



**ADVANCED VEHICLE TECHNOLOGIES, Inc.**

---

# AVT - 841 & 842 & 843

## Multiple Interface

### Volume 1

J1850 VPW

CAN [2-wire], High Speed

CAN [Single Wire CAN (SWC) ]

CAN ISO 15765 support

K-Line communications

LIN communications

AVT-841:        Hardware revision "B1"  
AVT-842:        Hardware revision "A1"  
AVT-843:        Hardware revision "A2"  
Firmware Version "4.3"  
23 May 2011

# Table of Contents

<b>1. INTRODUCTION</b>	<b>6</b>
1.1 SPECIAL NOTE	6
1.2 HARDWARE	6
1.3 FIRMWARE	6
1.3.1 <i>Determining Firmware Version</i>	7
1.4 COMMANDS AND RESPONSES	7
<b>2. GLOSSARY</b>	<b>7</b>
<b>3. AVT-84X OPERATION</b>	<b>8</b>
<b>4. HOST COMPUTER CONNECTION</b>	<b>8</b>
4.1 AVT-841 CONNECTION TO HOST COMPUTER	8
4.1.1 <i>AVT-841 Baud Rate to Host Computer</i>	9
4.2 AVT-842 CONNECTION TO HOST COMPUTER	9
4.3 AVT-843 CONNECTION TO HOST COMPUTER	10
4.3.1 <i>Ethernet Setup</i>	10
4.3.2 <i>Ethernet IP Addressing Modes</i>	12
4.3.3 <i>AutoIP Addressing</i>	12
4.4 PACKET COMMUNICATIONS WITH HOST COMPUTER	13
<b>5. VEHICLE CONNECTION</b>	<b>14</b>
5.1 POWER REQUIREMENTS	14
5.1.1 <i>Ground</i>	15
5.1.2 <i>Input Voltage</i>	15
5.1.3 <i>Power Consumption</i>	15
5.2 JUMPERS	15
<b>6. ADC CONNECTION</b>	<b>15</b>
6.1 ADC READINGS	16
<b>7. CAN MODE</b>	<b>16</b>
7.1 CAN SECONDARY OPERATIONAL MODES	17
7.2 CAN0 - 2-WIRE CAN	17
7.3 CAN4 - SINGLE WIRE CAN (SWC)	17
7.3.1 <i>Jumper JP1</i>	17
7.4 CAN0 CHANNEL NUMBER	17
7.5 CAN4 CHANNEL NUMBER	17
7.6 LIN SECONDARY OPERATIONAL MODE CHANNEL NUMBER	18
7.7 KWP SECONDARY OPERATIONAL MODE CHANNEL NUMBER	18
7.8 CAN COMMUNICATIONS GENERAL NOTES	18
7.8.1 <i>Disabled</i>	18
7.8.2 <i>Normal</i>	18
7.8.3 <i>Listen Only</i>	18
7.8.4 <i>Time Stamps</i>	19
7.9 ACCEPTANCE ID AND MASK CONFIGURATION	20
7.10 ACCEPTANCE ID AND MASK OPERATION	20
7.10.1 <i>ID/Mask mode = 2</i>	21
7.10.2 <i>ID/Mask mode = 4</i>	21
7.10.3 <i>ID/Mask mode = 8</i>	21
7.10.4 <i>ID/Mask Example #1</i>	22
7.10.5 <i>ID/Mask Example #2</i>	22
7.10.6 <i>ID/Mask Example #3</i>	23

7.10.7	ID/Mask Example #4.....	23
7.10.8	ID/Mask Example #5.....	23
7.10.9	Summary.....	24
7.11	SETTING UP A CAN CHANNEL FOR OPERATION .....	24
7.11.1	Communications Example.....	25
7.12	CHANNEL ACTIVITY.....	25
7.13	PERIODIC MESSAGE SUPPORT .....	25
7.13.1	Organization of Periodic Messages.....	26
7.13.2	Periodic Message Master Timer .....	26
7.13.3	Type1 Periodic Messages.....	26
7.13.4	Type2 Periodic Messages.....	27
7.13.5	Periodic Message Commands .....	28
7.14	ISO 15765 SUPPORT .....	29
7.14.1	ISO 15765 Terms and/or Definitions .....	29
7.14.2	ISO 15765 Receive Operations - General Notes.....	29
7.14.3	ISO 15765 Receive Operations - Brief Description .....	30
7.14.4	ISO 15765 Receive Operations - Examples.....	30
7.14.5	ISO 15765 Transmit Operations - General Notes.....	33
7.14.6	ISO 15765 Transmit Operations - Brief Description .....	33
7.14.7	ISO 15765 Transmit Operations - Examples .....	34
7.14.8	ISO 15765 Questions and Engineering Support .....	36
7.15	AUTO BLOCK TRANSMIT.....	36
7.15.1	Operation Description.....	36
7.15.2	Command Descriptions (non-volatile parameters).....	37
7.15.3	Command Description (control).....	39
7.15.4	Operation Example.....	41
<b>8.</b>	<b>LIN OPERATIONS – IN CAN MODE .....</b>	<b>42</b>
8.1	JUMPER JP2.....	42
8.2	COMMUNICATIONS.....	42
8.2.1	Message Length.....	42
8.2.2	Checksum.....	42
8.2.3	ID Byte Only Message .....	43
8.2.4	Communications Example .....	43
8.2.5	Time Stamp .....	44
8.3	PERIODIC MESSAGE SUPPORT .....	45
8.3.1	Organization of Periodic Messages.....	45
8.3.2	Periodic Message Master Timer.....	45
8.3.3	Type1 Periodic Messages .....	45
8.3.4	Type2 Periodic Messages .....	46
8.3.5	Periodic Message Commands.....	48
8.4	ABIC SUPPORT.....	48
8.5	COMMANDS AND RESPONSES .....	48
<b>9.</b>	<b>KWP OPERATIONS – IN CAN MODE.....</b>	<b>48</b>
9.1	JUMPER JP2.....	48
9.2	COMMUNICATIONS.....	48
9.3	OPERATION COMMANDS .....	49
9.3.1	Communications Example .....	49
9.3.2	Time Stamp .....	49
9.3.3	Fast Transmit.....	50
9.4	PERIODIC MESSAGE SUPPORT .....	50
9.4.1	Organization of Periodic Messages.....	50
9.4.2	Periodic Message Master Timer.....	51
9.4.3	Type1 Periodic Messages .....	51
9.4.4	Type2 Periodic Messages .....	52
9.4.5	Periodic Message Commands.....	53

<b>10.</b>	<b>VPW MODE.....</b>	<b>53</b>
10.1	JUMPER JP3.....	54
10.2	COMMUNICATIONS.....	54
10.2.1	<i>Communications Example - Not Block Transfer.....</i>	<i>54</i>
10.2.2	<i>Time Stamp.....</i>	<i>55</i>
10.3	MESSAGE FILTERING.....	56
10.3.1	<i>Example Network Message.....</i>	<i>57</i>
10.3.2	<i>Example #1.....</i>	<i>57</i>
10.3.3	<i>Example #2.....</i>	<i>57</i>
10.3.4	<i>Example #3.....</i>	<i>57</i>
10.3.5	<i>Example #4.....</i>	<i>57</i>
10.4	MASK / MATCH / RESPOND FUNCTION .....	58
10.4.1	<i>Operational Overview .....</i>	<i>58</i>
10.4.2	<i>Command Summary .....</i>	<i>58</i>
10.4.3	<i>Example.....</i>	<i>58</i>
10.5	PERIODIC MESSAGE SUPPORT .....	59
10.5.1	<i>Organization of Periodic Messages .....</i>	<i>59</i>
10.5.2	<i>Periodic Message Master Timer .....</i>	<i>59</i>
10.5.3	<i>Type1 Periodic Messages.....</i>	<i>60</i>
10.5.4	<i>Type2 Periodic Messages.....</i>	<i>60</i>
10.5.5	<i>Periodic Message Commands .....</i>	<i>61</i>
10.6	BLOCK TRANSMIT EXAMPLE.....	62
10.7	BLOCK RECEIVE EXAMPLE .....	62
<b>11.</b>	<b>KWP MODE.....</b>	<b>63</b>
11.1	JUMPER JP2.....	63
11.2	COMMUNICATIONS.....	63
11.2.1	<i>Communications Example.....</i>	<i>64</i>
11.2.2	<i>Time Stamp.....</i>	<i>64</i>
11.3	INITIALIZATION.....	65
11.3.1	<i>CARB Mode Initialization .....</i>	<i>66</i>
11.3.2	<i>FAST Initialization .....</i>	<i>66</i>
11.4	PERIODIC MESSAGE SUPPORT .....	67
11.4.1	<i>Organization of Periodic Messages .....</i>	<i>67</i>
11.4.2	<i>Periodic Message Master Timer .....</i>	<i>67</i>
11.4.3	<i>Type1 Periodic Messages.....</i>	<i>68</i>
11.4.4	<i>Type2 Periodic Messages.....</i>	<i>68</i>
11.4.5	<i>Periodic Message Commands .....</i>	<i>69</i>
<b>12.</b>	<b>AVT-84X FIELD REFLASHING .....</b>	<b>70</b>
12.1	AVT-84X REFLASHING - AVT PROVIDED APPLICATION.....	70
<b>13.</b>	<b>IDLE MODE - COMMANDS.....</b>	<b>72</b>
13.1	IDLE MODE - RESPONSES .....	72
13.2	OTHER RESPONSES.....	72
<b>14.</b>	<b>CAN MODE - COMMANDS.....</b>	<b>74</b>
14.1	CAN MODE - RESPONSES .....	85
<b>15.</b>	<b>LIN OPERATIONS IN CAN MODE - COMMANDS .....</b>	<b>100</b>
15.1	LIN OPERATIONS IN CAN MODE - RESPONSES .....	103
<b>16.</b>	<b>KWP OPERATIONS IN CAN MODE - COMMANDS.....</b>	<b>109</b>
16.1	KWP OPERATIONS IN CAN MODE - RESPONSES .....	112
<b>17.</b>	<b>VPW MODE - COMMANDS .....</b>	<b>117</b>

17.1	VPW MODE - RESPONSES .....	122
<b>18.</b>	<b>KWP MODE - COMMANDS .....</b>	<b>129</b>
18.1	KWP MODE - RESPONSES .....	134
<b>19.</b>	<b>APPENDIX A .....</b>	<b>142</b>
<b>20.</b>	<b>APPENDIX B .....</b>	<b>145</b>
<b>21.</b>	<b>QUESTIONS ?? .....</b>	<b>148</b>
<b>22.</b>	<b>BIT MAP FOR IDS, MASKS, COMMANDS, ETC.....</b>	<b>149</b>

## 1. Introduction

This document describes the AVT-841, AVT-842, and AVT-843 hardware and firmware.

The AVT-841, 842, and 843 are nearly identical units. The only difference is the means by which they connect to and communicate with the host computer.

AVT-841: RS-232 serial communications with host computer.

AVT-842: USB communications with the host computer.

AVT-843: Ethernet communications with host computer.

They are multiple network interfaces for in-vehicle networks. The operational firmware supports J1850 VPW (GM Class 2 compliant); 2-wire CAN; Single Wire CAN (SWC); LIN; and K-line communication operations.

Additional operational modes and functions are under development.

Some simultaneous operations are supported. Some are not.

### 1.1 *Special Note*

References in this document to “AVT-84x” mean the AVT-841, 842, or 843 - depending on the unit being used.

References to specific models are made where necessary.

All three units can be considered identical; unless otherwise noted.

### 1.2 *Hardware*

Refer to our web site for the most up-to-date information about the hardware status of each board.

The AVT-841 board revision stands at “B1.”

[Normally a stock item.]

Hardware status: [www.AVT-HQ.com/841\\_hw.htm](http://www.AVT-HQ.com/841_hw.htm)

The AVT-842 board revision stands at “A1.”

[To be replaced by the AVT-852 when stock is depleted.]

Hardware status: [www.AVT-HQ.com/842\\_hw.htm](http://www.AVT-HQ.com/842_hw.htm)

The AVT-843 board revision stands at “A2.”

[To be replaced by the AVT-853 when stock is depleted.]

Hardware status: [www.AVT-HQ.com/843\\_hw.htm](http://www.AVT-HQ.com/843_hw.htm)

### 1.3 *Firmware*

The firmware version stands at “4.3”.

Refer to our web site for the most up-to-date information about AVT-84x firmware versions:

[www.AVT-HQ.com/841\\_asm.htm](http://www.AVT-HQ.com/841_asm.htm)

### 1.3.1 Determining Firmware Version

Perform the following to determine the version of firmware in your unit.

[This simplified method applies to version 3.0 and later.]

- Connect to a host computer running the Hex Terminal or equivalent.
- Power on the 84x interface unit.
- The power-on notification is:
  - \$91 \$12 indicates idle mode operation.
  - \$92 \$04 \$xx where “xx” is the firmware version.
  - Example: “xx” = \$25 indicates version 2.5.
- Or, at any time, send the \$B0 command.
- The response will be: \$92 \$04 \$xx where “xx” is the firmware version.

### 1.4 Commands and Responses

A list of commands, responses, error codes, notes, etc. is provided at the end of this document.

## 2. Glossary

Common terms, abbreviations, acronyms, and more.

\$ sign	Indicates a hex number.
ADC	Analog to Digital Converter or Conversion.
CAN	Controller Area Network
CAN0	CAN channel 0
CAN4	CAN channel 4
EEPROM	Electrically Erasable Programmable Read Only Memory. Usually with the form of small rows and sectors. Erase and program operations are usually done for one sector at a time.
FLASH	A form of EEPROM. Usually with the form of large rows and sectors. Erase and program operations are usually done for one sector at a time.
ISP	Means “ISO 15765” process or processing.
IDE	ID Extended. When this bit = “0” the CAN frame uses an 11-bit ID. When this bit = “1” the CAN frame uses a 29-bit ID; extended ID.
ISO 11898	An ISO specification for 2-wire CAN physical layer.
ISO 15765	An ISO specification dealing with the formatting of data in the CAN frame data field. Also used in sending blocks of data using CAN. Also known as Multi-Frame Messaging (MFM) or Segmented Messages. This specification also deals with other CAN network issues.
J2411	An SAE specification for Single Wire CAN (SWC).
K-line	Single wire communications protocol. Refer to ISO 9141, ISO 9141-2, and ISO 14230 for more information.

LIN	Local Interconnect Network.
RTR	Remote Transmission Request. When this bit = “1” it indicates a frame that is requesting a remote node to transmit an answering frame.
SRR	Substitute Remote Request. A fixed recessive bit that only exists in extended frames (IDE = 1, 29-bit ID).
TVS	Transient Voltage Suppression.
Type1	Type1 Periodic Message support, CAN, each message operates independently.
Type2	Type2 Periodic Message support, CAN, messages within a group operate sequentially.
SWC	Single Wire CAN (SAE J2411).
XOR	Bit-wise logical exclusive ‘OR’.

### 3. AVT-84x Operation

The AVT-84x does not have a power switch. The unit powers up and begins operations as soon as it is plugged into a vehicle or other power source is applied.

On power-up, the interface will, almost immediately, report to the host computer: \$91 \$12 and \$92 \$04 \$xx (where “xx” is the unit firmware version number). This is the idle state, where the AVT-84x is waiting for the host computer to issue a mode switch command. No network communications are supported while in the idle state. Refer to Section 12 for a list of commands supported while in the idle state.

### 4. Host Computer Connection

The AVT-841 uses an RS-232 serial communications connection to the host computer. Detailed information in the following paragraphs.

The AVT-842 uses a USB connection to the host computer. Detailed information in the following paragraphs.

The AVT-843 uses an Ethernet connection to the host computer. Detailed information in the following paragraphs.

#### 4.1 AVT-841 Connection to Host Computer

The AVT-841 serial connection to the host computer com port is standard RS-232. It should be mated to a host computer using a straight through wired 9-pin cable (NOT a null modem cable).

The AVT-841 uses RTS/CTS hardware handshaking for flow control. The host computer must be configured for RTS/CTS hardware handshaking to proper operations.

Software flow control (XON/XOFF) cannot be used since communications between the AVT-841 and the host computer is binary data.

For reference, the signals of the AVT-841 DE-9S connector pins are listed below.

Pins #1, #6, and #9 are tied together on the AVT-841 board through 10 ohm resistors.



<u>Pin #</u>	<u>Signal</u>	<u>Direction</u>
1	RI	in to host computer
2	TX data	out of AVT-841
3	RX data	in to AVT-841
4	DTR	out of host computer
5	GND	
6	DSR	in to host computer
7	CTS	in to AVT-841
8	RTS	out of AVT-841
9	CD	in to host computer

AVT-841 RS-232 Connector Signals

#### 4.1.1 AVT-841 Baud Rate to Host Computer

The serial communications baud rate between the AVT-841 and the host computer is 57.6 kbaud (factory default).

The user can set the host communications baud rate by changing a value stored in EEPROM. Available baud rates are listed here.

19.2 kbaud  
38.4 kbaud  
57.6 kbaud [default]  
115.2 kbaud

Refer to the commands section for information about the “52 67 xx” command to change the host baud rate. Also, refer to the User’s Manual, Volume 2 for detailed EEPROM information.

Note that this baud rate setting has nothing to do with vehicle network or Ethernet network communications.

#### 4.2 AVT-842 Connection to Host Computer

The AVT-842 uses the FTDI embedded USB to serial converter device (FT2232). USB 1.1 operations are supported. Connection to the host computer is with a standard USB A/B cable.

Note that the USB side of the AVT-842 is USB bus powered. You can connect the AVT-842 to your host computer and it will be ‘discovered’ by the host and the port will be enumerated.

USB driver software is provided by FTDI (from their web site). AVT has tested and recommends using the FTDI Virtual Com Port (VCP) drivers for communications between the user application and the AVT-842 board. A separate document is available showing how to install the AVT-842 USB drivers. That document is posted on the AVT web site, Product Documents page.

On the AVT-842 board, the microcontroller and the FTDI USB device communicate via an internal serial link. The factory default for the internal serial link baud rate is 57.6 kbaud.

The user can set the host communications baud rate by changing a value stored in EEPROM. Available baud rates are listed here.

19.2 kbaud

38.4 kbaud  
57.6 kbaud [default]  
115.2 kbaud

Refer to the commands section for information about the “52 67 xx” command to change the host baud rate. Also, refer to the User’s Manual, Volume 2 for detailed EEPROM information.

Note that this baud rate setting has nothing to do with vehicle network.

### **4.3 AVT-843 Connection to Host Computer**

The AVT-843 uses a Lantronix XPort embedded serial server to provide an Ethernet connection to the host computer. On the AVT-843 this appears as an RJ-45 connector. It is 10/100 Ethernet; TCP/IP; and configured with a static IP address = 192.168.1.70 (factory default).

The user can change the IP address of the AVT-843 unit. Detailed information about this is provided in the Section 4.3.1 and Appendices “A” and “B.”

On the AVT-843 board, the microcontroller and the XPort device communicate via an internal serial link. The factory default baud rate is 57.6 kbaud.

The user can set the host communications baud rate by changing a value stored in EEPROM. Available baud rates are listed here.

19.2 kbaud  
38.4 kbaud  
57.6 kbaud [default]  
115.2 kbaud

Refer to the commands section for information about the “52 67 xx” command to change the host baud rate. Also, refer to the User’s Manual, Volume 2 for detailed EEPROM information.

Note that the baud rate setting of the XPort serial server must be changed to match that of the AVT-843 microcontroller. Refer to Section 4.3.1 and Appendices “A” and “B” for information on setting the XPort parameters.

Note that this baud rate setting has nothing to do with vehicle network or Ethernet network communications.

#### **4.3.1 Ethernet Setup**

The AVT-843 XPort can be reconfigured in the field by the user. Basic information is provided here. More information is provided in Appendices “A” and “B” at the end of this document.

Use care when changing the configuration. It is possible to corrupt the setup such that communications with the AVT-843 will be totally nonfunctional.

Ethernet communications with the AVT-843 use TCP/IP. The factory default device address is listed here as well as the various port numbers, depending on the type of communications to be established.

Note that several different communications sessions are possible with the AVT-843, depending on what is to be accomplished and what port is used. A session can be established with the AVT-843 XPort to change settings such as the Ethernet IP Address (port 9999). In normal use, an Ethernet session will be opened with the AVT-843 to communicate with the vehicle network (port 10001).

The AVT-843 uses the Lantronix “XPort” device. Detailed information about the XPort device, configuration tools, and more can be obtained from AVT or from the Lantronix web site ([www.Lantronix.com](http://www.Lantronix.com)).

### **4.3.1.1 Ethernet IP Address**

The factory default IP address of the AVT-843 is static and is set to:  
192.168.1.70

The factory default net mask setting is: 255.255.255.0

Depending on the particular network environment in which the AVT-843 is being used, the setting of the net mask may not be important. Rule of thumb: if connected to a busy network, and the AVT-843 is using static IP addressing, set the net mask to 255.255.255.0.

### **4.3.1.2 Hardware or MAC Address**

The hardware or MAC address of the AVT-843 can be found on the serial number sticker on the XPort device - which looks like an RJ-45 connector. The MAC address will start with: “00-20-4A” for Lantronix.

### **4.3.1.3 Vehicle Network Interface Port**

Communications with the AVT-843 vehicle network interface is via port 10001.

All communications with the AVT-843 vehicle interface is in binary bytes [not ASCII hex]. Refer to Section 4.4 for a description of the ‘packetized’ communications protocol between the AVT-843 and the host computer. All communications with the AVT-843 follow the exact same rules and formats as that of the AVT-841 and other AVT interface equipment.

### **4.3.1.4 Telnet Setup Port**

The configuration of the AVT-843 XPort device can be examined and changed using the Telnet application via port 9999.

To start a Telnet setup session with the AVT-843 unit perform the following (on a Win98/NT/XP computer):

- Start Menu
- Run
- Type into the command line:  
telnet 192.168.1.70 9999
- Click OK
- When the session banner from the XPort is displayed, you must hit ENTER within 5 seconds or else the session will time out and disconnect.

Refer to Appendix “A” at the end of this document for a listing of a Telnet session with an AVT-843 XPort device. Factory default settings are shown.

### **4.3.1.5 Web Page Setup**

The configuration of the AVT-843 XPort device can be examined and changed using a web browser. The setup screen is an HTML web page.

To establish a web page session with the AVT-843 XPort, enter the following into the web browser address line:        `http://192.168.1.70`

The setup form will appear. The first page is the configuration summary page. Select either “Server Properties” or “Port Properties” to change the configuration. After making changes, select “Update Settings.” The AVT-843 XPort will store the new settings and then reboot. Wait 1 to 2 minutes for the AVT-843 XPort to complete the reboot before attempting to access the unit with the new settings in-use.

Refer to Appendix “B” at the end of this document for a complete listing of all setup information available when using the AVT-843 XPort configuration web page.

### **4.3.2 Ethernet IP Addressing Modes**

Three IP addressing modes are available on the AVT-843 XPort.

- Static
- DHCP
- ARP

#### ***4.3.2.1 Static IP Addressing***

The factory default addressing mode for the AVT-843 XPort is static and the address is set to 192.168.1.70. In static mode the Ethernet address of the AVT-843 is always the same and does not change when power is cycled.

#### ***4.3.2.2 DHCP Addressing***

Setting the AVT-843 IP address to 0.0.0.0 will enable DHCP (Dynamic Host Configuration Protocol) function.

In this mode, the AVT-843 XPort will, on power-up, search for a DHCP server. If one is found it will obtain its IP address, gateway address, and subnet mask from the DHCP server.

If a DHCP server is not found, the AVT-843 XPort will then switch to AutoIP addressing, described in the next section.

An AVT-843 IP address of 0.0.1.0 enables DHCP addressing and disables AutoIP addressing.

### **4.3.3 AutoIP Addressing**

“AutoIP is an alternative to DHCP that allows hosts to automatically obtain an IP address in smaller networks that may not have a DHCP server.”

[Quoted from Lantronix XPort User Manual, revision A 3/03, page 3-4.]

AutoIP addressing is only enabled if the AVT-843 XPort IP address is set to 0.0.0.0 and no DHCP server is found.

If, on power-up, the AVT-843 XPort cannot find a DHCP server it will automatically assign itself an AutoIP address (range: 169.254.0.1 to 169.254.255.1). It will then send out an ARP (Address Resolution Protocol) request onto the network to see if any other node already has that address. If no conflict is found, the AVT-843 XPort will use that address until the next power-on reset or reboot.

If an address conflict is found (another node is discovered to already have that address) then the AVT-843 XPort will select another AutoIP address, send out another ARP request. The process will continue until it finds an address that is not being used.

#### **4.4 Packet Communications with Host Computer**

Communications between the host computer and the AVT-84x, in both directions, uses a “packet” protocol. This is the same protocol or method used by all AVT interface hardware.

- The first byte of a packet is the header byte.
- The header byte upper nibble (first hex digit) indicates what the packet is about.
- The header byte lower nibble (second hex digit) is the count of bytes to follow.
- If the header byte upper nibble is a zero (0) then the packet is a message to or from the network.
- This protocol is limited to 15 bytes following the header byte (lower nibble = \$F).
- Some messages require more than 15 bytes. For such a situation there are two “alternate” header formats, which are of the form:

11 xx

12 xx yy

These alternate header formats only apply to messages to or from the network.

- If the byte count is more than \$0F but equal to or less than \$FF the packet will be of the form:
  - 11 xx rr ss tt ...
  - \$11 indicates first alternate header format.
  - \$xx indicates the count of bytes to follow (not including the “xx” byte).
  - \$rr ss tt ... the packet data, including the message to/from the network.
- If the byte count is more than \$FF but less than or equal to \$FF FF the packet will be of the form:
  - 12 xx yy rr ss tt
  - \$12 indicates second alternate header format.
  - \$xx yy indicates the count of bytes to follow (not including the xx yy bytes).
  - \$rr ss tt ... the packet data, including the message to/from the network.
- Example #1
  - Turn on the time stamp feature in CAN mode.
  - Command: 52 08 01
  - Header byte upper nibble “5” indicates a configuration command.
  - Header byte lower nibble “2” indicates two bytes follow.
  - \$08 is the time stamp command.
  - \$01 commands enable time stamps.
- Example #2
  - Send a block of 348 bytes onto the VPW network.
  - Command: 12 01 5C rr ss tt vv ...
  - Header byte = \$12, alternate header format #2.
  - \$01 5C = 348 bytes
  - rr ss tt vv ... are the 348 message bytes.

- Example #3  
Receive an 18 byte message from the VPW network.  
Response: 11 13 rr ss tt vv ...  
Header byte = \$11, alternate header format #1.  
\$13 = 19 bytes follow  
rr = is the receive status byte (indicates if any error were detected, etc.)  
ss tt vv = actual message from the network.

Additional information about the AVT protocol is available at the beginning of the “Master Commands and Responses” document available from our web site at:  
[www.AVT-HQ.com/download.htm#Notes](http://www.AVT-HQ.com/download.htm#Notes)

## 5. Vehicle Connection

The vehicle or network connector is an industry standard DA-15P connector and requires a DA-15S mate. The pin out for the vehicle / network connector is shown below.

<u>Pin #</u>	<u>Description</u>	<u>Notes</u>
1	SWC	Bi-directional This signal goes through JP1
2	J1850 VPW bus +	Bi-directional This signal goes through JP3
4	Ground	pins #4 and #5 are connected together internally
3	[CAN4_H]	[only on rev. “T” unit]
5	Ground	pins #4 and #5 are connected together internally
6	CAN0_H	Bi-directional
7	K-Line used for LIN and KWP modes	Bi-directional This signal goes through JP2
11	[CAN4_L]	[only on rev. “T” unit]
13	V-Batt supply	Sourced by external equipment (vehicle)
14	CAN0_L	Bi-directional

P3 (the DA-15P connector on the AVT-84x board)

All other pins on the DA-15P vehicle network connector are “reserved.”  
The user should not connect anything to unused or reserved pins.

### 5.1 Power Requirements

The AVT-84x board requires a nominal +12 VDC power supply; usually provided by the vehicle or any suitable external power supply.

### 5.1.1 Ground

Common ground is required between the AVT-84x board and the subject vehicle or module. P3 pins #4 and/or #5 are ground. They are connected together internally on the AVT-84x board. One is needed for normal operations.

### 5.1.2 Input Voltage

V-Batt, the external power supply is applied to P3 pin #13.

Note that V-Batt is used to power the board. It also is the supply for the VPW signal; the pull-up and reference voltage for K-Line communications; and for Single Wire CAN (SWC) signal.

For most normal operations, V-Batt supply can range from +7 to +24 VDC.  
(The absolute maximum input voltage is +26.5 VDC.)

### 5.1.3 Power Consumption

Power consumption for each unit is listed here.

AVT-841: 750 milliwatts (nominal).

AVT-842: 1 watt (nominal).

AVT-843: 3.5 watts (nominal).

## 5.2 Jumpers

On the AVT-84x board are three jumpers: JP1, JP2, JP3.

They are for:

Single Wire CAN (SWC) network:	JP1
K-line network:	JP2
VPW network:	JP3.

Note that AVT-841 revision “A” boards do not have JP3. The VPW network line is always connected to pin #2 of P3 - the DA-15P “Vehicle” connector.

## 6. ADC Connection

The four position screw terminal block is used to access the three Analog to Digital Converter (ADC) channels.

- Terminal #1: ADC channel #1.
- Terminal #2: ADC channel #2.
- Terminal #3: ADC channel #3.
- Terminal #4: ground.

The input voltage range for all three channels is: 0 to +5 volts.

The inputs are passively pulled to ground through a 24.9 K ohm resistor. In most cases, with no input applied, the reading for a channel will generally stay below a reading of about \$03.

The inputs are ESD protected with a 27 volt TVS diode.

The inputs are over and under voltage protected. That does not mean the inputs can withstand abuse. Damage may still occur if subjected to a voltage outside the range of 0 to +5v with respect to ground.

## 6.1 ADC Readings

The ADC output conversion value range is: \$00 to \$FF. Full scale is \$FF.

Input voltage of 0.0v equals a reading of \$00.

Input voltage of 5.0v equals a reading of \$FF.

The conversion is linear, monotonic, with no missing codes.

There is no input filtering of the signal and readings are not averaged.

The ADC channels are read continuously and the most recent value is stored. The command to read an ADC channel returns the value most recently read. ADC conversions are not synchronized with commands from the host to take an ADC reading.

The 52 58 0x command is used to read a specified ADC channel; where x = channel number 1 to 3.

The 52 59 xx command is used to disable or enable and set the rate at which all three ADC channels are reported to the host. Parameter 'xx' is the number of timer ticks between reports.

The timer for the 5x 29 command is set by the 5x 63 command.

## 7. CAN Mode

Enter CAN mode with the \$E1 99 command.

The report \$91 10 indicates the AVT-84x has entered CAN operations.

The report \$83 11 00 00 indicates that CAN channel 0 is disabled.

The report \$83 11 04 00 indicates that CAN channel 4 is disabled.

The report \$91 19 indicates that LIN mode of operation is active.

The AVT-84x supports operations of two simultaneous CAN channels and one LIN or KWP channel when in CAN mode.

CAN0 is a 2-wire CAN channel that is ISO 11898 compliant.

It is channel 0.

CAN4 is a Single Wire CAN (SWC) channel that is J2411 compliant.

It is channel 4.

LIN mode is based on revisions 1.2, 1.3, and 2.0.

It is channel 5.

KWP mode is based on ISO 9141, ISO 9141-2, and ISO 14230.

It is channel 6.

When CAN mode is first entered, the following defaults are set:

- CAN0 and CAN4 operations are disabled.
- CAN0 baud rate is set to 500 kbaud.
- CAN4 baud rate is set to 33.333 kbaud.
- ID/Mask mode is set to 4 for both channels.
- All IDs are set for 11-bit with a value of \$07FF.
- All masks are set to zeros (must match condition).
- LIN mode is enabled and the LIN bus speed is 9600 baud.



### **7.1 CAN Secondary Operational Modes**

When operating in CAN mode, one of two secondary operating modes are supported. Either LIN mode or KWP (Key Word Protocol) mode can be selected to operate simultaneously with both CAN channels.

One secondary operational mode can be supported: LIN or KWP; not both.

After entering CAN mode, LIN mode is enabled as the default secondary mode.  
(Same as the 52 69 01 command.)

KWP mode can be selected using the 52 69 02 command. LIN mode is disabled.

The 52 69 00 command disables both LIN and KWP secondary modes.

### **7.2 CAN0 - 2-wire CAN**

CAN0 is a high speed 2-wire CAN channel that is ISO 11898 compliant. It uses the Philips TJA1050 transceiver. The CAN\_H signal is routed to the D-15 network connector pin #6. The CAN\_L signal is routed to the D-15 network connector pin #14.

The AVT-84x board has been designed to support a variety of network termination schemes for CAN0. The factory default is the split termination consisting of two 60 ohm resistors in series across the CAN\_H and CAN\_L signal lines. The mid-point of the two termination resistors is routed through a 10 ohm resistor and a 10,000 pF ceramic capacitor to ground. This configuration provides the standard 120 ohm DC termination and provides good common mode noise rejection.

Other termination configurations, including Ford compliant AC termination, are available - contact the factory for details.

### **7.3 CAN4 - Single Wire CAN (SWC)**

CAN4 is a low speed single wire CAN channel that is SAE J2411 compliant. It uses the Philips AU5790 transceiver. The SWC signal is routed through jumper JP1 to the D-15 network connector pin #1.

The factory default for jumper JP1 is OFF (to prevent possible damage in the event the AVT-84x is connected to a vehicle that does not have the SWC signal on pin #1).

#### **7.3.1 Jumper JP1**

Jumper JP1 on the AVT-84x board connects / disconnects the Single Wire CAN (SWC) line from pin #1 of P3; the DA-15P "Vehicle" connector. Usually, the factory default position of JP1 is installed.

### **7.4 CAN0 Channel Number**

CAN0 is designated channel "0".

Note: Bits in the upper nibble of the channel number have special meaning for some commands and responses.

### **7.5 CAN4 Channel Number**

CAN4 is designated channel "4".

Note: Bits in the upper nibble of the channel number have special meaning for some commands and responses.

### **7.6 LIN Secondary Operational Mode Channel Number**

When LIN mode is active as a secondary operational mode, it is designated channel “5”.

ABIC operations (in LIN mode only) are designated by channel “\$15”.

### **7.7 KWP Secondary Operational Mode Channel Number**

When KWP mode is active as a secondary operational mode, it is designated channel “6”.

### **7.8 CAN Communications General Notes**

A CAN network has to consist of at least two functioning CAN nodes.

(The AVT-84x can be one of the nodes).

Each CAN channel of the AVT-84x is independent of the other channel.

(This applies to all channel parameters.)

Each CAN channel of the AVT-84x has three operating modes:

Disabled

Normal

Listen only.

Messages to and from the network are of the form: \$0x yy rr ss tt vv ... where “x” is the count of bytes to follow. Refer to Sections 4.4, 7.3, and 13 for detailed information about CAN messages and packets to and from the network.

#### **7.8.1 Disabled**

The CAN channel can not receive any messages and it can not transmit any messages.

Command: 73 11 0x 00      Status report: 83 11 0x 00

#### **7.8.2 Normal**

The CAN channel will receive all messages from the network. It will assert the CAN frame ACK bit for all frames it receives without error. Only those frames it receives, where the message ID matches the acceptance ID according to the mask and associated rules, are passed to the host. Refer to Section 7.5 for a discussion.

The CAN channel can transmit messages.

Command: 73 11 0x 01      Status report: 83 11 0x 01

#### **7.8.3 Listen Only**

The CAN channel can only receive messages. It can not transmit messages and it can not assert the CAN frame ACK bit.

The CAN channel will receive all messages from the network. It can not assert the CAN frame ACK bit. Only those frames it receives, where the message ID matches the acceptance ID according to the mask and associated rules, are passed to the host. Refer to Section 7.5 for a discussion.

The CAN channel can only monitor received messages.

Command: 73 11 0x 02      Status report: 83 11 0x 02

#### 7.8.4 Time Stamps

Time stamps for both transmit acknowledgement and received messages can be disabled or enabled using the 5x 08 command.

Transmit ack: the time stamp is a two byte value immediately after the packet header byte but before the CAN channel number.

Receive message: the time stamp is a two-byte value immediately after the packet header byte but before the CAN channel number.

In CAN mode only, there are two sources for the time stamp.

The 52 08 00 command disables all time stamps.

The 52 08 01 command enables time stamps where the time stamp clock is separate for each CAN channel and separate from the LIN channel.

In this case the time stamp is a 16-bit free running counter that is driven by the baud clock for that channel. In other words, the time stamp is the inverse of the CAN channel baud rate.

Example #1: Baud rate is 500 kbaud. The time stamp interval is 2 microseconds.

Example #2: Baud rate is 33.333 kbaud. The time stamp interval is 30 microseconds.

The time stamp rolls over at \$FFFF.

The 52 08 02 command enables time stamps where the time stamp clock is the same for both CAN channels and the LIN channel. The time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at \$FFFF.

##### 7.8.4.1 Transmit Acknowledgment Examples

Time stamps disabled:

02 0x 0y:      Transmit ack.  
                   x:      CAN channel, 0 or 4.  
                   y:      transmit buffer number.

Time stamps enabled:

04 rr ss 0x 0y:      Transmit ack.  
                           rr ss:    time stamp, high byte, low byte.  
                           x:      CAN channel, 0 or 4.  
                           y:      transmit buffer number.

##### 7.8.4.2 Receive Message Examples

Time stamps disabled:

0p xy tt vv ww zz mm nn ... :  
           p:      count of bytes to follow.  
           x:            b7:    IDE  
                           0:      11-bit ID  
                           1:      29-bit ID  
                           b6:    RTR  
                           0:      normal frame

```

1:      RTR true, remote transmit request
b5:    0
b4:    0
y:      CAN channel, 0 or 4.
tt vv:  11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data field.

```

Time stamps enabled:

```

0p rr ss xy tt vv ww zz mm nn ... :
p:      count of bytes to follow.
rr ss:  time stamp.
x:      b7:  IDE
          0:  11-bit ID
          1:  29-bit ID
b6:     RTR
          0:  normal frame
          1:  RTR true, remote transmit request
b5:    0
b4:    0
y:      CAN channel, 0 or 4.
tt vv:  11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data field.

```

## 7.9 Acceptance ID and Mask Configuration

Each CAN channel of the AVT-84x is independent of the other channel.

Each CAN channel of the AVT-84x has three ID/Mask modes:

Mode 2.

The CAN channel has two 32-bit acceptance IDs and two corresponding 32-bit masks.

Mode 4.

The CAN channel has four 16-bit acceptance IDs and four corresponding 16-bit masks.

Mode 8.

The CAN channel has eight 8-bit acceptance IDs and eight corresponding 8-bit masks.

Acceptance IDs and masks are numbered sequentially starting at 0.

For example, in mode 4, the acceptance IDs and masks are numbered 0, 1, 2, and 3.

## 7.10 Acceptance ID and Mask Operation

Acceptance IDs and Masks are associated as pairs.

Acceptance ID0 is paired to Mask0; acceptance ID1 is paired to Mask1, etc.

A zero in a bit position in a mask is a “Must Match” condition for the acceptance ID.

A one in a bit position in a mask is a “Don’t Care” condition for the acceptance ID.

If the acceptance ID and mask are shorter than the actual message ID, the acceptance ID, mask, and message ID are all aligned to the left starting with the Most Significant Bit and go to the right.

Message ID bits with no corresponding acceptance ID or mask bits are “Don’t Care.”

How the acceptance IDs and masks operate.

- A message is received from the network.
- The message ID is passed through Mask0.
- The result is compared to acceptance ID0.
- If there is a match, the message is passed to the host.
- If there is not a match, the process is repeated for acceptance ID1 and Mask1, etc. until all acceptance ID and mask pairs are exhausted, at which point the message is discarded.

For modes 2 and 4, when setting up the acceptance ID, the user must specify in the command if the desired message ID is 11-bit or 29-bit. Refer to the CAN Commands and Responses later in this document; Section 13.

For modes 2 and 4, when the acceptance ID is for a 29-bit message, the SRR bit in the acceptance ID is set to “1” and the mask location for the SRR bit is set to Must Match.

For modes 2 and 4 the operator can program the RTR bit as desired. The associated mask RTR bit can also be configured as desired.

In mode 8, the size of the message ID (11 or 29-bit) and the state of the RTR bit are irrelevant.

If all messages being received use an 11-bit ID, modes 4 or 8 are recommended.

As a possible aid, a bit pattern diagram to help compute acceptance IDs and masks for given message IDs is provided at the end of this document.

A few examples are provided below to assist in understanding message ID, acceptance ID, and mask operations.

ID and mask bits, when specified in the command, are right justified.

#### **7.10.1 ID/Mask mode = 2**

29-bit IDs: All ID bits (ID28 : ID00) can be specified.  
Control bits IDE and RTR can be specified.

11-bit IDs: All ID bits (ID10 : ID00) can be specified.  
Control bits IDE and RTR can be specified.

#### **7.10.2 ID/Mask mode = 4**

29-bit IDs: ID bits (ID28 : ID15) can be specified.  
Control bits IDE and RTR can be specified.

11-bit IDs: All ID bits (ID10 : ID00) can be specified.  
Control bits IDE and RTR can be specified.

#### **7.10.3 ID/Mask mode = 8**

29-bit IDs: ID bits (ID28 : ID21) can be specified.  
Control bits can not be specified.

11-bit IDs: ID bits (ID10 : ID03) can be specified.  
Control bits can not be specified.

#### **7.10.4 ID/Mask Example #1**

Channel CAN0.

ID/Mask mode = 2.

Acceptance ID and mask are 32-bit values.

ID and mask #0.

Desired message is a 29-bit ID = 12 34 56 78.

RTR bit is 0 (do not receive RTR frames).

The following commands are used.

```
; set CAN0, ID/mask mode to 2
73 2B 00 02

; set CAN0, acceptance ID0, 29-bit, RTR=0
77 2A 80 00 12 34 56 78

; set CAN0, Mask0, all bits are "must match"
77 2C 00 00 00 00 00 00
```

Only network messages with a 29-bit ID = 12 34 56 78 and RTR = 0 will be received.

(It is assumed the operator has completed all other necessary channel commands.)

#### **7.10.5 ID/Mask Example #2**

Channel CAN4.

ID/Mask mode = 2.

Acceptance ID and mask are 32-bit values.

ID and mask #1.

Desired messages are 11-bit ID = 073x.

RTR bit is 0 (do not receive RTR frames).

The following commands are used.

```
; set CAN4, ID/mask mode to 2
73 2B 04 02

; set CAN4, acceptance ID1, 11-bit, RTR=0
75 2A 04 01 07 30

; set CAN4, Mask1, low order 4 bits are "don't care," all other bits are "must match."
75 2C 04 01 00 0F
```

All network messages with 11-bit IDs of the form 073x will be received.

(It is assumed the operator has completed all other necessary channel commands.)

Note that since the desired message IDs are 11-bit, using ID/Mask mode of 2 is a very inefficient use of resources; but it does work.

**7.10.6 ID/Mask Example #3**

Channel CAN4.

ID/Mask mode = 4.

Acceptance ID and mask are 16-bit values.

ID and mask #2.

Desired messages are 11-bit ID = 07Ex.

RTR bit is x (receive frames where RTR = 0 or 1).

The following commands are used.

; set CAN4, ID/mask mode to 4

73 2B 04 04

; set CAN4, acceptance ID2, 11-bit, RTR=0

75 2A 04 02 07 E0

; set CAN4, Mask2, low order 4 bits are “don’t care,” all other bits are “must match.”

75 2C 44 02 00 0F

All network messages with 11-bit IDs of the form 07Ex will be received, including RTR frames.

(It is assumed the operator has completed all other necessary channel commands.)

This is very similar to Example #2, but twice as many acceptance IDs and masks are available.

**7.10.7 ID/Mask Example #4**

Channel CAN0.

ID/Mask mode = 4.

Acceptance ID and mask are 16-bit values.

ID and mask #3.

Desired messages are 29-bit ID = 2B CC xx xx [or] 2B CD xx xx

RTR bit is 0 (do not receive RTR frames).

IDE and RTR mask bits are “must match.”

The following commands are used.

; set CAN0, ID/mask mode to 4

73 2B 00 04

; set CAN0, acceptance ID3, 29-bit, RTR=0

75 2A 80 03 2B CC

; set CAN0, Mask3, bit 0 is don’t care (ID bit 15)

75 2C 00 03 00 01

Only network messages with 29-bit ID = “2B CC xx xx” and “2B CD xx xx” will be received, no RTR frames. (It is assumed the operator has completed all other necessary channel commands.)

**7.10.8 ID/Mask Example #5**

Channel CAN0.

ID/Mask mode = 8.

Acceptance ID and mask are 8-bit values.

ID and mask #6.

Desired messages have the upper 8 bits equal to A5.

IDE bit is “don’t care.”

(Will receive both 29-bit and 11-bit messages, if they meet acceptance criteria)

RTR bit is “don’t care.”

(Will receive both non-RTR and RTR frames.)

Send the following commands.

    ; set CAN0, ID/mask mode to 8

73 2B 00 08

    ; set CAN0, acceptance ID6

74 2A 00 06 A5

    ; set CAN0, Mask6, mask bit #2 is don’t care, all others are must match.

74 2C 00 06 04

All network messages with 11-bit IDs in the following ranges will be received

03 08 to 03 0F

03 28 to 03 2F

All network messages with 29-bit IDs in the following ranges will be received.

28 40 00 00 to 28 7F FF FF

29 40 00 00 to 29 7F FF FF

### **7.10.9 Summary**

As can be seen, the ID/Mask mode, the acceptance ID, and the masks provide a powerful tool set that gives the user a very flexible means of receiving only those CAN network messages that are of interest. However, setting the mode, acceptance IDs, and masks is a non-trivial effort that requires some thought and analysis based on the particular application.

## **7.11 Setting up a CAN channel for operation**

The following sequence is recommended for setting a CAN channel for operations.

1. Research and examine the message IDs you want to receive.
2. Figure out which ID/Mask mode best suits your requirements.
3. Enter CAN mode. Leave the CAN channel disabled during setup.
4. Set the CAN channel baud rate.
5. Set the ID/Mask mode.
6. Set the acceptance ID(s).
7. Set the mask(s). [Optional.]
8. Enable the CAN channel.



### 7.11.1 Communications Example

The following example is using CAN0, 11-bit IDs, receive all messages of the form 07 Ex, transmit ID = 07 80, send one message, receive one message.

```
; enter CAN mode
E1 99

; set CAN0 to 500 kbaud
73 0A 00 02

; set CAN0 ID/Mask mode = 4
73 2B 00 04

; set CAN0 ID0 = 07 E0
75 2A 00 00 07 E0

; set CAN0 Mask0 low order 4-bits are "don't care."
75 2C 00 00 00 0F

; enable CAN0 for normal operations
73 11 00 01

; send a message with 5 bytes in the data field to ID = 07 80
08 00 07 80 04 11 22 33 44

; receive the transmit ack = 02 00 01

; receive a message from the network = 0B 00 07 E3 05 AA BB CC DD EE 00 00
```

### 7.12 Channel Activity

The “Channel Activity” function is available for both CAN channels, independently. When the function is enabled for a CAN channel and the AVT-84x receives a CAN frame, a counter is incremented and the message is immediately discarded. This continues until the counter reaches a value of \$FF. The counter will not rollover and will not automatically reset.

The host computer can query for network activity on that CAN channel at any time. The AVT-84x will issue a response with the number of CAN frames received (up to the maximum of \$FF) since the last query and then the counter is immediately reset.

The “73 3B xx yy” command disables/enables the “Channel Activity” function of the specified CAN channel.

The “72 3C xx” command reads and resets the activity counter for the specified CAN channel.

Note that the CAN channel must be properly set up to receive CAN messages for this function to work.

### 7.13 Periodic Message Support

Each CAN channel has the ability to transmit as many as thirty-two messages automatically. The operator defines and sets up the desired periodic messages, enables them, and the AVT-84x unit will then transmit those messages, at the defined interval, without any operator intervention.

A common use for this capability is for the “Tester Present” message that some ECUs (Electronic Control Units) require when in diagnostic mode. Another use would be in a simulation scenario.

Messages within a group can be set to operate in Type1 or Type2 mode.

The AVT-84x will not issue a transmit ack when a periodic message is transmitted.

### 7.13.1 Organization of Periodic Messages

Each CAN channel (0 and 4) has available two groups (1 and 2) of periodic messages with sixteen messages in each group.

For each CAN channel, the periodic messages are numbered from \$01 to \$20.

Group1:        messages \$01 thru \$10

Group2        messages \$11 thru \$20

Each message is independently disabled or enabled (7x 1B command).

Each message has its own time interval (7x 1A command); valid only in Type1 operations.

### 7.13.2 Periodic Message Master Timer

There is one timer that governs:

    The Analog To Digital (ATD) functions.

    Type1 periodic messages.

    Type2 periodic messages.

The time interval for that timer is set with the 52 63 xx command.

The available settings are:

    98.30 msec [Default]

    49.15 msec

    20.48 msec

    10.24 msec

    5.12 msec

### 7.13.3 Type1 Periodic Messages

Type1 periodic messages operate independently of each other.

When Type1 operations are enabled, each enabled message in that group operates according to its own interval count.

    The message is set up (ID and data field are defined).

    The interval count is defined.

    The message is enabled.

    The group is enabled for Type1 operations.

#### 7.13.3.1 Type1 Example

Want to send two messages on CAN0 at 500 kbaud. One message about every 500 msec. The other message about every 1 second. Using CAN0, Group1, Type1 operations, here is a sequence of commands to do this. It is assumed that this is from a reset condition.

1. ; Enter CAN mode  
    E1 99
2. ; Set CAN0 baud rate to 500 kbps  
    73 0A 00 02

3. ; Enable CAN0 normal operations  
73 11 00 01
4. ; Set the master timer to 98.30 msec  
52 63 01
5. ; Define periodic message #01, ID = 0246, data = 03 A3 B4 C5  
79 18 01 00 02 46 03 A3 B4 C5
6. ; Set periodic message #01 for an interval count of 10, actual interval = 0.983 msec  
74 1B 00 01 0A
7. ; Enable periodic message #01  
74 1A 00 01 01
8. ; Note that nothing will be transmitted until the group control is set to Type1
9. ; Define periodic message #06, ID = 0498, data = 04 1A 2B 3C 4D  
7A 18 06 00 04 98 04 1A 2B 3C 4D
10. ; Set periodic message #06 for an interval count of 5, actual interval = 0.4915 msec  
74 1B 00 06 05
11. ; Enable periodic message #06  
74 1A 00 06 01
12. ; Enable CAN0, Group1, for Type1 operations  
; At this point all enabled messages in CAN0, Group1, will begin transmission according to their own independent schedule  
74 0C 00 01 01

#### 7.13.4 Type2 Periodic Messages

Type2 periodic messages are transmitted in sequence, within the group.

When more than one message in a group is defined and enabled, and the group operating mode is set for Type2 operations (7x 0C command) then those messages will be transmitted, in sequence, using the interval count of the first message in the group (regardless if that first message is used or not).

For Group1 messages, only message \$01 interval count is used.

For Group2 messages, only message \$11 interval count is used.

The sequential messages are setup.

All are in the same group.

The interval count is defined. Only use the interval count of the first message in the group.

The messages are enabled.

The group is enabled for Type2 operations.

##### 7.13.4.1 Type2 Example

Want to send three messages on CAN4 at 33.333 kbaud. One message every 2.5 seconds. Using CAN4, Group2, Type2 operations, here is a sequence of commands to do this. It is assumed that this is from a reset condition.

1. ; Enter CAN mode  
E1 99
2. ; Set CAN4 baud rate to 33.333 kbps  
73 0A 04 0A
3. ; Enable CAN4 normal operations  
73 11 04 01
4. ; Enable CAN4 SWC transceiver for normal operations  
72 12 03
5. ; Set the master timer to 98.30 msec  
52 63 01
6. ; Define periodic message #0B, ID = 0123, data = AA 34 56  
78 18 0B 04 01 23 AA 34 56
7. ; Enable periodic message #0B  
74 1A 04 0B 01
8. ; Note that nothing will be transmitted until the group control is set to Type2
9. ; Define periodic message #0D, ID = 0234, data = BB 78 9A BC  
79 18 0D 04 02 34 BB 78 9A BC
10. ; Enable periodic message #0D  
74 1A 04 0D 01
11. ; Define periodic message #0F, ID = 0345, data = CC 34 56 78 9A BC  
7A 18 0F 03 45 CC 34 56 78 9A BC
12. ; Enable periodic message #0F  
74 1A 04 0F 01
13. ; Set the CAN4, Group2, Type2 interval for a count of \$19 (decimal 25)  
; the actual interval = 2.4575 sec.      must use message \$09 timer  
74 1B 04 09 19
14. ; Enable CAN4, Group2, for Type2 operations  
; At this point all enabled messages in CAN4, Group2, will begin transmitting in sequence,  
one message every 2.4575 seconds.  
74 0C 04 02 02

### 7.13.5 Periodic Message Commands

All commands are listed in the Commands and Responses section at the end of this document.  
A brief summary of them is provided here.

- 5x 63      Timer interval
- 7x 0C      Periodic message group operation control (disabled, Type1, Type2)
- 7x 18      Define a periodic message
- 7x 1A      Periodic message disable/enable
- 7x 1B      Periodic message interval

- 7x 1C      Disable all periodic messages  
              Disable all groups

### **7.14 ISO 15765 Support**

This may also be known as Multi-Frame Messaging (MFM) or Segmented Messages. This protocol is often used in diagnostic modes for all messages and is particularly useful when moving large blocks of data.

Firmware version 0.6 provides full ISO 15765 processing capability for both transmitting and receiving messages for both CAN0 and CAN4 channels. When ISO 15765 processing has been enabled for a CAN channel, that CAN channel has the ability to receive or transmit single frame and multi-frame messages up to the maximum of 4095 data bytes. Also available is “Address Extension” (AE) support.

#### **7.14.1 ISO 15765 Terms and/or Definitions**

AE:      Address Extension byte as defined in the ISO 15765 specification. If being used, it resides in the first byte, or byte0, of the CAN frame data field. AE is used at the discretion of the system designer.

PCI:      Protocol Control Information byte. The PCI byte is the first byte, or byte0, in the CAN frame data field; or, if “AE” is being used, the PCI byte is byte1 in CAN frame data field. The PCI byte tells the receiver how to process the CAN frame.

Padding:      Some networks require that all CAN frames be full length, that is the data field of the frame consist of 8 bytes - hence the frame is padded.

Flow Control:      A CAN frame with protocol handshaking information.

Separation Time:      A time parameter in the Flow Control frame. Some networks require the Separation Time to have a specific value.

#### **7.14.2 ISO 15765 Receive Operations - General Notes**

When ISO 15765 processing is enabled, the AVT-84x will not send the PCI byte to the host, unless an error is encountered.

When receiving a multi-frame message, the AVT-84x will automatically compose and send a Flow Control frame, if necessary. The AVT-84x will never send a transmit ack (02 0x 0y) to the host when a Flow Control frame is transmitted.

During receive operations, the padding command, 7x 27, only affects transmitted Flow Control frames associated with receiving a multi-frame message.

The 7x 27 command allows the user to change the value of the pad byte, if enabled.

Some networks use “FF” and others use “00” as the pad byte. The user is free to specify any byte value.

Pad bytes are always stripped from a received frame; regardless of the state of the padding command.

Some networks require the Flow Control frame Separation Time to have a specific value.

Many networks will accept the default Separation Time parameter.

The user can change the Flow Control separation time using the 7x 0E command.

[If a received Flow Control frame has a separation time of zero or an invalid time, then a default value of 2 milliseconds is used for transmitted frames. This value can be changed using the 7x 34 command.]

This was changed in firmware version 4.3. A received flow control separation time of 0 milliseconds is now valid and frames will be sent as quickly as possible, without delay.

Error checking is implemented and the host is notified when an error is encountered. A list of the error codes is provided in the Commands and Responses section of this document.

### 7.14.3 ISO 15765 Receive Operations - Brief Description

When ISO 15765 processing is enabled on a given CAN channel, all incoming messages (subject to the usual CAN frame ID filtering) are processed according to the 'rules' of ISO 15765.

The PCI byte is stripped from the data field and decoded.

If the CAN message is a single frame, the frame is sent to the host (PCI byte omitted).

If the CAN message(s) are part of a multi-frame sequence, the AVT-84x will save the ID, save the AE byte (if enabled), remove the PCI bytes, remove any pad bytes, and buffer the inbound data. The AVT-84x handles all handshaking with the downstream module. When the complete message is received, the AVT-84x forwards the ID and block of data to the host using the usual 'packet' convention:

0x rr ss tt vv mm nn ...	'x' is the count of bytes to follow.
or	
11 xx rr ss tt vv mm nn ...	'xx' is the count of bytes to follow.
or	
12 xx yy rr ss tt vv mm nn ...	'xx yy' is the count of bytes to follow.

There are several commands associated with setting up and enabling ISO 15765 processing. The commands are listed in the Commands and Responses section at the end of this document.

Note that when ISO 15765 processing is enabled for a specified CAN channel, all messages received by that CAN channel are expected to be ISO 15765 formatted and are processed accordingly.

### 7.14.4 ISO 15765 Receive Operations - Examples

To use this capability, the user must:

Set the acceptance ID(s) for the message(s) expected.

7x 2A ...

Set the acceptance mask mode.

7x 2B ...

Set the acceptance mask(s).

7x 2C ...

Enable or disable message padding.

7x 27 ...

Enable or disable Address Extension (AE).

73 30 0x 0y

Specify the flow control ID (and AE byte, if enabled).

7x 0F ...

Enable ISO 15765 processing.

73 26 0x 01

Enable the CAN channel.

73 11 0x 01

#### **7.14.4.1 ISO 15765 Receive Operations - Example #1**

The user wants to set up the AVT-84x to receive ISO 15765 formatted messages from a CAN module. The specifics are:

- 2-wire CAN at 500k baud.
- 11-bit message IDs.
- No “AE”.
- Receive message ID = 357.
- Transmit message ID = 246.  
(The flow control frame ID is usually the same as the transmit message ID.)

Note that CAN0 must be used. Here are the commands to set up CAN0, in sequence:

1. Switch to CAN mode  
E1 99
2. Set baud rate to 500 kbaud  
73 0A 00 02
3. Set acceptance ID0 to 357  
75 2A 00 00 03 57
4. Set mask mode to 4 (qty. 4 16-bit masks)  
73 2B 00 04
5. Set mask0 to all bits must match  
75 2C 00 00 00 00
6. Disable padding  
73 27 00 00
7. Disable AE  
73 30 00 00
8. Set Flow Control ID = 246  
74 0F 00 02 46
9. Set Flow Control separation time  
(this is the default value and does not need to be set, if it has not been changed before)  
73 0E 00 0A
10. Enable ISO 15765 processing for CAN0  
73 26 00 01
11. Enable CAN0 operations  
73 11 00 01

At this point the AVT-84x will receive all network messages with matching ID, process them according to ISO 15765 protocol, and send the results to the host.

#### **7.14.4.2 ISO 15765 Receive Operations - Example #2**

The user wants to set up the AVT-84x to receive ISO 15765 formatted messages from a CAN module. The specifics are:

- Single Wire CAN (SWC) at 33.333 kbaud.
- 29-bit message IDs.
- No “AE”.
- Receive message ID = 13 57 9A CE.
- Transmit message ID = 02 46 8B DF.  
(The flow control frame ID is usually the same as the transmit message ID.)

Note that CAN4 must be used. Here are the commands to set up CAN4, in sequence:

1. Switch to CAN mode  
E1 99
2. Set baud rate to 33.333 kbaud  
73 0A 04 0A
3. Set acceptance ID0 to 13 57 9A CE  
77 2A 84 00 13 57 9A CE
4. Set mask mode to 2 (qty. 2 32-bit masks)  
73 2B 04 02
5. Set mask0 to all bits must match  
77 2C 04 00 00 00 00 00
6. Disable padding  
73 27 04 00
7. Disable AE  
73 30 04 00
8. Set Flow Control ID = 02 46 8B DE  
76 0F 84 02 46 8B DF
9. Set Flow Control separation time  
(“0A” is the default value, change it only if the network requires a different value)  
73 0E 04 0A
10. Enable ISO 15765 processing for CAN4  
73 26 04 01
11. Set CAN4 transceiver to normal mode  
72 12 03
12. Enable CAN4 operations  
73 11 04 01

At this point the AVT-84x will receive all network messages with matching ID, process them according to ISO 15765 protocol, and send the results to the host.



### 7.14.5 ISO 15765 Transmit Operations - General Notes

When ISO 15765 processing is enabled, the AVT-84x will handle everything required to transmit a message to the network conforming to the ISO 15765 protocol standard.

The user only has to provide:

- Message ID and length.
- “AE” byte, only if AE function is enabled.
- All of the data.

All of that information is contained in the transmit command.

Be sure to OMIT PCI byte(s) and / or pad bytes from the transmit command.

If any problems are encountered - one or more error messages will be sent to the host.

If padding is enabled, the AVT-84x will compute the required number of pad bytes and insert them into the data field of the correct frame.

When ISO 15765 processing is enabled for a specified CAN channel, all messages transmitted by that CAN channel will be ISO 15765 formatted when transmitted to the network.

When ISO 15765 processing is disabled:

- The minimum number of data bytes in a CAN message is 0.
- The maximum number of data bytes in a CAN message is 8.
- The transmit format “0x ....” will always work.
- The alternate transmit command formats (“11 xx” and “12 xx yy”) are available and will work.

When ISO 15765 processing is enabled:

- The minimum number of data bytes in a CAN message is 0.
- The maximum number of data bytes in a CAN message is 4095.
- The transmit format “0x ....” will not always work.
- The alternate transmit command formats (“11 xx” and “12 xx yy”) are available and will work.

### 7.14.6 ISO 15765 Transmit Operations - Brief Description

Once the AVT-84x receives a complete transmit command from the host, it will perform the following.

- Check the command.
- Store the data.
- Determine if the transmit operation requires a single frame or multiple frames.
- Compute the required “PCI” byte and insert it into the data field.
- Compose the frame or frames including the “PCI” byte and “AE” byte (if enabled).
- Pad the data field of the frame (if enabled).
- Wait for and decode the received flow control frame (if necessary).
- Send one transmit ack to the host (if enabled) when the entire transaction is complete.

OMIT all PCI byte(s) from the transmit command.

OMIT any pad byte(s) from the transmit command.

Include the “AE” byte just once, in the correct place of the command, if “AE” is enabled for that CAN channel.

#### **7.14.7 ISO 15765 Transmit Operations - Examples**

To use this capability, the user must:

Set the acceptance ID.

For the expected flow control message.

[Only necessary if transmitting more data than will fit in a single frame.]

7x 2A ...

Set the acceptance mask mode.

For the expected flow control message.

7x 2B ...

Set the acceptance mask(s).

For the expected flow control message.

7x 2C ...

Enable or disable message padding.

7x 27 ...

Enable or disable Address Extension (AE).

7x 30 ...

Enable ISO 15765 processing.

73 26 0x 01

Enable the CAN channel.

73 11 0x 01

##### **7.14.7.1 ISO 15765 Transmit Operations - Example #1**

The user wants to set up the AVT-84x to transmit ISO 15765 formatted messages to a CAN module. The specifics are:

- 2-wire CAN at 500k baud.
- 11-bit message IDs.
- No “AE”.
- Receive message ID = 357 (flow control ID).

Note that CAN0 must be used. Here are the commands to set up CAN0, in sequence:

1. Switch to CAN mode

E1 99

2. Set baud rate to 500 kbaud

73 0A 00 02

3. Set acceptance ID0 to 357

75 2A 00 00 03 57

4. Set mask mode to 4 (qty. 4 16-bit masks)

73 2B 00 04

5. Set mask0 to all bits must match  
75 2C 00 00 00 00
6. Disable padding  
73 27 00 00
7. Disable AE  
73 30 00 00
8. Enable ISO 15765 processing for CAN0  
73 26 00 01
9. Enable CAN0 operations  
73 11 00 01
10. Transmit a message with ID = 02 57 and 8 bytes of data.  
0B 00 02 57 11 22 33 44 55 66 77 88

At this point the AVT-84x will, according to the ISO 15765 protocol, compose the required CAN frames and transmit them to the network. When the transaction is complete, the AVT-84x will send one transmit ack to the host (if transmit acks are enabled).

#### **7.14.7.2 ISO 15765 Transmit Operations - Example #2**

The user wants to set up the AVT-84x to transmit ISO 15765 formatted messages to a CAN module. The specifics are:

- Single Wire CAN (SWC) at 33.333 kbaud.
- 29-bit message IDs.
- No “AE”.
- Receive message ID = 13 57 9A CE (flow control ID).

Note that CAN4 must be used. Here are the commands to set up CAN4, in sequence:

1. Switch to CAN mode  
E1 99
2. Set baud rate to 33.333 kbaud  
73 0A 04 0A
3. Set acceptance ID0 to 13 57 9A CE  
77 2A 84 00 13 57 9A CE
4. Set mask mode to 2 (qty. 2 32-bit masks)  
73 2B 04 02
5. Set mask0 to all bits must match  
77 2C 04 00 00 00 00 00
6. Disable padding  
73 27 04 00
7. Disable AE  
73 30 04 00

8. Enable ISO 15765 processing for CAN4  
73 26 04 01
9. Set CAN4 transceiver to normal mode  
72 12 03
10. Enable CAN4 operations  
73 11 04 01
11. Transmit a message with ID = 1A 2B 3C 4D and 18 (decimal) bytes of data.  
11 17 00 1A 2B 3C 4D 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18  
[Note that the alternate format header “11 xx” had to be used in this case.]

At this point the AVT-84x will, according to the ISO 15765 protocol, compose the required CAN frames and transmit them to the network. When the transaction is complete, the AVT-84x will send one transmit ack to the host (if transmit acks are enabled).

#### **7.14.8 ISO 15765 Questions and Engineering Support**

Some may find the ISO 15765 protocol and its implementation in the AVT-84x to be confusing and difficult. It needn't be. Contact the factory if you have questions - we will do our best to help.

#### **7.15 Auto Block Transmit**

A special mode of operation known as “Auto Block Transmit” (abbreviated as “ABX”) is available for the AVT-84x in CAN mode. The ABX mode was primarily intended for fully automated operations but can be adapted to a number of CAN applications.

The contents of Manual Supplement “84x\_s01a.pdf” is included here.

##### **7.15.1 Operation Description**

The ABX function gives an AVT-84x unit the capability of transmitting a sequence of CAN frames containing a total of up to 32 KBytes (32 768 bytes) of data - without host computer intervention.

Note: This function is only available in CAN mode. However, it is available to either CAN0 or CAN4 channels (separately, simultaneous operations are not permitted).

The operator (with a host computer) sets up and stores CAN data and CAN transmit parameters into non-volatile memory of the AVT-84x unit. This is done using the commands described below.

Then, using one simple command (issued by the host computer or stored as an auto start command) the AVT-84x will begin transmitting the stored CAN data. The data is transmitted according to the stored CAN parameters. The AVT-84x unit composes CAN frames and transmits them, in sequence, until all of the data is transmitted (or the operation is terminated by the host computer).

Presented below are:

- All commands involving non-volatile parameters are described first.
- The control command (does not involve non-volatile parameters) is then described.
- Lastly, a brief example is described.

### 7.15.2 Command Descriptions (non-volatile parameters)

The following commands involve querying for or storing various Auto Block Transmit (ABX) parameters. All of these parameters are stored in non-volatile memory of the AVT-84x.

All of these parameters should be initialized by the user prior to invoking the ABX function.

ABX operational parameters are stored in EEPROM space.

ABX data is stored in FLASH space.

#### 7.15.2.1 ABX Separation Count 7x 36

The 7x 36 command queries for and sets the CAN frame ‘separation count’. This is the time between CAN frames that the AVT-84x unit is transmitting.

This parameter sets a count value. The AVT-84x unit will wait by ‘counting’ the value specified by this command before transmitting the next CAN frame.

There are two ‘sources’ for this count: milliseconds or ‘loops’.

Setting the separation count to be milliseconds results in the AVT-84x ‘counting’ the specified number of milliseconds between transmitted CAN frames.

A ‘loop’ is the time it takes the AVT-84x firmware to make a complete loop in the firmware, in CAN mode. This time is variable, but empirical measurements in a lightly loaded environment reveal the ‘loop’ time to be approximately 45 microseconds. The user should keep in mind that a full size 11-bit CAN frame at 500 kbaud occupies approximately 186 microseconds on the CAN bus.

71 36:                    query for setting  
74 36 0r xx yy:        command  
                             0r = 01 = millisecond count  
                             0r = 02 = ‘loop’ count  
                             xx yy = count value  
                             default:        84 36 02 00 0A  
                             (‘loop’ with count of 10 decimal ~ 450 microseconds between CAN frames)

#### 7.15.2.2 ABX Message ID 7x 37

This command sets the transmit ID and related parameters for the transmitted CAN frame.

71 37:                    query for setting  
74 37 mm rr ss:        command  
                             mm:    b7 = IDE = 0 = 11-bit ID  
                                             b6 = RTR = should be set to 0, but user can define  
                                             all other bits (b5 to b0) are 0  
                             rr ss:    11-bit ID, right justified  
76 37 mm rr ss tt vv:    command  
                             mm:    b7 = IDE = 1 = 29-bit ID  
                                             b6 = RTR = should be set to 0, but user can define  
                                             all other bits (b5 to b0) are 0  
                             rr ss tt vv:    29-bit ID, right justified

### 7.15.2.3 ABX Data 76 38

This command queries for or stores the CAN frame data. All of this data is stored in the AVT-84x unit in non-volatile FLASH space.

The address range is \$0000 thru \$7FFF (16 KBytes = 32 768 bytes).

When transmitting, all data is read and transmitted starting at address \$0000.

The total number of data bytes transmitted is set by the 7x 39 command, described below.

Reading stored data is easy and very flexible.

- Data can be read starting at any address in the range and for any number of bytes specified (so long as the resulting address does not exceed the address range).

Storing or writing data has special rules.

- Data is stored in ‘sectors’ where a ‘sector’ is a maximum of 512 bytes.
- Data must be stored starting at a sector start address.
- A sector start address is on an even 512 byte boundary. In a 16-bit address, a sector start address has bits 8:0 all set to zero. In binary, a valid sector start address will be of the form:

xxxx xxx0 0000 0000

In hex, some example valid sector start addresses are (where ‘x’ is any value):

\$ x000  
\$ x200  
\$ x400  
\$ x600  
\$ x800  
\$ xA00  
\$ xC00  
\$ xE00

- It is not necessary to completely fill a sector. However, it is not allowed to specify a start address in a sector that is not the sector start address.
- Any data not specified when storing a sector is automatically filled with \$FF bytes.
- To completely fill the available 32 KBytes of ABX data would require the host computer to send 64 (decimal) commands to the AVT-84x unit.

Note that both the response to a query and the command have the unique format where the data does not contain a “header” byte. The data immediately follows the command (when storing) or the response (when making a query).

#### General format of the query and command.

76 38 0r ss tt kk ll

0r = 01 = store data

ss tt = sector start address

kk ll = count of bytes to immediately follow (if a command)  
or count of bytes requested (if a query)

#### Examples

query for stored data; send this query:

76 38 02 ss tt kk ll

request to read stored data  
start reading at address \$ ss tt  
read and send back to the host \$ kk ll number of bytes

the AVT-84x will respond with:

86 38 02 ss tt kk ll ..... the requested number of bytes will immediately follow.

store 512 bytes of data, or fewer; send this command:

76 38 01 ss 00 kk ll ..... the data must follow immediately

command to store data  
start storing the data at address \$ ss 00  
(which must be a valid sector start address)  
\$ kk ll specifies how many bytes the host computer  
will send immediately following the command  
(\$ kk ll valid range is \$0001 to \$0200)

After the AVT-84x receives the command and all expected data bytes, the AVT-84x will respond with:

86 38 01 ss tt kk ll

#### **7.15.2.4 ABX Byte Count 7x 39**

This parameter is the count of bytes the ABX function is going to read from non-volatile memory, fill CAN frames, and transmit.

Do not confuse this with the term ‘count’ used in any other command.

This is the count of the total number of data bytes that the ABX function will transmit in sequential CAN frames.

71 39:                    query for stored value

73 39 xx yy:            store the data count  
                             xx yy = number of bytes to transmit, total

#### **7.15.3 Command Description (control)**

There is only one command needed to commence an Auto Block Transmit operation. Described below.

(Once started, the function will run to completion unless the user issues a disable command.)

Remember that all parameters and data are stored in non-volatile memory on the AVT-84x unit. Thus, once an AVT-84x unit has been initialized, the user can start an Auto Block Transmit operation at any time. There is no need to re-initialize the ABX parameters just because the AVT-84x unit has been reset or power cycled. Note that all of this can be automated using stored “Auto Start Commands”.

### 7.15.3.1 ABX Control

### 7x 3A

Prior to issuing this command the user must have initialized all ABX operation parameters using the previously described commands.

(Note that the function can be invoked without initializing the stored parameters - it would work and likely be rather ugly.)

71 3A: query for operational status, both CAN channels

72 3A 0x: query for operational status for CAN channel 'x'

73 3A 0x 0y: set CAN channel 'x' operation to value 'y'

#### Examples

query for status of both CAN channels; send this query:

71 3A

receive these responses

83 3A 00 00 CAN0 is disabled

83 3A 04 00 CAN4 is disabled

query for status of CAN4; send this query:

72 3A 04

receive this response:

83 3A 04 00 CAN4 is disabled

enable the ABX function for channel CAN0; send this command:

73 3A 00 01

receive this response:

83 3A 00 01 CAN0 is enabled

Note: The ABX function using CAN0 will begin immediately.

when the transfer completes, receive this response:

83 3A 00 00 CAN0 is disabled

to disable an ABX operation that is in progress on CAN0; issue this command:

73 3A 00 00

Note: The ABX function will be halted immediately.

It can be restarted, but it can not be resumed.



#### 7.15.4 Operation Example

I want to set the following ABX parameters:

- set CAN ID = 03 57 (11-bit, no RTR)
- separation time to 20 milliseconds
- store some data
- total data count to 100 bytes

Then I will set the AVT-84x unit, as noted here, and transmit the block.

- CAN0 at 500 kbaud
- do not receive any CAN frames
- enable CAN0 for operations.

Communications from host computer to AVT-84x unit

```

; reset the unit
F1 A5

; enter CAN mode
E1 99

; set ABX CAN ID = 03 57, 11-bit, RTR = 0
74 37 00 03 57

; set ABX separation to milliseconds and a count of 20 (decimal)
74 36 01 00 14

; store some ABX data, start address = $0000, byte count to store = $0020
76 38 01 00 00 00 20
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

; note that the rest of the data, by default, = $FF

; set ABX transmit count to 100 bytes
73 39 00 64

; at this point all ABX parameters are defined and stored


; set CAN0 baud rate to 500 kbaud
73 0A 00 02

; will not define any acceptance ID masks
; will not define any acceptance IDs
; enable CAN0 for operations
73 11 00 01


; invoke the ABX function, using channel CAN0
73 3A 00 01

```

## 8. LIN operations – in CAN mode

As of firmware version 1.0 LIN communications are supported. At the present, LIN mode is only active when in CAN mode of operation. LIN mode is always enabled when in CAN mode. However, CAN0 and CAN4 channels can be individually enabled or disabled.

LIN mode operates simultaneously with CAN mode. LIN is designated as channel 5.

To enter LIN mode - enter CAN mode using the \$E1 99 command. The following responses or reports will be received:

\$91 10 indicates the AVT-84x has entered CAN operations.

\$83 11 00 00 indicates that CAN channel 0 is disabled.

\$83 11 04 00 indicates that CAN channel 4 is disabled.

\$91 19 indicates that LIN mode of operation is active.

LIN mode uses the AVT-84x K-line for communications.

LIN mode can be disabled with the 52 69 00 command.

### 8.1 Jumper JP2

Jumper JP2 on the AVT-84x board connects / disconnects the K-line (LIN bus) from pin #7 of P3; the DA-15P “Vehicle” connector.

### 8.2 Communications

Unless otherwise commanded, the AVT-84x will passively receive all messages from the LIN bus.

The AVT-84x is capable of communicating on (or transmitting to) the LIN bus as a Master without data, as a Master with data, or as a Slave with data. Those functions are only invoked by command from the user (through the host computer).

#### 8.2.1 Message Length

At this time, messages are received from the network and expected message length is computed based on the message ID. This operation is in accordance with LIN protocol specification revisions 1.2 and 1.3.

LIN protocol specification revision 2.0 has eliminated the relationship between message ID and expected frame length. Therefore, if a message ID and frame length is encountered that do not agree with LIN protocol specification revisions 1.2 or 1.3, a receive error may be encountered.

As of firmware version 1.1 the new “5x 28” command can be used to enable or disable LIN message processing using the ID byte. The default condition is enabled.

When ID byte processing is enabled the ID byte is used to determined expected message length.

When ID byte processing is disabled, maximum frame time is used to determine the end of a LIN message frame. Maximum frame time, in milliseconds, can be set using the “5x 52” command.

#### 8.2.2 Checksum

Both “Classic” and “Enhanced” checksum methods are available through the 5x 5A command.

### 8.2.3 ID Byte Only Message

If the Master on a LIN bus transmits the ID byte and no module on the bus responds, then the message is an “ID byte only” message. In firmware versions 1.0 through 1.4, if the AVT-84x interface received an ID byte only message, an error packet was sent to the host and the ID byte only message was discarded.

There was no way to “see” the ID byte and no way to suppress those error packets.

In firmware version 1.5 and later, the “5x 66” command and function were implemented.

The “52 66 00” command is the default condition. Operation is: ID byte only messages are discarded and nothing is sent to the host.

The “52 66 01” command causes the AVT-84x to notify the host that an ID byte only message was received and report the ID byte. The format of the notification is (time stamps disabled):

```

03 05 03 xx
or 03 05 83 xx
      03      means from the network, 3 bytes follow
      05      channel 5 = LIN
      03 or 83 receive status byte, frame timeout bit may or may not be set
                  message too short and checksum error bits are set
      xx      the received ID byte

```

### 8.2.4 Communications Example

This example is to enter CAN mode, receive a message from the LIN network (passively) and to send messages to the LIN network in the three possible methods. Time stamps are disabled.

```

; enter CAN mode
E1 99

; receive a LIN network message (passively)
05 05 00 C4 78 9A
; 0 indicates “from” the network
; 5 count of bytes to follow
; 05 channel 5 - LIN
; 00 status byte, no bits set indicates no errors detected
; C4 message ID
; 78 9A message data field

; act as a Master without data -- this elicits a response from a Slave node
03 05 01 25
; 0 indicates “to” the network
; 3 count of bytes to follow
; 05 channel 5 - LIN
; 01 master node
; 25 message ID

; act as a Master with data -- this sends a complete message onto the network
0B 05 01 B4 11 22 33 44 55 66 77 88
; 0 indicates “to” the network
; B count of bytes to follow = $B = 11 decimal
; 05 channel 5 - LIN

```

```

;      01 master node
;      B4 message ID
;      11 22 33 44 55 66 77 88 message data

; act as a Slave -- the node will wait for the Master to request data from the specified ID
05 05 00 C4 11 22
;      0 indicates "to" the network
;      5 count of bytes to follow
;      05 channel 5 - LIN
;      00 slave node
;      C4 message ID
;      11 22 message data

```

### 8.2.5 Time Stamp

Time stamps for both the transmit ack and received messages can be disabled or enabled using the 5x 08 command.

Transmit ack: the time stamp is a two byte value immediately after the packet header byte(s); but before the LIN channel number (05).

Receive message: the time stamp is a two-byte value immediately after the packet header byte(s); but before the LIN channel number (05).

The 52 08 00 command disables all time stamps.

The 52 08 01 command enables time stamps where the time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at \$FFFF.

#### 8.2.5.1 Receive Message Examples

When time stamps are disabled a receive message example is:

```

08 05 00 25 11 22 33 44
    08 header byte, indicates from the network, 8 bytes follow.
    05 channel 5 - LIN
    00 status byte indicating no errors detected.
    25 message ID.
    11 22 33 44 message bytes.

```

When time stamps are enabled a receive message example is:

```

0A xx yy 05 00 25 11 22 33 44
    0A header byte, indicates from the network, $A or decimal 10 bytes follow.
    xx yy time stamp (xx is the high byte, yy is the low byte).
    05 channel 5 - LIN
    00 status byte indicating no errors detected.
    25 message ID.
    11 22 33 44 message bytes.

```

#### 8.2.5.2 Transmit Ack Examples

When time stamps are disabled a transmit ack example is:

```

02 05 40

```

02 header byte, indicates from the network, 2 bytes follow.  
05 channel 5 - LIN  
40 status byte, bit 5 set, indicates “from this node”.

When time stamps are enabled a transmit ack example is:

04 xx yy 05 60  
04 header byte, indicates from the network, 4 bytes follow.  
xx yy time stamp (xx is the high byte, yy is the low byte).  
05 channel 5 - LIN  
40 status byte, bit 5 set, indicates “from this node”.

### **8.3 Periodic Message Support**

When LIN mode is active, the AVT-84x has the ability to transmit as many as ten (\$0A) messages automatically. The operator defines and sets up the desired periodic messages, enables them, and the AVT-84x unit will then transmit those messages, at the defined interval, without any operator intervention.

The AVT-84x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01) is enabled.

#### **8.3.1 Organization of Periodic Messages**

In LIN mode there is only one group of periodic messages: Group1.

All ten periodic messages are in Group1.

All ten periodic messages can be set to operate in Type1 or Type2 mode.

The periodic messages are numbered \$01 to \$0A (inclusive).

Each message is independently disabled or enabled (7x 1B 05 command).

Each message has its own time interval (7x 1A 05 command); valid only in Type1 operations.

#### **8.3.2 Periodic Message Master Timer**

There is one timer that governs:

The Analog To Digital (ATD) functions.

Type1 periodic messages.

Type2 periodic messages.

The time interval for that timer is set with the 52 63 xx command.

The available settings are:

98.30 msec [Default]  
49.15 msec  
20.48 msec  
10.24 msec  
5.12 msec

#### **8.3.3 Type1 Periodic Messages**

Type1 periodic messages operate independently of each other.

When Type1 operations are enabled, each enabled message in that group operates according to its own interval count.

The message is set up.  
The interval count is defined.  
The message is enabled.  
The group is enabled for Type1 operations.

### 8.3.3.1 Type1 Example

Want to send two messages automatically and independently. One message about every 500 msec. The other message about every 1 second. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

Note: LIN is channel 05.

1. ; LIN mode is only available in CAN mode  
; Enter CAN mode  
E1 99
2. ; Enable LIN operations (this is the default condition)  
52 69 01
3. ; Set the master timer to 98.30 msec  
52 63 01
4. ; Define periodic message #01.  
The message is: Master, ID = 25, data = 68 6A F1 3F  
79 18 01 05 01 25 68 6A F1 3F
5. ; Set periodic message #01 for an interval count of 10, actual interval = 0.983 msec  
74 1B 05 01 0A
6. ; Enable periodic message #01  
74 1A 05 01 01
7. ; Note that nothing will be transmitted until the group control is set to Type1
8. ; Define periodic message #06.  
The message is: Slave, ID = 37, data = 68 6A F1 01  
79 18 06 05 00 37 68 6A F1 01
9. ; Set periodic message #06 for an interval count of 5, actual interval = 0.4915 msec  
74 1B 05 06 05
10. ; Enable periodic message #06  
74 1A 05 06 01
11. ; Enable Group1 for Type1 operations  
; At this point all enabled messages in Group1, will begin transmission according to their own independent schedule  
74 0C 05 01 01

### 8.3.4 Type2 Periodic Messages

Type2 periodic messages are transmitted in sequence, within the group.

When more than one message in a group is defined and enabled, and the group operating mode is set for Type2 operations (7x 0C command) then those messages will be transmitted, in sequence, using the interval count of the first message in the group (regardless if that first message is used or not).

For Group1 messages, only message \$01 interval count is used.

The sequential messages are setup.

All are in the same group.

The interval count is defined. Only use the interval count of the first message in the group.

The messages are enabled.

The group is enabled for Type2 operations.

#### **8.3.4.1 Type2 Example**

Want to send three messages in sequence one message every 2.5 seconds. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

1. ; LIN mode is only available in CAN mode  
; Enter CAN mode  
E1 99
2. ; Enable LIN operations (this is the default condition)  
52 69 01
3. ; Set the master timer to 98.30 msec  
52 63 01
4. ; Define periodic message #02  
79 18 02 05 01 48 6B 10 41 0D
5. ; Enable periodic message #02  
74 1A 05 02 01
6. ; Note that nothing will be transmitted until the group control is set to Type2
7. ; Define periodic message #04  
79 18 04 05 00 48 6B 10 41 0D
8. ; Enable periodic message #04  
74 1A 05 04 01
9. ; Define periodic message #07  
7A 18 07 05 01 48 6B 10 41 0D 67
10. ; Enable periodic message #07  
74 1A 05 07 01
11. ; Set Group1 Type2 interval for a count of \$19 (decimal 25)  
; the actual interval = 2.4575 sec.          must use message \$01 timer  
74 1B 05 01 19
12. ; Enable Group1 for Type2 operations  
; At this point all enabled messages in Group1, will begin transmitting in sequence, one message every 2.4575 seconds.  
74 0C 05 01 02

### 8.3.5 Periodic Message Commands

All commands are listed in the Commands and Responses section at the end of this document. A brief summary of them is provided here.

- 5x 63      Timer interval
- 7x 0C      Periodic message group operation control (disabled, Type1, Type2)
- 7x 18      Define a periodic message
- 7x 1A      Periodic message disable/enable
- 7x 1B      Periodic message interval
- 7x 1C      Disable all periodic messages  
                 Disable all groups

### 8.4 ABIC Support

In LIN mode, there is support for transmitting messages to an “ABIC” module.

Please refer to the section: “LIN operations in CAN mode – Commands” for detailed information on how to format a transmit command for an ABIC module.

The response format is described in the section: “LIN operations in CAN mode – Responses”.

### 8.5 Commands and Responses

Refer to Section 14 for a complete list of LIN mode commands and responses.

## 9. KWP operations – in CAN mode

As of firmware version 2.2, KWP communications can be enabled while in CAN mode. KWP communications uses the K-line. When KWP secondary mode is enabled while in CAN mode, the LIN mode is disabled.

The secondary KWP mode of operations, while in CAN mode, is almost identical to KWP operations. Refer to Sections 9 and 11.

The biggest difference of using KWP mode while in CAN mode is that KWP commands are designated as channel “06”.

To enable secondary KWP mode, use the 52 69 02 command. The responses 62 69 02 and 91 0F will be issued to indicate that the command was processed and that KWP mode is now operational.

### 9.1 Jumper JP2

Jumper JP2 on the AVT-84x board connects / disconnects the K-line from pin #7 of P3; the DA-15P “Vehicle” connector.

### 9.2 Communications

Unless otherwise commanded, the AVT-84x will passively receive all messages from the K-line.



### 9.3 Operation Commands

Refer to the commands and responses for both CAN mode and KWP mode.

#### 9.3.1 Communications Example

This example is to enter CAN mode, receive a message from the K-line and to transmit a message on to the K-line.

```
; enter CAN mode
E1 99

; enable secondary mode as KWP
52 69 02

; receive a message from the K-line
05 06 00 C4 78 9A
;      0 indicates "from" the network
;      5 count of bytes to follow
;      06 channel 6 – KWP
;      00 status byte, no bits set indicates no errors detected
;      C4 78 9A message data field

; transmit a message onto the K-line
0B 06 A2 B4 11 22 33 44 55 66 77 88
;      0 indicates "to" the network
;      B count of bytes to follow = $B = 11 decimal
;      06 channel 6 – KWP
;      A2 B4 11 22 33 44 55 66 77 88 message data
```

#### 9.3.2 Time Stamp

Time stamps for both the transmit ack and received messages can be disabled or enabled using the 5x 08 command.

Transmit ack: the time stamp is a two byte value immediately after the packet header byte(s); but before the KWP channel number (06).

Receive message: the time stamp is a two-byte value immediately after the packet header byte(s); but before the KWP channel number (06).

The 52 08 00 command disables all time stamps.

The 52 08 01 command enables time stamps where the time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at \$FFFF.

##### 9.3.2.1 Receive Message Examples

When time stamps are disabled a receive message example is:

```
08 06 00 25 11 22 33 44
    08 header byte, indicates from the network, 8 bytes follow.
    06 channel 6 – KWP
    00 status byte indicating no errors detected.
```

25 message ID.  
11 22 33 44 message bytes.

When time stamps are enabled a receive message example is:

0A xx yy 06 00 25 11 22 33 44  
0A header byte, indicates from the network, \$A or decimal 10 bytes follow.  
xx yy time stamp (xx is the high byte, yy is the low byte).  
06 channel 6 – KWP  
00 status byte indicating no errors detected.  
25 11 22 33 44 message bytes.

### **9.3.2.2 Transmit Ack Examples**

When time stamps are disabled a transmit ack example is:

02 05 40  
02 header byte, indicates from the network, 2 bytes follow.  
06 channel 6 – KWP  
40 status byte, bit 5 set, indicates “from this node”.

When time stamps are enabled a transmit ack example is:

04 xx yy 05 60  
04 header byte, indicates from the network, 4 bytes follow.  
xx yy time stamp (xx is the high byte, yy is the low byte).  
05 channel 5 - LIN channel.  
40 status byte, bit 5 set, indicates “from this node”.

### **9.3.3 Fast Transmit**

During so-called “normal” transmission of a message onto the K-line, the AVT-84x inserts a default value of 5 milliseconds between bytes. This value can be changed using the “5x 27” command. A value of zero can be set using the “52 27 00” command.

However, due to internal processing, even when this value is set to zero, the AVT-84x unit still inserts a small delay between transmitted bytes. For some applications this (small) delay is unacceptable.

The “52 6C 01 Fast Transmit” command minimizes the delay between transmitted bytes to a much smaller amount of time.

## **9.4 Periodic Message Support**

When KWP mode is active, the AVT-84x has the ability to transmit as many as ten (\$0A) messages automatically. The operator defines and sets up the desired periodic messages, enables them, and the AVT-84x unit will then transmit those messages, at the defined interval, without any operator intervention.

The AVT-84x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01) is enabled.

### **9.4.1 Organization of Periodic Messages**

In KWP mode there is only one group of periodic messages: Group1.

All ten periodic messages are in Group1.

All ten periodic messages can be set to operate in Type1 or Type2 mode.

The periodic messages are numbered \$01 to \$0A (inclusive).

Each message is independently disabled or enabled (7x 1B 05 command).

Each message has its own time interval (7x 1A 05 command); valid only in Type1 operations.

#### **9.4.2 Periodic Message Master Timer**

There is one timer that governs:

- The Analog To Digital (ATD) functions.

- Type1 periodic messages.

- Type2 periodic messages.

The time interval for that timer is set with the 52 63 xx command.

The available settings are:

- 98.30 msec [Default]

- 49.15 msec

- 20.48 msec

- 10.24 msec

- 5.12 msec

#### **9.4.3 Type1 Periodic Messages**

Type1 periodic messages operate independently of each other.

When Type1 operations are enabled, each enabled message in that group operates according to its own interval count.

- The message is set up.

- The interval count is defined.

- The message is enabled.

- The group is enabled for Type1 operations.

##### **9.4.3.1 Type1 Example**

Want to send two messages automatically and independently. One message about every 500 msec. The other message about every 1 second. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

Note: KWP is channel 06.

- 12. ; KWP mode is only available in CAN mode

  - ; Enter CAN mode

    - E1 99

- 13. ; Enable KWP operations

  - 52 69 02

- 14. ; Set the master timer to 98.30 msec

  - 52 63 01

- 15. ; Define periodic message #01.  
The message is: 25 68 6A F1 3F  
78 18 01 06 25 68 6A F1 3F
- 16. ; Set periodic message #01 for an interval count of 10, actual interval = 0.983 msec  
74 1B 06 01 0A
- 17. ; Enable periodic message #01  
74 1A 06 01 01
- 18. ; Note that nothing will be transmitted until the group control is set to Type1
- 19. ; Define periodic message #03.  
The message is: 37 68 6A F1 01  
78 18 03 06 37 68 6A F1 01
- 20. ; Set periodic message #03 for an interval count of 5, actual interval = 0.4915 msec  
74 1B 06 03 05
- 21. ; Enable periodic message #03  
74 1A 06 03 01
- 22. ; Enable Group1 for Type1 operations  
; At this point all enabled messages in Group1, will begin transmission according to their  
own independent schedule  
74 0C 06 01 01

#### **9.4.4 Type2 Periodic Messages**

Type2 periodic messages are transmitted in sequence, within the group.

When more than one message in a group is defined and enabled, and the group operating mode is set for Type2 operations (7x 0C command) then those messages will be transmitted, in sequence, using the interval count of the first message in the group (regardless if that first message is used or not).

For Group1 messages, only message \$01 interval count is used.

The sequential messages are setup.

All are in the same group.

The interval count is defined. Only use the interval count of the first message in the group.

The messages are enabled.

The group is enabled for Type2 operations.

##### **9.4.4.1 Type2 Example**

Want to send three messages in sequence one message every 2.5 seconds. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

- 13. ; KWP mode is only available in CAN mode  
; Enter CAN mode  
E1 99
- 14. ; Enable KWP operations  
52 69 02

15. ; Set the master timer to 98.30 msec  
52 63 01
16. ; Define periodic message #02  
79 18 02 06 AB 48 6B 10 41 0D
17. ; Enable periodic message #02  
74 1A 06 02 01
18. ; Note that nothing will be transmitted until the group control is set to Type2
19. ; Define periodic message #04  
79 18 04 06 BC 48 6B 10 41 0D
20. ; Enable periodic message #04  
74 1A 06 04 01
21. ; Define periodic message #07  
7A 18 07 06 D4 48 6B 10 41 0D 67
22. ; Enable periodic message #07  
74 1A 06 07 01
23. ; Set Group1 Type2 interval for a count of \$19 (decimal 25)  
; the actual interval = 2.4575 sec.          must use message \$01 timer  
74 1B 06 01 19
24. ; Enable Group1 for Type2 operations  
; At this point all enabled messages in Group1, will begin transmitting in sequence, one  
message every 2.4575 seconds.  
74 0C 06 01 02

#### 9.4.5 Periodic Message Commands

All commands are listed in the Commands and Responses section at the end of this document.  
A brief summary of them is provided here.

- 5x 63      Timer interval
- 7x 0C      Periodic message group operation control (disabled, Type1, Type2)
- 7x 18      Define a periodic message
- 7x 1A      Periodic message disable/enable
- 7x 1B      Periodic message interval
- 7x 1C      Disable all periodic messages  
                Disable all groups

## 10. VPW Mode

Enter VPW mode with the \$E1 33 command.

The report \$91 07 indicates the AVT-84x has entered VPW operations.

The AVT-84x supports J1850 VPW operations in both 1X and 4X speed modes. It also supports GM block transfers of up to 4112 bytes.

When VPW mode is first entered, the following defaults are set:

- VPW operations are enabled.
- 1X mode is enabled.
- Receive network messages are enabled.
- Match bytes are disabled.
- Transmit acks are enabled and consist of the bytes “01 60.”

### 10.1 Jumper JP3

Jumper JP3 on the AVT-84x board connects / disconnects the VPW network from pin #2 of P3; the DA-15P “Vehicle” connector. Usually, the factory default position of JP3 is installed.

AVT-841 revision “A” boards do not have jumper JP3. The VPW network is always connected to pin #2 of P3; the DA-15P “Vehicle” connector.

### 10.2 Communications

J1850 VPW messages consist of a maximum of 11 bytes.

The AVT-84x handles the CRC byte. The host computer should never send it and the AVT-84x will never report it.

J1850 VPW communications on GM vehicles generally follow this format:

- byte #1: Priority / Type byte.
- byte #2: Destination address (can be function or physical).
- byte #3: Source address (always physical).
- byte #4: Extended address byte.
- byte #5 on: The data field.

Messages to and from the network are of the form: \$0x yy rr ss tt vv ... where “x” is the count of bytes to follow. Refer to Section 4.4 and the Commands Section 15 for detailed information about messages to and from the network.

#### 10.2.1 Communications Example - Not Block Transfer

This example is to enter VPW mode, send a message to the network, receive a message from the network.

```
; enter VPW mode
E1 33

; send a message to the network (OBD-II RPM request)
05 68 6A F1 01 0C

; explanation:
; 0 indicates “to” the network
; 5 is the count of bytes to follow
; 68 is the priority / type byte
; 6A is the destination address (functional, in this case)
; F1 is the source address (an OBD-II tool, the AVT-84x, in this case)
; 01 is the mode
; 0C is the PID, which is a request for engine RPM
```

```

; receive the transmit ack = 01 60
;   0 indicates "from" the network
;   1 count of bytes to follow
;       all messages from the network have a receive status byte
;       immediately after the header byte
;   60 is the receive status byte
;   refer to Section 15.1 for the bit map of the receive status byte

; receive a message from the network = 08 00 48 6B 10 41 0C xx yy
; explanation:
;   0 indicates from the network
;   8 is the count of bytes to follow
;   00 is the receive status byte and indicates no errors
;   48 is the priority / type byte
;   6B is the destination address (functional, in this case)
;   10 is the source address (engine ECU)
;   41 is a response to a mode 1 request
;   0C is the PID
;   xx yy is the engine RPM

```

Note that sending or receiving blocks of data are handled using the alternate header formats. Refer to Section 4.4 and Section 15 for detailed information.

### 10.2.2 Time Stamp

Time stamps for both the transmit ack and received messages can be disabled or enabled using the 5x 08 command.

Transmit ack: the time stamp is a two byte value immediately after the packet header byte(s); but before the status byte.

Receive message: the time stamp is a two-byte value immediately after the packet header byte(s); but before the status byte.

The 52 08 00 command disables all time stamps.

The 52 08 01 command enables time stamps where the time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at \$FFFF.

#### 10.2.2.1 Receive Message Examples

When time stamps are disabled a receive message example is:

```

06 00 11 22 33 44
    06 header byte, indicates from the network, 6 bytes follow.
    00 status byte indicating no errors detected.
    11 22 33 44 message bytes.

```

When time stamps are enabled a receive message example is:

```

08 xx yy 00 11 22 33 44
    08 header byte, indicates from the network, 8 bytes follow.
    xx yy time stamp (xx is the high byte, yy is the low byte).

```

00 status byte indicating no errors detected.  
11 22 33 44 message bytes.

### **10.2.2.2 Transmit Ack Examples**

When time stamps are disabled a transmit ack example is:

01 60  
01 header byte, indicates from the network, 1 byte follows.  
60 status byte, bits 4 and 5 set, indicates “from this device” and “transmit success”

When time stamps are enabled a transmit ack example is:

03 xx yy 60  
03 header byte, indicates from the network, 3 bytes follow.  
xx yy time stamp (xx is the high byte, yy is the low byte).  
60 status byte, bits 4 and 5 set, indicates “from this device” and “transmit success”

## **10.3 Message Filtering**

Messages received from the network, that are NOT blocks, have the form:

0x ww pp dd ss mm nn rr ...

- “0” indicates ‘from the network’.
- “x” is the count of bytes to follow.
- “ww” is the receive status byte (usually equal to \$00).
- “pt” is the first byte of the actual message and is known as the Priority/Type byte.
- “dd” is the destination address (either functional or physical).
- “ss” is the source address (always physical).
- “mm nn rr ...” are the remaining bytes of the message.

Messages received from the network that are blocks, have the form:

11 yy ww pp dd ss mm nn rr ...

or

12 xx yy ww pp dd ss mm nn rr ...

- “11” indicates ‘from the network’.
- “yy” is the count of bytes to follow.
- or
- “12” indicates ‘from the network’.
- “xx yy” is the count of bytes to follow.
- “ww” is the receive status byte (usually equal to \$00).
- “pt” is the first byte of the actual message and is known as the Priority/Type byte.
- “dd” is the destination byte. (It is either a functional or physical address.)
- “ss” is the source. (It is always a physical address.)
- “mm nn rr ...” are the remaining bytes of the message.

The AVT-84x firmware permits the host computer to specify one or two filter or match bytes, one for the destination byte and one for the source byte.



Command 5x 5B is used to set the destination match byte.

Command 5x 5C is used to set the source match byte.

Both bytes default to value \$00, which means don't care or don't check.

Any other value (\$01 to \$FF) enables the filter or match feature for that byte.

If both bytes are defined, the filtering function is a logical "AND" operation.

The filter function applies to regular network messages and block messages.

### 10.3.1 Example Network Message

Assume the following message appears on the network:

E5 F2 10 3B 4C 5D 6E 7F A1 B2

"E5" is the priority/type byte.

"F2" is the destination byte.

"10" is the source byte.

### 10.3.2 Example #1

Both filter bytes are set to \$00 (default).

The host will receive this packet from the AVT-84x.

0B 00 E5 F2 10 3B 4C 5D 6E 7F A1 B2

### 10.3.3 Example #2

Host sends: 52 5B F1

AVT-84x response: 62 5B F1

The destination filter byte is set to \$F1.

Only messages with destination byte equal to \$F1 will be passed to the host.

If the message above is received from the network, the AVT-84x will determine that the destination bytes do not match and will throw out the message. The host will not receive anything.

### 10.3.4 Example #3

Host sends: 52 5B 00

AVT-84x response: 62 5B 00

Host sends: 52 5C 20

AVT-84x response: 62 5C 20

The destination filter byte is cleared (don't care).

The source filter byte is set to \$20.

Only messages with source byte equal to \$20 will be passed to the host.

If the message above is received from the network, the AVT-84x will determine that the source bytes do not match and will throw out the message. The host will not receive anything.

### 10.3.5 Example #4

Host sends: 52 5B F2

AVT-84x response: 62 5B F2

Host sends: 52 5C 10

AVT-84x response: 62 5C 10

The destination filter byte is set to \$F2.

The source filter byte is set to \$10.

Only messages with the destination byte equal \$F2 “AND” source byte equal \$10 will be passed to the host.

The host will receive this packet from the AVT-84x.

0B 00 E5 F2 10 3B 4C 5D 6E 7F A1 B2

## 10.4 Mask / Match / Respond Function

The AVT-84x, in VPW mode, Mask/Match/Respond (MMR) function allows the user to define a message mask and match byte sequence. If the match is successful, the Response is queued for immediate execution.

An operational overview, a summary of the commands, and an example follow.

### 10.4.1 Operational Overview

This function operates on all network messages and includes both ‘received’ and ‘transmitted’ messages (relative to the AVT-84x interface and the VPW network). The message filtering function, described just above, does NOT affect the operation of this function.

A VPW network message passes through the AVT-84x. If the length of the message is equal to or longer than the defined Mask/Match sequence length, the message is checked. Starting with the first byte of the message, each message byte is logically “ANDed” with the corresponding mask byte. The result of that operation is then compared to the corresponding match byte. If there is a match, the test continues until all defined mask and match bytes are processed. If there is a “no match” the test terminates immediately.

If the test is successful, the respond command is queued for immediate processing. The “Respond” is any command that can be issued by the user/host computer.

### 10.4.2 Command Summary

Command: 5x 75  
Define the mask. Default is \$FF.

Command: 5x 76  
Define the match byte sequence.

Command: 5x 77  
Define the response.

Command: 5x 78  
Disable/Enable the MMR function.

### 10.4.3 Example

Assume you want the AVT-84x unit to switch to 4X mode as soon as the message “\$AA \$BB \$CC \$DD \$EE” passes by. You want an exact message match, no message ambiguity allowed. The command to switch to 4X mode is: “\$C1 \$01”.

To set-up and enable the function, you would issue the following commands to the AVT-84x.

```
; define the mask
56 75 FF FF FF FF FF

; define the match
56 76 AA BB CC DD EE

; define the response
53 77 C1 01

; enable the function
52 78 01
```

At this point, if any message passes by that has the byte sequence “AA BB CC DD EE” (or longer), and it must be an exact match, the command “C1 01” will be issued that will cause the AVT-84x to switch to 4X mode. The AVT-84x will also issue the command response “C1 01” to the host computer telling the host computer that it successfully completed execution of the “C1 01” command.

### **10.5 Periodic Message Support**

In VPW mode, the AVT-84x has the ability to transmit as many as ten (\$0A) messages automatically. The operator defines and sets up the desired periodic messages, enables them, and the AVT-84x unit will then transmit those messages, at the defined interval, without any operator intervention.

A common use for this capability is for the “Tester Present” message that some ECUs (Electronic Control Units) require when in diagnostic mode. Another use would be in a simulation scenario.

The AVT-84x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01) is enabled.

#### **10.5.1 Organization of Periodic Messages**

In VPW mode there is only one group of periodic messages: Group1.

All ten periodic messages are in Group1.

All ten periodic messages can be set to operate in Type1 or Type2 mode.

The periodic messages are numbered \$01 to \$0A (inclusive).

Each message is independently disabled or enabled (7x 1B command).

Each message has its own time interval (7x 1A command); valid only in Type1 operations.

#### **10.5.2 Periodic Message Master Timer**

There is one timer that governs:

- The Analog To Digital (ATD) functions.
- Type1 periodic messages.
- Type2 periodic messages.

The time interval for that timer is set with the 52 63 xx command.

The available settings are:

- 98.30 msec [Default]
- 49.15 msec

20.48 msec  
10.24 msec  
5.12 msec

### 10.5.3 Type1 Periodic Messages

Type1 periodic messages operate independently of each other.

When Type1 operations are enabled, each enabled message in that group operates according to its own interval count.

The message is set up.  
The interval count is defined.  
The message is enabled.  
The group is enabled for Type1 operations.

#### 10.5.3.1 Type1 Example

Want to send two messages automatically and independently. One message about every 500 msec. The other message about every 1 second. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

1. ; Enter VPW mode  
E1 33
2. ; Set the master timer to 98.30 msec  
52 63 01
3. ; Define periodic message #01. (The message is: 68 6A F1 3F.)  
76 18 01 68 6A F1 3F
4. ; Set periodic message #01 for an interval count of 10, actual interval = 0.983 msec  
73 1B 01 0A
5. ; Enable periodic message #01  
73 1A 01 01
6. ; Note that nothing will be transmitted until the group control is set to Type1
7. ; Define periodic message #06. (The message is: 68 6A F1 01 0C.)  
77 18 06 68 6A F1 01 0C
8. ; Set periodic message #06 for an interval count of 5, actual interval = 0.4915 msec  
73 1B 06 05
9. ; Enable periodic message #06  
73 1A 06 01
10. ; Enable Group1 for Type1 operations  
; At this point all enabled messages in Group1, will begin transmission according to their own independent schedule  
73 0C 01 01

### 10.5.4 Type2 Periodic Messages

Type2 periodic messages are transmitted in sequence, within the group.

When more than one message in a group is defined and enabled, and the group operating mode is set for Type2 operations (7x 0C command) then those messages will be transmitted, in sequence, using the interval count of the first message in the group (regardless if that first message is used or not).

For Group1 messages, only message \$01 interval count is used.

The sequential messages are setup.

All are in the same group.

The interval count is defined. Only use the interval count of the first message in the group.

The messages are enabled.

The group is enabled for Type2 operations.

#### **10.5.4.1 Type2 Example**

Want to send three messages in sequence one message every 2.5 seconds. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

1. ; Enter VPW mode  
E1 33
2. ; Set the master timer to 98.30 msec  
52 63 01
3. ; Define periodic message #03  
78 18 03 48 6B 10 41 0D 23
4. ; Enable periodic message #03  
73 1A 03 01
5. ; Note that nothing will be transmitted until the group control is set to Type2
6. ; Define periodic message #05  
78 18 05 48 6B 10 41 0D 45
7. ; Enable periodic message #05  
73 1A 05 01
8. ; Define periodic message #07  
78 18 07 48 6B 10 41 0D 67
9. ; Enable periodic message #07  
73 1A 07 01
10. ; Set Group1 Type2 interval for a count of \$19 (decimal 25)  
; the actual interval = 2.4575 sec.            must use message \$01 timer  
73 1B 01 19
11. ; Enable Group1 for Type2 operations  
; At this point all enabled messages in Group1, will begin transmitting in sequence, one  
message every 2.4575 seconds.  
73 0C 01 02

#### **10.5.5 Periodic Message Commands**

All commands are listed in the Commands and Responses section at the end of this document.  
A brief summary of them is provided here.

- 5x 63      Timer interval
- 7x 0C      Periodic message group operation control (disabled, Type1, Type2)
- 7x 18      Define a periodic message
- 7x 1A      Periodic message disable/enable
- 7x 1B      Periodic message interval
- 7x 1C      Disable all periodic messages  
                Disable all groups

### **10.6 Block Transmit Example**

The AVT-84x supports transmitting VPW messages in block mode. The maximum message length supported is 4112 bytes = 4096 + 16 bytes. The hex equivalent is: \$1010.

The following is an example of transmitting a maximum length message in block mode. Note that 4112 is the actual number of message bytes.

Command sent to AVT-84x:  
12 10 10

Followed immediately by the 4112 message bytes.

When transmission of the block to the network is complete, the AVT-85x will send a transmit acknowledgement (if enabled) to the host. The acknowledgement is of the form: "F3 pp rr ss". A description of that acknowledgement is listed at the end of the VPW Responses, Section 17.1.

The AVT-84x waits 3 seconds for the entire message to be received from the host. The message is buffered before being sent to the network.

If the host sends a message larger than 4112 bytes, an error message is generated and the message is flushed.

If all expected bytes are not received within 3 seconds, an error message is generated and the message is flushed.

### **10.7 Block Receive Example**

The following is an example of receiving a block of data 2048 bytes long. Note that the maximum number of bytes that can be received during a block transfer is 4112 bytes and is a count of the actual number of message bytes.

Response from AVT-84x:  
12 08 01

The very next byte is the receive status byte.

The 2048 message bytes follow immediately.

The AVT-84x buffers a message from the network before sending it to the host.

The maximum size message that the AVT-84x can receive is 4112 bytes. If a larger message is received, the AVT-84x will generate an error message to inform the host of the actual size of the message received. The AVT-84x will then send the host the first 4112 bytes received during the block transfer. Any and all bytes over the 4112 byte limit are lost. The receive status byte, most significant bit (b7) is set to indicate that the received network message was too long.

The following is an example of receiving a network message that was too long.

Response received from AVT-84x:

23 53 xx yy

xx yy = actual count of block

Response received from AVT-84x:

12 10 01

The very next byte is the receive status byte (msb set).

The first 4112 message bytes follow immediately.

## 11. KWP Mode

KWP (Key Word Protocol) is available when operating in CAN mode or as KWP mode only.

To enter KWP only mode, use the \$E1 DD command.

The report \$91 0F indicates the AVT-84x has entered KWP operations.

This mode is known and referred to as “KWP” mode in this document and for the AVT-84x interface units. (KWP is from Key Word Protocol 2000 - the ISO 14230 standard.)

To enable KWP mode while in CAN mode use the 52 69 02 command to enable KWP mode, disable LIN mode, and still allow CAN0 and CAN4 operations.

[If operating in CAN mode, KWP operations are designated channel 6.]

KWP mode is communications compliant with the following standards:

ISO 9141

ISO 9141-2

ISO 14230.

The AVT-84x only supports the K-line. It does not support the L-line.

The AVT-84x uses the Vishay-Siliconix Si9241AEY K-line transceiver.

A single 1 K ohm resistor is used to passively pull-up the K-line to V-Batt potential.

[An alternate configuration for the pull-up resistors is available. A second 1 K ohm resistor can be installed in parallel for an equivalent pull-up resistance of 500 ohms.]

### 11.1 Jumper JP2

Jumper JP2 on the AVT-84x board connects / disconnects the K-line from pin #7 of P3; the DA-15P “Vehicle” connector.

### 11.2 Communications

K-line messages, according to ISO 14230 have a maximum length of 259 bytes (including checksum byte).

The AVT-84x is capable of receiving K-line network messages up to the full length of 259 bytes. The AVT-84x will check the message against the received checksum, discard the checksum byte and then forward the received message to the host computer. The user can disable or enable sending the received checksum to the host.

The AVT-84x is capable of transmitting K-line network messages up to the full length of 259 bytes. The AVT-84x will compute and append the checksum byte, unless the user disables that function.

Messages to and from the network are of the form: \$0x yy rr ss tt vv ... where “x” is the count of bytes to follow. Refer to Sections 4.4 and 16 for detailed information about the format of messages to and from the network.

### 11.2.1 Communications Example

This example is to enter KWP mode, send a message to the network, receive a message from the network.

```
; enter KWP mode
E1 DD

; send a message to the network (OBD-II RPM request)
05 68 6A F1 01 0C

; explanation:
;   0 indicates “to” the network
;   5 is the count of bytes to follow
;   68 is the priority / type byte
;   6A is the destination address (functional, in this case)
;   F1 is the source address (an OBD-II tool, the AVT-84x, in this case)
;   01 is the mode
;   0C is the PID, which is a request for engine RPM

; receive the transmit ack = 01 60
;   0 indicates “from” the network
;   1 count of bytes to follow
;   all messages from the network have a receive status byte
;   immediately after the header byte
;   60 is the receive status byte
;   refer to Section 16.1 for the bit map of the receive status byte

; receive a message from the network = 08 00 48 6B 10 41 0C xx yy

; explanation:
;   0 indicates from the network
;   8 is the count of bytes to follow
;   00 is the receive status byte and indicates no errors
;   48 is the priority / type byte
;   6B is the destination address (functional, in this case)
;   10 is the source address (engine ECU)
;   41 is a response to a mode 1 request
;   0C is the PID
;   xx yy is the engine RPM
```

Sending or receiving messages of more than 15 bytes are handled using the alternate header formats. Refer to Section 4.4 and the Commands Section 16 for detailed information.

### 11.2.2 Time Stamp

Time stamps for both the transmit ack and received messages can be disabled or enabled using the 5x 08 command.



Transmit ack: the time stamp is a two byte value immediately after the packet header byte(s); but before the status byte.

Receive message: the time stamp is a two-byte value immediately after the packet header byte(s); but before the status byte.

The 52 08 00 command disables all time stamps.

The 52 08 01 command enables time stamps where the time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at \$FFFF.

#### ***11.2.2.1 Receive Message Examples***

When time stamps are disabled a receive message example is:

06 00 11 22 33 44  
06 header byte, indicates from the network, 6 bytes follow.  
00 status byte indicating no errors detected.  
11 22 33 44 message bytes.

When time stamps are enabled a receive message example is:

08 xx yy 00 11 22 33 44  
08 header byte, indicates from the network, 8 bytes follow.  
xx yy time stamp (xx is the high byte, yy is the low byte).  
00 status byte indicating no errors detected.  
11 22 33 44 message bytes.

#### ***11.2.2.2 Transmit Ack Examples***

When time stamps are disabled a transmit ack example is:

01 60  
01 header byte, indicates from the network, 1 byte follows.  
60 status byte, bits 4 and 5 set, indicates “from this device” and “transmit success”

When time stamps are enabled a transmit ack example is:

03 xx yy 60  
03 header byte, indicates from the network, 3 bytes follow.  
xx yy time stamp (xx is the high byte, yy is the low byte).  
60 status byte, bits 4 and 5 set, indicates “from this device” and “transmit success”

### ***11.3 Initialization***

Depending on the particular application, K-line communications with a vehicle and/or module may require initialization. Initialization is essentially a logical function. The AVT-84x unit (also known as the off-board tester) announces itself to the module it wishes to communicate with, requests a communications session, and, if successful, communicates with that module.

The three specifications (ISO 9141, ISO 9141-2, and ISO 14230) and various manufacturer requirements call out no fewer than 3 initialization methods or schemes. AVT-84x firmware version 0.9 supports two methods with some user definable parameters.

The initialization schemes currently supported are:

- CARB mode (5-baud with a fixed communications baud rates of 10.4 kbaud).

- FAST mode (user defines down time, up time, and baud rate).

A brief description of each method and command follows.

### 11.3.1 CARB Mode Initialization

The user can set the following CARB mode parameters prior to an initialization attempt.

- Length of time the K-line must be idle prior to starting an initialization attempt (W5).  
Command is “53 46 xx yy” where “xx yy” is in milliseconds.  
Default is 300 milliseconds (in accordance with ISO 9141-2 and ISO 14230).
- 5-Baud address using the “52 13 xx” command.  
Default is \$33 (in accordance with ISO 9141-2 and ISO 14230).
- Communications baud rate using the “53 03 xx yy” command.  
Default is 10.4 kbaud (in accordance with ISO 9141-2 and ISO 14230).

CARB mode initialization is invoked with the “61 11” command.

There is at least a 2 second delay from invoking CARB mode initialization until the AVT-84x will respond.

If the initialization attempt was successful, the AVT-84x will respond with “71 11”.

The user can query for the keyword, or two key bytes, using the “51 2C” command.

If the initialization attempt fails, the AVT-84x will respond with a “22 54 xx” error code and then the 71 00 initialization attempt failure report.

A complete listing of the “xx” error code is in the KWP response section of this document.

### 11.3.2 FAST Initialization

The user can set the following FAST mode parameters prior to an initialization attempt.

- Length of time the K-line must be idle prior to starting an initialization attempt (W5).  
Command is “53 46 xx yy” where “xx yy” is in milliseconds.  
Default is 300 milliseconds (in accordance with ISO 9141-2 and ISO 14230).
- Communications baud rate using the “53 03 xx yy” command.  
Default is 10.4 kbaud (in accordance with ISO 14230).
- K-line low time is set with the “52 47 xx” command, where “xx” is in milliseconds.  
Default is 25 milliseconds (in accordance with ISO 14230).
- K-line high time is set with the “52 58 xx” command, where “xx” is in milliseconds.  
Default is 25 milliseconds (in accordance with ISO 14230).

FAST mode initialization is invoked with the “6x 13” command where the “Start Communications” message is included in the command. A common FAST initialization command is:

65 13 81 10 F1 81

The Start Communications message is “81 10 F1 81”

The AVT-84x computes and appends the checksum.

If the initialization attempt was successful the AVT-84x will respond with “71 11” the initialization attempt success report. It will also respond with the “01 60” which is the transmit ack for the Start

Communications message and the downstream module will then respond to the Start Communications message.

If something goes wrong during the initialization attempt, the AVT-84x will likely respond with a “22 54 xx” error code and then the 71 00 initialization attempt failure report.

A complete listing of the “xx” error code is in the KWP response section of this document.

If the downstream module fails to respond to the Start Communications message, the user should consider that to be a failure; even though the AVT-84x will not respond with an error code. Note that the AVT-84x unit will respond with the “71 11” success report - this only indicates that the AVT-84x unit was able to generate the Fast Initialization sequence but does not indicate anything about the downstream module.

### **11.4 Periodic Message Support**

In KWP mode, the AVT-84x has the ability to transmit as many as ten (\$0A) messages automatically. The operator defines and sets up the desired periodic messages, enables them, and the AVT-84x unit will then transmit those messages, at the defined interval, without any operator intervention.

A common use for this capability is for the “Tester Present” message that some ECUs (Electronic Control Units) require when in diagnostic mode. Another use would be in a simulation scenario.

The AVT-84x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01) is enabled.

#### **11.4.1 Organization of Periodic Messages**

In KWP mode there is only one group of periodic messages: Group1.

All ten periodic messages are in Group1.

All ten periodic messages can be set to operate in Type1 or Type2 mode.

The periodic messages are numbered \$01 to \$0A (inclusive).

Each message is independently disabled or enabled (7x 1B command).

Each message has its own time interval (7x 1A command); valid only in Type1 operations.

#### **11.4.2 Periodic Message Master Timer**

There is one timer that governs:

- The Analog To Digital (ATD) functions.

- Type1 periodic messages.

- Type2 periodic messages.

The time interval for that timer is set with the 52 63 xx command.

The available settings are:

- 98.30 msec [Default]

- 49.15 msec

- 20.48 msec

- 10.24 msec

- 5.12 msec

### 11.4.3 Type1 Periodic Messages

Type1 periodic messages operate independently of each other.

When Type1 operations are enabled, each enabled message in that group operates according to its own interval count.

- The message is set up.
- The interval count is defined.
- The message is enabled.
- The group is enabled for Type1 operations.

#### 11.4.3.1 Type1 Example

Want to send two messages automatically and independently. One message about every 500 msec. The other message about every 1 second. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

- 23. ; Enter KWP mode  
E1 DD
- 24. ; Set the master timer to 98.30 msec  
52 63 01
- 25. ; Define periodic message #01. (The message is: 68 6A F1 3F.)  
76 18 01 68 6A F1 3F
- 26. ; Set periodic message #01 for an interval count of 10, actual interval = 0.983 msec  
73 1B 01 0A
- 27. ; Enable periodic message #01  
73 1A 01 01
- 28. ; Note that nothing will be transmitted until the group control is set to Type1
- 29. ; Define periodic message #06. (The message is: 68 6A F1 01 0C.)  
77 18 06 68 6A F1 01 0C
- 30. ; Set periodic message #06 for an interval count of 5, actual interval = 0.4915 msec  
73 1B 06 05
- 31. ; Enable periodic message #06  
73 1A 06 01
- 32. ; Enable Group1 for Type1 operations  
; At this point all enabled messages in Group1, will begin transmission according to their own independent schedule  
73 0C 01 01

### 11.4.4 Type2 Periodic Messages

Type2 periodic messages are transmitted in sequence, within the group.

When more than one message in a group is defined and enabled, and the group operating mode is set for Type2 operations (7x 0C command) then those messages will be transmitted, in sequence, using the interval count of the first message in the group (regardless if that first message is used or not).

For Group1 messages, only message \$01 interval count is used.

The sequential messages are setup.

All are in the same group.

The interval count is defined. Only use the interval count of the first message in the group.

The messages are enabled.

The group is enabled for Type2 operations.

#### **11.4.4.1 Type2 Example**

Want to send three messages in sequence one message every 2.5 seconds. Here is a sequence of commands to do this. It is assumed that this is from a reset condition.

25. ; Enter KWP mode

E1 DD

26. ; Set the master timer to 98.30 msec

52 63 01

27. ; Define periodic message #03

78 18 03 48 6B 10 41 0D 23

28. ; Enable periodic message #03

73 1A 03 01

29. ; Note that nothing will be transmitted until the group control is set to Type2

30. ; Define periodic message #05

78 18 05 48 6B 10 41 0D 45

31. ; Enable periodic message #05

73 1A 05 01

32. ; Define periodic message #07

78 18 07 48 6B 10 41 0D 67

33. ; Enable periodic message #07

73 1A 07 01

34. ; Set Group1 Type2 interval for a count of \$19 (decimal 25)

; the actual interval = 2.4575 sec.          must use message \$01 timer

73 1B 01 19

35. ; Enable Group1 for Type2 operations

; At this point all enabled messages in Group1, will begin transmitting in sequence, one message every 2.4575 seconds.

73 0C 01 02

#### **11.4.5 Periodic Message Commands**

All commands are listed in the Commands and Responses section at the end of this document.

A brief summary of them is provided here.

- 5x 63      Timer interval
- 7x 0C      Periodic message group operation control (disabled, Type1, Type2)

- 7x 18 Define a periodic message
- 7x 1A Periodic message disable/enable
- 7x 1B Periodic message interval
- 7x 1C Disable all periodic messages  
Disable all groups

## **12. AVT-84x Field reFLASHing**

All AVT-84x units can be reFLASHed in the field to permit updating the unit operating firmware.

### ***12.1 AVT-84x reFLASHing - AVT Provided Application***

AVT can provide stand alone applications for a host PC to reFLASH an AVT-841, 842, or 843 unit.

Host operating systems supported are: WIN98, WIN-NT, WIN-XP, WIN2000.

The reFLASH applications are called:

- AVT-841\_reFLASH2.exe (obsolete)
- AVT-841\_reFLASH3.exe
- AVT-842\_reFLASH3.exe
- AVT-843\_reFLASH3.exe

They are supplied in a zip file. Simply unzip the file into a directory or folder of your choice.

There is nothing else to install and the registry is not changed.

Double click on the executable (.exe) to launch the application. Running the application should be self explanatory.

The AVT-84x firmware is likely to be released with the following naming convention.

The “XX” in the name will be the version number.

- AVT-841\_vXX\_main\_only.pf3
- AVT-841\_vXX\_complete.pf3

**13. Idle Mode - Commands****B: Firmware version.**

B0: Request firmware version number.

**D: Operational mode.**

D0: Request operational mode report.

**E: Mode switch.**

E1 33: Switch to VPW mode.

E1 99: Switch to CAN mode.

E1 DD: Switch to KWP mode.

**F: Model Query and Reset**

F0: Query for model number.

F1 A5: Restart the AVT-84x (a form of software reset).

**13.1 Idle Mode - Responses****2: Error reports.**22 34 xx: Command time-out.  
xx: header byte of offending command.

-----

22 77 xx: Switch mode error. "xx" = specific error byte.

01: start address equals \$0000.

02: start address equals \$FFFF.

03: start address less than or equal to \$8000.

04: start address equal to or greater than \$BFFF.

05: expected checksum equals \$0000.

06: expected checksum equals \$FFFF.

07: byte count to sum = \$0000.

08: checksums are not equal.

**3: Invalid command.**31 xx: Invalid command.  
xx: header byte of offending command.**13.2 Other Responses****2: Error reports.**



- 21 70: Backdoor is disabled.
- 21 71: Backdoor access attempt failed.
- 21 84: Command buffer mode fault.

## 14. CAN Mode - Commands

*High nibble, bits b7 - b4: Command type.*

0: Packet for message to be transmitted to the network.

### Format "0x"

0p xy tt vv ww zz mm nn ... :

p: count of bytes to follow.

x: b7: IDE.

0: 11-bit ID.

1: 29-bit ID.

b6: RTR.

0: normal frame.

1: RTR true, remote transmit request.

b5: 0

b4: 0

y: Channel number: 0, 4, 5, 6.

tt vv: 11-bit ID, right justified.

tt vv ww zz: 29-bit ID, right justified.

mm nn ...: data.

### Format "11 pp"

11 pp xy tt vv ww zz mm nn ... :

pp: count of bytes to follow.

x: b7: IDE.

0: 11-bit ID.

1: 29-bit ID.

b6: RTR.

0: normal frame.

1: RTR true, remote transmit request.

b5: 0

b4: 0

y: Channel number: 0, 4, 5, 6.

tt vv: 11-bit ID, right justified.

tt vv ww zz: 29-bit ID, right justified.

mm nn ...: data.

### Format "12 pp qq"

12 pp qq xy tt vv ww zz mm nn ... :

pp qq: count of bytes to follow.

x: b7: IDE.

0: 11-bit ID.

1: 29-bit ID.

b6: RTR.

0: normal frame.

1: RTR true, remote transmit request.

---

b5:	0
b4:	0
y:	Channel number: 0, 4, 5, 6.
tt vv:	11-bit ID, right justified.
tt vv ww zz:	29-bit ID, right justified.
mm nn ...:	data.

Data byte count limitations

When ISO 15765 is disabled, maximum is 8 data bytes.

When ISO 15765 is enabled, maximum is 4095 data bytes.

2: Reset.

21 04:	Reset FIFO #1.
21 05:	Reset FIFO #2.
21 0A:	Reset CAN0.
21 0B:	Reset CAN4.

3: \_\_\_\_\_4: \_\_\_\_\_5: Configuration.

51 08:	Request time stamp status.
52 08 00:	Disable time stamps. [Default]
52 08 01:	Enable time stamps. For CAN0 and CAN4 the time stamp interval is the inverse of the baud rate for that channel. For KWP or LIN, the time stamp is 1 millisecond resolution.
52 08 02:	Enable time stamps. For all channels the time stamp is 1 millisecond resolution.

-----

51 40:	Transmit acks query.
52 40 00:	Do not send transmit acks to host.
52 40 01:	Send transmit acks to host. [Default]

-----

52 4C xx:	Command processing delay. Delay is 'xx' timer ticks (5x 63 command). Only useful between commands; does not otherwise affect operations.
-----------	------------------------------------------------------------------------------------------------------------------------------------------------

-----

52 58 01:	Read ADC channel #1 (terminal #1).
52 58 02:	Read ADC channel #2 (terminal #2).

---

---

52 58 03: Read ADC channel #3 (terminal #3).

-----

51 59: Query for status of periodic ADC reports.

52 59 00: Disable periodic ADC reports. [Default]

52 59 xx: Enable periodic ADC reports.

Report interval is 'xx' timer ticks (5x 63 command).

-----

51 63: Master timer status query.

52 63 xx: Master timer setting.

xx: 01 98.30 msec. [Default]

02 49.15 msec.

03 20.48 msec.

04 10.24 msec.

05 5.12 msec.

-----

51 67: Query for host baud rate setting.

52 67 01: Set host baud rate to 19.2 kbaud.

52 67 02: Set host baud rate to 38.4 kbaud.

52 67 03: Set host baud rate to 57.6 kbaud.

52 67 04: Set host baud rate to 115.2 kbaud.

Note: New setting does not take affect until unit is reset;  
either power-on reset or software reset (F1 A5).

-----

51 69: Query for secondary operational mode.

52 69 00: Disable both KWP and LIN secondary modes.

52 69 01: Enable LIN secondary operations. [Default.]

52 69 02: Enable KWP secondary operations.

-----

51 6A: Query for red LED blink rate.

52 6A xx: Set red LED blink rate.

00 = red LED off.

xx = red LED blink rate; interval is 174.8 msec.

FF = red LED on.

-----

51 6E: Microcontroller resonator query.

6: \_\_\_\_\_

7: CAN configuration.

71 0A: Request baud rate settings for all CAN channels.

---

---

72 0A 0x:	Request baud rate setting for CAN channel. x: CAN channel, 0 or 4.
73 0A 0x yy:	Set baud rate for CAN channel. x: CAN channel number: 0 or 4. yy: 00: User specified using 74 0B 0x rr ss command. 01: 1 Mbps. 02: 500 Kbps. [Default for CAN0.] 03: 250 Kbps. 04: 125 Kbps. 0A: 33.333 Kbps. [Default for CAN4]. 0B: 83.333 Kbps.

---

71 0B:	Request Bit Timing Register (BTR) settings for all CAN channels.
72 0B 0x:	Request BTR settings for CAN channel. x: CAN channel number: 0 or 4.
74 0B 0x rr ss:	Set Bit Timing Registers (BTR) for CAN channel. x: CAN channel, 0 or 4. rr: Bit Timing Register 0 setting. ss: Bit Timing Register 1 setting.

---

Note: Values loaded into BTR0 and BTR1 depend on the value of the external resonator that is installed. Use the 51 6E query to determine what value external resonator is installed. If the 4 MHz resonator is installed and 1 Mbaud operation is selected, the 25 MHz internal PLL is selected as the CAN channel clock source. That source has more clock jitter than the external resonator. It is possible that could affect CAN communications.

---

71 0C:	Periodic message group operational control, status query. All channels, all groups, are reported.
72 0C 0y:	Status query. y: Channel number: 0, 4, 5, 6. All groups are reported.
73 0C 0y 0v:	Status query. y: Channel number: 0, 4, 5, 6. v: Group, 1 or 2.
74 0C 0y 0v 0w:	Periodic message group operational control command. y: Channel number: 0, 4, 5, 6. v: Group, 1 or 2. w: Mode. 0: Disabled. 1: Type1 enabled. 2: Type2 enabled.

---

71 0E:	Outbound flow control separation time query; both channels.
--------	-------------------------------------------------------------

---

---

72 0E 0r: Outbound flow control separation time query.  
           r: CAN channel, 0 or 4.

73 0E 0r ss: Set outbound flow control separation time.  
           r: CAN channel, 0 or 4.  
           ss: Separation time. [Default = 0A]  
               (There are rules in ISO 15765 for setting this parameter.)

-----

71 0F: Outbound flow control ID query; both channels.

72 0F 0s: Outbound flow control ID query.  
           s: CAN channel, 0 or 4.

74 0F rs tt uu: Set outbound flow control ID, 11-bit; no AE byte.  
           r: 0 = 11-bit ID.  
           s: CAN channel, 0 or 4.  
           tt uu: 11-bit ID.

75 0F rs tt uu ae: Set outbound flow control ID, 11-bit, with AE byte.  
           r: 0 = 11-bit ID.  
           s: CAN channel, 0 or 4.  
           tt uu: 11-bit ID.  
           ae: AE byte.

76 0F rs tt uu vv ww: Set outbound flow control ID, 29-bit; no AE byte.  
           r: 8 = 29-bit ID.  
           s: CAN channel, 0 or 4.  
           tt uu vv ww: 29-bit ID.

77 0F rs tt uu vv ww ae: Set outbound flow control ID, 11-bit, with AE byte.  
           r: 8 = 29-bit ID.  
           s: CAN channel, 0 or 4.  
           tt uu vv ww: 29-bit ID.  
           ae: AE byte.

-----

71 11: Request operational mode status for all CAN channels.

72 11 0x: Request operational mode status for CAN channel.  
           x: CAN channel, 0 or 4.

73 11 0x 0y: Set operational mode for CAN channel.  
           x: CAN channel, 0 or 4.  
           y: 0: Disabled. [Default for CAN0 and CAN4.]  
               1: Enabled for normal operations.  
               2: Enabled for listen only operations.

-----

71 12: Request Single Wire CAN (SWC) transceiver status. CAN4 only.

72 12 0x: Set SWC transceiver mode. CAN4 only.  
           x: 0: Sleep mode.  
               1: High speed mode.  
               2: Wake up mode.  
               3: Normal mode. [Default.]

---

---

-----  
 71 17: Single Wire CAN (SWC) transceiver status request. CAN0 only.  
 72 17 0y: Set SWC transceiver mode. CAN0 only.  
           y:     0:     Sleep mode.  
               1:     High speed mode.  
               2:     Wake up mode.  
               3:     Normal mode. [Default.]

-----  
 NOTE: The periodic message setup command (7x 18) has the CAN channel and message number fields reversed as compared to all other commands.

-----  
 73 18 vv 0y: Periodic message setup query.  
               vv:    Message number, \$01 to \$10.  
               y:     Channel number: 0, 4, 5, 6.

7x 18 vv xy tt vv ww zz mm nn ...     Periodic message setup command.  
               vv:    Message number, \$01 to \$20.  
               x:     b7:    IDE.  
                       0:     11-bit ID.  
                       1:     29-bit ID.  
                   b6:    RTR.  
                       0:     normal frame.  
                       1:     RTR true, remote transmit request.  
                   b5:    0  
                   b4:    0  
               y:     Channel number: 0, 4, 5, 6.  
               tt vv:   11-bit ID, right justified.  
               tt vv ww zz: 29-bit ID, right justified.  
               mm nn ...: data field.

-----  
 73 1A 0x yy: Periodic message disable/enable status query.  
               x:     Channel number: 0, 4, 5, 6.  
               yy:    Message number, \$01 to \$20.

74 1A 0x yy 0v: Periodic message disable/enable command.  
               x:     Channel number: 0, 4, 5, 6.  
               yy:    Message number, \$01 to \$20.  
               v:     0     disabled.  
                   1     enabled.

-----  
 73 1B 0x yy: Periodic message interval count status query.  
               x:     Channel number: 0, 4, 5, 6.  
               yy:    Message number, \$01 to \$20.

---

74 1B 0x yy vv:	Periodic message interval count command. x: Channel number: 0, 4, 5, 6. yy: Message number, \$01 to \$20. vv: interval count.
-----	
72 1C 0x:	Disable all periodic messages of one CAN channel. x: Channel number: 0, 4, 5, 6.
72 1C EE:	Disable all periodic messages, all CAN channels. (Note: the setup for each periodic message is not affected.)
-----	
71 26:	I5P operations status query, both CAN channels.
72 26 0s:	I5P operations status query. s: CAN channel, 0 or 4.
73 26 0s 0t:	I5P operations. s: CAN channel, 0 or 4. t: 0: disabled. 1: enabled.
-----	
71 27:	Outbound message padding status query, both CAN channels.
72 27 0s:	Outbound message padding status query. s: CAN channel, 0 or 4.
73 27 0s 0t:	Outbound message padding. s: CAN channel, 0 or 4. t: 0: disabled. 1: enabled.
74 27 0s 0t vv:	Outbound message padding. s: CAN channel, 0 or 4. t: 0: disabled. 1: enabled. vv: pad byte.
-----	
71 2A:	Report all acceptance IDs for all CAN channels.
72 2A 0x:	Report all acceptance IDs for CAN channel. x: CAN channel, 0 or 4.
73 2A 0x 0z:	Report specified acceptance ID. x: CAN channel, 0 or 4. z: Acceptance ID number; from 00 on up. Number depends on ID/Mask mode.
7x 2A xy 0z rr ss tt vv:	Set acceptance ID. x: b7: IDE. 0: 11-bit ID. 1: 29-bit ID. b6: RTR.

---



---

0: normal frame.  
 1: RTR true, remote transmit request.  
 b5: 0  
 b4: 0  
 y: CAN channel, 0 or 4.  
 z: Acceptance ID number; from 00 on up.  
 Number depends on ID/Mask mode.  
 rr: Acceptance ID value when ID/Mask mode = 8.  
 rr ss: Acceptance ID value when ID/Mask mode = 4.  
 rr ss: Acceptance ID value when ID/Mask mode = 2  
 and IDE = 0 (11-bit).  
 rr ss tt vv: Acceptance ID value when ID/Mask mode = 2  
 and IDE = 1 (29-bit).

-----

71 2B: Report ID/Mask mode for all CAN channels.  
 72 2B 0x: Report ID/Mask mode for CAN channel.  
           x: CAN channel, 0 or 4.  
 73 2B 0x 0y: Set ID/Mask mode for CAN channel.  
           x: CAN channel, 0 or 4.  
           y: 2: Two 32-bit IDs and masks.  
              4: Four 16-bit IDs and masks.  
              8: Eight 8-bit IDs and masks.

-----

71 2C: Report all masks for all CAN channels.  
 72 2C 0x: Report all masks for CAN channel.  
           x: CAN channel, 0 or 4.  
 73 2C 0x 0z: Report specified mask.  
           x: CAN channel, 0 or 4.  
           z: Mask number; from 00 on up.  
              Number depends on ID/Mask mode.  
 7x 2C xy 0z rr ss tt vv: Set mask.  
           x: b7: IDE bit.  
                   0: must match.  
                   1: don't care.  
              b6: RTR bit.  
                   0: must match.  
                   1: don't care.  
              b5: 0  
              b4: 0  
           y: CAN channel, 0 or 4.  
           z: Mask number; from 00 on up.  
              Number depends on ID/Mask mode.  
           rr: Mask value when ID/Mask mode = 8.  
           rr ss: Mask value when ID/Mask mode = 4.  
           rr ss: Mask value when ID/Mask mode = 2

---

---

and acceptance ID has IDE = 0 (11-bit).

rr ss tt vv: Mask value when ID/Mask mode = 2.  
acceptance ID has IDE = 1 (29-bit).

-----  
73 2D 0x 0y: Acceptance ID register direct read.  
x: CAN channel, 0 or 4.  
y: Acceptance ID register number, 0 to 7.  
74 2D 0x 0y zz: Acceptance ID register direct write.  
x: CAN channel, 0 or 4.  
y: Acceptance ID register number, 0 to 7.  
zz: Register value to write.

-----  
73 2E 0x 0y: Mask register direct read.  
x: CAN channel, 0 or 4.  
y: Mask register number, 0 to 7.  
74 2E 0x 0y zz: Mask register direct write.  
x: CAN channel, 0 or 4.  
y: Mask register number, 0 to 7.  
zz: Register value to write.

-----  
71 30: "AE" byte; disable / enable status query, both CAN channels.  
72 30 0s: "AE" byte; disable / enable status query.  
s: CAN channel, 0 or 4.  
73 30 0s 0t: "AE" byte; disable / enable command.  
s: CAN channel, 0 or 4.  
t: 0: disable.  
1: enable.

-----  
71 32: Query for first frame data length (I5P ops only), both channels.  
72 32 0x: Query for first frame data length (I5P ops only).  
x: CAN channel 0 or 4.  
73 32 0x 0y: Set first frame data field length.  
x: CAN channel 0 or 4.  
y: Data field length, 4 to 8.

-----  
71 33: Query for CAN I5P receive buffer time-out value.  
72 33 xx: Set CAN I5P receive buffer time-out value.  
Time is in 174.8 msec increments.  
Both CAN0 and CAN4 channels use this value.

-----  
71 34: Query for CAN I5P inbound flow control separation time

---

---

	default value.
72 34 xx:	Set CAN I5P inbound flow control separation time value. Time is in milliseconds. [Default = 2 msec.] This value is only used when a received inbound flow control frame separation time has value of 00 or is invalid (by ISO 15765). Both CAN0 and CAN4 channels use this value. Note: With firmware version 4.3 a received separation time of 0 milliseconds is valid. This function has no affect.
-----	
71 36:	ABX separation time query.
74 36 0r xx yy:	ABX separation time command. r = 01 = millisecond count r = 02 = 'loop' count. [Default] (loop time is about 45 microseconds) xx yy = count. [Default = \$00 0A]
-----	
71 37:	ABX transmit ID query.
7x 37 m0 nn rr ss tt:	ABX transmit ID command. m: b7: IDE. 0: 11-bit ID. 1: 29-bit ID. b6: RTR. 0: normal frame. 1: RTR true, remote transmit request. b5: 0 b4: 0 nn rr: 11-bit ID, right justified. (IDE = 0). nn rr ss tt: 29-bit ID, right justified. (IDE = 1).
-----	
76 38 0r ss tt kk ll:	ABX data, read or store. r = 1: store the data into FLASH. r = 2: read data from FLASH. ss tt: start address, must be on 512 byte boundary if storing. start address can be any valid number if reading. kk ll: number of bytes to store, number of bytes to follow (r = 1). number of bytes to read (r = 2).
-----	
71 39:	ABX data count query.
73 39 xx yy:	ABX data count command. xx yy: count of bytes to be transmitted.
-----	
71 3A	ABX control status query, both channels.

---

---

72 3A 0r: ABX control status query for specified CAN channel.  
 73 3A 0r 0s: ABX control command.  
           r: 0 = CAN channel 0.  
           r: 4 = CAN channel 4.  
           s: 0 = disable / terminate operations.  
           s: 1 = enable / start operations.

-----  
 71 3B: CAN channel activity status query, both CAN channels.  
 72 3B 0x: CAN channel activity status query for CAN channel "x".  
 73 3B 0x 0y: CAN channel activity command.  
               x = CAN channel: either "0" or "4"  
               y = 0 = disable. [Default]  
               y = 1 = enable.

-----  
 71 3C: CAN channel activity query, both CAN channels.  
 72 3C 0x: CAN channel activity query for specified CAN channel.  
           x = CAN channel: either "0" or "4"

8: \_\_\_\_\_

9: \_\_\_\_\_

A: \_\_\_\_\_

B: Firmware version.

B0: Request firmware version number.

C: \_\_\_\_\_

D: Operational mode.

D0: Request operational mode report.

E: Mode switch.

E1 33: Switch to VPW mode.

E1 99: Switch to CAN mode.

E1 DD: Switch to KWP mode.

F: Model Query and Reset

F0: Query for model number.

F1 A5: Restart the AVT-84x (a form of software reset).

**14.1 CAN Mode - Responses***High nibble, bits b7 - b4: Command type.*0: Packet for message received from the network.

0p rr ss xy tt vv ww zz mm nn ... :

p: count of bytes to follow.

rr ss: time stamp. [Optional.]

x: b7: IDE.

0: 11-bit ID.

1: 29-bit ID.

b6: RTR.

0: normal frame.

1: RTR true, remote transmit request.

b5: 0

b4: 0

y: Channel number: 0, 4, 5, 6.

tt vv: 11-bit ID, right justified.

tt vv ww zz: 29-bit ID, right justified.

mm nn ...: data field.

02 0y 0z: Transmit ack.

y: Channel number: 0, 4, 5, 6.

z: Transmit buffer number.

04 0y 0z rr ss: Transmit ack.

y: Channel number: 0, 4, 5, 6.

z: Transmit buffer number.

rr ss: Time stamp

2: Error reports.

21 0E: Transmit command too long.

-----  
22 2C xx: Serial comms with host error.

xx:

b7: transmit data register empty.

b6: transmit complete.

b5: receive data register full.

b4: idle.

b3: overrun.

b2: noise flag.

b1: framing error.

b0: parity fault.  
-----

---

22 34 xx:	Command time-out. xx: header byte of offending command. [0.5 seconds.]
-----	
21 35:	Time out reading bytes from 12 xx yy command (less than 12 bytes). [3 seconds.]
-----	
21 5B:	Time out trying to send received block to host. [3 seconds.]
-----	
22 5F xx	CAN ISO 15765 processing error. 00:
	10:
	11: CAN0; DLC too short, < 1.
	12: CAN0; DLC too short, < 2.
	13: CAN0; Unknown frame.
	14: CAN0; DLC too short; < SF_DL.
	15: CAN0; Consecutive frame, receive buffer not in-use.
	16: CAN0; Consecutive frame, sequence number error.
	17: CAN0; First frame, buffer not idle.
	18: CAN0; First frame, DLC too short, < 2.
	19: CAN0; First frame, DLC too short, < 3.
	1A: CAN0; Received flow control frame, not expecting one.
	1B: CAN0; Flow control frame, DLC too short, < 3.
	1C: CAN0; Flow control frame, DLC too short, < 4.
	1D: CAN0; Time out sending buffer to host.
	1E: CAN0; Receive buffer time out.
	1F: CAN0; Transmit buffer time out.
	20:
	21: CAN0; Buffer forward time out.
	22: CAN0; Flow control, transmit, transmitter not available.
	23: CAN0; Flow control, transmit, watchdog time out.
	24: CAN0; Count = 0 in 11/12 transmit command.
	25: CAN0; Time out reading data (converting to 0x command).
	26:
	27: CAN0; Time out reading 11-bit ID bytes.
	28: CAN0; Time out reading 29-bit ID bytes.
	29: CAN0; Time out reading "AE" byte.
	2A: CAN0; Time out reading data.
	2B: CAN0; Command too short. (0x xmt cmd, no AE, 11-bit)
	2C: CAN0; Command too short. (0x xmt cmd, with AE, 11-bit)
	2D: CAN0; Command too short. (0x xmt cmd, no AE, 29-bit)

---

---

2E:	CAN0; Command too short. (0x xmt cmd, with AE, 29-bit)
2F:	CAN0; Transmitter not enabled.
30:	
31:	
32:	
33:	CAN0; Command too long, 11/12, command flushed.
34:	
35:	CAN0; Transmitter not available, command flushed.
36:	CAN0; Ran out of data on first frame, command flushed.
37:	CAN0; Invalid separation time received (\$80 to \$F0).
38:	CAN0; Invalid separation time received (\$FA to \$FF).
39:	CAN0; Flow status = 2; transaction aborted.
3A:	CAN0; Flow status undefined; transaction aborted.
3B:	CAN0; Flow block size not 00.
3C:	CAN0; Time-out waiting for FIFO2 to go empty.
3D:	
3E:	
3F:	
40:	
50:	
51:	CAN4; DLC too short, < 1.
52:	CAN4; DLC too short, < 2.
53:	CAN4; Unknown frame.
54:	CAN4; DLC too short; < SF_DL.
55:	CAN4; Consecutive frame, receive buffer not in-use.
56:	CAN4; Consecutive frame, sequence number error.
57:	CAN4; First frame, buffer not idle.
58:	CAN4; First frame, DLC too short, < 2.
59:	CAN4; First frame, DLC too short, < 3.
5A:	CAN4; Received flow control frame, not expecting one.
5B:	CAN4; Flow control frame, DLC too short, < 3.
5C:	CAN4; Flow control frame, DLC too short, < 4.
5D:	CAN4; Time out sending buffer to host.
5E:	CAN4; Receive buffer time out.
5F:	CAN4; Transmit buffer time out.
60:	
61:	CAN4; Buffer forward time out.
62:	CAN4; Flow control, transmit, transmitter not available.
63:	CAN4; Flow control, transmit, watchdog time out.
64:	CAN4; Count = 0 in 11/12 transmit command.
65:	CAN4; Time out reading data (converting to 0x command).
66:	
67:	CAN4; Time out reading 11-bit ID bytes.

---

---

68: CAN4; Time out reading 29-bit ID bytes.  
 69: CAN4; Time out reading "AE" byte.  
 6A: CAN4; Time out reading data.  
 6B: CAN4; Command too short. (0x xmt cmd, no AE, 11-bit)  
 6C: CAN4; Command too short. (0x xmt cmd, with AE, 11-bit)  
 6D: CAN4; Command too short. (0x xmt cmd, no AE, 29-bit)  
 6E: CAN4; Command too short. (0x xmt cmd, with AE, 29-bit)  
 6F: CAN4; Transmitter not enabled.  
  
 70:  
 71:  
 72:  
 73: CAN4; Command too long, 11/12, command flushed.  
 74: Transmit command dump timeout.  
 75: CAN4; Transmitter not available, command flushed.  
 76: CAN4; Ran out of data on first frame, command flushed.  
 77: CAN4; Invalid separation time received (\$80 to \$F0).  
 78: CAN4; Invalid separation time received (\$FA to \$FF).  
 79: CAN4; Flow status = 2; transaction aborted.  
 7A: CAN4; Flow status undefined; transaction aborted.  
 7B: CAN4; Flow block size not 00.  
 7C: CAN4; Time-out waiting for FIFO2 to go empty.  
 7D:  
 7E:  
 7F:

-----

22 77 xx: Switch mode error. "xx" = specific error byte.

01: start address equals \$0000.  
 02: start address equals \$FFFF.  
 03: start address less than or equal to \$8000.  
 04: start address equal to or greater than \$BFFF.  
 05: expected checksum equals \$0000.  
 06: expected checksum equals \$FFFF.  
 07: byte count to sum = \$0000.  
 08: checksums are not equal.

-----

21 79: No instruction trap.

-----

21 7A: COP fail reset.

-----

21 7B: Clock monitor reset.

-----



---

22 7F xx	CAN processing error.
00:	
01:	CAN0; 0x transmit processing error.
02:	CAN4; 0x transmit processing error.
03:	Invalid CAN channel number.
04:	CAN0; channel not configured to transmit.
05:	
06:	CAN0; non-I5P transmit command too short, 11-bit.
07:	CAN0; non-I5P transmit command too long, 11-bit.
08:	CAN0; non-I5P transmit command too short, 29-bit.
09:	CAN0; non-I5P transmit command too long, 29-bit.
0A:	CAN0; non-I5P 11/12 transmit command too long (gross check).
0B:	CAN0; fail to enter sleep mode.
0C:	CAN0; fail to enter init mode.
0D:	CAN0; fail to exit init mode (enable listen).
0E:	CAN0; fail to exit init mode (enable normal).
0F:	
10:	CAN0; time out reading 11/12 transmit command.
11:	Time out reading 11/12 transmit command flag byte.
12:	CAN0; 11/12 transmit processing error.
13:	CAN4; 11/12 transmit processing error.
14:	Invalid CAN channel number, 11/12 transmit processing.
15:	LIN; 11/12 transmit processing error.
16:	KWP; 11/12 transmit processing error.
17:	
18:	
20:	
30:	
40:	n/a
41:	n/a
42:	n/a
43:	n/a
44:	CAN4; channel not configured to transmit.
45:	
46:	CAN4; non-I5P transmit command too short, 11-bit.
47:	CAN4; non-I5P transmit command too long, 11-bit.
48:	CAN4; non-I5P transmit command too short, 29-bit.
49:	CAN4; non-I5P transmit command too long, 29-bit.
4A:	CAN4; non-I5P 11/12 transmit command too long (gross check).
4B:	CAN4; fail to enter sleep mode.
4C:	CAN4; fail to enter init mode.
4D:	CAN4; fail to exit init mode (enable listen).
4E:	CAN4; fail to exit init mode (enable normal).

---

---

4F:

50: CAN4; time out reading 11/12 transmit command.  
51: CAN0; mask mode not equal mask status, 7x\_2B.  
52: CAN4; mask mode not equal mask status, 7x\_2B.  
53: Mask mode not equal mask status, 7x\_2B.  
54: Invalid id mode in CAN\_rpt\_all\_ids.  
55: Invalid mask mode in CAN\_rpt\_all\_masks.  
56: Invalid CAN channel in 7x\_2C.  
57: Invalid CAN channel in 7x\_2C.  
58: Invalid mask number for mask mode in 7x\_2C.  
59: Invalid mask mode in 7x\_2C.  
5A: CAN0; Invalid mask number in 7x\_2C.  
5B: CAN4; Invalid mask number in 7x\_2C.  
5C: Invalid channel number in 7x\_2C.  
5D: Incorrect header byte in 7x\_2C. (Mode 2, 11-bit.)  
5E: Incorrect header byte in 7x\_2C. (Mode 2, 29-bit.)  
5F: Incorrect header byte in 7x\_2C. (Mode 4.)  
  
60: Incorrect header byte in 7x\_2C. (Mode 8.)  
61: Mask mode error in 7x\_2C.  
62:  
63:  
64:  
65:  
66:  
67:  
68:  
69:  
6A:  
6B: Invalid CAN channel in 7x\_2A.  
6C: Invalid CAN channel in 7x\_2A.  
6D: Invalid mask number for mask mode in 7x\_2A.  
6E: Invalid mask mode in 7x\_2A.  
6F: Invalid mask number in 7x\_2A.  
  
70: Invalid mask number in 7x\_2A.  
71: Invalid channel number in 7x\_2A.  
72: Incorrect header byte in 7x\_2A. (Mode 2, 11-bit.)  
73: Incorrect header byte in 7x\_2A. (Mode 2, 29-bit.)  
74: Incorrect header byte in 7x\_2A. (Mode 4.)  
75: Incorrect header byte in 7x\_2A. (Mode 8.)  
76: Invalid mask mode in 7x\_2A.  
77:

-----  
23 81 xx yy    CAN0 error report.

---

---

xx:

- b7: 0
- b6: 0
- b5: 0
- b4: DLC > 8 in ISO 15765 receive manager
- b3: DLC > 8 in 7x\_18 routine
- b2: DLC > 8 in non-ISO 15765 receive manager
- b1: 0
- b0: CAN error interrupt

yy: Copy of CAN0 rflg register.

- b7: wake up interrupt flag.
- b6: CAN status change interrupt flag.
- b5: receiver status bit 1.
- b4: receiver status bit 0.
- b3: transmitter status bit 1.
- b2: transmitter status bit 0.
- b1: overrun interrupt flag.
- b0: receive buffer full flag.

#### receive status bits

- 00: receive ok; receive error count between 0 and 96
- 01: receive warning; receive error count between 97 and 127
- 10: receive error; receive error count greater than 127
- 11: bus off, transmit error count greater than 255

#### transmit status bits

- 00: transmit ok; transmit error count between 0 and 96
- 01: transmit warning; transmit error count between 97 and 127
- 10: transmit error; transmit error count greater than 127
- 11: bus off, transmit error count greater than 255

-----  
 23 82 xx yy CAN4 error report.

xx:

- b7: 0
- b6: 0
- b5: 0
- b4: DLC > 8 in ISO 15765 receive manager
- b3: DLC > 8 in 7x\_18 routine
- b2: DLC > 8 in non-ISO 15765 receive manager
- b1: 0
- b0: CAN error interrupt

yy: Copy of CAN4 rflg register.

- b7: wake up interrupt flag.
- b6: CAN status change interrupt flag.

---

---

b5: receiver status bit 1.  
 b4: receiver status bit 0.  
 b3: transmitter status bit 1.  
 b2: transmitter status bit 0.  
 b1: overrun interrupt flag.  
 b0: receive buffer full flag.

-----  
 21 84: Command buffer mode fault.

3: Command error.

31 xx: xx = Header byte of message in error.

3:

4:

5: Configuration reports.

62 08 00: Time stamps disabled. [Default.]

62 08 01: Time stamps enabled.

CAN0 and CAN4 the time stamp interval is the inverse of the baud rate for that channel.

LIN the time stamp is 1 millisecond resolution.

62 08 02: Time stamps enabled.

For all channels the time stamp is 1 millisecond resolution.

-----  
 62 40 00: Transmit acks to host are disabled.

62 40 01: Transmit acks to host are enabled. [Default.]

-----  
 62 4C xx: Command processing delay.

Delay is 'xx' timer ticks (5x 63 command).

-----  
 63 58 01 xx: ADC channel #1 reading.

63 58 02 xx: ADC channel #2 reading.

63 58 03 xx: ADC channel #3 reading.

-----  
 64 58 xx yy zz: Periodic ADC report.

xx = ADC channel #1 reading.

yy = ADC channel #2 reading.

zz = ADC channel #3 reading.

---

-----  
 62 59 00: Periodic ADC reports are disabled. [Default]  
 62 59 xx: Periodic ADC reports are enabled.  
 Report interval is 'xx' timer ticks (5x 63 command).

-----  
 62 63 xx: Master timer setting.  
           xx: 01 98.30 msec.  
               02 49.15 msec.  
               03 20.48 msec.  
               04 10.24 msec.  
               05 5.12 msec.

-----  
 62 67 01: Host baud rate is set for 19.2 kbaud.  
 62 67 02: Host baud rate is set for 38.4 kbaud.  
 62 67 03: Host baud rate is set for 57.6 kbaud.  
 62 67 04: Host baud rate is set for 115.2 kbaud.

-----  
 62 69 00: KWP and LIN secondary modes disabled.  
 62 69 01: LIN secondary operations enabled.  
 62 69 02: KWP secondary operations enabled.

-----  
 62 6A xx: Red LED blink rate.  
           00 = red LED off.  
           xx = red LED blink rate; interval is 174.8 msec.  
           FF = red LED on.

-----  
 62 6E 01: 4 MHz resonator installed.  
 62 6E 02: 8 MHz resonator installed.

6:

7: CAN configuration reports.

83 0A 0x yy: Baud rate for CAN channel.  
               x: CAN channel, 0 or 4.  
               yy: 00: User specified using 74 0B 0x rr ss command.  
                   01: 1 Mbps.  
                   02: 500 Kbps. [Default for CAN0.]  
                   03: 250 Kbps.  
                   04: 125 Kbps.  
                   0A: 33.333 Kbps. [Default for CAN4].

---

0B: 83.333 Kbps.

-----  
 84 0B 0x rr ss: Bit Timing Registers (BTR) for CAN channel.  
 x: CAN channel, 0 or 4.  
 rr: Bit Timing Register 0 setting.  
 ss: Bit Timing Register 1 setting.

-----  
 84 0C 0y 0v 0w: Periodic message group operation status.  
 y: CAN channel, 0 or 4.  
 v: Group, 1 or 2.  
 w: Mode.  
 0: Disabled.  
 1: Type1 enabled.  
 2: Type2 enabled.

-----  
 83 0E 0r ss: Outbound flow control separation time.  
 r: CAN channel, 0 or 4.  
 ss: Separation time.

-----  
 84 0F rs tt uu: Outbound flow control ID, 11-bit; no AE byte.  
 r: 0 = 11-bit ID.  
 s: CAN channel, 0 or 4.  
 tt uu: 11-bit ID.

85 0F rs tt uu ae: Outbound flow control ID, 11-bit, with AE byte.  
 r: 0 = 11-bit ID.  
 s: CAN channel, 0 or 4.  
 tt uu: 11-bit ID.  
 ae: AE byte.

86 0F rs tt uu vv ww: Outbound flow control ID, 29-bit; no AE byte.  
 r: 8 = 29-bit ID.  
 s: CAN channel, 0 or 4.  
 tt uu vv ww: 29-bit ID.

87 0F rs tt uu vv ww ae: Outbound flow control ID, 11-bit, with AE byte.  
 r: 8 = 29-bit ID.  
 s: CAN channel, 0 or 4.  
 tt uu vv ww: 29-bit ID.  
 ae: AE byte.

-----  
 83 11 0x 0y: Operational mode for CAN channel.  
 x: CAN channel, 0 or 4.  
 y: 0: Disabled. [Default for CAN0 and CAN4.]  
 1: Enabled for normal operations.

---

---

2: Enabled for listen only operations.

-----  
 82 12 0x: Set SWC transceiver mode. CAN4 only.  
 x: 0: Sleep mode.  
 1: High speed mode.  
 2: Wake up mode.  
 3: Normal mode. [Default.]

-----  
 82 17 0x: SWC transceiver mode. CAN0 only.  
 x: 0: Sleep mode.  
 1: High speed mode.  
 2: Wake up mode.  
 3: Normal mode. [Default.]

-----  
 NOTE: The periodic message setup command (7x 18) have the CAN channel and message number fields reversed as compared to all other commands.

-----  
 8x 18 vv xy tt vv ww zz mm nn ... Periodic message setup.  
 vv: Message number, \$01 to \$20.  
 x: b7: IDE.  
 0: 11-bit ID.  
 1: 29-bit ID.  
 b6: RTR.  
 0: normal frame.  
 1: RTR true, remote transmit request.  
 b5: 0  
 b4: 0  
 y: CAN channel, 0 or 4.  
 tt vv: 11-bit ID, right justified.  
 tt vv ww zz: 29-bit ID, right justified.  
 mm nn ...: data field.

-----  
 84 1A 0x yy 0v: Periodic message disable/enable status.  
 x: CAN channel, 0 or 4.  
 yy: Message number, \$01 to \$20.  
 v: 0 disabled.  
 1 enabled.

-----  
 84 1B 0x yy vv: Periodic message interval count.  
 x: CAN channel, 0 or 4.  
 yy: Message number, \$01 to \$20.

---

---

 vv: interval count.

-----  
 82 1C 0x: All periodic messages of CAN channel 'x' disabled.  
 82 1C EE: All periodic messages, all CAN channels, disabled.

-----  
 83 26 0s 0t: I5P operations.  
           s: CAN channel, 0 or 4.  
           t: 0: disabled.  
               1: enabled.

-----  
 83 27 0s 0t: Outbound message padding.  
           s: CAN channel, 0 or 4.  
           t: 0: disabled.  
               1: enabled.

84 27 0s 0t vv: Outbound message padding.  
           s: CAN channel, 0 or 4.  
           t: 0: disabled.  
               1: enabled.  
           vv: pad byte (only if enabled).

-----  
 8x 2A xy 0z rr ss tt vv: Report acceptance ID.  
           x: b7: IDE.  
                   0: 11-bit ID.  
                   1: 29-bit ID.  
           b6: RTR.  
                   0: normal frame.  
                   1: RTR true, remote transmit request.  
           b5: 0  
           b4: 0  
           y: CAN channel, 0 or 4.  
           z: Acceptance ID number.  
               Number depends on ID/Mask mode.  
           rr: Acceptance ID value when ID/Mask mode = 8.  
           rr ss: Acceptance ID value when ID/Mask mode = 4.  
           rr ss: Acceptance ID value when ID/Mask mode = 2.  
                   and IDE = 0 (11-bit).  
           rr ss tt vv: Acceptance ID value when ID/Mask mode = 2  
                           and IDE = 1 (29-bit).

-----  
 83 2B 0x 0y: Report ID/Mask mode for CAN channel.  
           x: CAN channel, 0 or 4.  
           y: 2: Two 32-bit IDs and masks.

---



- 
- 4: Four 16-bit IDs and masks.  
8: Eight 8-bit IDs and masks.

-----  
8x 2C xy 0z rr ss tt vv:

Report mask.

- x: b7: IDE bit.  
0: must match.  
1: don't care.  
b6: RTR bit.  
0: must match.  
1: don't care.  
b5: 0  
b4: 0  
y: CAN channel, 0 or 4.  
z: Mask number.  
Number depends on ID/Mask mode.  
rr: Mask value when ID/Mask mode = 8.  
rr ss: Mask value when ID/Mask mode = 4.  
rr ss: Mask value when ID/Mask mode = 2  
and acceptance ID has IDE = 0 (11-bit).  
rr ss tt vv: Mask value when ID/Mask mode = 2  
and acceptance ID has IDE = 1 (29-bit).

-----  
84 2D 0x 0y zz:

Report acceptance ID register.

- x: CAN channel, 0 or 4.  
y: Acceptance ID register number, 0 to 7.  
zz: Register value read.

-----  
84 2E 0x 0y zz:

Report mask register.

- x: CAN channel, 0 or 4.  
y: Mask register number, 0 to 7.  
zz: Register value read.

-----  
83 30 0s 0t:

“AE” byte; disable / enable status report.

- s: CAN channel, 0 or 4.  
t: 0: disabled.  
1: enabled.

-----  
83 32 0x 0y:

First frame data field length.

- x: CAN channel 0 or 4.  
y: Data field length, 4 to 8.
-

---

82 33 xx:	CAN I5P receive buffer time-out value. Time is in 174.8 msec increments. Both CAN0 and CAN4 channels use this value.
-----	
82 34 xx:	CAN I5P inbound flow control separation time default value. Time is in milliseconds. This value is only used when a received inbound flow control frame separation time has value of 00 or is invalid (by ISO 15765). Both CAN0 and CAN4 channels use this value.
-----	
84 36 0r xx yy:	ABX separation time. r = 01 = millisecond count r = 02 = 'loop' count. (loop time is about 45 microseconds) xx yy = count.
-----	
8x 37 m0 nn rr ss tt:	ABX transmit ID. m: b7: IDE. 0: 11-bit ID. 1: 29-bit ID. b6: RTR. 0: normal frame. 1: RTR true, remote transmit request. b5: 0 b4: 0 nn rr: 11-bit ID, right justified. (IDE = 0). nn rr ss tt: 29-bit ID, right justified. (IDE = 1).
-----	
86 38 02 ss tt kk ll:	ABX data, read from FLASH. ss tt: start address. kk ll: number of bytes to follow. Specified number of bytes will immediately follow this response.
-----	
83 39 xx yy:	ABX data count. xx yy: count of bytes to be transmitted.
-----	
83 3A 0r 0s:	ABX control status. r: 0 = CAN channel 0. r: 4 = CAN channel 4. s: 0 = operations are disabled. s: 1 = operations are enabled.

---

-----  
83 3B 0x 0y:

CAN channel activity status.

x = CAN channel: either “0” or “4”

y = 0 = disabled.

y = 1 = enabled.

-----  
83 3C 0x yy:

CAN channel activity status.

x = CAN channel: either “0” or “4”

yy = frame count since last query.

8:\_\_\_\_\_

9:     Board status information.

92 04 xx:     Firmware version report. Firmware version is ‘xx’.

91 07:        VPW operations.

91 08:        DLC initialization complete.

91 0C:        FIFO reset.

91 0F:        KWP operations.

91 10:        CAN operations.

91 19:        LIN operations.

91 24:        CAN0 reset.

91 25:        CAN4 reset.

A:\_\_\_\_\_

B:\_\_\_\_\_

C:\_\_\_\_\_

D:\_\_\_\_\_

E:\_\_\_\_\_

F:\_\_\_\_\_

## 15. LIN operations in CAN mode - Commands

*High nibble, bits b7 - b4: Command type.*

### Note:

Only LIN unique commands are listed.  
Some CAN mode commands may be applicable.  
Refer to the CAN mode section for additional commands, responses, and information.

### 0: LIN packet for transmission to the network

0y 05 pp qq rr ss ...  
 x: count of bytes to follow  
 05: channel 5 - LIN channel  
 pp: 00 if slave  
     01 if master  
 qq message ID  
 rr ss ... message data [optional]

### 0: ABIC packet for transmission to the network (long form supported)

0x 15 pp qq rr ss tt ...  
 x: count of bytes to follow

11 yy 15 pp qq rr ss tt ...  
 yy: count of bytes to follow  
 15: channel 15 – ABIC  
 pp: count of bytes expected from ABIC device  
 qq: PID (protected ID)  
 rr: CMD (ABIC module command)  
 ss tt ... message data [optional]

### 1: Alternate header formats, packet for transmission to the network

Alternate form #1 for long messages.

11 yy zz rr ss tt ...  
 yy: number of bytes to follow  
 zz: channel number  
     LIN = 05  
     ABIC = 15  
 rr: control  
     LIN: 00 = slave; 01 = master  
     ABIC = count of expected response bytes from ABIC  
 ss tt: message bytes

Alternate form #2 for long messages.

12 xx yy 06 rr ss tt ...  
 xx yy: number of bytes to follow  
 zz: channel number  
     LIN = 05

ABIC = 15  
 rr: control  
 LIN: 00 = slave; 01 = master  
 ABIC = count of expected response bytes from ABIC  
 ss tt: message bytes

All forms are equal in ascending order.

0x = 11 0x = 12 00 0x  
 11 xx = 12 00 xx  
 12 xx yy

2:

3:

4:

5: Configuration.

51 01: Request received checksum forwarding status.  
 52 01 00: Do not send received checksum to host. [Default]  
 52 01 01: Send received checksum to host.

-----  
 51 02: Receive buffer timeout status query.  
 52 02 xx: Receive buffer timeout set to “xx” milliseconds, based on last received byte.  
 [Default = \$FF = 255 milliseconds, effectively disabled]

-----  
 51 06: Request transmit message echo status.  
 52 06 00: Do not echo transmitted messages. [Default]  
 52 06 01: Echo transmitted messages.

-----  
 51 19: Query for transmit checksum status.  
 52 19 00: Do not append a checksum to a frame transmitted to the K-line.  
 52 19 01: Append a checksum to a frame transmitted to the K-line. [Default]

-----  
 51 24: Network messages query.  
 52 24 00: Do not receive any network messages.  
 52 24 01: Receive network messages. [Default]

-----  
 51 27: Query for P4 time; transmit message inter-byte time.

---

52 27 xx:	Set P4 time to “xx” where “xx” is in increments of approximately 30 microseconds. [Default = \$03]
<hr/>	
51 28:	Query for receive ID byte processing status.
52 28 00:	Disable receive ID byte processing. Use the maximum frame time to determine the end of a received message.
52 28 01:	Enable receive ID byte processing. [Default] Use the received frame ID byte to determine expected message length.
<hr/>	
51 3C:	Receive buffer auto-termination status query.
52 3C 00:	Disable receive buffer auto-termination function.
52 3C 01:	Enable receive buffer auto-termination function. [Default] (Auto-terminate function only applies to the receive function of network messages being transmitted by the AVT-84x. e.g. reception of the echo.)
<hr/>	
51 50:	Query for LIN bus baud rate.
52 50 01:	LIN bus baud rate is 2400 baud
52 50 02:	LIN bus baud rate is 9600 baud. [Default]
52 50 03:	LIN bus baud rate is 19200 baud.
54 50 04 xx yy:	LIN bus baud rate is set by user and equal to: 24 000 000 / (16 * xxyy) [all values shown are decimal] Example: for LIN bus baud rate = 9600; xxyy = \$00 9C (hex) = 156 (decimal)
<hr/>	
51 52:	Query for maximum frame time, in milliseconds.
52 52 xx:	Set maximum frame time to \$xx milliseconds.
<hr/>	
51 5A:	Query for checksum method.
52 5A 00:	Use LIN “classic” checksum method.
52 5A 01:	Use LIN revision 2.0 “enhanced” checksum method.
<hr/>	
51 66:	Query for ID byte only message operation.
52 66 00:	Suppress (discard) ID byte only messages.
52 66 01:	Inform host of an ID byte only message and send the ID byte.
<hr/>	
53 6B xx yy:	Transmit a break to the network with duration = “xx yy” microseconds.

6: \_\_\_\_\_

7: \_\_\_\_\_

Periodic messages are supported for LIN operations.

Refer to the CAN commands section 14 for command information.

8: \_\_\_\_\_9: \_\_\_\_\_A: \_\_\_\_\_B: \_\_\_\_\_C: \_\_\_\_\_D: \_\_\_\_\_E: \_\_\_\_\_F: \_\_\_\_\_**15.1 LIN operations in CAN mode - Responses***High nibble, bits b7 - b4: Response type.***Note:**

Only LIN unique responses are listed.

Some CAN mode commands may be applicable.

Refer to the CAN mode section for additional commands, responses, and information.

0: Valid LIN message packet received from the network

0x 05 pp qq rr ss tt ...

x: count of bytes to follow.

05: channel 5 - LIN channel.

pp: message status byte; bit map, bit set indicates:

b7: frame time out

b6: from this node

b5: receive message too long

b4: buffer closed by break

---

b3: buffer opened without break  
 b2: synch byte error  
 b1: receive message too short or  
     actual length not equal to expected length  
 b0: checksum error  
 qq message ID  
 rr ss tt ... message bytes.

1: Alternate header format for packet received from the network

Alternate form #1 for long messages.

11 xx 06 pp rr ss tt ...

xx: count of bytes to follow.  
 06: the channel number  
 pp: message status byte, as defined above.  
 rr ss tt ... message bytes.

0: Valid ABIC message packet received from the network (long form supported)

0x 15 pp qq rr ss tt ...

x: count of bytes to follow.

or

11 yy 15 pp qq rr ss tt ...

yy: count of bytes to follow.

15: channel 15 – ABIC message.  
 pp: message status byte; bit map, bit set indicates:  
     b7: frame time out  
     b6: from this node  
     b5: receive message too long  
     b4: buffer closed by break  
     b3: buffer opened without break  
     b2: synch byte error  
     b1: receive message too short or  
         actual length not equal to expected length  
     b0: checksum error  
 qq PID (protected ID)  
 rr CMD (ABIC module command)  
 ss tt ... message bytes.

2: Error reports

24 81 rr ss tt

rr:

b7: received byte not equal transmitted  
 b6: received command not equal transmitted  
 b5: received pid not equal transmitted  
 b4: received sync not equal transmitted  
 b3: transmitted state unknown  
 b2: unknown command byte

---



---

b1: 0  
b0: transmitted processing return code error

ss:

b7: 0  
b6: state watchdog timeout  
b5: echo error in tm6  
b4: echo error in tm6  
b3: transmit length error in tm6  
b2: state error in tm6  
b1: state error in tm4  
b0: state error in tm2

tt:

b7: 0  
b6: 0  
b5: 0  
b4: 0  
b3: 0  
b2: 0  
b1: timeout waiting for strobe  
b0: zero length receive message

-----  
25 86 rr ss tt vv

rr:

b7: short to ground detected  
b6: transmit command processing return code error  
b5: error decoding length bits, receive manager #1  
b4: no receive buffer available (break received)  
b3: receive byte not equal transmit byte (master)  
b2: receive buffer mode error  
b1: 0  
b0: no receive buffer available (no break received)

ss:

b7: transmit watchdog time-out  
b6: error in transmit master routine  
b5: transmit command too short  
b4: baud rate index = 0  
b3: transmit command control byte error  
b2: transmit command slave no data  
b1: 0  
b0: received a zero length message

tt:

b7: 0

---

---

b6: 0  
 b5: 0  
 b4: error decoding length bits, receive manager #2  
 b3: master mode is unknown  
 b2: receive byte not equal transmit byte (slave)  
 b1: slave mode error  
 b0: lin\_m\_state time-out

vv:

b7: transmit data register empty.  
 b6: transmit complete.  
 b5: receive data register full.  
 b4: idle.  
 b3: overrun.  
 b2: noise flag.  
 b1: framing error.  
 b0: parity fault.

3:\_\_\_\_\_

4:\_\_\_\_\_

5:\_\_\_\_\_

6: Configuration reports

62 01 00: Send received checksum to host disabled. [Default]  
 62 01 01: Send received checksum to host enabled.

-----

62 02 xx: Receive buffer timeout based on last received byte.  
 Set to “xx” milliseconds

-----

62 06 00: Echo of transmitted messages disabled. [Default.]  
 62 06 01: Echo of transmitted messages enabled.

-----

62 19 00: Checksum is not appended to a frame transmitted to the K-line.  
 62 19 01: Checksum is appended to a frame transmitted to the K-line.

-----

62 24 00: Do not receive any network messages.  
 62 24 01: Receive network messages. [Default.]

---

62 27 xx: P4 time, transmit message inter-byte time.  
 “xx” is in increments of approximately 30 microseconds.  
 [Default = \$03 ≈ 90 microseconds.]

---

62 28 00: Receive ID byte processing disabled.  
 62 28 01: Receive ID byte processing enabled. [Default]

---

62 3C 00: Receive buffer auto-terminate disabled.  
 62 3C 01: Receive buffer auto-terminate enabled.

---

62 50 01: LIN bus baud rate is 2400 baud  
 62 50 02: LIN bus baud rate is 9600 baud. [Default]  
 62 50 03: LIN bus baud rate is 19200 baud.  
 64 50 04 xx yy: LIN bus baud rate is equal to:  $24\,000\,000 / (16 * xxyy)$  [all decimal]

---

62 52 xx: Maximum frame time is \$xx milliseconds.

---

62 5A 00: Use LIN “classic” checksum method.  
 62 5A 01: Use LIN revision 2.0 “enhanced” checksum method.

---

62 66 00: Suppress (discard) ID byte only messages.  
 62 66 01: Inform host of an ID byte only message and send the ID byte.

---

63 6B xx yy: Break of duration “xx yy” microseconds was transmitted to the network.

---

62 6E 04: 4 MHz external resonator is installed.  
 62 6E 08: 8 MHz external resonator is installed.

7: \_\_\_\_\_

8: \_\_\_\_\_

9: \_\_\_\_\_

A: \_\_\_\_\_

B:\_\_\_\_\_

C:\_\_\_\_\_

D:\_\_\_\_\_

E:\_\_\_\_\_

F:\_\_\_\_\_

## 16. KWP operations in CAN mode - Commands

*High nibble, bits b7 - b4: Command type.*

### Note:

Only KWP unique commands are listed.  
Some CAN mode commands may be applicable.  
Some KWP mode commands may be applicable.  
Refer to the CAN mode and/or KWP sections for additional commands, responses, and information.

### 0: Packet for transmission to the network.

0x 06 yy zz ... : x is message length; yy zz ... message bytes.

### 1: Alternate header formats, packet for transmission to the network.

Alternate form #1 for long messages.

11 xx 06 rr ss tt ... xx is the number of bytes to follow.

06 is the channel number

rr ss tt ... are the message bytes.

Alternate form #2 for long messages.

12 xx yy 06 rr ss tt ... xx yy is the number of bytes to follow.

06 is the channel number

rr ss tt ... are the message bytes.

Maximum length is 4112 message bytes (\$1010).

All forms are equal in ascending order. [0x or 11 xx or 12 xx yy].

### 2: Reset.

21 04: Reset FIFO #1.

21 05: Reset FIFO #2.

### 3: \_\_\_\_\_

### 4: \_\_\_\_\_

### 5: Configuration.

51 01: Request received checksum forwarding status.

52 01 00: Do not send received checksum to host. [Default]

52 01 01: Send received checksum to host.

-----  
51 03: Request baud rate divisor value.

53 03 xx yy: K-line bus baud rate is set by user and equal to:

24 000 000 / (16 \* xxyy) [all values shown are decimal]

---

Example: for K-line bus baud rate = 10400;  
xyyy = \$00 90 (hex) = 144 (decimal)

-----  
51 06: Request transmit message echo status.  
52 06 00: Do not echo transmitted messages. [Default]  
52 06 01: Echo transmitted messages.

-----  
51 13: Query for 5-baud address.  
52 13 xx: Set 5-baud address to \$xx. [Default = \$33]

-----  
51 19: Query for transmit checksum status.  
52 19 00: Do not append a checksum to a frame transmitted to the K-line.  
52 19 01: Append a checksum to a frame transmitted to the K-line. [Default]

-----  
51 24: Network messages query.  
52 24 00: Do not receive any network messages.  
52 24 01: Receive network messages. [Default]

-----  
51 27: Query for P4 time; transmit message inter-byte time.  
52 27 xx: Set P4 time to “xx” milliseconds. [Default = \$05]

-----  
51 2A: Query for P3 time; end of receive to start of transmit time.  
52 2A xx: Set P3 time to “xx” milliseconds. [Default = \$37 = 55]

-----  
51 2B: Query for receive buffer expiration time.  
52 2B xx: Set receive buffer expiration time to “xx” milliseconds. [Default = \$17 = 23]

-----  
51 2C: Query for the two key bytes (key word).

-----  
51 46: Query for W5, the bus idle time prior starting an initialization attempt.  
53 46 xx yy: Set time W5 to “xx yy” milliseconds. [Default = 300]

-----  
51 47: Query for FAST initialization low time.  
52 47 xx: Set FAST initialization low time to “xx” milliseconds. [Default = 25]

-----  
51 48: Query for FAST initialization high time.

---

---

52 48 xx: Set FAST initialization high time to “xx” milliseconds. [Default = 25]

-----

51 4B: Query for status of type of transmit checksum.  
 52 4B 00: Transmit checksum is “normal” (sum of bytes). [Default]  
 52 4B 01: Transmit checksum is 2’s complement.

-----

51 57: Query for data bits and parity type.  
 52 57 xx: Set parity type and frame length.  
           All are one start bit and one stop bit.  
           xx: 01 – 8 data bits, no parity (frame length = 10). [Default]  
           xx: 02 – 8 data bits, even parity (frame length = 11).  
           xx: 03 – 8 data bits, odd parity (frame length = 11).  
           xx: 04 – 7 data bits, even parity (frame length = 10).  
           xx: 05 – 7 data bits, odd parity (frame length = 10).

-----

51 6C: Query for ‘fast transmit’ status.  
 52 6C 00: Fast transmit disabled. [Default]  
 52 6C 01: Fast transmit enabled.

6: Initialization

61 11: CARB mode 5-baud initialization.

-----

6x 13 yy zz ...: FAST initialization.  
                   x: count of bytes to follow.  
                   yy zz ... : start communications message.

7: \_\_\_\_\_

Periodic messages are supported for KWP operations.  
 Refer to the CAN commands section 14 for command information.

8: \_\_\_\_\_

9: \_\_\_\_\_

A: \_\_\_\_\_

B: \_\_\_\_\_

C: \_\_\_\_\_

D: \_\_\_\_\_

E: \_\_\_\_\_

F: \_\_\_\_\_

**16.1 KWP operations in CAN Mode - Responses***High nibble, bits b7 - b4: Response type.*Note:

Only KWP unique responses are listed.  
 Some CAN mode responses may be applicable.  
 Some KWP mode responses may be applicable.  
 Refer to the CAN and/or KWP sections for additional commands, responses, and information.

0: Valid message packet received from the network.

0x 06 pp rr ss tt ...

x: count of bytes to follow.  
 06: the channel number  
 pp: message status byte; bit map, bit set indicates:  
     b7:  
     b6: From this device.  
     b5: Transmission successful.  
     b4: Lost arbitration.  
     b3:  
     b2:  
     b1:  
     b0: Checksum error.  
 rr ss tt ... message bytes.

1: Alternate header format for packet received from the network.

Alternate form #1 for long messages (blocks)

11 xx 06 pp rr ss tt ...

xx: count of bytes to follow.  
 06: the channel number  
 pp: message status byte, as defined above.  
 rr ss tt ... message bytes.

Alternate form #2 for long messages (blocks).

12 xx yy 06 pp rr ss tt ...

xx yy: count of bytes to follow.



---

06:	the channel number
pp:	message status byte, as defined above.
rr ss tt ...	message bytes.

## 2: Error reports.

21 34: Time out reading bytes from 11 xx command (less than 12 bytes).  
[3 seconds.]

-----  
22 4A xx: Message of length 1 received.  
xx: the one byte that was received.

-----  
22 54 xx Initialization attempt error codes.

xx:

00:

01: Retry interval not expired.

02: Idle state wait time (W5) failure.

03: Time out while trying to send 5-baud address.

04: Synch byte received with errors.

05: Time out waiting for synch byte.

06: Synch byte not \$55.

07: Key byte 1 received with errors.

08: Time out waiting for key byte 1.

09: Key byte 2 received with errors.

0A: Time out waiting for key byte 2.

0B: Time out waiting for W4.

0C: Inverted key byte 2 echo received with errors.

0D: Time out waiting for inverted key byte 2 echo.

0E: Inverted address byte received with errors.

0F: Time out waiting for inverted address byte.

10: Inverted address byte received in error, not equal to expected.

11: Unknown return code from initialization attempt.

12: 5-baud start bit error.

13: 5-baud, sending "0" bit error.

14: 5-baud, sending "1" bit error.

15: 5-baud sending stop bit error.

16: Inverted key byte 2 echo received in error; not equal to expected.

17: K-line not low during T\_low.

18: Time out waiting for T\_low.

19: K-line not high during T\_high.

1A: Time out waiting 1 msec at beginning of T\_high.

1B: K-line not high during rest of T\_high.

1C: Time out waiting for rest of T\_high.

1D:

1E:

1F:

-----  
24 85 xx yy zz: KWP error.

xx:

b7: 0  
 b6: short to ground detected.  
 b5: periodic message error.  
 b4: 0  
 b3: zero length periodic message found.  
 b2: zero length periodic message found.  
 b1: transmit watchdog time out.  
 b0: no receive buffers available.

yy:

b7: 0  
 b6: 0  
 b5: 0  
 b4: 0  
 b3: 0  
 b2: 0  
 b1: 0  
 b0: 0

zz:

b7: transmit data register empty.  
 b6: transmit complete.  
 b5: receive data register full.  
 b4: idle.  
 b3: overrun.  
 b2: noise.  
 b1: framing error.  
 b0: parity error.

3: \_\_\_\_\_

4: \_\_\_\_\_

5: \_\_\_\_\_

6: Configuration reports.

62 01 00: Send received checksum to host disabled. [Default]  
 62 01 01: Send received checksum to host enabled.

-----  
 63 03 xx yy: Baud rate divisor is “xx yy”.

-----  
 62 06 00: Echo of transmitted messages disabled. [Default.]  
 62 06 01: Echo of transmitted messages enabled.

-----  
 62 13 xx: 5-baud address is “xx”.

-----  
 62 19 00: Checksum is not appended to a frame transmitted to the K-line.  
 62 19 01: Checksum is appended to a frame transmitted to the K-line.

-----  
 62 24 00: Do not receive any network messages.  
 62 24 01: Receive network messages. [Default.]

-----  
 62 27 xx: P4 time, transmit message inter-byte time, is “xx” milliseconds.

-----  
 62 2A xx: P3 time, end of receive to start of transmit time, is “xx” milliseconds.

-----  
 62 2B xx: Receive buffer expiration time is “xx” milliseconds.

-----  
 63 2C xx yy: Key bytes from 5-baud initialization attempt; “xx” and “yy”.

-----  
 63 46 xx yy: Bus idle time (W5) prior to initialization attempt = “xx yy” milliseconds.

-----  
 62 47 xx: FAST initialization low time = “xx” milliseconds.

-----  
 62 48 xx: FAST initialization high time = “xx” milliseconds.

-----  
 62 4B 00: Transmit checksum is “normal.” [Default]  
 62 4B 01: Transmit checksum is 2’s complement.

-----  
 62 57 xx: Parity type and frame length.  
           All are one start bit and one stop bit.  
       xx: 01 – 8 data bits, no parity (frame length = 10). [Default]

---

xx: 02 – 8 data bits, even parity (frame length = 11).  
 xx: 03 – 8 data bits, odd parity (frame length = 11).  
 xx: 04 – 7 data bits, even parity (frame length = 10).  
 xx: 05 – 7 data bits, odd parity (frame length = 10).

-----  
 62 6C 00: Fast transmit disabled.  
 62 6C 01: Fast transmit enabled.

7: Initialization attempt response

71 00: Initialization attempt failure.

-----  
 71 11: Initialization attempt success.

8: \_\_\_\_\_

9: \_\_\_\_\_

A: \_\_\_\_\_

B: \_\_\_\_\_

C: \_\_\_\_\_

D: \_\_\_\_\_

E: \_\_\_\_\_

F: \_\_\_\_\_

## 17. VPW Mode - Commands

*High nibble, bits b7 - b4: Command type.*

### 0: Packet for transmission to the network.

0x yy zz ... : x is message length; yy zz ... message bytes.

### 1: Alternate header formats, packet for transmission to the network.

Alternate form #1 for long messages (blocks).

11 xx rr ss tt ...      xx is the number of bytes to follow.  
                                  rr ss tt ... are the message bytes.

Alternate form #2 for long messages (blocks).

12 xx yy rr ss tt ...      xx yy is the number of bytes to follow.  
                                  rr ss tt ... are the message bytes.

Maximum length is 4112 message bytes (\$1010).

All forms are equal in ascending order. [0x or 11 xx or 12 xx yy].

### 2: Reset.

21 03:      Reset DLC.  
 21 04:      Reset FIFO #1.  
 21 05:      Reset FIFO #2.

### 3:

### 4:

### 5: Configuration.

51 06:      Request transmit message echo status.  
 52 06 00:    Do not echo transmitted messages. [Default]  
 52 06 01:    Echo transmitted messages.

-----  
 51 08:      Request time stamp status.  
 52 08 00:    Disable time stamps. [Default]  
 52 08 01:    Enable time stamps. The time stamp is 1 millisecond resolution.

-----  
 51 24:      Network messages query.  
 52 24 00:    Do not receive any network messages.  
 52 24 01:    Receive network messages. [Default]

-----  
 53 35 xx yy: Direct communication with DLC.

---

xx - Transmit Data or Configuration byte.  
yy - Command byte.

-----  
51 40: Transmit acks query.  
52 40 00: Do not send transmit acks to host.  
52 40 01: Send transmit acks to host. [Default]

-----  
52 4C xx: Command processing delay.  
Delay is 'xx' timer ticks (5x 63 command).  
Only useful between commands; does not otherwise affect operations.

-----  
52 58 01: Read ADC channel #1 (terminal #1).  
52 58 02: Read ADC channel #2 (terminal #2).  
52 58 03: Read ADC channel #3 (terminal #3).

-----  
51 59: Query for status of periodic ADC reports.  
52 59 00: Disable periodic ADC reports. [Default]  
52 59 xx: Enable periodic ADC reports.  
Report interval is 'xx' timer ticks (5x 63 command).

-----  
51 5B: Query for destination filter byte.  
52 5B xx: Set destination filter byte to value "xx".  
[Default = \$00, disabled.]

-----  
51 5C: Query for source filter byte.  
52 5C xx: Set source filter byte to value "xx".  
[Default = \$00, disabled.]

-----  
51 63: Master timer status query.  
52 63 xx: Master timer setting.  
          xx: 01 98.30 msec. [Default]  
              02 49.15 msec.  
              03 20.48 msec.  
              04 10.24 msec.  
              05 5.12 msec.

-----  
51 67: Query for host baud rate setting.  
52 67 01: Set host baud rate to 19.2 kbaud.  
52 67 02: Set host baud rate to 38.4 kbaud.

---

---

52 67 03: Set host baud rate to 57.6 kbaud.  
 52 67 04: Set host baud rate to 115.2 kbaud.  
 Note: New setting does not take affect until unit is reset;  
 either power-on reset or software reset (F1 A5).

-----  
 51 6A: Query for red LED blink rate.  
 52 6A xx: Set red LED blink rate.  
 00 = red LED off.  
 xx = red LED blink rate; interval is 174.8 msec.  
 FF = red LED on.

-----  
 51 6E: Microcontroller resonator query.

-----  
 MMR = Mask/Match/Respond

-----  
 51 75: MMR function mask query.  
 5x 75 yy zz ... MMR function mask definition.  
 x – count of bytes to follow  
 yy zz ... mask bytes

-----  
 51 76: MMR function match query.  
 5x 76 yy zz ... MMR function match definition.  
 x – count of bytes to follow  
 yy zz ... match bytes

-----  
 51 77: MMR function respond query.  
 5x 77 yy zz ... MMR function respond definition.  
 x – count of bytes to follow  
 yy zz ... command bytes

-----  
 51 78: MMR function status query.  
 52 78 00: Disable MMR function.  
 52 78 01: Enable MMR function.

6: \_\_\_\_\_

7: \_\_\_\_\_

71 0C: Periodic message group operational control.  
 Status query; Group1 is reported.

---

---

72 0C 0x:	Status query
	x: 1 = Group1 (only).
73 0C 0x 0y:	Periodic message group operational control command.
	x: 1 = Group1 (only).
	y: Mode.
	0: Disabled.
	1: Type1 enabled.
	2: Type2 enabled.
<hr/>	
72 18 xx:	Periodic message setup query.
	xx: Message number, \$01 to \$0A.
7x 18 xx mm nn pp ...	Periodic message setup command.
	xx: Message number, \$01 to \$0A.
	mm nn pp ...: The message.
<hr/>	
72 1A xx:	Periodic message disable/enable status query.
	xx: Message number, \$01 to \$0A.
73 1A xx 0v:	Periodic message disable/enable command.
	xx: Message number, \$01 to \$0A.
	v: 0 disabled.
	1 enabled.
<hr/>	
72 1B xx:	Periodic message interval count status query.
	xx: Message number, \$01 to \$0A.
73 1B xx yy:	Periodic message interval count command.
	xx: Message number, \$01 to \$0A.
	yy: Interval = 'yy' times the timer (5x 63 command).
<hr/>	
71 1C:	Disable all periodic messages.
	(Note: the setup for each periodic message is not affected.)

8: \_\_\_\_\_

9: \_\_\_\_\_

A: \_\_\_\_\_

B: Firmware version.

B0: Request firmware version number.



C: Network speed.

C0: Request speed mode status.  
C1 00: Select normal (1X) mode.  
C1 01: Select high speed (4X) mode.

D: Operational mode.

D0: Request operational mode report.

E: Mode switch.

E1 33: Switch to VPW mode.  
E1 99: Switch to CAN mode.  
E1 DD: Switch to KWP mode.

F: Model Query and Reset

F0: Query for model number.  
F1 A5: Restart the AVT-84x (a form of software reset).

**17.1 VPW Mode - Responses***High nibble, bits b7 - b4: Response type.*0: Valid message packet received from the network.

0x pp rr ss tt ...

x: count of bytes to follow.  
 pp: message status byte; bit map, bit set indicates:  
     b0: CRC error.  
     b1: Incomplete message.  
     b2: Break received.  
     b3: IFR data.  
     b4: Lost arbitration.  
     b5: Transmission successful.  
     b6: From this device.  
     b7: Bad message, bad status, or receive block too big.  
 rr ss tt ... message bytes.

1: Alternate header format for packet received from the network.

Alternate form #1 for long messages (blocks)

11 xx pp rr ss tt ...

xx: count of bytes to follow.  
 pp: message status byte, as defined above.  
 rr ss tt ... message bytes.

Alternate form #2 for long messages (blocks).

12 xx yy pp rr ss tt ...

xx yy: count of bytes to follow.  
 pp: message status byte, as defined above.  
 rr ss tt ... message bytes.

2: Error reports.

21 0E: Transmit command too long.

-----  
 22 2C xx: Serial comms with host error.

xx:  
     b7: transmit data register empty.  
     b6: transmit complete.  
     b5: receive data register full.  
     b4: idle.  
     b3: overrun.  
     b2: noise flag.  
     b1: framing error.  
     b0: parity fault.

-----

---

21 34:	Time out reading bytes from 11 xx command (less than 12 bytes). [3 seconds.]
-----	
22 34 xx:	Command time-out. xx: header byte of offending command. [0.5 seconds.]
-----	
21 35:	Time out reading bytes from 12 xx yy command (less than 12 bytes). [3 seconds.]
-----	
21 36:	Time out storing a block > 11 bytes. [5 seconds.]
-----	
22 37 xx:	Error during block transmit. xx: DLC status byte.
-----	
22 38 xx:	Time out error during block transmit. xx: DLC status byte. [5 seconds.]
-----	
23 53 xx yy zz:	Received block too big (greater than 4112 bytes). xx yy: Count of bytes received. zz: Receive status byte.
-----	
21 5A:	Received block is too short (less than 3 bytes).
-----	
21 5B:	Time out trying to send received block to host. [3 seconds.]
-----	
22 77 xx:	Switch mode error. "xx" = specific error byte.
01:	start address equals \$0000.
02:	start address equals \$FFFF.
03:	start address less than or equal to \$8000.
04:	start address equal to or greater than \$BFFF.
05:	expected checksum equals \$0000.
06:	expected checksum equals \$FFFF.
07:	byte count to sum = \$0000.
08:	checksums are not equal.

---

-----  
 21 79:        No instruction trap.

-----  
 21 7A:        COP fail reset.

-----  
 21 7B:        Clock monitor reset.

-----  
 23 83 xx yy:  VPW error.

  xx:

    b7:    Short to high detected.  
     b6:    Short to ground detected.  
     b5:    0  
     b4:    0  
     b3:    0  
     b2:    SPTE not set, attempted SPI transmit.  
     b1:    DLC no receive buffer available.  
     b0:    DLC receive FIFO overflow.

  yy:

    b7:    0  
     b6:    0  
     b5:    RFS7 DLC interrupt.  
     b4:    RFS6 DLC interrupt.  
     b3:    RFS5 DLC interrupt.  
     b2:    RFS4 DLC interrupt.  
     b1:    RFS1 DLC interrupt.  
     b0:    error during block transmit.

-----  
 21 84:        Command buffer mode fault.

### 3:    Command error.

  31 xx:        xx = Header byte of message in error.

4:    \_\_\_\_\_

5:    \_\_\_\_\_

### 6:    Configuration reports.

  62 06 00:    Echo of transmitted messages disabled. [Default.]

---

62 06 01: Echo of transmitted messages enabled.

-----  
62 08 00: Time stamps are disabled. [Default]

62 08 01: Time stamps are enabled.

-----  
62 24 00: Do not receive any network messages.

62 24 01: Receive network messages. [Default.]

-----  
63 35 xx yy: Direct communication with DLC.

xx - Returned status byte.

yy - Returned data byte.

-----  
62 40 00: Transmit acks to host are disabled.

62 40 01: Transmit acks to host are enabled. [Default.]

-----  
62 4C xx: Command processing delay.

Delay is 'xx' timer ticks (5x 63 command).

-----  
63 58 01 xx: ADC channel #1 reading.

63 58 02 xx: ADC channel #2 reading.

63 58 03 xx: ADC channel #3 reading.

-----  
64 58 xx yy zz: Periodic ADC report.

xx = ADC channel #1 reading.

yy = ADC channel #2 reading.

zz = ADC channel #3 reading.

-----  
62 59 00: Periodic ADC reports are disabled. [Default]

62 59 xx: Periodic ADC reports are enabled.

Report interval is 'xx' times the timer (5x 63 command).

-----  
62 5B xx: Destination filter byte is set to value "xx". [Default = 00, disabled.]

-----  
62 5C xx: Source filter byte is set to value "xx". [Default = 00, disabled.]

-----  
62 63 xx: Master timer setting.

---

---

xx:	01	98.30 msec.
	02	49.15 msec.
	03	20.48 msec.
	04	10.24 msec.
	05	5.12 msec.

-----  
62 67 01: Host baud rate is set for 19.2 kbaud.  
62 67 02: Host baud rate is set for 38.4 kbaud.  
62 67 03: Host baud rate is set for 57.6 kbaud.  
62 67 04: Host baud rate is set for 115.2 kbaud.

-----  
62 6A xx: Red LED blink rate.  
00 = red LED off.  
xx = red LED blink rate; interval is 174.8 msec.  
FF = red LED on.

-----  
62 6E 01: 4 MHz resonator installed.  
62 6E 02: 8 MHz resonator installed.

-----  
6x 75 yy zz ... MMR function mask definition.  
x – count of bytes to follow  
yy zz ... mask bytes

-----  
6x 76 yy zz ... MMR function match definition.  
x – count of bytes to follow  
yy zz ... match bytes

-----  
6x 77 yy zz ... MMR function response definition.  
x – count of bytes to follow  
yy zz ... command bytes

-----  
62 78 00: MMR function disabled..  
62 78 01: MMR function enabled.

7:

8:

83 0C 0x 0y: Periodic message group operation status.  
x: Group, 1 or 2.

---

y: Mode.  
 0: Disabled.  
 1: Type1 enabled.  
 2: Type2 enabled.

-----  
 8x 18 xx mm nn pp... Periodic message setup.  
 xx: Message number, \$01 to \$0A.  
 mm nn pp ...: The message.

-----  
 83 1A xx 0y: Periodic message disable/enable status.  
 xx: Message number, \$01 to \$0A.  
 y: 0 disabled.  
 1 enabled.

-----  
 83 1B xx yy: Periodic message interval count.  
 xx: Message number, \$01 to \$0A.  
 yy: interval count.

-----  
 81 1C: All periodic messages disabled.

9: Board status information.

92 04 xx: Firmware version report. Firmware version is 'xx'.  
 91 07: VPW operations.  
 91 08: DLC initialization complete.  
 91 0C: FIFO reset.  
 91 0F: KWP operations.  
 91 10: CAN operations.  
 91 19: LIN operations.

A: \_\_\_\_\_

B: \_\_\_\_\_

C: Network speed report.

C1 00: Normal (1X) mode selected. [Default.]  
 C1 01: High speed (4X) mode selected.

D: \_\_\_\_\_

E: \_\_\_\_\_

F:

F3 pp rr ss     Block transmit acknowledgement  
3:                count of bytes to follow.  
pp:               message status byte; bit map, bit set indicates:  
b7:               Bad message, bad status, or receive block too big.  
b6:               From this device.  
b5:               Transmission successful.  
b4:               Lost arbitration.  
b3:               IFR data.  
b2:               Break received.  
b1:               Incomplete message.  
b0:               CRC error.  
rr ss:            count of bytes transmitted (in hex).

Example:

F3 60 01 04  
60                indicates successful transmission from this interface  
                    no errors detected  
0104              count of bytes transmitted (260 decimal)



## 18. KWP Mode - Commands

*High nibble, bits b7 - b4: Command type.*

### 0: Packet for transmission to the network.

0x yy zz ... : x is message length; yy zz ... message bytes.

### 1: Alternate header formats, packet for transmission to the network.

Alternate form #1 for long messages (blocks).

11 xx rr ss tt ...      xx is the number of bytes to follow.  
                                  rr ss tt ... are the message bytes.

Alternate form #2 for long messages (blocks).

12 xx yy rr ss tt ...      xx yy is the number of bytes to follow.  
                                  rr ss tt ... are the message bytes.

Maximum length is 4112 message bytes (\$1010).

All forms are equal in ascending order. [0x or 11 xx or 12 xx yy].

### 2: Reset.

21 04:      Reset FIFO #1.

21 05:      Reset FIFO #2.

### 3: \_\_\_\_\_

### 4: \_\_\_\_\_

### 5: Configuration.

51 01:      Request received checksum forwarding status.

52 01 00:      Do not send received checksum to host. [Default]

52 01 01:      Send received checksum to host.

-----  
 51 03:      Request baud rate divisor value.

53 03 xx yy:      K-line bus baud rate is set by user and equal to:

24 000 000 / (16 \* xxyy)      [all values shown are decimal]

Example: for K-line bus baud rate = 10400;

xxyy = \$00 90 (hex) = 144 (decimal)

-----  
 51 06:      Request transmit message echo status.

52 06 00:      Do not echo transmitted messages. [Default]

52 06 01:      Echo transmitted messages.

---

51 08:	Request time stamp status.
52 08 00:	Disable time stamps. [Default]
52 08 01:	Enable time stamps. The time stamp is 1 millisecond resolution.
-----	
51 13:	Query for 5-baud address.
52 13 xx:	Set 5-baud address to \$xx. [Default = \$33]
-----	
51 19:	Query for transmit checksum status.
52 19 00:	Do not append a checksum to a frame transmitted to the K-line.
52 19 01:	Append a checksum to a frame transmitted to the K-line. [Default]
-----	
51 24:	Network messages query.
52 24 00:	Do not receive any network messages.
52 24 01:	Receive network messages. [Default]
-----	
51 27:	Query for P4 time; transmit message inter-byte time.
52 27 xx:	Set P4 time to “xx” milliseconds. [Default = \$05]
-----	
51 2A:	Query for P3 time; end of receive to start of transmit time.
52 2A xx:	Set P3 time to “xx” milliseconds. [Default = \$37 = 55]
-----	
51 2B:	Query for receive buffer expiration time.
52 2B xx:	Set receive buffer expiration time to “xx” milliseconds. [Default = \$17 = 23]
-----	
51 2C:	Query for the two key bytes (keyword).
-----	
51 40:	Query for status of transmit acks.
52 40 00:	Do not send transmit acks to host.
52 40 01:	Send transmit acks to host. [Default]
-----	
51 46:	Query for W5, the bus idle time prior starting an initialization attempt.
53 46 xx yy:	Set time W5 to “xx yy” milliseconds. [Default = 300]
-----	
51 47:	Query for FAST initialization low time.
52 47 xx:	Set FAST initialization low time to “xx” milliseconds. [Default = 25]
-----	

---

---

51 48: Query for FAST initialization high time.  
 52 48 xx: Set FAST initialization high time to “xx” milliseconds. [Default = 25]

-----

51 4B: Query for status of type of transmit checksum.  
 52 4B 00: Transmit checksum is “normal” (sum of bytes). [Default]  
 52 4B 01: Transmit checksum is 2’s complement.

-----

52 4C xx: Command processing delay.  
 Delay is ‘xx’ timer ticks (5x 63 command).  
 Only useful between commands; does not otherwise affect operations.

-----

51 57: Query for data bits and parity type.  
 52 57 xx: Set parity type and frame length.  
           All are one start bit and one stop bit.  
       xx: 01 – 8 data bits, no parity (frame length = 10). [Default]  
       xx: 02 – 8 data bits, even parity (frame length = 11).  
       xx: 03 – 8 data bits, odd parity (frame length = 11).  
       xx: 04 – 7 data bits, even parity (frame length = 10).  
       xx: 05 – 7 data bits, odd parity (frame length = 10).

-----

52 58 01: Read ADC channel #1 (terminal #1).  
 52 58 02: Read ADC channel #2 (terminal #2).  
 52 58 03: Read ADC channel #3 (terminal #3).

-----

51 59: Query for status of periodic ADC reports.  
 52 59 00: Disable periodic ADC reports. [Default]  
 52 59 xx: Enable periodic ADC reports.  
           Report interval is ‘xx’ timer ticks (5x 63 command).

-----

51 63: Master timer status query.  
 52 63 xx: Master timer setting.  
           xx: 01 98.30 msec. [Default]  
               02 49.15 msec.  
               03 20.48 msec.  
               04 10.24 msec.  
               05 5.12 msec.

-----

51 67: Query for host baud rate setting.  
 52 67 01: Set host baud rate to 19.2 kbaud.  
 52 67 02: Set host baud rate to 38.4 kbaud.

---

---

52 67 03: Set host baud rate to 57.6 kbaud.  
 52 67 04: Set host baud rate to 115.2 kbaud.  
 Note: New setting does not take affect until unit is reset;  
 either power-on reset or software reset (F1 A5).

-----  
 51 6A: Query for red LED blink rate.  
 52 6A xx: Set red LED blink rate.  
 00 = red LED off.  
 xx = red LED blink rate; interval is 174.8 msec.  
 FF = red LED on.

-----  
 51 6C: Query for "Fast Transmit" status  
 52 6C 00: Disable "Fast Transmit". [Default]  
 52 6C 01: Enable "Fast Transmit".

-----  
 51 6E: Microcontroller resonator query.

#### 6: Initialization

61 11: CARB mode 5-baud initialization.

-----  
 6x 13 yy zz ...: FAST initialization.  
 x: count of bytes to follow.  
 yy zz ... : start communications message.

#### 7:

71 0C: Periodic message group operational control.  
 Status query; Group1 is reported.  
 72 0C 0x: Status query  
 x: 1 = Group1 (only).  
 73 0C 0x 0y: Periodic message group operational control command.  
 x: 1 = Group1 (only).  
 y: Mode.  
 0: Disabled.  
 1: Type1 enabled.  
 2: Type2 enabled.

-----  
 72 18 xx: Periodic message setup query.  
 xx: Message number, \$01 to \$0A.  
 7x 18 xx mm nn pp ... Periodic message setup command.  
 xx: Message number, \$01 to \$0A.  
 mm nn pp ...: The message.

---

-----	
72 1A xx:	Periodic message disable/enable status query. xx: Message number, \$01 to \$0A.
73 1A xx 0v:	Periodic message disable/enable command. xx: Message number, \$01 to \$0A. v: 0 disabled. 1 enabled.
-----	
72 1B xx:	Periodic message interval count status query. xx: Message number, \$01 to \$0A.
73 1B xx yy:	Periodic message interval count command. xx: Message number, \$01 to \$0A. yy: Interval = 'yy' times the timer (5x 63 command).
-----	
71 1C:	Disable all periodic messages. (Note: the setup for each periodic message is not affected.)

8: \_\_\_\_\_

9: \_\_\_\_\_

A: \_\_\_\_\_

B: Firmware version.

B0: Request firmware version number.

C: \_\_\_\_\_

D: Operational mode.

D0: Request operational mode report.

E: Mode switch.

E1 33: Switch to VPW mode.

E1 99: Switch to CAN mode.

E1 DD: Switch to KWP mode.

F: Model Query and Reset

F0: Query for model number.

F1 A5: Restart the AVT-84x (a form of software reset).

**18.1 KWP Mode - Responses***High nibble, bits b7 - b4: Response type.*0: Valid message packet received from the network.

0x pp rr ss tt ...

x: count of bytes to follow.  
 pp: message status byte; bit map, bit set indicates:  
     b7:  
     b6: From this device.  
     b5: Transmission successful.  
     b4: Lost arbitration.  
     b3:  
     b2:  
     b1:  
     b0: Checksum error.  
 rr ss tt ... message bytes.

1: Alternate header format for packet received from the network.

Alternate form #1 for long messages (blocks)

11 xx pp rr ss tt ...

xx: count of bytes to follow.  
 pp: message status byte, as defined above.  
 rr ss tt ... message bytes.

Alternate form #2 for long messages (blocks).

12 xx yy pp rr ss tt ...

xx yy: count of bytes to follow.  
 pp: message status byte, as defined above.  
 rr ss tt ... message bytes.

2: Error reports.

21 0E: Transmit command too long.

-----  
22 2C xx: Serial comms with host error.

xx:

b7: transmit data register empty.  
 b6: transmit complete.  
 b5: receive data register full.  
 b4: idle.  
 b3: overrun.  
 b2: noise flag.  
 b1: framing error.  
 b0: parity fault.

-----  
22 34 xx: Command time-out.

---

xx: header byte of offending command.  
[0.5 seconds.]

-----

21 35: Time out reading bytes from 12 xx yy command (less than 12 bytes).  
[3 seconds.]

-----

22 4A xx: Message of length 1 received.  
xx: the one byte that was received.

-----

22 54 xx Initialization attempt error codes.  
xx:

00:  
01: Retry interval not expired.  
02: Idle state wait time (W5) failure.  
03: Time out while trying to send 5-baud address.  
04: Synch byte received with errors.  
05: Time out waiting for synch byte.  
06: Synch byte not \$55.  
07: Key byte 1 received with errors.  
08: Time out waiting for key byte 1.  
09: Key byte 2 received with errors.  
0A: Time out waiting for key byte 2.  
0B: Time out waiting for W4.  
0C: Inverted key byte 2 echo received with errors.  
0D: Time out waiting for inverted key byte 2 echo.  
0E: Inverted address byte received with errors.  
0F: Time out waiting for inverted address byte.

10: Inverted address byte received in error, not equal to expected.  
11: Unknown return code from initialization attempt.  
12: 5-baud start bit error.  
13: 5-baud, sending "0" bit error.  
14: 5-baud, sending "1" bit error.  
15: 5-baud sending stop bit error.  
16: Inverted key byte 2 echo received in error; not equal to expected.  
17: K-line not low during T\_low.  
18: Time out waiting for T\_low.  
19: K-line not high during T\_high.  
1A: Time out waiting 1 msec at beginning of T\_high.  
1B: K-line not high during rest of T\_high.  
1C: Time out waiting for rest of T\_high.  
1D:  
1E:  
1F:

-----  
21 5B:        Time out trying to send received block to host.  
              [3 seconds.]

-----  
22 77 xx:     Switch mode error. "xx" = specific error byte.  
          01:    start address equals \$0000.  
          02:    start address equals \$FFFF.  
          03:    start address less than or equal to \$8000.  
          04:    start address equal to or greater than \$BFFF.  
          05:    expected checksum equals \$0000.  
          06:    expected checksum equals \$FFFF.  
          07:    byte count to sum = \$0000.  
          08:    checksums are not equal.

-----  
21 79:        No instruction trap.

-----  
21 7A:        COP fail reset.

-----  
21 7B:        Clock monitor reset.

-----  
21 84:        Command buffer mode fault.

-----  
24 85 xx yy zz:     KWP error.  
          xx:  
            b7:    0  
            b6:    short to ground detected.  
            b5:    periodic message error.  
            b4:    0  
            b3:    zero length periodic message found.  
            b2:    zero length periodic message found.  
            b1:    transmit watchdog time out.  
            b0:    no receive buffers available.  
  
          yy:  
            b7:    0  
            b6:    0  
            b5:    0  
            b4:    0  
            b3:    0  
            b2:    0



---

b1: 0  
b0: 0

zz: SCI1\_error

b7: transmit data register empty.  
b6: transmit complete.  
b5: receive data register full.  
b4: idle.  
b3: overrun.  
b2: noise.  
b1: framing error.  
b0: parity error.

3: Command error.

31 xx: xx = Header byte of message in error.

4: \_\_\_\_\_

5: \_\_\_\_\_

6: Configuration reports.

62 01 00: Send received checksum to host disabled. [Default]  
62 01 01: Send received checksum to host enabled.

-----  
63 03 xx yy: Baud rate divisor is “xx yy”.

-----  
62 06 00: Echo of transmitted messages disabled. [Default.]  
62 06 01: Echo of transmitted messages enabled.

-----  
62 08 00: Time stamps are disabled. [Default]  
62 08 01: Time stamps are enabled.

-----  
62 13 xx: 5-baud address is “xx”.

-----  
62 19 00: Checksum is not appended to a frame transmitted to the K-line.  
62 19 01: Checksum is appended to a frame transmitted to the K-line.

-----  
62 24 00: Do not receive any network messages.  
62 24 01: Receive network messages. [Default.]

---

-----  
 62 27 xx: P4 time, transmit message inter-byte time, is “xx” milliseconds.

-----  
 62 2A xx: P3 time, end of receive to start of transmit time, is “xx” milliseconds.

-----  
 62 2B xx: Receive buffer expiration time is “xx” milliseconds.

-----  
 63 2C xx yy: Key bytes from 5-baud initialization attempt; “xx” and “yy”.

-----  
 62 40 00: Transmit acks to host are disabled.

62 40 01: Transmit acks to host are enabled. [Default.]

-----  
 63 46 xx yy: Bus idle time (W5) prior to initialization attempt = “xx yy” milliseconds.

-----  
 62 47 xx: FAST initialization low time = “xx” milliseconds.

-----  
 62 48 xx: FAST initialization high time = “xx” milliseconds.

-----  
 62 4B 00: Transmit checksum is “normal.” [Default]

62 4B 01: Transmit checksum is 2’s complement.

-----  
 62 4C xx: Command processing delay.  
 Delay is ‘xx’ timer ticks (5x 63 command).

-----  
 62 57 xx: Parity type and frame length.  
           All are one start bit and one stop bit.  
       xx: 01 – 8 data bits, no parity (frame length = 10). [Default]  
       xx: 02 – 8 data bits, even parity (frame length = 11).  
       xx: 03 – 8 data bits, odd parity (frame length = 11).  
       xx: 04 – 7 data bits, even parity (frame length = 10).  
       xx: 05 – 7 data bits, odd parity (frame length = 10).

-----  
 63 58 01 xx: ADC channel #1 reading.

63 58 02 xx: ADC channel #2 reading.

63 58 03 xx: ADC channel #3 reading.

64 58 xx yy zz:      Periodic ADC report.  
                          xx = ADC channel #1 reading.  
                          yy = ADC channel #2 reading.  
                          zz = ADC channel #3 reading.

-----  
 62 59 00:      Periodic ADC reports are disabled. [Default]  
 62 59 xx:      Periodic ADC reports are enabled.  
                  Report interval is 'xx' times the timer (5x 63 command).

-----  
 62 63 xx:      Master timer setting.  
                  xx:    01      98.30 msec.  
                         02      49.15 msec.  
                         03      20.48 msec.  
                         04      10.24 msec.  
                         05      5.12 msec.

-----  
 62 67 01:      Host baud rate is set for 19.2 kbaud.  
 62 67 02:      Host baud rate is set for 38.4 kbaud.  
 62 67 03:      Host baud rate is set for 57.6 kbaud.  
 62 67 04:      Host baud rate is set for 115.2 kbaud.

-----  
 62 6A xx:      Red LED blink rate.  
                  00 = red LED off.  
                  xx = red LED blink rate; interval is 174.8 msec.  
                  FF = red LED on.

-----  
 62 6C 00:      "Fast Transmit" disabled.  
 62 6C 01:      "Fast Transmit" enabled.

-----  
 62 6E 01:      4 MHz resonator installed.  
 62 6E 02:      8 MHz resonator installed.

7: Initialization attempt response

71 00:      Initialization attempt failure.

-----  
 71 11:      Initialization attempt success.

8:

83 0C 0x 0y:      Periodic message group operation status.

---

x: Group, 1 or 2.  
y: Mode.  
0: Disabled.  
1: Type1 enabled.  
2: Type2 enabled.

-----  
8x 18 xx mm nn pp... Periodic message setup.  
xx: Message number, \$01 to \$0A.  
mm nn pp ...: The message.

-----  
83 1A xx 0y: Periodic message disable/enable status.  
xx: Message number, \$01 to \$0A.  
y: 0 disabled.  
1 enabled.

-----  
83 1B xx yy: Periodic message interval count.  
xx: Message number, \$01 to \$0A.  
yy: interval count.

-----  
81 1C: All periodic messages disabled.

9: Board status information.

92 04 xx: Firmware version report. Firmware version is 'xx'.  
91 07: VPW operations.  
91 08: DLC initialization complete.  
91 0C: FIFO reset.  
91 0F: KWP operations.  
91 10: CAN operations.  
91 19: LIN operations.

A: \_\_\_\_\_

B: \_\_\_\_\_

C: \_\_\_\_\_

D: \_\_\_\_\_

E: \_\_\_\_\_

F:\_\_\_\_\_

## 19. Appendix A

A Telnet listing of the factory default settings for an AVT-843 unit is provided here:

Select option "0" to change the server settings:

- IP Address; default: 192.168.1.70.
- Gateway IP Address; not used.
- Netmask. The factory default is:  
8 host bits [255.255.255.0]
- Telnet configuration password; not used.

Select option "9" to save the new settings, exit, and reboot.

It is not recommended that Channel 1 configuration be changed.

=====

To start a TELNET session

From the START menu, select run, enter this command:

telnet 192.168.1.70 9999

"telnet" is the application name.

"192.168.1.70" is the IP address of the AVT-843 XPort module.

"9999" is the port the "telnet" application will connect to.

As soon as the session starts you have 5 seconds to hit <Enter> or else the connection will time-out and you will have to start again.

After you hit <Enter> The overall configuration of the AVT-843 XPort module will scroll by very quickly. You will then get the main menu, shown here:

Change Setup:

```

0 Server
1 Channel 1
3 E-mail
5 Expert
6 Security
7 Factory defaults
8 Exit without save
9 Save and exit      Your choice ?

```

If you do not make a selection, but instead just hit <Enter> the AVT-843 XPort module will display the full unit configuration. An example listing is below (3 sections down).

=====

At the menu prompt, select "0" and hit <Enter>.

You can walk thru the Server settings.

The recommended and factory settings are shown in parentheses, as shown here.

You should only need to change the IP Address and Netmask settings.

(Hint: do not enter leading zeros for the IP address. Enter it as 192.168.1.70)

IP Address : (192) .(168) .(001) .(070)  
 Set Gateway IP Address (N) N  
 Netmask: Number of Bits for Host Part (0=default) (8)  
 Change telnet config password (N) N

=====

At the menu prompt, select "1" and hit <Enter>.  
 You can walk thru the Channel 1 settings.  
 The recommended and factory settings are shown in parentheses, as shown here.

We strongly recommend that you use these settings and do not change them.

Change Setup:  
 0 Server  
 1 Channel 1  
 3 E-mail  
 5 Expert  
 6 Security  
 7 Factory defaults  
 8 Exit without save  
 9 Save and exit                      Your choice ?

Baudrate (57600) ?  
 I/F Mode (4C) ?  
 Flow (02) ?  
 Port No (10001) ?  
 ConnectMode (C0) ?  
 Remote IP Address : (000) .(000) .(000) .(000)  
 Remote Port (0) ?  
 DisConnMode (00) ?  
 FlushMode (00) ?  
 DisConnTime (00:00) ?:  
 SendChar 1 (00) ?  
 SendChar 2 (00) ?

=====

When done changing the IP Address, and possibly verifying the Server and Channel 1 settings, do not forget to select <9> from the menu to save the settings.

When you select <9> and hit <Enter> the AVT-843 XPort will save the configuration and reboot - that will terminate the telnet session.

Wait at least 20 seconds for the AVT-843 XPort unit to finish saving and rebooting before trying to establish a connection to the AVT-843 unit.

=====

At the menu prompt, do not make a selection, just hit <Enter>.  
 The XPort will display the full configuration, as shown here.

Change Setup:  
 0 Server

---

1 Channel 1  
3 E-mail  
5 Expert  
6 Security  
7 Factory defaults  
8 Exit without save  
9 Save and exit            Your choice ?

\*\*\* basic parameters

Hardware: Ethernet TPI

IP addr 192.168.1.70, no gateway set, netmask 255.255.255.000

\*\*\* Security

SNMP is            enabled

SNMP Community Name: public

Telnet Setup is    enabled

TFTP Download is   enabled

Port 77FEh is      enabled

Web Server is      enabled

ECHO is            disabled

Enhanced Password is disabled

Port 77F0h is      enabled

\*\*\* Channel 1

Baudrate 57600, I/F Mode 4C, Flow 02

Port 10001

Remote IP Addr: --- none ---, Port 00000

Connect Mode : C0

Disconn Mode : 00

Flush Mode : 00

\*\*\* Expert

TCP Keepalive : 45s

ARP cache timeout: 600s

High CPU performance: disabled

Monitor Mode @ bootup : enabled

HTTP Port Number : 80

SMTP Port Number : 25

\*\*\*\*\* E-mail \*\*\*\*\*

Mail server: 0.0.0.0

Unit :

Domain :

Recipient 1:

Recipient 2:

\*\*\* Trigger 1

Serial Sequence: 00,00

CP1: X

CP2: X

CP3: X

Message :

Priority: L

Min. notification interval: 1 s

Re-notification interval : 0 s



---

\*\*\* Trigger 2  
 Serial Sequence: 00,00  
 CP1: X  
 CP2: X  
 CP3: X  
 Message :  
 Priority: L  
 Min. notification interval: 1 s  
 Re-notification interval : 0 s

\*\*\* Trigger 3  
 Serial Sequence: 00,00  
 CP1: X  
 CP2: X  
 CP3: X  
 Message :  
 Priority: L  
 Min. notification interval: 1 s  
 Re-notification interval : 0 s

=====

## 20. Appendix B

A listing of the AVT-843 XPort device, setup web page, factory default settings are provided in this appendix. [Please see the notes at the end of this Section.]

To log into the AVT-843 XPort to view and/or change the configuration, use a web browser and enter the following address: <http://192.168.1.70> (the factory default address)

The first web page to come up is the “Port Properties” page. It is not recommended that changes be made to this page.

Factory default configuration information for the first three “Menu” buttons on the left hand side is provided below. The buttons are “Unit Configuration”, “Server Properties”, and “Port Properties”.

The “Unit Configuration” page displays unit configuration only. No changes can be made from this page.

If it is necessary to change any operational parameters, go to the appropriate page, change the parameters, and then save them by selecting “Update Settings”. The AVT-843 XPort will save the new parameters and then reboot. It takes about 1 minute for the AVT-843 XPort to complete a reboot and return to normal operations, with the new parameters in-use.

### Unit Configuration Page

Select the button “Unit Configuration” to view current configuration of the AVT-843 XPort. No changes are permitted from this page. Factory default settings are shown below.

#### Server Configuration

Product:	XPort Device Server
Model:	Ethernet 1 Channel

---

Firmware version:	V1.50 <i>(may be different / higher)</i>
Hardware Address:	This is the so-called "MAC" address. It is unique to each AVT-843 XPort device.
IP Address:	192.168.1.70
Subnet Mask:	255.255.255.0
Gateway Address:	0.0.0.0

**Port Configuration**

Local Port Number:	10001
Remote Port Number:	[blank]
Serial Port Speed:	57600
Flow Control:	02
Interface Mode:	4C
Connect Mode:	C0
Disconnect Mode:	00
Flush Mode:	00
Pack Control:	00
UDP Datagram Type:	[Not used]

**Server Properties Page**

Select the button "Server Properties" to view and change the AVT-843 XPort properties. Factory default settings are shown here.

**Server Properties**

IP Address:	192.168.1.70	
Subnet Mask:	255.255.255.0	
Gateway Address:	0.0.0.0	
High Performance:	Disable	[this field may not exist]
Telnet Password:	XXXX	[none assigned]

**Port Properties Page**

Select the button "Port Properties" to view and change the AVT-843 XPort communications port properties. Factory default settings are shown here.

[It is NOT recommended that changes be made to these parameters.]

**Serial Port Settings**

Serial Protocol:	RS232
Speed:	57600
Character Size:	8
Parity:	None
Stop bit:	1
Flow Control:	CTS/RTS (Hardware)

**Connect Mode Settings**

---

UDP Datagram Mode:	Disable
UDP Datagram Type:	[blank]
Incoming Connection:	Accept unconditional
Response:	Nothing (quiet)
Startup:	No Active Connection Startup

Dedicated Connection

Remote IP Address:	[blank]
Remote Port:	[blank]
Local Port:	10001

Flush Mode Input Buffer (Line to Network)

On Active Connection:	Disable
On Passive Connection:	Disable
At Time To Disconnect:	Disable

Flush Mode Input Buffer (Network to Line)

On Active Connection:	Disable
On Passive Connection:	Disable
At Time To Disconnect:	Disable

Packing Algorithm

Packing Algorithm:	Disable
Idle Time:	Force Transmit 12 ms
Trailing Characters:	None
Send Immediate After Sendchars:	Disable
Sendchar Define 2-Byte Sequence:	Disable
Send Character 01:	00
Send Character 02:	00

Additional Settings

Disconnect Mode:	Ignore DTR
Check for CTRL-D to Disconnect:	Disable
Port Password:	Disable
Telnet Mode:	Disable
Inactivity Timeout:	Enable
Inactivity Timer:	0:0
Port Password:	[blank]

**Notes:**

I tried several different versions of web browsers to try and access the web page of the AVT-843 XPort. Only one worked. Your results will likely differ. (Both browsers and the embedded web page in the Xport have been updated and changed. The last time I checked, things are working much better using the browser interface of the Xport.)

It is easy to identify when a browser doesn't work. The server configuration page may load, but the selection buttons on the left hand side have no affect on the display.

In my testing, only Netscape version 7.0 worked. However, if I loaded the page, left the page, and then try to load the page again, the load was corrupted. It was necessary to completely shutdown Netscape (thus closing the Java background task manager) before launching Netscape again, and logging into the AVT-843 XPort.

I tried these browsers and they did not work:

- Internet Explorer version 4.0
- Internet Explorer version 5.50
- Internet Explorer version 6.0
- Netscape version 4.7.

For my tests the host operating system was MS-Windows 98SE.  
I have not tried any other operating systems.

**21. Questions ??**

Contact the factory by e-mail, phone, or fax.

Contact information is provided here and on the bottom of page 1.

Post: 1509 Manor View Road  
Davidsonville, MD 21035 USA

Phone: +1-410-798-4038

Fax: +1-410-798-4308

E-mail: Support@AVT-HQ.com

Web site: www.AVT-HQ.com

**22. Bit Map for IDs, Masks, Commands, etc.**

A worksheet that may help in determining acceptance IDs, masks, and command values is on the next page.

bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	.
29																																	
bit																																	
ID																																	
32																																	
bit																																	
mask																																	
16																																	
bit																																	
mask																																	
8																																	
bit																																	
mask																																	
11																																	
bit																																	
ID																																	
16																																	
bit																																	
mask																																	
8																																	
bit																																	
mask																																	
.																																	

Bit map form