AUTHOR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Department** |  | **Signature** |  | **Print Name** |  | **Date** |
| R&D |  |  |  |  |  |  |

SIGNATURES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Department** |  | **Signature** |  | **Print Name** |  | **Date** |
| R&D |  |  |  |  |  |  |

Revision History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rev.** |  | **Date** |  | **Initials** |  | **Description** |
| 0.1 |  | 06-Apr-2009 |  | KW |  | Initial draft derived from 15000309 Rev 0.5, added author and blanked fields |
| 0.2 |  | 07-Sep-2009 |  | KW |  | Updated to reflect code changes. |
| A |  | 16-Oct-2009 |  | DAT |  | Initial Release |
| A.1 |  | 19-Jul-2019 |  | KW |  | More corrections from dry runs, cleanup of Software Update section |
| A.2 |  | 23-Jul-2019 |  | KW |  | More corrections from a dry run, added additional test for Program Page Request CRC |
| A.3 |  | 24-Jul-2019 |  | KW |  | Swapped the + / - pin connections for J8 |
| A.4 |  | 10-Sep-2019 |  | KW |  | Corrected Program Page Command Timeout from 400ms to 500ms to reflect code |
| B |  | 26-Sep-2019 |  | DAT |  | Updated to Revision B |

References

1. 15000286 – DYONICS II RS485 Accessory Protocol

Equipment required for verification

1. Footswitch (REF 72201092)
2. POWERMINI MDU with Hand Controls (REF 72201500)
3. DYOINCS II EIP Shaver System running TestApp
4. RS-485 to RS-232 Serial Converter (used to convert the A/B RS485 differential lines into a RS232 serial output that can be read by a PC)
5. Computer with Windows 7/10 with an RS232 port and Serial Port Monitoring Program (e.g. RealTerm) installed and EZView software installed
6. EZ-Tap Pro
7. Host computer with Windows 2000/XP and the following software and hardware installed on it:
   1. Microsoft Embedded Visual C++ 4.0 Debugger for the Shaver Application
   2. Source code for Shaver application

| **Description** | **SKU/Model Number** | **Serial Number** | **Software** |
| --- | --- | --- | --- |
| DYONICS POWER II EIP | 72200873 |  |  |
| DYONICS POWER II Footswitch | 72201092 |  |  |
| DYONICS POWERMINI MDU with Hand Controls | 72201500 |  |  |
| RS-485 to RS-232 |  |  |  |
| Serial Monitoring Software |  |  |  |
| EZ-Tap Pro with EZView |  |  |  |

Table of Contents

[1 Overview 3](#_Toc20383412)

[2 Protocol Verification 3](#_Toc20383413)

[3 SUMMARY 17](#_Toc20383414)

[3.1 Notes 17](#_Toc20383415)

[3.2 Overall Pass/Fail Status 18](#_Toc20383416)

RS485 Accessory Protocol Verification

# Overview

The purpose of this document is to:

* + Show the traceability of the verification procedures against the DYONICS II RS485 Accessory Protocol
  + Provide a summary of the Verification procedures
  + Provide a summary of the Verification results
  + Provide a link to more detailed internal engineering verification reports when required

Verification is defined as the process by which the design output meets the design input requirements. The results of the design verification, including identification of the design, method(s), the date, and the individual(s) performing the verification, shall be documented in the DHF. Design verification shall be traceable to product specifications.

The verification is traceable to the following functional requirement and specification documents:

Dyonics II RS485 Accessory Protocol – Document # 15000286

# Protocol Verification

To setup for the Dyonics II EIP RS485 Protocol verification:

* Connect the RS-485 to RS-232 Serial Converter to one side of the EZ-Tap Pro.
* Connect the other side of the EZ-Tap Pro to the Computer.
* Connect the USB cable from the EZ-Tap Pro to the Computer.
* Start Serial Monitoring application on the Computer. Configure the application for 57600 baud, 8N1 (8 bit data, no parity, 1 stop bit). Set the Serial Monitoring Software to display Hexadecimal values and open up the COM port.
* Start the EZView application.

There are several elements used in the configuration. They are:

|  |  |
| --- | --- |
| Element | Description |
| Dbg-Shaver | Using Microsoft Embedded Visual C++ 4.0 to debug the main shaver app |
| Monitor | Using a Serial Monitoring program on the computer, to display the RS-485 serial data. |
| EZView | Using an EZ-Tap Pro with EZView program to monitor specific timing of the RS-485 protocol. |

Each test uses one or more of the elements above.

The following is a list of all the tests. The POWERMINI MDU is the RS485 slave in all these tests:

| **Section Number** | **Verification Procedure Summary** | **Summary Results**  **(may include links to other verification reports)** | **Pass / Fail** | **Initials / Date** |
| --- | --- | --- | --- | --- |
| 2.1 a.  2.2.1. | Verify that 8 bit bytes are sent over the bus using no parity and one stop bit at 19200.  Verify that bits 9-0 of the response to the Accessory ID request for the Footswitch is 1. | Monitor:  Hooked up the pins 8 & 11 of J8 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjusted the connection settings to 19200 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port.  Shaver and Monitor:  Plugged the Footswitch into the Shaver. Set a break point in the last line of the function FootSwitch::ResetVariableFootPedals(). Started the Shaver Application and waited till it stopped at the breakpoint. Verified that following serial byte sequence occurred:  0xB1 🡪 0x64 0x01  Removed all wires and cables from the shaver. | **P / F**  **P / F** |  |
|  | Verify that 8 bit bytes are sent over the bus using no parity and one stop bit at 57600 baud.  Verify that bits 9-0 of the response to the Accessory ID request for the POWERMINI MDU is 2. | Monitor:  Hooked up the pins 1 & 14 of J1 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjusted the connection settings to 57600 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port.  Shaver and Monitor:  Plugged a POWERMINI MDU in Port A of the Shaver. Set a break point in the last line of the function CControl::SendSerialRequests(). Started the Shaver Application and waited till it stopped at the breakpoint. Verified that following serial byte sequence occurred:  0xB1 🡪 0x54 0x02  0x00 🡪 0x25 0x04  Removed all wires and cables from the shaver. | **P / F**  **P / F** |  |
| 2.1 b. | Verify that the master initiates all communication by sending command requests and the only time a slave transmits on the bus is in response to a master command. | Shaver:  Plugged a POWERMINI MDU in Port A of the Shaver. Set a break point in the last line of the function CControl::SendSerialRequests(). Start the Shaver Application and waited till it stopped at the breakpoint.  Monitor:  Hooked up the pins 1 & 14 of J1 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjusted the connection settings to 57600 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port and verified that no serial activity takes place while the shaver is not sending requests to the POWERMINI MDU.  Shaver and Monitor:  Removed the breakpoint and allowed the shaver program to run and verified that serial activity takes places.  Removed all wires and cables from the shaver. | **P / F** |  |
| 2.1 c.  2.1.d.  2.1.e.  2.2. |  | Monitor and EZView:  Hooked up the pins 8 & 11 of J8 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjusted the connection settings to 19200 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port and started EZView capture.  Shaver:  Plugged the Footswitch into the Shaver.  Modified the shaver code software by inserting the following code segment:  {  SnWord wRequest;  static SnByte pbReqs[6] = {0x00,0xB1,0xD2,0x63,0xE4,0x55};  SnQByte qCnt;  for (qCnt = 0; qCnt < 6; qCnt++) {  wRequest = FOOT\_CMD(pbReqs[qCnt]);  SendSerialRequest(&wRequest, 10);  }  }  Right before the return TRUE; line of CControl::Init() in Control.cpp.  Set a break point at return TRUE; line of CControl::Init() in Control.cpp. Started the Shaver Application and waited till it stops at the breakpoint.  Monitor and EZ-View:  Stopped the captures.  Removed the inserted code and removed all wires and cables from the shaver. |  |  |
|  | Verify that command requests are one byte. | The serial capture showed that each command request was one byte. | **P / F** |  |
|  | Verify that slave responses are two bytes. | The serial capture showed that each slave response was two bytes. | **P / F** |  |
|  | Verify slave responses are within 3 milliseconds of receiving the command request for commands 0-2. | The EZView capture showed that slave responses were within 3 milliseconds of receiving the command request for commands 0-2. | **P / F** |  |
|  | Verify slave responses are within 8 milliseconds of receiving the command request for commands 3-5. | The EZView capture showed that slave responses were within 8 milliseconds of receiving the command request for commands 3-5. | **P / F** |  |
| 2.1 f. 1)  2.1 f. 2)  2.1 f. 3) |  | Monitor:  Hooked up the pins 8 & 11 of J8 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjusted the connection settings to 19200 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port.  Shaver and Monitor:  Plugged the Footswitch into the Shaver.  Modified the shaver code software by inserting the following code segment:  {  SnWord wRequest;  wRequest = FOOT\_CMD(SERIAL\_CMD\_DEV\_TYPE);  SendSerialRequest(&wRequest, 10);  wRequest = FOOT\_CMD(0xB2);  SendSerialRequest(&wRequest, 10);  }  Right before the return TRUE; line of CControl::Init() in Control.cpp.  Set a break point at the return TRUE; line of CControl::Init() in Control.cpp. Started the Shaver Application and waited till it stops at the breakpoint. Verified that following serial byte sequence occurred:  0xB1 🡪 0x64 0x01  0xB2 🡪 0x00 0x00  Removed the inserted code and remove all wires and cables from the shaver. | **P / F** |  |
|  | Verify that bits 3-0 are used to encode outgoing commands and that command value 0xF gets a NAK response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word of 0x6401 indicates that it is Accessory ID #1 which corresponds to the footswitch.  The command byte 0xB2 contains the 0x2 command in bits 3-0 and Bit 10 is clear indicating a NAK response. | **P / F** |  |
|  | Verify that bits 6-4 are used as ECC bits for outgoing commands. | The command byte 0xB1 contains the 0x1 command in bits 3-0 and bits 6-4 contain the value 0x3 which is the correct ECC value when bits 3-0 are the value 0x1. | **P / F** |  |
|  | Verify that bit 7 is used as the parity bit for outgoing commands. | The command byte 0xB1 contains the parity bit set in bit 7 which is the correct parity when bits 6-0 are the value 0x31. | **P / F** |  |
|  | Verify that return value is stored in bits 9-0 in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x6401. Bits 9-0 are the value 0x1 which corresponds to the footswitch. | **P / F** |  |
|  | Verify that bit 10 is used as the ACK/NAK bit in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x6401. Bit 10 is set indicating that the command was request was valid. | **P / F** |  |
|  | Verify that bits 11 through 14 are used as ECC bits in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x6401. Bits 14-11 contain the value 0xC which is the correct ECC value when bits 9-0 are the value 0x1. | **P / F** |  |
|  | Verify that bit 15 is used as a parity bit in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x6401. Bit 15, (the parity bit), is clear which is the correct parity when bits 14-0 are the value 0x6401. | **P / F** |  |
|  | Verify that for erroneous requests, the NAK bit is set, bits 9-0 are cleared, and parity and ECC bits set accordingly. | The command byte 0xB2 does not have a valid ECC bits set for the command 0x2 in bits 3-0. The response word to the invalid request is 0x0000. Bit 10 is clear which corresponds to a NAK. | **P / F** |  |
| 2.1 c.  2.1.d.  2.1.e.  2.2. |  | Monitor and EZView:  Hooked up the pins 1 & 14 of J1 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjusted the connection settings to 57600 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port and started EZView capture.  Shaver:  Plugged a POWERMINI MDU in Port A of the Shaver.  Modified the shaver code software by inserting the following code segment:  {  SnWord wRequest;  static SnByte pbReqs[12] =  {0x00,0xB1,0xD2,0x63,0xE4,0x55,0x36,0x87,  0x78,0xC9,0x1B,0x2D};  SnQByte qCnt;  for (qCnt = 0; qCnt < 12; qCnt++) {  wRequest = HAND\_PORT\_CMD(PORTA, pbReqs[qCnt]);  SendSerialRequest(&wRequest, 4);  }  }  Right before the return TRUE; line of CControl::Init() in Control.cpp.  Set a break point at return TRUE; line of CControl::Init() in Control.cpp. Started the Shaver Application and waited till it stops at the breakpoint.  Monitor and EZ-View:  Stopped the captures.  Removed the inserted code and removed all wires and cables from the shaver. |  |  |
|  | Verify that command requests are one byte. | The serial capture showed that each command request was one byte. | **P / F** |  |
|  | Verify that slave responses are two bytes. | The serial capture showed that each slave response was two bytes. | **P / F** |  |
|  | Verify slave responses are within 3 milliseconds of receiving the command request. | The EZView capture showed that slave responses were within 3 milliseconds of receiving the command request. | **P / F** |  |
| 2.1 f. 1)  2.1 f. 2)  2.1 f. 3) |  | Monitor:  Hooked up the pins 1 & 14 of J1 on the EIP Board to pin - & + of the converter. Started the serial monitor program and adjust the connection settings to 57600 baud, 8 bit, no parity, and 1 stop bit. Opened up the COM port.  Shaver and Monitor:  Plugged a POWERMINI MDU in Port A of the Shaver.  Modified the Shaver code software by inserting the following code segment:  {  SnWord wRequest;  wRequest = HAND\_PORT\_CMD(PORTA, SERIAL\_CMD\_DEV\_TYPE);  SendSerialRequest(&wRequest, 4);  wRequest = HAND\_PORT\_CMD(PORTA, 0xFF);  SendSerialRequest(&wRequest, 4);  }  Right before the return TRUE; line of CControl::Init() in Control.cpp.  Set a break point at return TRUE; line of CControl::Init() in Control.cpp. Started the Shaver Application and waited till it stopped at the breakpoint. Verified that following serial byte sequence occurred:  0xB1 🡪 0x54 0x02  0xFF 🡪 0x00 0x00  Removed the inserted code and removed all wires and cables from the shaver. | **P / F** |  |
|  | Verify that bits 3-0 are used to encode outgoing commands and that command value 0xF gets a NAK response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word of 0x5402 indicates that it is Accessory ID #2 which corresponds to the POWERMINI MDU w/Hand Controls.  The command byte 0xFF contains the 0xF command in bits 3-0 and Bit 10 is clear indicating a NAK response. | **P / F** |  |
|  | Verify that bits 6-4 are used as ECC bits for outgoing commands. | The command byte 0xB1 contains the 0x1 command in bits 3-0 and bits 6-4 contain the value 0x3 which is the correct ECC value when bits 3-0 are the value 0x1. | **P / F** |  |
|  | Verify that bit 7 is used as the parity bit for outgoing commands. | The command byte 0xB1 contains the parity bit set in bit 7 which is the correct parity when bits 6-0 are the value 0x31. | **P / F** |  |
|  | Verify that return value is stored in bits 9-0 in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x5402. Bits 9-0 are the value 0x2 which corresponds to the POWERMINI MDU w/Hand Controls. | **P / F** |  |
|  | Verify that bit 10 is used as the ACK/NAK bit in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x5402. Bit 10 is set indicating that the command was request was valid. | **P / F** |  |
|  | Verify that bits 11 through 14 are used as ECC bits in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x5402. Bits 14-11 contain the value 0xA which is the correct ECC value when bits 9-0 are the value 0x2. | **P / F** |  |
|  | Verify that bit 15 is used as a parity bit in a slave response. | The command byte 0xB1 contains the 0x1 command in bits 3-0 which is the Accessory ID request. The response word to the Accessory ID request command is 0x5402. Bit 15, (the parity bit), is clear which is the correct parity when bits 14-0 are the value 0x5402. | **P / F** |  |
|  | Verify that for erroneous requests, the NAK bit is set, bits 9-0 are cleared, and parity and ECC bits set accordingly. | The command byte 0xFF contains the 0xF command in bits 3-0 which is not a valid request. The response word to the invalid request is 0x0000. Bit 10 is clear which corresponds to a NAK. | **P / F** |  |
| 2.2.2  2.2.3  2.2.4 |  | Monitor:  Hooked up the pins 1 & 14 of J1 – Port A on the EIP Board to pin - & + of the converter.  PC:  Programed a USB Flash drive with an image of the POWERMINI MDU Software  Shaver:  Powered off the Shaver. Plugged the POWERMINI MDU into Port A of the Shaver. Powered on the Shaver. Pressed the Continue button when the Check Serial Number Screen was displayed.  Monitor and EZView:  Started the Serial Monitoring and EZView capture.  Shaver:  Pressed the Start button to initiate the software upgrade. The software upgrade started. At the completion of the update the Shaver reported that the MDU was being reset and that Update was complete.  Monitor, EZView and Shaver:  Stopped the Serial Monitoring and EZView capture. Powered off the Shaver. Reviewed the Serial Monitoring and EZView capture data. |  |  |
|  | Verify the Software Update begins with Software Update Start Request. | The serial capture began with:  0x1B 🡪 FC 00 | **P / F** |  |
|  | Verify that the POWERMINI MDU ACKs the Software Update Start Request. | The response word to the Software Update Start Request (0x1B) was 0xFC00. Bit 10 is set indicating that the command was ACKed. | **P / F** |  |
|  | Verify that only Program Page Command Requests are after the Software Update Start command and before the Reset command. | The serial capture showed that after the Software Update Start Command (0x1B), only Program Page Command Requests (0x9C) appear before the Reset Request (0x2D). | **P / F** |  |
|  | Verify that the POWERMINI MDU Program Page Request is of the form: 0x9C, MS Page #, LS Page #, 128 Bytes of Data and CRC Byte. | The format of every Program Page Command Request was the following:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Program Page (0x9C) | MS Byte of 128 byte Page # | LS Byte of 128 byte Page # | 128 Bytes of Flash Data | CRC Byte | | **P / F** |  |
|  | Verify that the POWERMINI MDU ACKs the Program Page Command Requests.. | The response word to every Program Page Command Request (0x9C) was 0xFC00. Bit 10 is set indicating that the command was ACKed. | **P / F** |  |
|  | Verify that the total size of Program Page Requests is 132 bytes. | The total number of bytes of the Program Page Command Requests was 132. | **P / F** |  |
|  | Verify that the Program Page request receives a response within 500ms after the request was sent. | Reviewed the EZView capture to verify that the POWERMINI MDU responded to each of the Program Page Command Requests 0x9C within 500ms. | **P / F** |  |
|  | Verify that the POWERMINI MDU ACKs the Reset Command Request. | The response word to the Reset Command Request (0x2D) was 0xFC00. Bit 10 is set indicating that the command was request was ACKed. | **P / F** |  |
|  | Verify that Command Requests sent at least 4 seconds after the Reset Command Request receives ACKs. | Powered off the Shaver, removed the USB drive and powered the Shaver back on and the POWERMINI MDU resumes communications with the Shaver. | **P / F** |  |
| 2.2.2  2.2.3  2.2.4 |  | Monitor:  Hooked up the pins 1 & 14 of J1 – Port A or J6 – Port B on the EIP Board to pin - & + of the converter. Started the Serial Monitoring capture.  Shaver and Monitor:  Plugged a Reliant MDU w/Hand Controls in Port A of the Shaver.  Modified the shaver code software by inserting the following code segment:  {  SnByte pbBuf[134];  SnWord wRequest;  SnByte bCnt;  wRequest = HAND\_PORT\_CMD(PORTA, 0x1B);  SendSerialRequests(1, &wRequest, 4);  \*(SnWord\*)pbBuf = HAND\_PORT\_CMD(PORTA, SERIAL\_CMD\_REQ\_12);  pbBuf[2] = SERIAL\_CMD\_REQ\_12;  pbBuf[3] = 0;  pbBuf[4] = 69;  for (bCnt = 0; bCnt < 128; bCnt++) {  pbBuf[bCnt + 5] = bCnt;  }  pbBuf[128 + 5] = CrcMemChunk(&pbBuf[2], 128 + 3, 0);  m\_hDriver->SerialPageToDevice(134, pbBuf);  pbBuf[128 + 5] = 0x00;  m\_hDriver->SerialPageToDevice(134, pbBuf);  wRequest = HAND\_PORT\_CMD(PORTA, 0x2D);  SendSerialRequests(1, &wRequest, 4);  }  Right before the return TRUE; line of CControl::Init() in Control.cpp.  Set a break point at return TRUE; line of CControl::Init() in Control.cpp. Started the Shaver Application and waited till it stops at the breakpoint.  Monitor:  Stopped the capture. |  |  |
|  | Verify that response to first Program Page request is an ACK. | The Program Page request had a calculated CRC from the Shaver of 0xD2, which got a response word of 0xFC00. Bit 10 is set indicating that the command was request got an ACK. | **P / F** |  |
|  | Verify that response to second Program Page request (with the invalid CRC of 0x00) is a NAK. | The Program Page request had a CRC of 0x00, which got a response word of 0x0000. Bit 10 is clear indicating that the command was request got a NAK. | **P / F** |  |

# SUMMARY

## Notes

## Overall Pass/Fail Status

|  |  |
| --- | --- |
| **Overall Pass / Fail Status** |  |
| **Date** |  |
| **Signature** |  |
| **Printed Name** |  |
| **Department** |  |
| **Title** |  |