diagnostics. With the exception of the Multi-Drop Peripheral Interface, when a unit is first powered up, it will transmit a reset message which consists of the header character, an exclamation point, and then a carriage return. The user can verify this transmission by watching the red LED flash as power is applied. This reset indicator can be used to test the receive routine running on the host as well as confirm that a physical link has been established with the host's receiver. When transmitting a command from the host to a data module, the red LED will turn on after receiving the correct header character and then turn off after receiving a carriage return. This will appear as a quick flash and can be used to verify a physical link with the host's transmitter.

A common user mistake will be to leave out the carriage return at the end of a command string. If this should occur, the data module will continue to wait for the carriage return until eventually the on-board watchdog timer expires (after approximately 1.2 seconds) and causes the error message to be transmitted to the host. This 1.2 second delay will be visually noticeable by the red LED staying on longer then a simple flash, as is usually the case, and can determine if the error indicator is reflecting a bad command string, or simply one which is missing a carriage return.

The flowchart shown in figure 3 details the procedure used by a data module when receiving a typical packet from the host. Note that this is just an example and that the actual number of characters being received will vary from command to command. As mentioned above, the red LED turns on after reception of a valid header character and can be used to confirm a proper link on the transmit line of the host. So in other words, when sending a command string to a data module, if the LED flashes, the header character has been successfully received as expected with no communications error. Consequently, if the data module is responding to the command packet with the error message, it can be assumed that the physical link is good, and that the error lies in the construction of the actual data packet transmitted by the host and not a fault in the communications wiring.