ADVANCED VEHICLE TECHNOLOGIES, Inc.

AVT - 852 & 853

Multiple Interface

Volume 1

CAN0: 2-wire (high speed)

CAN4: 2-wire (high speed) or Single Wire CAN (SWC)

CAN ISO 15765 support (both CAN channels)

LIN1: LIN or K-Line communications

LIN0: LIN communications

J1850 VPW (optional)

AVT-852: PC board revision A1
AVT-853: PC board revision F
AVT-853: PC board revision G
Firmware Version 3.6 (0A)
18 October 2018

1509 Manor View Road, Davidsonville, MD 21035 USA
+1-410-798-4038 (voice) support@AVT-HQ.com
www.AVT-HQ.com

Table of Contents

1. Introduction

This document describes the AVT-852 and AVT-853 hardware and firmware.

The AVT-852 and 853 are nearly identical units. They have the following differences.

AVT-852: USB communications with the host computer.
LIN0 (channel 7 does not exist).

AVT-853: Ethernet communications with host computer.
LIN0 (channel 7 has been added).

They are multiple network interfaces for in-vehicle networks. The hardware and operational firmware support the following networks/protocols:

- J1850 VPW (GM Class 2 compliant) with 4x mode.
- 2-wire CAN; ISO 11898-2 : 2016;
  (known as CAN0, channel number 0).
- 2-wire CAN; ISO 11898-2 : 2016;
  or
  Single Wire CAN (SWC); J2411;
  (known as CAN4, channel number 4).
- LIN or K-line; (LIN1 is channel 5, KWP is channel 6).
- LIN; (LIN0 is channel 7; AVT-853 board rev. D and later).

NOTE:
CAN0 and CAN4 do NOT support low speed fault tolerant CAN; ISO 11898-3 : 2006.

Some simultaneous operations are supported. Some are not.
Refer to the 5x 69 command in Section 16.

1.1 AVT-85x Note

References in this document to "AVT-85x" mean the AVT-852 or 853 - depending on the unit being used.

References to specific models are made where necessary.

The two interfaces are very nearly identical with two exceptions:

AVT-853 is Ethernet;	AVT-852 is USB.
AVT-853 has a second LIN0 channel;	AVT-852 does not.

1.2 Hardware

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

The current AVT-852 PC board revision status is shown on the title page of this document.
[Normally a stock item.]

Hardware status: www.AVT-HQ.com/852_hw.htm

The current AVT-853 PC board revision status is shown on the title page of this document.
[Normally a stock item.]

Hardware status: www.AVT-HQ.com/853_hw.htm

1.3 Firmware

Refer to our web site for the most up-to-date information about AVT-85x firmware versions:

www.AVT-HQ.com/852_asm.htm

1.3.1 Determining Firmware Version

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

- Connect to a host computer running the Hex Terminal or equivalent.
- Power on the 85x interface unit.
- The power-on notification is:
  $91 $27 indicates idle mode operation.
  $92 $04 $xx where xx is the firmware version.
  Example: xx = $01 indicates version 0.1.
- At any time, send the $B0 command.
- The response will be: $92 $04 $xx where xx is the firmware version.
- At any time, send the $B1 $01 command.
- The response will be: $93 $04 $xx $yy where xx yy is the extended firmware version.

1.3.2 Determining Model Number

Perform the following to determine the model number of your hardware.

- Connect to a host computer running the Hex Terminal or equivalent.
- Power on the 85x interface unit.
- Send the $F0 command.
- The response will be: $93 $28 $xx $yy where xxyy forms the model number. (eg. 0853)

1.3.3 Determining Board Revision Level

There are two revision levels in this product family.

- "Board" revision level. This can be determined by looking at the bottom of the PC board (not the component side). Written in copper is the board revision level and date.
- The "Circuit Configuration" revision level is written on the top (component side) of the PC board, in black marker, in the little white "rev" block.

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 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.
I5P	Means ISO 15765 process or processing. (Short acronym only used in this manual.)
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 11898-2	ISO specification for 2-wire CAN physical layer; high speed.
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-85x Operation

The AVT-85x 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. The AVT-852 will operate from USB power.

However, some communications modes will not function properly if the interface is only powered from the USB connection.

On power-up, the interface will, almost immediately, report to the host computer: $91 $27 and $92 $04 $xx (where xx is the unit firmware version number). This is the idle state, where the AVT-85x 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 14 for a list of commands supported while in the idle state.

4. Host Computer Connection

The AVT-852 uses a USB connection to the host computer.
Detailed information in the following paragraphs.

The AVT-853 uses an Ethernet connection to the host computer.
Detailed information in the following paragraphs.

4.1 AVT-852 Connection to Host Computer

The AVT-852 uses the FTDI embedded USB converter device (FT245BL). USB 1.1 operations are supported. Connection to the host computer is with a standard USB A/B cable.

The AVT-852 is powered by either the USB bus or the vehicle supply. You can connect the AVT-852 to a host computer and it will be discovered by the host and the port will be enumerated. (Vehicle power is required for VPW, Single Wire CAN, K-line, and LIN communications.)

USB driver software is provided by FTDI. AVT has tested and (currently) recommends using the FTDI Virtual Com Port (VCP) drivers for communications between the user application and the AVT-852 board. A separate document is available showing how to install the AVT-852 USB drivers. That document is posted on the AVT web site, Product Documents page.

Note that the virtual com port will show baud rate – it does not really exist. Select any baud rate.

4.2 AVT-853 Connection to Host Computer

The AVT-853 uses a Lantronix XPort embedded serial server to provide an Ethernet connection to the host computer. On the AVT-853 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-853 unit. Detailed information about this is provided in Section 4.2.1 and Appendices A and B.

On the AVT-853 board, the microcontroller and the XPort device communicate via an internal serial link. The factory default baud rate is 230.4 kbaud.

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

19.2 kbaud
38.4 kbaud
57.6 kbaud
115.2 kbaud
230.4 kbaud [default]

460.8 kbaud
921.6 kbaud

Refer to Section 16 for information about the 52 67 xx command (while in CAN mode) to change the host baud rate.

Note that the baud rate setting of the XPort serial server must be changed to match that of the AVT-853 microcontroller. Refer to Section 4.2.1 and Appendices A and B for information on setting the XPort parameters.

Obtain and read the document "AVT-853_Internal_Baud_Rate.pdf" for the procedure to change the internal baud rate. The document is available from our web site. A direct link is:
http://www.avt-hq.com/AVT-853_Internal_Baud_Rate.pdf

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

Refer to the User's Manual, Volume 2 for detailed EEPROM information.

4.2.1 Ethernet Setup

The AVT-853 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-853 will be totally nonfunctional.

Ethernet communications with the AVT-853 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-853, depending on what is to be accomplished and what port is used. A session can be established with the AVT-853 XPort to change settings such as the Ethernet IP Address (port 9999). In normal use, an Ethernet session will be opened with the AVT-853 to communicate with the vehicle network (port 10001).

The AVT-853 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).

4.2.1.1 Ethernet IP Address

The factory default IP address of the AVT-853 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-853 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-853 is using static IP addressing, set the net mask to 255.255.255.0.

4.2.1.2 Hardware or MAC Address

The hardware or MAC address of the AVT-853 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.

4.2.1.3 Vehicle Network Interface Port

Communications with the AVT-853 vehicle network interface is via port 10001.

All communications with the AVT-853 vehicle interface is in binary bytes [not ASCII hex]. Refer to Section 4.3 for a description of the 'packetized' communications protocol between the AVT-853 and the host computer. All communications with the AVT-853 follow the exact same rules and formats as that of the AVT-852 and all other AVT interface equipment.

4.2.1.4 Telnet Setup Port

The configuration of the AVT-853 XPort device can be examined and changed using the Telnet application via port 9999.

To start a Telnet setup session with the AVT-853 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-853 XPort device. Factory default settings are shown.

4.2.1.5 Web Page Setup

The configuration of the AVT-853 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-853 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-853 XPort will store the new settings and then reboot. Wait 1 to 2 minutes for the AVT-853 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-853 XPort configuration web page.

4.2.2 Ethernet IP Addressing Modes

Three IP addressing modes are available on the AVT-853 XPort.

- Static
- DHCP
- ARP

4.2.2.1 Static IP Addressing

The factory default addressing mode for the AVT-853 XPort is static and the address is set to 192.168.1.70. In static mode the Ethernet address of the AVT-853 is always the same and does not change when power is cycled.

4.2.2.2 DHCP Addressing

Setting the AVT-853 IP address to 0.0.0.0 will enable DHCP (Dynamic Host Configuration Protocol) function.

In this mode, the AVT-853 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-853 XPort will then switch to AutoIP addressing, described in the next Section.

An AVT-853 IP address of 0.0.1.0 enables DHCP addressing and disables AutoIP addressing.

4.2.3 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-853 XPort IP address is set to 0.0.0.0 and no DHCP server is found.

If, on power-up, the AVT-853 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-853 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-853 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.

4.3 Packet Communications with Host Computer

Communications between the host computer and the AVT-85x, 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

4.4 AVT-853 – Preventing Loss Of Data

With previous versions of firmware, during periods of high CAN bus traffic, it was possible for a byte to be "lost" in moving from the AVT-853 microcontroller to the control computer. This would cause the control computer to get out of synch with the AVT-853 (because the length of each packet is indicated in the header byte). This could usually only happen when traffic, on both CAN busses, was very heavy and all of it was being received by the AVT-853 and sent to the control computer.

Significant changes were made to the AVT-85x firmware to prevent this from happening. If, at one instant in time, there is so much data moving that is not room for a complete CAN frame to be stored and processed, that CAN frame will be discarded. A frame counter will be incremented, when that happens. There is also a related lost frame timer.

The user controls the lost frame counter and lost frame timer through the "7x 50" command.

If the lost frame counter function is enabled, and if the number of lost frames exceeds the user designated threshold – then the user will be notified when that threshold is exceeded.

Similarly, there is a lost frame timer that the user can enable and set the time-out value. If the lost frame counter is not zero when the timer expires, the user is notified as to the number of lost frames. After the user is notified as to the number of lost frames (for either the threshold or timer reasons), the lost frame counter is reset and operations start over.

Lost frame counter and timer function command.

71 50: Query for Lost Frame counter and/or timer function status
73 50 xx yy: Set the Lost Frame counter and timer functions.
xx is for the counter threshold function
xx = 00: count threshold function disabled
xx = 01 to FF: count threshold set point
yy is for the timer function
yy = 00: timer function disabled
yy = 01 to FF: time to wait, in milliseconds, to report lost frame count.

5. Vehicle Connection

The vehicle or network connector (P3) is an industry standard DA-15P connector and requires a DA-15S mate. The pin / signal assignments for the vehicle / network connector are listed here.

Pins that are Not Listed are reserved.
The user should not connect anything to those pins.

PC board revision level is labeled in copper on the bottom of the PC board.
(The revision level on the top of the board, in the white block, is the board configuration.)

5.1 AVT-852 Board Revision "A1"

Pin #	Description	Notes
1	CAN4_SWC	Bi-directional (only when CAN4 is configured as Single Wire CAN) [This signal goes through JP1]
2	J1850 VPW bus +	Bi-directional [This signal goes through JP3]
3	CAN4_H	Bi-directional (only when CAN4 is configured as 2-wire CAN)

4	Ground	pins #4 and #5 are connected together internally
5	Ground	pins #4 and #5 are connected together internally
6	CAN0_H	Bi-directional
7	K-Line LIN (channel 5) KWP (channel 6)	Bi-directional [This signal goes through JP2]
11	CAN4_L	Bi-directional (only when CAN4 is configured as 2-wire CAN)
13	V-Batt supply	Sourced by external equipment (vehicle)
14	CAN0_L	Bi-directional

P3 (the DA-15P connector on the AVT-852 board)

5.2 AVT-853 Board Revision "B1"

Pin #	Description	Notes
1	CAN4_SWC	Bi-directional (only when CAN4 is configured as Single Wire CAN) [This signal goes through JP1]
2	J1850 VPW bus +	Bi-directional [This signal goes through JP3]
3	CAN4_H	Bi-directional (only when CAN4 is configured as 2-wire CAN)
4	Ground	pins #4 and #5 are connected together internally
5	Ground	pins #4 and #5 are connected together internally
6	CAN0_H	Bi-directional
7	K-Line LIN (channel 5) KWP (channel 6)	Bi-directional [This signal goes through JP2]
11	CAN4_L	Bi-directional (only when CAN4 is configured as 2-wire CAN)
13	V-Batt supply	Sourced by external equipment (vehicle)
14	CAN0_L	Bi-directional

P3 (the DA-15P connector on the AVT-853 board)

5.3 AVT-853 Board Revision "D", "F", and "G"

Pin #	Description	Notes
1	CAN4_SWC	Bi-directional (when CAN4 is selected as Single Wire CAN)
2	J1850 VPW bus +	Bi-directional
3	CAN4_H	Bi-directional (when CAN4 is selected as 2-wire CAN)
4	Ground	pins #4 and #5 are connected together internally
5	Ground	pins #4 and #5 are connected together internally
6	CAN0_H	Bi-directional
7	K-Line LIN1 (channel 5) KWP (channel 6)	Bi-directional
11	CAN4_L	Bi-directional (when CAN4 is selected as 2-wire CAN)
13	V-Batt supply	Sourced by external equipment (vehicle)
14	CAN0_L	Bi-directional
15	LIN0 (channel 7)	Bi-directional

P3 (the DA-15P connector on the AVT-853 board)

5.4 Power Requirements

The AVT-85x board requires a nominal +12 VDC power supply; usually provided by the vehicle or any suitable external power supply.

5.4.1 Ground

Common ground is required between the AVT-85x board and the subject vehicle or module. P3 pins #4 and #5 are ground. They are connected together internally on the AVT-85x board. Only one is needed for normal operations.

5.4.2 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.)

5.4.3 Power Dissipation

Power dissipation for each unit is listed here. Measured with a supply voltage of 13.0 VDC.

AVT-852: 1 watt (nominal).

AVT-853: 2.75 watts (nominal).

5.5 Shunts

Note: These shunts only existed on board revision "B1". They were removed on later board revisions.

On the AVT-85x board are three shunts or jumpers: JP1, JP2, JP3.

They are for:

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

6. ADC Connection

Unless otherwise requested, the analog to digital input capability of the AVT-85x boards is NOT installed. The following notes only apply to those boards where the functionality is installed.

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-85x 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 29 indicates that LIN0 mode of operation is active.
[rev. “D” and “F” boards only]
The report $91 19 indicates that LIN1 mode of operation is active.

7.1 AVT-852 Board Revision “A1”

The AVT-852 board revision “A1” supports operations of two simultaneous CAN channels and one LIN or KWP channel when in CAN mode.

CAN0

2-wire high speed CAN; ISO 11898-2.
Channel number: 0.

CAN4

Configured at the factory as either:
2-wire high speed CAN; ISO 11898-2.
or
Single wire CAN (SWC); J2411.
Channel number: 4.

LIN

Supports revisions: 1.2, 1.3, 2.0, 2.1, 2.2A.
Channel number: 5.

KWP

Specifications: ISO 9141, ISO 9141-2, and ISO 14230.
Channel number: 6.

7.2 AVT-853 Board Revision “B1”

The AVT-853 board revision “B1” supports operations of two simultaneous CAN channels and one LIN or KWP channel when in CAN mode.

CAN0

2-wire high speed CAN; ISO 11898-2.
Channel number: 0.

CAN4

Configured at the factory as either:
2-wire high speed CAN; ISO 11898-2.
or
Single wire CAN (SWC); J2411.
Channel number: 4.

LIN
Supports revisions: 1.2, 1.3, 2.0, 2.1, 2.2A.
Channel number: 5.

KWP
Specifications: ISO 9141, ISO 9141-2, and ISO 14230.
Channel number: 6.

7.3 AVT-853 Board Revision "D" and "F"

The AVT-853 board revision "D" and "F" support operations of two simultaneous CAN channels, one LIN or KWP channel, and one dedicated LIN channel – all when in CAN mode.

CAN0

2-wire high speed CAN; ISO 11898-2.
Channel number: 0.

CAN4

Software selectable to be either:
2-wire high speed CAN; ISO 11898-2.
or
Single Wire CAN (SWC); J2411.
Channel number: 4.

LIN1

Supports revisions: 1.2, 1.3, 2.0, 2.1, 2.2A.
Same transceiver as KWP (channel 6).
Channel number: 5.

KWP

Specifications: ISO 9141, ISO 9141-2, and ISO 14230.
Same transceiver as LIN1 (channel 5).
Channel number: 6.

LIN0

Supports revisions: 1.2, 1.3, 2.0, 2.1, 2.2A.
Channel number: 7.

7.4 CAN Mode Default Configuration

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 determined by hardware revision and configuration.
  (It may be either 500 kbaud or 33.3333 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).
- LIN1 mode is enabled and the LIN1 bus speed is 9600 baud.
- LIN0 mode is enabled and the LIN0 bus speed is 9600 baud.
  (Only if LIN0 exists on the board.)

7.5 CAN0 - 2-wire CAN

CAN0 is a high speed 2-wire CAN channel that is ISO 11898-2 : 2016 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-85x board has been designed to support several different 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.

7.5.1 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.

7.6 CAN4 - 2-wire CAN

For AVT-852 revision “A1” and AVT-853 revision “B1” boards, CAN4 can be configured at the factory for 2-wire high speed CAN ISO 11898-2.

For AVT-853 revision “D” and “F” boards, CAN4 can be software selected to be 2-wire high speed CAN ISO 11898-2.

CAN0 is a high speed 2-wire CAN channel that is ISO 11898-2 : 2016 compliant.
It uses the Philips TJA1050 transceiver.
The CAN_H signal is routed to the D-15 network connector pin #3.
The CAN_L signal is routed to the D-15 network connector pin #11.

The AVT-85x board has been designed to support several different network termination schemes for CAN4. 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.

7.7 CAN4 - Single Wire CAN (SWC)

For AVT-852 revision "A1" and AVT-853 revision "B1" boards, CAN4 can be configured at the factory for Single Wire CAN (SWC) J2411.

For AVT-853 revision "D" and "F" boards, CAN4 can be software selected to be Single Wire CAN (SWC) J2411.

CAN4 is Single Wire CAN (SWC) that is SAE J2411 compliant.
It uses the Philips AU5790 transceiver.

AVT-853 revision "B1" boards only: shunt JP1 connects / disconnects the SWC bus from pin #1 of P3 (the DA-15P network connector).

For AVT-853 revision "D" and "F" boards, the SWC signal is routed directly to P3 (the D-15 network connector) pin #1.

7.7.1 Shunt JP1

Shunt JP1 only exists on the AVT-852 revision "A1" and AVT-853 revision "B1" boards.

Shunt JP1 on the AVT-85x board connects / disconnects the Single Wire CAN (SWC) line from pin #1 of P3; the DA-15P Vehicle connector. When SWC is installed, the factory default is JP1 is installed.

7.7.2 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.

7.8 CAN Secondary Operational Modes

When operating in CAN mode, three secondary operating modes are supported. There are limitations to the combinations but all operate independently of and simultaneously to both CAN channels.

7.8.1 Board revision "B1" Allowed Secondary Modes

Only one secondary operational mode can be supported: LIN or KWP.

After entering CAN mode, LIN mode is enabled as the default secondary mode.

LIN mode can be selected using the 52 69 01 command. KWP mode is disabled.

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

Both modes are disabled with the 52 69 00 command.

7.8.1.1 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.

7.8.1.2 KWP Secondary Operational Mode Channel Number

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

7.8.2 Board revisions "D" and "F" Allowed Secondary Modes

There are multiple secondary operational modes supported.

After entering CAN mode, LIN1 and LIN0 modes are enabled.

The allowed secondary modes are:

- None 52 69 00
- LIN1 only 52 69 01
- KWP only 52 69 02
- LIN0 only 52 69 04
- LIN1 and LIN0 52 69 05
- KWP and LIN0 52 69 06

7.8.2.1 LIN1 Secondary Operational Mode Channel Number

When LIN1 mode is active as a secondary operational mode, it is designated channel 5.

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

7.8.2.2 KWP Secondary Operational Mode Channel Number

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

7.8.2.3 LI0N Secondary Operational Mode Channel Number

When LIN0 mode is active as a secondary operational mode, it is designated channel 7.

7.9 CAN Communications General Notes

A CAN network has to consist of at least two functioning CAN nodes.
(The AVT-85x can be one of the nodes).

Each CAN channel of the AVT-85x is independent of the other channel.
(This applies to all channel parameters.)

Each CAN channel of the AVT-85x 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 4.3 and 16 for detailed information about CAN messages and packets to and from the network.

7.9.1 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

7.9.2 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 7.9 for a discussion.

The CAN channel can transmit messages.

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

7.9.3 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 7.9 for a discussion.

The CAN channel can only monitor received messages.

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

7.9.4 Transmit Command

The fields and construction of a transmit command are shown here. The transmit command is also explained in the Commands and Responses – Section 16.

There are three forms of the transmit command. The number of bytes in the transmit command determines the format of the command to use.

7.9.4.1 Transmit Command Format 0x

The 0x form of the transmit command can be used when the byte count following the header is $0F or less. Refer to the beginning of Section 16 for a complete description of the transmit command.

7.9.4.2 Transmit Command Format 11 xx

The 11 xx form of the transmit command can be used when the byte count following the header is $FF or less. Refer to the beginning of Section 16 for a complete description of the transmit command.

7.9.4.3 Transmit Command Format 12 xx yy

The 12 xx yy form of the transmit command can be used when the byte count following the header is $1004 or less (that limit is imposed by ISO15765). Refer to the beginning of Section 16 for a complete description of the transmit command.

7.9.4.4 Byte Count Limits

The total number of message data bytes permitted in a transmit command depends on whether or not ISO 15765 processing is enabled or specified for the transmit command.

When ISO 15765 is disabled and not specified, maximum is 8 data bytes.
When ISO 15765 is enabled or specified, maximum is 4095 data bytes.

7.9.4.5 Pacing Note 1

The AVT-853 will process a transmit command from the host as quickly as possible. It is also possible that the host can send transmit commands to the AVT-85x so quickly that the transmitted CAN frames cause problems for the downstream module. This can happen with ISO15765 processing enabled or disabled.

A pacing timer can be used to add a delay between transmitted CAN frames. The 7x 3F command sets the value of the pacing timer. The timer interval is one millisecond. There is an independent pacing timer for each CAN channel. Refer to the 7x 3F command in Section 16.

NOTE: This is NOT the same pacing timer used for multi-frame messages when ISO15765 processing is enabled (which is the 7x 35 command). Refer to Section 7.13.8.1 for information about that timer.

7.9.5 Transmit – Preserving Order of Transmit Commands

If the user issues a series of transmit commands, faster than can be actually transmitted to the CAN bus, it is possible that the messages will appear on the CAN bus out of order as compared to the order of the commands.

To preserve order of transmit commands, set bit 5 of the channel byte in the transmit command. This will force the AVT-853 to use only transmit buffer0 (the highest priority transmit buffer) and maintain the order of the messages.

7.9.6 Receive Response

There are two possible 'from the network' responses that the AVT-85x can send to the host.

1. A CAN message from the network (from another CAN node).
2. A transmit acknowledgement; or transmit ack.

Both are described at the beginning of Section 16.1.

Regarding messages from the CAN network - there are three possible forms of that receive response. The number of bytes in the receive response determines the format used by the AVT-85x interface.

7.9.6.1 Receive Response Format 0x

The 0x form of the receive response is used when the byte count of the response (not including the header byte) is $0F or less. Refer to Section 16.1 for a description of all bytes in the packet.

7.9.6.2 Receive Response Format 11 xx

The 11 xx form of the receive response is used when the byte count of the response (not including the header byte) is $FF or less. Refer to Section 16.1 for a description of all bytes in the packet.

7.9.6.3 Format 12 xx yy

The 12 xx yy form of the receive response is used when the byte count of the response (not including the header byte) is $FFFF or less. Refer to Section 16.1 for a description of all bytes in the packet.

7.9.7 Time Stamps

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

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

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

The 52 08 00 command disables all time stamps.
The 53 08 0x 00 command disables time stamps for channel x.

The 52 08 01 command enables time stamps for all channels.
The 53 08 0x 01 command enables time stamps for channel x.

The time stamp clock is separate for each CAN channel and separate from the other channels.

The time stamp is a 16-bit free running counter that is driven by the baud clock for that CAN 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 for all channels.
The 53 08 0x 02 command enables time stamps for channel x.
The time stamp clock is the same for both CAN channels and all other channels.

The time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at $FFFF.

7.9.7.1 Transmit Acknowledgment Description

Time stamps disabled:

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

Time stamps enabled:

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

7.9.7.2 Receive Message Description

Time stamps disabled:

0x qr tt vv ww zz mm nn ... :

- x: count of bytes to follow.
- q: 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
- r: channel: 0, 4.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data.

Time stamps enabled:

0x jj kk qr tt vv ww zz mm nn ... :

- x: count of bytes to follow.
- jj kk: time stamp [optional]
- q: 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
- r: channel: 0, 4.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data.

7.10 Acceptance ID and Mask

7.10.1 Configuration

Each CAN channel of the AVT-85x is independent of the other channel.

Each CAN channel of the AVT-85x 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.

7.10.2 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 in Section 1.6 and Responses in Section 16.1.

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.

7.10.2.1 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.

7.10.2.2 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.

7.10.2.3 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.

7.10.2.4 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.)

7.10.2.5 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.

7.10.2.6 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.

7.10.2.7 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 = 1B CC xx xx [or] 1B 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.)

7.10.2.8 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.

Mask bit 2 set to don't care demonstrates how two ranges of message ID's will be accepted.

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

```
05 08  to  05 0F
05 28  to  05 2F
```

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

```
14 20 00 00  to  14 3F FF FF
14 A0 00 00  to  14 AF FF FF
```

7.10.3 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 Periodic Message Support

As of firmware version 2.4 (0A) the organization of the periodic messages for both CAN channels has been changed significantly. For previous versions of firmware, the user should consult an earlier version of this manual.

~~There are a total of $64 (100 decimal) periodic messages for CAN mode~~.

As of firmware version 2.7 (0A) there are a total of $58 (88 decimal) periodic messages for CAN mode.

The periodic messages can be dynamically allocated across the two CAN channels and two groups for each channel. The operator defines and sets up the desired periodic messages, enables them, and the AVT-85x unit will then transmit those messages, at the defined interval, without any operator intervention.

The AVT-85x will not issue a transmit ack when a periodic message is transmitted.

NOTE: The periodic messages for the two LIN channels and the KWP channel have not been changed. Refer to the respective sections of this manual for information about those.

7.12.1 Summary Of Changes

There are two CAN channels (numbered 0 and 4). For each CAN channel there are two groups (numbered 1 and 2) of periodic messages. The user can allocate the available periodic messages among those four groups. Each group must have a minimum of one message assigned. That means, in the extreme, three groups could have assigned one message each and the fourth group would have $55 (85 decimal) messages assigned.

To be backward compatible with previous versions of firmware, CAN0 group 1 and 2 and CAN4 group 1 are all assigned $10 (16 decimal) periodic messages. The balance of $28 (40 decimal) are assigned to CAN4 group 2.

7.12.2 Organization Of Periodic Messages

There are a total of $58 (88 decimal) CAN periodic messages available.

The periodic messages are numbered $01 to $58.

The periodic messages are assigned, from lowest message number to higher, to CAN0 group 1, CAN0 group 2, CAN4 group 1, and CAN4 group 2 – in that order.

The first message assigned to CAN0 group 1 is always message number $01.

The ’72 4C yy’ command sets the first message assigned to CAN4 group 1; where ‘yy’ is the message number.

The ’73 4D 0x yy’ command assigns the first message number to CANx group 2.

The user should first assign the CAN4 group 1 message using the ’72 4C yy’ command.

Then, the user should assign the first message of CAN0 group 2 using the ’73 4D 00 yy’.

Then, the user should assign the first message of CAN4 group 2 using the ’73 4D 04 yy’.

The message number used in the ‘7x 4C’ and ‘7x 4D’ commands are the actual or absolute message number in the range $01 to $58.

7.12.3 The Default Configuration

When CAN mode is first entered, the CAN periodic messages are assigned as follows.

CAN0 group 1: $01

CAN0 group 2:	$11	(same as ’73 4D 00 11’ command)
CAN4 group 1:	$21	(same as ’72 4C 21’ command)
CAN4 group 2:	$31	(same as ’73 4D 04 31’ command)

This means:

Messages $01 to $10 are assigned to CAN0 group 1.
Messages $11 to $20 are assigned to CAN0 group 2.
Messages $21 to $30 are assigned to CAN4 group 1.
Messages $31 to $58 are assigned to CAN4 group 2.

With the exception of the last group, the number of messages assigned to each group mimics the situation found in previous versions of firmware.

Here is an attempt to illustrate the organization of the periodic message table.

Msg Num (hex)	Boundary (command)	[default configuration shown]
01	← CAN0 group 1 (fixed)	
02		
…		
0F		
10		
11	← CAN0 group2 (73 4D 00 11)	
12		
…		
20		
21	← CAN4 group1 (72 4C 21)	
22		
…		
30		
31	← CAN4 group2 (73 4D 04 31)	
32		
…		
3F		
40		
…		
50		
…		
58		

7.12.4 Dynamic Allocation

The user is free to dynamically allocate the messages. This can be done while messages are defined, enabled, and operating.

The ‘7x 4C’ and ‘7x 4D’ commands do not affect the contents of any messages.

The user should send the ‘7x 4C’ command before sending any ‘7x 4D’ commands.

If the user moves the CAN4 group 1 boundary (the ‘7x 4C’ command) such that either group 2 boundary is affected, the affected boundary will be moved to the mid-point of the available range. A ’83 4D 0x yy’ response will be issued to inform the user that the affected boundary was moved.

The user can not (or should not be able to) move any of the boundaries in a manner that will put them out of order.

7.12.5 Message Numbering In The Periodic Message Commands

To maintain backward compatibility the following commands:

7x 18	(periodic message setup)
7x 1A	(periodic message enable)
7x 1B	(periodic message timer/count)
7x 40	(periodic message 'ARC' function)
7x 47	(periodic message 'RUP' function)
7x 4A	(periodic message 'CAC function)

Are subject to the following message numbering rules.

If channel CAN0 is specified, the message numbers are **absolute** and the range is: $01 to $58. The specified message number directly accesses the table.

This means all messages can be accessed when CAN0 is specified.

If channel CAN4 is specified, the message numbers are **relative** to the CAN4 group1 setting (the '7x 4C' command). The message number range is dynamic and is: $01 to {$58 - $xx + $01} where 'xx' is the value from the '72 4C xx' command.

This means only messages assigned to CAN4 can be accessed when CAN4 is specified.

7.12.6 Group Operations

Each CAN channel (0 and 4) is assigned two groups (1 and 2) of periodic messages.

The available messages are assigned to the four groups using the '7x 4C' and '7x 4D' commands.

Each group can operate as Type1 or Type2. Type operations are described below.

Each message is independently disabled or enabled; '7x 1B' command.

Each message has its own time interval; '7x 1A'command. (For Type1 operations.)

7.12.7 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

7.12.8 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.

7.12.8.1 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 sec
   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 sec
    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

7.12.9 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.

7.12.9.1 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

7.12.10 Periodic Message Commands

All commands are listed in Section 16. A brief summary 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
- 7x 4C Specify the first periodic message assigned to CAN4 group 1.
- 7x 4D Specify the first periodic message assigned to CANx group 2.

7.13 Periodic Message Special Functions

There are several special functions available for all CAN periodic messages operating in Type1 mode. These special functions were developed specifically at customer request. Each of the functions are described below.

Each function is available to every CAN periodic message. Each function and each periodic message are independent. In other words, one periodic message can have one function enabled and another periodic message can have another function enabled.

Only one mode is allowed to be enabled for any given periodic message. If you attempt to enable more than one mode, the last mode command will be the one enabled.

For all of these functions, the data field of a periodic message can be changed ‘on the fly’. You do NOT need to disable the message or the function to change anything.

7.13.1 CAN Frame Data Definition

Each CAN frame can contain up to 8 data bytes.

In the following discussion, Data0 is the first data byte in the CAN frame; or the first data byte onto the network; or the first data byte after the message ID.

Likewise Data7 is the last data byte of the CAN frame.

Within a byte, the bits are numbered from 0 (least significant bit) to 7 (most significant).

7.13.2 ARC Function Description

"ARC" = Asynchronous Rolling Counter.

A basic description of the operation of this function follows. If you require detailed information, contact me.

The counter is bits 3:2 of Data0. Increment the counter. Add the counter bits to the MainHighVltCntctCmd bits and two's complement them. The resulting two bits are "MainHighVltCntctCmdProtVal". Write those bits to 1:0 of Data1.

The message is then queued for transmission.

7.13.2.1 ARC Function Command

Details of the ARC function command.

73 40 0x yy

query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 40 0x yy 0v

Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.

7.13.3 RUP Function Description

"RUP" = Rolling UPper nibble.

A basic description of the operation of this function follows. If you require detailed information, contact me.

Mode1:
The counter is bits 7:4 of Data6. Increment the counter. Compute the J1850 CRC of the message data including Data0 to Data6. Write the resulting CRC to Data7. The message is then queued for transmission.

Mode2:
The counter is bits 7:4 of the byte specified in the command. Increment the counter. If enabled, compute the J1850 CRC of the message data including Data0 to and including the counter byte. If enabled, write the resulting CRC to data byte after the counter. The message is then queued for transmission.

7.13.3.1 RUP Function Command

Detailed explanation of the RUP function command.

73 47 0x yy

query for status

x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 47 0x yy 0v
Disable / Enable command – Mode1.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.

75 47 0x yy 0v wz
Disable / Enable command – Mode2.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 2 to enable.
w = 0 to exclude the CRC byte; = 1 to include the CRC byte.
z = location of the counter,
byte0 to byte6 if the CRC byte is included,
byte0 to byte7 if the CRC byte is excluded.

7.13.4 CAC Function Description

"CAC" = Counter And Checksum.

A basic description of the operation of this function follows. If you require detailed information, contact me.

The counter is bits 3:0 of Data1. Increment the counter. Compute the checksum of the message using Exclusive OR, every byte of the message from Data1 to the end. The end is determined by the number of data bytes written to the data field in the periodic message setup command. Exclusive OR the result with a constant value. Write the resulting checksum to Data0. The message is then queued for transmission.

7.13.4.1 CAC Function Command

Details of the CAC function command.

73 4A 0x yy
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 4A 0x yy 0v
Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.

75 4A 0x yy 0v rr
Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.

v = 0 to disable, = 1 to enable.
rr = is the constant used in the checksum computation.

Note: If you use the 74 4A form of the command, the constant is left at the value it was at when last written. The constant is initialized to $00 when you enter CAN mode. The constant value is only changed by using the 75 4A form of the command.

7.13.5 CIB Function Description

"CIB" = CAN Increment Byte.

A basic description of the operation of this function follows. If you require detailed information, contact me.

The counter is byte sized and the location in the CAN data field is specified in the command. The counter increment is specified in the command. The counter increment is 'usually' $0F. Increment the counter. If the counter was at $70, it should go to $8F. The message is then queued for transmission.

7.13.5.1 CIB Function Command

Details of the CIB function command.

73 4E 0x yy
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 4E 0x yy 0v
Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.

76 4E 0x yy 0v 0w rr
Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.
w is the location of the counter byte, 0 to 7.
rr is the increment amount.

Note: If you use the 74 4E form of the command, the location and increment amount are left at the values they were at when last written.

7.13.6 RC2 Function Description

"RC2" = Rolling Counter 2.

A basic description of the operation of this function follows. If you require detailed information, contact me.

The counter is bits 6:3 of Data4. Increment the counter. Compute the J1850 CRC of message data bytes 0 to 6 inclusive. Write the computed CRC to Data7. The message is then queued for transmission.

7.13.6.1 RC2 Function Command

Details of the RC2 function command.

73 4F 0x yy
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 4F 0x yy 0v
Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.

7.13.7 RC3 Function Description

"RC3" = Rolling Counter 3.

A basic description of the operation of this function follows. If you require detailed information, contact me.

The counter is bits 7:6 of Data6. Increment the counter. Clear bits 2:0 of Data6. Compute the checksum of the message, using 11-bit unsigned addition, from Data0 to Data6. Take the 11-bit CAN ID, divide by 8 and add that to the checksum. Write the lower 8-bits of the checksum Data7. Write the upper 3 bits of the checksum to bits 2:0 of Data6. The message is then queued for transmission.

7.13.7.1 RC3 Function Command

Details of the RC3 function command.

73 51 0x yy
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 51 0x yy 0v
Disable / Enable command.
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0 to disable, = 1 to enable.

7.13.8 RC4 Function Description

"RC4" = Rolling Counter 4.

A basic description of the operation of this function follows. If you require detailed information, contact me.

The counter is 4-bits. It is inserted into an upper or lower byte of one data byte in a CAN frame. The J1850 CRC is computed and inserted into the data byte following the counter byte.

7.13.8.1 RC4 Function Command

Details of the RC4 function command.

73 52 0x yy
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.

74 52 0x yy 0v
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0: disable
v = 1: enable

75 52 0x yy 0v rt
query for status
x is the CAN channel, 0 or 4.
yy is the periodic message number.
v = 0: disable
v = 1: enable
r = 0: counter is lower nibble
r = 1: counter is upper nibble
t: location of counter, 0 to 6

7.13.9 Periodic Pause Function

The Periodic Pause function, when enabled for a specific CAN channel, will inhibit all CAN periodic messages whenever an ISO15765 transaction is in-progress. Note that this only applies to ISO15765 transactions that require more than one CAN frame; in other words, if a multi-frame message transaction is in-progress on the CAN channel, no periodic messages will be queued for transmission.

7.13.9.1 Periodic Pause Function Command

Refer to the 7x 1F command in Section 16 for detailed information regarding the command format.

7.14 ISO 15765 Support

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

Full ISO 15765 processing capability is provided 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 Terms and 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.

Flow Control: A CAN frame with protocol handshaking information.

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.

Receive: A message from the CAN network to the host computer.
A message may be one or more CAN frames.

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

Transmit: A message from the host computer to the CAN network.
A message may be one or more CAN frames.

7.14.2 Modes of Operation

Firmware version 1.0 and later introduced three modes of ISO 15765 operations.

(Firmware version 0.9 and earlier only supported two ISO modes:
disabled (now called Mode0)
and
enabled (now called Mode1).

The three new ISO operating modes are described here.

7.14.2.1 Mode0

Mode0 is the default mode when CAN mode is entered. It can be selected using the command:
73 26 0x 00 (where x is the CAN channel number).

Mode0 means ISO 15765 processing is disabled for all receive messages (CAN frames from the CAN network).

Only messages specifically indicated by the host are ISO 15765 formatted before being transmitted to the CAN network. Refer to the beginning of Section 16 for information about formatting a transmit command.

7.14.2.2 Mode1

Mode1 means ISO 15765 processing is enabled for all transmit and receive messages (messages to and from the CAN network). This mode is selected using the command:
73 26 0x 01 (where x is the CAN channel number).

All messages through the AVT-85x interface (for the specified CAN channel) are treated and processed as ISO 15765 formatted messages. This means:

- All frames received from the CAN network are assumed to be ISO 15765 formatted and are processed accordingly.
- All transmit commands from the host are reformatted to meet ISO 15765 are transmitted to the CAN network.

7.14.2.3 Mode2

Mode2 means that only receive messages (messages from the CAN network) matching the expected receive ID will be ISO 15765 processed. This mode is selected using the command:

73 26 0x 02 (where x is the CAN channel number).

The 7x 29 command is used to specify the ID of the messages from the CAN network that are to be ISO 15765 processed. All other messages received from the CAN network are not ISO 15765 processed and are passed to the host.

Only messages specifically indicated by the host are ISO 15765 formatted before being transmitted to the CAN network. Refer to the beginning of Section 16 for information about formatting a transmit command.

7.14.3 Receive Operations - General Notes

When a receive message is ISO 15765 processed (Mode1 or Mode2), the AVT-85x will not send the PCI byte to the host, unless an error is encountered.

When receiving a multi-frame message, the AVT-85x will automatically compose and send a Flow Control frame. The AVT-85x will never send a transmit ack (02 0x 0y) to the host when a Flow Control frame is transmitted.

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

Error checking is implemented and the host is notified when an error is encountered. A list of the error codes is provided in the Responses Section 16.1.

7.14.3.1 Flow Control Frame

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.

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.

7.14.4 Receive Operations – Mode0

No frames received from the CAN network are subject to ISO 15765 processing.

All frames are passed directly to the host without modification.

7.14.5 Receive Operations – Mode1

All frames received from the CAN bus are processed according to the rules of ISO 15765. The usual CAN frame ID filtering function applies.

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-85x 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-85x handles all handshaking with the downstream module. When the complete message is received, the AVT-85x 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 Section 16.

If a CAN frame is received that is not ISO 15765 formatted, subsequent actions will be undefined.

7.14.6 Receive Operations – Mode2

Only frames received from the CAN bus that match the expected ID are processed according to the rules of ISO 15765. The usual CAN frame ID filtering function applies.

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-85x 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-85x handles all handshaking with the downstream module. When the complete message is received, the AVT-85x 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 Section 16.

7.14.7 Transmit Operations

When transmitting a message to the CAN network, there are two methods of specifying if the message should or should not be ISO 15765 formatted. The methods depend on the ISO 15765 Mode and are described below.

A transmit command is a command from the host containing a message for the CAN network.

7.14.7.1 Mode0

Only transmit commands from the host with bit4 of the CAN channel byte set (=1) are ISO 15765 formatted and then transmitted onto the CAN network. Refer to the beginning of Section 16 for information about formatting a transmit command.

All other transmit commands are transmitted onto the CAN network without modification.

7.14.7.2 Mode1

All transmit commands from the host are ISO 15765 formatted and then transmitted onto the CAN network.

7.14.7.3 Mode2

Only transmit commands from the host with bit4 of the CAN channel byte set (=1) are ISO 15765 formatted and then transmitted onto the CAN network. Refer to the beginning of Section 16 for information about formatting a transmit command.

All other transmit commands are transmitted onto the CAN network without modification.

7.14.8 Transmit Operations – General Notes

This section only applies to transmit commands (messages) that are to be reformatted to ISO 15765 prior to being transmitted onto the CAN network (as described in the previous section).

The AVT-85x 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 is used and the function is enabled)
all of the data

A properly constructed transmit command contains all of that information. If any problems are encountered - one or more error responses will be sent to the host.

When ISO 15765 processing is not used:

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 command 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 specified or 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 .... may not work.
The alternate transmit command formats (11 xx and 12 xx yy)
are available and will work.

After the host sends complete transmit command, the AVT-85x 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.

The user should:

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

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

If AE (Address Extension) is used for a particular application and module, then include the AE byte just once, in the correct place of several commands and enable AE support for the CAN channel being used.

7.14.8.1 Pacing Note 2

When transmitting a multi-frame message the AVT-85x will attempt to use the separation time specified in the flow control frame received from the downstream module.

If the downstream module responds with a flow control separation time of 00 the AVT-85x will transmit frames as fast as it can. It is possible an AVT-85x can transmit frames too quickly.

The 7x 35 pacing command will insert a programmable delay between transmitted frames. This delay will only be used if the module flow control frame specified a time of 00.

Empirical testing revealed the following results:

pacing count = $00 frame spacing ~ 258 microseconds
pacing count = $0A frame spacing ~ 559 microseconds
pacing count = $14 frame spacing ~ 1048 microseconds

7.14.9 Operation Examples

Set the acceptance mask mode.

7x 2B ...

Set the acceptance mask(s).

7x 2C ...

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

7x 2A ...

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 (Mode1)

or
73 26 0x 02 (Mode2)

Enable the CAN channel.
73 11 0x 01

7.14.9.1 Example #1 (Mode1)

The user wants to set up the AVT-85x to exchange ISO 15765 formatted messages with 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.)

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 mask mode to 4 (qty. 4 16-bit masks)
   73 2B 00 04
4. Set mask0 to all bits must match
   75 2C 00 00 00 00
5. Set acceptance ID0 to 357
   75 2A 00 00 03 57
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 Mode1 processing for CAN0
    73 26 00 01
11. Enable CAN0 operations
    73 11 00 01

12. Transmit an ISO 15765 formatted message with ID = 246 and 6 bytes of data
    09 00 02 46 11 22 33 44 55 66

At this point the AVT-85x will receive and transmit ISO 15765 formatted messages to/from the downstream module across the CAN network.

7.14.9.2 Example #2 (Mode2)

The user wants to set up the AVT-85x to receive all CAN network messages but only specified IDs are to be treated as ISO 15765 formatted. The specifics are:

- 2-wire CAN at 500k baud.
- 11-bit message IDs.
- No AE.
- Receive message ID (ISO 15765) = 357.
- Transmit message ID = 246.
  (The flow control frame ID is usually the same as the transmit message ID.)

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 mask mode to 4 (qty. 4 16-bit masks)
   73 2B 00 04
4. Set mask0 to all bits are don't care
   RTR bit is don't care
   only receive 11-bit ID frames
   75 2C 40 00 03 FF
5. Set acceptance ID0 to 00
   75 2A 00 00 00 00
6. Set expected ISO 15765 Mode2 receive ID
   74 29 00 03 57
7. Disable padding
   73 27 00 00
8. Disable AE
   73 30 00 00
9. Set Flow Control ID = 246
   74 0F 00 02 46
10. 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
11. Enable ISO 15765 Mode2 processing for CAN0
    73 26 00 02

12. Enable CAN0 operations
    73 11 00 01

13. Transmit an ISO 15765 formatted message with ID = 246 and 5 bytes of data
    08 10 02 46 11 22 33 44 55

At this point the AVT-85x will receive and transmit both non-ISO and ISO formatted messages to/from the downstream module across the CAN network.

7.14.9.3 Example #3 (Mode1)

The user wants to set up the AVT-85x to exchange ISO 15765 formatted messages with 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 mask mode to 2 (qty. 2 32-bit masks)
   73 2B 04 02

4. Set mask0 to all bits must match
   77 2C 04 00 00 00 00 00

5. Set acceptance ID0 to 13 57 9A CE
   77 2A 84 00 13 57 9A CE

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 Mode1 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

13. Transmit a message with ID = 02 46 8B DF and 18 (decimal) bytes of data.
    11 17 00 02 46 8B DF 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-85x will receive and transmit ISO 15765 formatted messages to/from the downstream module across the CAN network.

7.14.10 ISO 15765 Questions and Engineering Support

Some may find the ISO 15765 protocol and its implementation in the AVT-85x 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-85x 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 "85x_s01a.pdf" is included here.

7.15.1 Operation Description

The ABX function gives an AVT-85x 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-85x 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-85x will begin transmitting the stored CAN data. The data is transmitted according to the stored CAN parameters. The AVT-85x 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-85x.

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-85x unit is transmitting.

This parameter sets a count value. The AVT-85x 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-85x counting the specified number of milliseconds between transmitted CAN frames.

A loop is the time it takes the AVT-85x 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-85x 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-85x 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-85x 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-85x receives the command and all expected data bytes, the AVT-85x 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-85x unit. Thus, once an AVT-85x 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-85x 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

75 3A 0x 0y mm nn: set CAN channel x operation to value y, start address = mm nn.

77 3A 0x 0y mm nn pp qq: set CAN x operation to value y,
start address = mm nn and byte count = pp qq

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-85x 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-85x 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

7.16 Channel Activity

This function is available for both CAN channels, independently. When the function is enabled for a CAN channel and the AVT-85x 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-85x 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 0x yy command disables/enables the Channel Activity function of the specified CAN channel.

The 72 3C 0x 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.

7.17 ATD Read Function

This function was designed to read certain CAN messages that contain Analog To Digital (ATD) data and store the number of messages processed as well as the minimum and maximum values for each channel.

The CAN messages being monitored are assumed to be ISO 15765 formatted and therefore ISO 15765 processing has to be enabled to function properly.

The CAN messages subject to this function are immediately discarded after being read. The host never sees these messages.

7.17.1 Message Construction

The message of concern is ISO 15765 formatted and contains 8 data bytes. Therefore, it is a segmented message. Shown here is the ID and data bytes only. Not shown are the ISO 15765 protocol bytes.

ID	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

ID	11 or 29 bit
Data0, Data1, Data2:	fixed data field
Data3:	ATD channel number (lower nibble only)
Data4:Data5:	ATD data, unsigned 16-bit integer
Data6, Data7:	ignored

7.17.2 Set Up

The user specifies the ID of expected ATD messages using the 7x 43 command.

The user specifies the first three bytes of the data field using the 7x 44 command.

The function is enabled using the 7x 41 command.

An acceptance ID and associated mask must be set to receive the message of interest.

The ISO 15765 function must be enabled, in either Mode1 or Mode2 using the 7x 26 command.

If ISO 15765 Mode1 is enabled then the AVT-85x will check the receive IDs of all received messages to locate the one to be ATD read processed.

If ISO 15765 Mode2 is enabled, the Mode2 receive ID must be set to the same ID as the message of interest using the 7x 29 command.

7.17.3 Operation

After the function is set up, the user enables it. When the function is enabled, the count, the minimum, and the maximum values for each ATD monitor channel are cleared.

The user reads the status of one or more ATD monitor channels using the 7x 42 command. Each time a channel status is read the AVT-85x reports the number of messages processed, the minimum values seen, and the maximum value seen since the last time status was read. The count, the minimum, and the maximum are then cleared.

7.18 CAN Digital Output Function

This function was custom developed for a customer.

When enabled, a digital output is driven high or low depending on the outcome of a test.

Note that the digital output is not buffered. It is driven directly from an I/O pin of the AVT-853 microcontroller. This function requires a hardware modification to bring the I/O signal out for the user. A description of that hardware modification can be found in the version "2.6 (0B)" section of this page: http://www.avt-hq.com/852_asm.htm

There is one command for this function.

72 5A 0x: CAN Digital Output function status query
- x: channel: 0, 4

76 5A 0x 0r rr st vv: CAN Digital Output function setup
- x: channel: 0, 4
- rrr: 11-bit ID
- s: 0 = if bit-wise mask and match
  1 = if byte wise match only
- t: byte offset into data field, 0 to 7
- vv: mask and match value

78 5A 0x 0r rr rr rr st vv: CAN Digital Output function setup
- x: channel: 0, 4
- rrrrrrr: 29-bit ID
- s: 0 = if bit-wise mask and match
  1 = if byte wise match only
- t: byte offset into data field, 0 to 7
- vv: mask and match value

The logic for the digital output function is (for mask and match):

if the CAN ID matches
if (byte AND mask) = = mask
then output = 1, high
else output = 0, low

The logic for the digital output function is (for match only):

if the CAN ID matches
if byte = = match
then output = 1, high
else output = 0, low

8. LIN1 Operations – in CAN mode

LIN1 is only available when in CAN mode of operation.
LIN1 operation is completely independent of all other channels.
LIN1 operation is controlled by the 52 69 xx command.

LIN1 supports LIN revisions: 1.2, 1.3, 2.0, 2.1, 2.2A.

To use LIN1 mode, enter CAN mode using the $E1 99 command. The following responses will be received by the host:

$91 10 indicates the AVT-85x 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 29 [optional] indicates that LIN0 mode of operation is active.
$91 19 indicates that LIN1 mode of operation is active.

LIN1 mode uses the AVT-85x K-line for communications.

8.1 Shunt JP2

Shunt JP2 only exists on revision "B1" boards. It was removed from later board revisions.

AVT-853 revision "B1" boards only: shunt JP2 connects / disconnects the LIN1 bus from pin #7 of P3 (the DA-15P network connector).

8.2 Communications

Unless otherwise commanded, the AVT-85x will passively receive all messages from the LIN1 bus.

The AVT-85x is capable of communicating on (or transmitting to) the LIN1 bus as a Master without data, as a Master with data, or as a Slave with data.

8.2.1 Message Length

When LIN messages are received from the network, 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. This is know as "ID byte processing".

LIN protocol specification revision 2.0 and later 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.

The 5x 28 command can be used to enable or disable LIN message processing using the ID byte. The default condition is this function is enabled.

When ID byte processing is enabled [default] 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.

LIN revision 1.3 and earlier use the Classic checksum method.

LIN revision 2.0 and later use the Enhanced checksum.

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 with data, then the message is an ID byte only message. The default state is that the AVT-85x will throw out an ID byte only message and not tell the host.

The 5x 66 command selects whether or not the AVT-85x informs the host that an ID byte only was received.

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-85x 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-85x 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-85x unit will then transmit those messages, at the defined interval, without any operator intervention.

The AVT-85x will not generate a transmit ack when a periodic message is transmitted, unless the transmit acknowledgement or echo function is enabled (5x 6F command).

8.3.1 Modes of Operation

LIN periodic messages are defined as either Master or Slave messages (specified by bit0 of the fifth byte in the 7x 18 periodic message setup command).

A periodic message designated as Master can operate as Type1 or Type2 (using the 7x 1A command). These modes are described in Sections 8.3.4 and 8.3.5, below.

A periodic message designated as Slave can operate as Type1 or Type2. It may also be enabled as a slave response message. These modes are described in Sections 8.3.4, 8.3.5, and 8.3.6, below.

The 7x 1A enable/disable command has four modes for each periodic message:

mode 0:	disabled.
mode 1:	enabled for operation as Type1 or Type2 periodic message.
mode 2:	enabled for operation as a Slave response message only.
mode 3:	enabled for operation as both a Type1 or Type2 and Slave response message.

8.3.2 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.

8.3.3 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
2.56 msec

8.3.4 Type1 Periodic Message

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.

A periodic message designated as a Master will be queued for transmission when its timer expires. It will be transmitted as soon as possible after that.

A periodic message designated as a Slave will be que for transmission when its timer expires. However, it will not be transmitted until a matching ID byte is received from the LIN bus. The message will stay queued until then and thus prevent other periodic messages from being transmitted.

8.3.4.1 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 sec
   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 sec
   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

8.3.5 Type2 Periodic Message

Type2 periodic messages are transmitted in sequence, within the group.
(There is only one group in LIN mode.)

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 set up.
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.

A periodic message designated as a Master will be queued for transmission when the timer expires. It will be transmitted as soon as possible after that.

A periodic message designated as a Slave will be qued for transmission when its timer expires. However, it will not be transmitted until a matching ID byte is received from the LIN bus. The message will stay queued until then and thus prevent other periodic messages from being transmitted.

8.3.5.1 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

8.3.6 Slave Response Message

This mode of operation only applies to periodic messages that are designated as slave when setup.

If a slave response message is enabled for Type1 or Type2 operations, it operates as described in Sections 8.3.4 and 8.3.5, above.

When a periodic message is enabled for slave response message operation (the 7x 1A command) it operates independently of a timer. Every time an ID byte is received from the LIN bus, all periodic messages are searched. If a periodic message is enabled for Slave Response, and if its ID byte matches that just received from the LIN bus, then that message is immediately transmitted into the data field of the LIN frame in progress.

This will happen without host computer intervention. The host will not be informed that the message has been transmitted (by default). If the host/user wants to know when a slave response message has been transmitted, the 5x 6F command can be used to send one of two possible responses to the host.

Transmit acknowledgement (of the form):

03 05 xx yy

03 – is the packet header byte
05 – channel 5, LIN
xx – LIN receive status byte
yy – ID of the LIN message

Transmit echo (of the form):

0x 05 xx yy rr ss tt …

03 – is the packet header byte
05 – channel 5, LIN
xx – LIN receive status byte
yy – ID of the LIN message
rr ss tt … the data field of the LIN message

8.3.7 Periodic Message Commands

All commands are listed in Section 16. A brief summary 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 Periodic Message Special Function

There is one special function available for all LIN periodic messages operating in Type1 and Type2 modes. Not available for Slave Response messages. The special function was developed specifically at customer request. It is described below.

If this function is enabled or not, the data field of a periodic message can be changed 'on the fly'. You do NOT need to disable the message or the function to change anything.

8.4.1 LIN Frame Data Definition

Each LIN frame can contain up to 8 data bytes.

In the following discussion, Data0 is the first data byte in the LIN frame (after the ID byte). Likewise Data7 is the last byte of the LIN frame.

Within a byte, the bits are numbered from 0 (least significant bit) to 7 (most significant).

8.5 LIN1 Digital Output Function

This function was custom developed for a customer.

When enabled, a digital output is driven high or low depending on the outcome of a test.

There are two possible tests:
Mask and Match.
Match only.

Note that the digital output is not buffered. It is driven directly from an I/O pin of the AVT-853 microcontroller. This function requires a hardware modification to bring the I/O signal out for the user. A description of that hardware modification can be found in the version "2.6 (0B)" section of this page: http://www.avt-hq.com/852_asm.htm

8.5.1 LIN1 Digital Output Function Command

Details of the LIN1 Digital Output command.

51 89:	LIN1 (channel 5) Digital Output status query.
52 89 00:	disable the function.
52 89 rx:	enable the function. r = 0: Mask and Match operation. r = 1: Match Only operation. x = 1: Enable the function, send the LIN1 message to the user. x = 2: Enable the function, discard the LIN1 message.
51 8A:	LIN1 (channel 5) Digital Output setup query
54 8A rr 0x yy:	function setup rr: LIN ID of message to 'look' for x: offset into the data field, zero based yy: mask value

Mask and Match operation.
if ((byte AND mask) = = mask)
then output = 1, high
else output = 0, low

Match Only operation.
if (byte = = mask)
then output = 1, high
else output = 0, low

8.6 LIN Special Function #1 (LSF1)

The LIN Special Function #1 (LSF1).
(Also known as the "Counter0" function.)

When enabled, for a specific periodic message, for a specific LIN channel; the following action takes place each time the message is queued for transmission.

Bits 1:0 of Data2 are incremented. In other words, the lowest two bits of Data2 form a rolling counter that increments and rolls over. For example: 00, 01, 10, 11, 00 …

8.6.1 LIN Special Function #1 (LSF1) Command

Refer to the 7x 49 command in Section 16 for detailed information about the command format. Remember, the function is by LIN channel and periodic message number.

8.7 LIN Special Function #2

The LIN Special Function #2 (LSF2).

LIN data field is byte0 to byte7.

Data byte 1, upper nibble, is a 4-bit counter.

Data byte 2, is the J1850 CRC of bytes 0 to 1.

8.7.1 LIN Special Function #2 Command

Details of the LIN Special Function #2 command.

53 7F 0x 0y: LSF2 status query
- x: channel 5 or 7
- y: periodic message number $0 to $A

54 7F 0x 0y 0z: Set LSF2 status
- x: channel 5 or 7
- y: periodic message number $0 to $A
- z: 0 = function disabled
  1 = function enabled.

8.8 LIN1 Special Function #3

The LIN1 Special Function #1(LSF3).

Note: Only available for LIN1 (channel 5).

The user specifies an expected LIN byte sequence.

The user specifies whether each byte in the sequence is receive or transmit (relative to the AVT-85x interface).

The enable command takes effect as soon as the LIN bus is idle.

When the function is enabled, all LIN1 (channel 5) functions are suspended. This includes transmit, receive, periodic, and slave periodic functions.

The function commences as soon as a break is received. A synch byte is then expected.

If the watchdog timer expires while waiting for the synch byte, the watchdog is disabled, and the function goes back to waiting for the break symbol.

After the synch byte is received, every byte received from the LIN bus is compared to the expected value from the user defined sequence. Bytes designated to be transmitted are done so as specified in the sequence.

If anything happens that is not exactly as specified, the function aborts and the "21 92" error response is sent to the control computer.

If the function completes correctly, the success report "52 8B F0" is sent to the control computer.

The function watchdog timer resets every time a byte is received from the network.

If the watchdog timer expires, the function aborts and the "21 92" error response is sent to the control computer.

8.8.1 LIN Special Function #3 Command

Details of the LIN Special Function #3 commands.

51 8B:	LSF3 disable/enable status query
52 8B 00:	disable the function
52 8B 01:	enable the function
51 8C:	LSF3 byte sequence query
5x 8C rr ss ... :	rr ss ... is the user specified byte sequence
51 8D:	LSF3 rcv/xmt map query
53 8D rr ss:	bit map specifying if each byte is receive or transmit. bit0 is the first byte on the network (after the synch byte).
51 8E:	LSF3 watchdog reset value query
52 8E rr:	set the watchdog reset value to 'rr' milliseconds (default = 0x14, 20 milliseconds)

8.9 LIN Special Function #4 (LSF4)

The LIN Special Function #4 (LSF4).
(Also known as the “CounterX” function.)

Can be enabled for a specified LIN channel and specified periodic message.

For that periodic message, the user can specify the size of the rolling counter (1 to 4 bits), the byte location in the LIN frame (data byte 0 to 7) and the bit location (0 to 7) of the counter least significant bit (lsb).

Each time the periodic message is queued for transmission, the counter is updated and placed into the message at the specified location.

8.9.1 LIN Special Function #4 (LSF4) Command

Refer to the 5x 95 command in Section 16 for detailed information about the command format. Remember, the function is by LIN channel and periodic message number.

8.10 Short LIN1 (SLIN) Special Function

The “Short LIN1” (SLIN) special function only applies to LIN1, channel 5, pin #7 of the D15 Network connector.

When this function is enabled, all communications through channel #5 are transmitted using the following format:

Break symbol (13 to 14 bit times).
13 data bytes (are expected, but the AVT-85x does not enforce this requirement)
1 checksum byte

Each byte has the following format:

2400 baud
1 start bit
8 data bits
1 odd parity bit
1 stop bit

The checksum byte is computed using the classic LIN checksum algorithm.

NOTE

This is not LIN.
The LIN1 channel is used, but the message format is changed considerably from LIN.
For transmit commands and received responses an ID byte placeholder exists.
It should always be set to $00.
The ID byte is never transmitted.
The ID byte is not received.

The ‘5x 90’ command has been added to disable/enable this function.
Refer to Section 16.0 for command specifics.

When the function is enabled (using the ’52 90 01’ command), the AVT-85x interface sets the following parameters:

baud rate is changed to: 2400
byte format is changed to: 1 start, 8 data, 1 odd, 1 stop
classic LIN checksum algorithm is enabled
maximum frame time is set to 80 msec
receive buffer expiration time is set to 8 msec
ID byte processing is disabled

Periodic message $0A has been modified to accept a data field of 13 bytes.
The ‘5x 91’ command has been added to query/set the long data field of periodic message $0A.
Refer to Section 16.0 for command specifics.

8.10.1 Short LIN1 (SLIN) Set-Up Examples

Examples of setting up the Short LIN1 Special Function.

The AVT-85x is in CAN mode.

Enable LIN1 only:
52 69 01

Enable Short LINhort mode:
52 90 01

Transmit a message with 13 (decimal) data bytes:
11 10 05 01 00 01 02 03 04 05 06 07 08 09 10 11 12 13

where

11 - is the extended format header
10 - count of bytes to follow
05 - channel 5, LIN1
01 - transmit as master

00 - id (should be 00, but is not transmitted)
01 02 ... 12 13 - thirteen (decimal) data bytes of the message

Setup up periodic message $0A.
(note that the data is not stored with this command)
75 18 0A 05 01 00

Write the data to periodic message $0A
5E 91 01 02 03 04 05 06 07 08 09 10 11 12 13

Note:
You can change the data on-the-fly.
You do NOT need to disable the message.

Set periodic message $0A for every 500 msec
(assumed that the master clock is running at 98.30 msec ticks)
74 1B 05 0A 05

Enable periodic message $0A
74 1A 05 0A 01

8.11 ABIC Support

LIN1 supports communications with an ABIC module.

Refer to the transmit command explanation in Section 16 for information on how to format a transmit command for an ABIC module.

Refer to Section 16.1 for information on how an ABIC response is formatted.

8.12 Commands and Responses

Refer to Section 16 for a complete list of LIN1 mode commands and Section 16.1 for responses.

9. LIN0 operations – in CAN mode

LIN0 is only available when in CAN mode of operation.
LIN0 operation is completely independent of all other channels.
LIN0 operation is controlled by the 52 69 xx command.

LIN0 supports LIN revisions: 1.2, 1.3, 2.0, 2.1, 2.2A.

To use LIN0 mode, enter CAN mode using the $E1 99 command. The following responses will be received by the host:

$91 10 indicates the AVT-85x 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 29 [optional] indicates that LIN0 mode of operation is active.
$91 19 indicates that LIN1 mode of operation is active.

The $91 29 response indicates that LIN0 exists and is enabled.

9.1 LIN0 Operations Notes

LIN0 operations are nearly identical to LIN1 described in Section 8.

LIN0 does NOT support ABIC operations.

LIN0 is channel 7.

9.2 Commands and Responses

Refer to Section 16 for a complete list of LIN0 mode commands.

10. KWP operations – in CAN mode

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, LIN1 mode is disabled.

The secondary KWP mode of operations, while in CAN mode, is almost identical to KWP Stand-Alone operations. Refer to Section 11.

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

To enable secondary KWP mode, use the 52 69 02 or 52 69 06 command. The responses 62 69 0x and 91 0F will be issued to indicate that the command was processed and that KWP mode is now operational.

10.1 Shunt JP2

Shunt JP2 only exists on revision "B1" boards. It was removed from later board revisions.

AVT-853 revision "B1" boards only: shunt JP2 connects / disconnects the K-line from pin #7 of P3 (the DA-15P network connector).

10.2 Communications

Unless otherwise commanded, the AVT-85x will passively receive all messages from the K-line.

10.3 Operation Commands

Refer to Section 16 for all KWP (while in CAN) mode commands.

10.3.1 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 KWP as secondary mode
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
```

10.3.2 Time Stamp

Time stamps for both the transmit ack and received messages can be disabled or enabled.

The 52 08 0y command format will affect all channels (CAN, KWP, LIN0).

The 53 08 06 0y command format will only affect KWP channel 6.

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 time stamp is a 16-bit free running counter with 1 millisecond resolution. The time stamp rolls over at $FFFF.

10.3.2.1 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.
```

10.3.2.2 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.

10.3.3 Fast Transmit

During so-called normal transmission of a message onto the K-line, the AVT-85x 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.

Due to internal processing, even when this value is set to zero, the AVT-85x unit still inserts a small delay between transmitted bytes. For some applications this (small) delay may be unacceptable.

The 52 6C 01 fast transmit command minimizes the delay between transmitted bytes to a much smaller amount of time.

10.4 Periodic Message Support

When KWP mode is active, the AVT-85x 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-85x unit will then transmit those messages, at the defined interval, without any operator intervention.

The AVT-85x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01, or 53 06 06 01) is enabled.

10.4.1 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.

10.4.2 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
2.56 msec

10.4.3 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.

10.4.3.1 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.

1. ; KWP mode is only available in CAN mode
   ; Enter CAN mode
   E1 99
2. ; Enable KWP operations
   52 69 02
3. ; Set the master timer to 98.30 msec
   52 63 01
4. ; Define periodic message #01.
   ; the message is: 25 68 6A F1 3F
   78 18 01 06 25 68 6A F1 3F
5. ; Set periodic message #01 for an interval count of 10, actual interval = 0.983 sec
   74 1B 06 01 0A
6. ; Enable periodic message #01
   74 1A 06 01 01
7. ; Note that nothing will be transmitted until the group control is set to Type1
8. ; Define periodic message #03.
   ; the message is: 37 68 6A F1 01
   78 18 03 06 37 68 6A F1 01
9. ; Set periodic message #03 for an interval count of 5, actual interval = 0.4915 sec
   74 1B 06 03 05

10. ; Enable periodic message #03
    74 1A 06 03 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 06 01 01

10.4.4 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.

10.4.4.1 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.

1. ; KWP mode is only available in CAN mode
   ; Enter CAN mode
   E1 99

2. ; Enable KWP operations
   52 69 02

3. ; Set the master timer to 98.30 msec
   52 63 01

4. ; Define periodic message #02
   79 18 02 06 AB 48 6B 10 41 0D

5. ; Enable periodic message #02
   74 1A 06 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 06 BC 48 6B 10 41 0D

8. ; Enable periodic message #04
   74 1A 06 04 01

9. ; Define periodic message #07
   7A 18 07 06 D4 48 6B 10 41 0D 67

10. ; Enable periodic message #07
    74 1A 06 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 06 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 06 01 02

10.4.5 Periodic Message Commands

All commands are listed in Section 16. A brief summary 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

11. VPW Mode

Enter VPW mode with the $E1 33 command.
The report $91 07 indicates the AVT-85x has entered VPW operations.

The AVT-85x 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.

11.1 Shunt JP3

Shunt JP3 only exists on revision “B1” boards. It was removed from later board revisions.

AVT-853 revision “B1” boards only: shunt JP3 connects / disconnects the VPW bus from pin #2 of P3 (the DA-15P network connector).

11.2 Communications

J1850 VPW messages consist of a maximum of 11 bytes.

The AVT-85x handles the CRC byte. The host computer should never send it and the AVT-85x 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 4.3 and the Commands Section 17 for detailed information about messages to and from the network.

11.2.1 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-85x, 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 17.1 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 4.3 and Section 17 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 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-85x 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.

11.3.1 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.

11.3.2 Example #1

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

The host will receive this packet from the AVT-85x.

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

11.3.3 Example #2

Host sends: 52 5B F1
AVT-85x 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-85x will determine that the destination bytes do not match and will throw out the message. The host will not receive anything.

11.3.4 Example #3

Host sends: 52 5B 00
AVT-85x response: 62 5B 00

Host sends: 52 5C 20
AVT-85x 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-85x will determine that the source bytes do not match and will throw out the message. The host will not receive anything.

11.3.5 Example #4

Host sends: 52 5B F2
AVT-85x response: 62 5B F2

Host sends: 52 5C 10
AVT-85x 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-85x.

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

11.4 Mask / Match / Respond Function

The 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.

11.4.1 Operational Overview

This function operates on all network messages and includes both received and transmitted messages (relative to the AVT-85x interface and the VPW network). The message filtering function, described in the previous Section, does NOT affect the operation of this function.

A VPW network message passes through the AVT-85x. 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.

11.4.2 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.

11.4.3 Example

Assume you want the AVT-85x 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-85x.

; 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 is an exact match, the command C1 01 will be issued that will cause the AVT-85x to switch to 4X mode. The AVT-85x 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.

11.5 Periodic Message Support

In VPW mode, the AVT-85x 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-85x 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-85x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01) is enabled.

11.5.1 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.

11.5.2 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
- 2.56 msec

11.5.3 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.

11.5.3.1 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 sec
   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 sec
   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

11.5.4 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.

11.5.4.1 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

11.5.5 Periodic Message Commands

All commands are listed in Section 17. A brief summary 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

11.6 Block Transmit Example

The AVT-85x 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-85x:
12 10 10

Followed immediately by the 4112 message bytes.

The AVT-85x will wait for a maximum of 3 seconds for the entire message to be received from the host. The message is buffered before being sent to the network.

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.

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.

11.7 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-85x:
12 08 01

The very next byte is the receive status byte.

The 2048 message bytes follow immediately.

The AVT-85x buffers a message from the network before sending it to the host.

The maximum size message that the AVT-85x can receive is 4112 bytes. If a larger message is received, the AVT-85x will generate an error message to inform the host of the actual size of the message received. The AVT-85x 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-85x:
23 53 xx yy
xx yy = actual count of block

Response received from AVT-85x:
12 10 01

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

The first 4112 message bytes follow immediately.

12. KWP Stand Alone Mode

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

To enter KWP Stand Alone mode, use the $E1 DD command.
The report $91 0F indicates the AVT-85x has entered KWP operations.

This mode is known and referred to as KWP mode in this document and for the AVT-85x 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 LIN1 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-85x only supports the K-line. It does not support the L-line.

The AVT-85x 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.]

12.1 Shunt JP2

Shunt JP2 only exists on revision “B1” boards. It was removed from later board revisions.

AVT-853 revision “B1” boards only: shunt JP2 connects / disconnects the K-line from pin #1 of P3 (the DA-15P network connector).

12.2 Communications

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

The AVT-85x is capable of receiving K-line network messages up to the full length of 259 bytes. The AVT-85x 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-85x is capable of transmitting K-line network messages up to the full length of 259 bytes. The AVT-85x 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 4.3 18, and 18.1 for detailed information about the format of messages to and from the network.

12.2.1 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-85x, 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 17.1 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 Sections 4.3 and 18 for detailed information.

12.2.2 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.

12.2.2.1 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.

12.2.2.2 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.

12.3 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-85x 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-85x 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.

12.3.1 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-85x will respond.

If the initialization attempt was successful, the AVT-85x 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-85x will respond with a 22 54 xx error code and then the 71 00 initialization attempt failure report.
A complete listing of xx error codes are in the KWP response, Section 18.1.

12.3.2 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-85x computes and appends the checksum.

If the initialization attempt was successful the AVT-85x 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-85x will likely respond with a 22 54 xx error code and then the 71 00 initialization attempt failure report.
A complete listing of xx error codes are in the KWP response, Section 18.1.

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-85x will not respond with an error code. Note that the AVT-85x unit will respond with the 71 11 success report - this only indicates that the AVT-85x unit was able to generate the Fast Initialization sequence but does not indicate anything about the downstream module.

12.4 Mask / Match / Respond Function

The 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.

This function is also available for KWP operations while in CAN mode.

12.4.1 Operational Overview

This function operates on all network messages and includes both received and transmitted messages (relative to the AVT-85x interface and the network).

A network message passes through the AVT-85x. 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. It can be a configuration command or a message for transmission.

12.4.2 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.

12.4.3 Example

Assume you want the AVT-85x unit to transmit a message as soon as the message $AA $BB $CC $DD $EE passes by. You want an exact message match, no message ambiguity allowed.

To set-up and enable the function, you would issue the following commands to the AVT-85x.

; define the mask
56 75 FF FF FF FF FF

; define the match
56 76 AA BB CC DD EE

; define the response
05 11 22 33 44 55

; 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 is an exact match, the transmit command (defined above) will be issued that will cause the AVT-85x to transmit that message to the network.

12.5 Periodic Message Support

In KWP mode, the AVT-85x 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-85x 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-85x will not generate a transmit ack when a periodic message is transmitted, unless transmit forwarding (52 06 01) is enabled.

12.5.1 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.

12.5.2 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
2.56 msec

12.5.3 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.

12.5.3.1 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 KWP mode
   E1 DD
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 sec
   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 sec
   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

12.5.4 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.

12.5.4.1 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 KWP mode
   E1 DD
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

12.5.5 Periodic Message Commands

All commands are listed in Section 18. A brief summary 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

13. AVT-85x Field reFLASHing

All AVT-85x units can be reFLASHed in the field to permit updating the unit operating firmware.

13.1 AVT-85x reFLASHing - AVT Provided Application

AVT can provide stand alone applications for a host PC to reFLASH an AVT-852 or 853 unit.

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

The reFLASH applications are called:

- AVT-852_reFLASH3.exe
- AVT-853_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-85x firmware is likely to be released with the following naming convention.
The XX in the name will be the version number.

- AVT-85x_vXX.pf3

14. 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-85x (a form of software reset).

14.1 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.

14.2 Other Responses

2: Error reports.
- 21 70: Backdoor is disabled.

21 71: Backdoor access attempt failed.

21 84: Command buffer mode fault.

2: Status reports.

91 27: Idle state. Waiting for mode switch command.
92 04 xx: Firmware version report. Version is xx.
93 04 xx yy: Firmware version report. Version is xx yy.
93 28 0x yz: Model number report. xyz is the model number.

15. CAN Mode – Operational Notes

CAN0	is	channel 0.
CAN4	is	channel 4.
LIN1	is	channel 5.
KWP	is	channel 6.
LIN0	is	channel 7.

LIN1 and KWP are mutually exclusive modes. They share the same physical media (wire).
The only allowed states are: Both disabled.
LIN1 enabled (only).
KWP enabled (only).

Refer to the 5x 69 secondary mode command.

All other modes can be operated independently and simultaneously.

Some commands are not associated to a specific channel.

For those commands that are applicable to one or more channels, the applicable channels are listed above the command description.

16. CAN Mode – Commands

High nibble, shown at left, bits b7 - b4 indicates the Command type.
Low nibble, bits b3 – b0 indicates how many bytes are to follow.

All transmit command forms are equal in ascending order.

0x	=	11 0x	=	12 00 0x
		11 xx	=	12 00 xx
				12 xx yy

0: CAN packet for transmission to the network.

CAN0, CAN4

0x qr tt vv ww zz mm nn ... :

- x: count of bytes to follow.
- q:
  - 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 = transmit message 'as-is'.
    1 = format message for ISO 15765.
- r: channel: 0, 4.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data.

0: LIN packet for transmission to the network.

LIN1, LIN0

0x 0y 0p qq rr ss ...

- x: count of bytes to follow.
- y: channel: 5, 7.
- p:
  - 0: slave.
  - 1: master.
- qq message ID.
- rr ss ... message data [optional].

0: ABIC packet for transmission to the network.

LIN1

0x 15 0p qq rr ss tt ...

- x: count of bytes to follow.
- 15: ABIC; channel: 5.
- p:
  - 0: slave.
  - 1: master.

qq: PID (protected ID)
rr: CMD (ABIC module command)
ss tt ... message data [optional]

0: KWP packet for transmission to the network.

KWP

0x 06 qq rr ss ...
x: count of bytes to follow.
6: channel: 6.
qq rr ss ... message data.

1: CAN packet for transmission to the network; alternate header formats.

CAN0, CAN4

11 xx qr tt vv ww zz mm nn ... :
xx: count of bytes to follow.
q: 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 = transmit message ‘as-is’.
1 = format message for ISO 15765.
r: channel: 0, 4.
tt vv: 11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data.

CAN0, CAN4

12 xx yy qr tt vv ww zz mm nn ... :
xx yy: count of bytes to follow.
q: 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 = transmit message ‘as-is’.
1 = reformat message for ISO 15765.
r: channel: 0, 4.
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.

1: LIN packet for transmission to the network; alternate header formats.

LIN1, LIN0

11 xx 0z 0p qq rr ss ...

xx: count of bytes to follow.
z: channel: 5, 7.
p: 0: slave.
1: master.
qq message ID.
rr ss ... message data [optional].

LIN1, LIN0

12 xx yy 0z 0p qq rr ss ...

xx yy: count of bytes to follow.
z: channel: 5, 7.
p: 0: slave.
1: master.
qq message ID.
rr ss ... message data [optional].

1: ABIC packet for transmission to the network; alternate header formats.

LIN1

11 xx 15 0p qq rr ss tt ...

xx: count of bytes to follow.
15: ABIC; channel: 5.
p: 0: slave.
1: master.
qq: PID (protected ID)
rr: CMD (ABIC module command)
ss tt ... message data [optional]

LIN1

12 xx yy 15 0p qq rr ss tt ...

xx yy: count of bytes to follow.
15: ABIC; channel: 5.
p: 0: slave.
1: master.
qq: PID (protected ID)
rr: CMD (ABIC module command)
ss tt ... message data [optional]

1: KWP packet for transmission to the network; alternate header formats.

KWP
11 xx 06 qq rr ss ...
xx: count of bytes to follow.
6: channel: 6.
qq rr ss ... message data.

KWP
12 xx yy 06 qq rr ss ...
xx yy: count of bytes to follow.
6: channel: 6.
qq rr ss ... message data.

2: Reset.
21 0A: Reset CAN0.
21 0B: Reset CAN4.

3:

4:

5: Configuration.

LIN1, KWP, LIN0
51 01: Request received checksum forwarding status.
52 01 00: Do not send received checksum to host; all channels. [Default]
52 01 01: Send received checksum to host; all channels.
53 01 0x 0y: Disable / Enable received checksum forwarding for specified channel.
x: channel: 5, 6, 7.
y: 0 disable.
1 enable.

LIN1
51 02: Receive buffer timeout status query.
52 02 xx: Receive buffer timeout set to xx milliseconds, from last received byte.
[Default = $FF = 255 msec.]

KWP
51 03: Request K-line baud rate divisor value.
53 03 xx yy: K-line bus baud rate is set by user and equal to:
25 000 000 / (16 * xxyy) [all values shown are decimal]

Example: for K-line bus baud rate = 10400;
xxyy = $00 96 (hex) = 150 (decimal)

LIN1, KWP, LIN0

51 06: Request transmit message echo status.
52 06 00: Do not echo transmitted messages; all channels. [Default]
52 06 01: Echo transmitted messages; all channels.
53 06 0x 0y: Disable / Enable transmit echo for specified channel.
x: channel: 5, 6, 7.
y: 0 disable.
1 enable.

CAN0, CAN4, LIN1, KWP, LIN0

51 08: Time stamp status query.
52 08 00: Disable time stamps; all channels. [Default]
52 08 01: Enable time stamps; all channels.
For CAN0 and CAN4 the time stamp interval is the inverse of the baud rate for that channel.
For LIN1, KWP, and LIN0, the time stamp is 1 millisecond resolution.
52 08 02: Enable time stamps.
For all channels the time stamp is 1 millisecond resolution.
53 08 0x 0y: Disable / Enable time stamp for specified channel.
x: channel: 0, 4, 5, 6, 7.
y: 0 disable.
1 enable (as defined above).
2 enable (as defined above).

KWP

51 13: Query for 5-baud address.
52 13 xx: Set 5-baud address to $xx. [Default = $33]

LIN1, KWP, LIN0

51 19: Query for transmit checksum status.
52 19 00: Do not append checksum to transmitted message; all channels.
52 19 01: Append a checksum to transmitted message; all channels. [Default]
53 19 0x 0y: Disable / Enable transmitted checksum for specified channel.
x: channel: 5, 6, 7.
y: 0 disable.
1 enable.

LIN1, KWP, LIN0

51 24: Network messages query.

52 24 00: Do not receive any network messages; all channels.
52 24 01: Receive network messages. [Default]
53 24 0x 0y: Disable / Enable receive network messages for specified channel.
x: channel: 5, 6, 7.
y: 0 disable.
1 enable.

KWP

51 27: Query for P4 time; transmit message inter-byte time.
52 27 xx: Set P4 time to xx where xx is in milliseconds
[Default = $05]

LIN1

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 = $02]

LIN1, LIN0

51 28: Query for receive ID byte processing status.
52 28 00: Disable receive ID byte processing; all channels.
Use the maximum frame time to determine the end of a received message.
52 28 01: Enable receive ID byte processing; all channels. [Default]
Use the received frame ID byte to determine expected message length.
53 28 0x 0y: Disable / Enable ID byte processing for specified channel.
x: channel: 5, 7.
y: 0 disable.
1 enable.

KWP

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]

KWP

51 2B: Query for receive buffer expiration time.
52 2B xx: Set receive buffer expiration time to xx milliseconds. [Default = $17 = 23]

KWP

51 2C: Query for the two key bytes (key word).

KWP

52 30 yy: Send a break onto the K-line for 'yy' milliseconds.

CAN0, CAN4, LIN1, KWP, LIN0

51 40: Transmit acks query.
52 40 00: Do not send transmit acks to host; all channels.
52 40 01: Send transmit acks to host; all channels. [Default]
53 40 0x 0y: Disable / Enable transmit acks for specified channel.
x: channel: 0, 4, 5, 6, 7.
y: 0 disable.
1 enable.

KWP

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 = $012D = 301]

KWP

51 47: Query for FAST initialization low time.
52 47 xx: Set FAST initialization low time to xx milliseconds. [Default = $19 = 25]

KWP

51 48: Query for FAST initialization high time.
52 48 xx: Set FAST initialization high time to xx milliseconds. [Default = $19 = 25]

KWP

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 4B 02: Transmit checksum is XOR.

---

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

LIN1

51 50: Query for LIN1 bus baud rate.
52 50 01: LIN1 bus baud rate is 2400 baud
52 50 02: LIN1 bus baud rate is 9600 baud. [Default]
52 50 03: LIN1 bus baud rate is 19200 baud.
54 50 04 xx yy: LIN1 bus baud rate is set by user and equal to:
25 000 000 / (16 * xxyy) [all values shown are decimal]

Example: for LIN1 bus baud rate = 9600;
xxyy = $00 A3 (hex) = 163 (decimal)

LIN1

51 52: Query for maximum frame time, in milliseconds.
52 52 xx: Set maximum frame time to $xx milliseconds. [Default = $14 = 20]

KWP

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: Periodic ADC reports status query.
52 59 00: Disable periodic ADC reports. [Default]
52 59 xx: Enable periodic ADC reports.
Report interval is xx timer ticks (5x 63 command).

LIN1, LIN0

51 5A: Query for checksum method.
52 5A 00: Use LIN Classic checksum method; all channels.
52 5A 01: Use LIN revision 2.0 Enhanced checksum method; all channels.
53 5A 0x 0y: Select checksum type for specified channel.
x: channel: 5, 7.
y: 0 Classic method.
1 Enhanced method.

---

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.
06 2.56 msec.

LIN1, LIN0

51 66: Query for ID byte only message operation.
52 66 00: Suppress (discard) ID byte only messages; all channels.
52 66 01: Inform host of an ID byte only message and send the ID byte; all channels.
53 66 0x 0y: Disable / Enable ID byte only message for specified channel.
x: channel: 5, 7.
y: 0 disable.
1 enable.

AVT-853 only

51 67: Internal baud rate setting query.
52 67 01: Set internal baud rate to 19.2 kbaud.
52 67 02: Set internal baud rate to 38.4 kbaud.
52 67 04: Set internal baud rate to 57.6 kbaud.
52 67 08: Set internal baud rate to 115.2 kbaud.
52 67 FF: Set internal baud rate to 230.4 kbaud.
52 67 20: Set internal baud rate to 460.8 kbaud.
52 67 40: Set internal baud rate to 921.6 kbaud.
New setting does not take affect until reset;
either power-on reset or software reset (F1 A5).
XPort baud rate must be changed to match.

LIN1, KWP, LIN0

51 69: Secondary operational mode query.
52 69 00: Disable all secondary modes.
52 69 01: Enable LIN1 secondary operations.
52 69 02: Enable KWP secondary operations.
52 69 04: Enable LIN0 secondary operations.
52 69 05: Enable LIN1 and LIN0 secondary operations. [Default.]
52 69 06: Enable KWP and LIN0 secondary operations.

---

51 6A: Red LED blink rate query.
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.

LIN1

53 6B xx yy: Transmit a break to the network with duration = xx yy microseconds

(time is approximate)

KWP

51 6C: Query for fast transmit status.
52 6C 00: Fast transmit disabled. [Default]
52 6C 01: Fast transmit enabled.

LIN1, LIN0

51 6F: Query for slave response message ack/echo status.
52 6F 00: Disable; all channels.
52 6F 01: Enable and ack with ID; all channels.
52 6F 02: Enable and echo complete transmitted message; all channels.
53 6F 0x 0y: Disable / Enable for specified channel.

- x: channel: 5, 7.
- y: 0 disable.
  1 enable as described above.
  2 enable as described above.

MMR = Mask/Match/Respond
Refer to Section 12.4

KWP

51 75: MMR function mask query.
5x 75 yy zz … MMR function mask definition.
x – count of bytes to follow
yy zz … mask bytes

KWP

51 76: MMR function match query.
5x 76 yy zz … MMR function match definition.
x – count of bytes to follow
yy zz … match bytes

KWP

51 77: MMR function respond query.
5x 77 yy zz … MMR function respond definition.
x – count of bytes to follow
yy zz … command bytes

KWP

51 78: MMR function status query.
52 78 00: Disable MMR function.
52 78 01: Enable MMR function.

LIN1

51 7D: Status query.
52 7D 00: Suppress 'noise' only error responses (25 86 00 00 00 x4). [Default]
52 7D 01: Do not suppress any '25 86 rr ss tt vv' error responses.

LIN1, LIN0

53 7F 0x 0y: LIN Special Function #2 (LSF2) status query.
x: channel: 5, 7.
y: periodic message number $0 to $A.

54 7F 0x 0y 0v: LIN Special Function #2 (LSF2) disable / enable command.
x: channel: 5, 7.
y: periodic message number $0 to $A.
v: 0 = disable.
1 = enable.

LIN0

51 80: Receive buffer timeout status query.
52 80 xx: Receive buffer timeout set to xx milliseconds, based on last received byte.
[Default = $FF = 255 milliseconds.]

LIN0

51 81: Query for LIN0 bus baud rate.
52 81 01: LIN0 bus baud rate is 2400 baud
52 81 02: LIN0 bus baud rate is 9600 baud. [Default]
52 81 03: LIN0 bus baud rate is 19200 baud.
54 81 04 xx yy: LIN0 bus baud rate is set by user and equal to:
25 000 000 / (16 * xxyy) [all values shown are decimal]
Example: for LIN0 bus baud rate = 9600;
xxyy = $00 A3 (hex) = 163 (decimal)

LIN0

51 82: Query for maximum frame time, in milliseconds
52 52 xx: Set maximum frame time to $xx milliseconds. [Default = $14 = 20]

LIN0

53 83 xx yy: Transmit a break to the network with duration = xx yy microseconds
(time is approximate)

LIN0

51 84: Query for P4 time; transmit message inter-byte time
52 84 xx: Set P4 time to xx where xx is in increments of approximately 30 microseconds. [Default = $02]

LIN1

51 89: LIN1 (channel 5) Digital Output status query.
52 89 00: disable the function.
52 89 rx: enable the function
r = 0: Mask and Match operation.
r = 1: Match Only operation.
x = 1: Enable the function, send the LIN1 message to the user.
x = 2: Enable the function, discard the LIN1 message.

LIN1

51 8A: LIN1 (channel 5) Digital Output setup query.
54 8A rr 0x yy: Function setup.
rr: LIN ID of message to 'look' for.
x: offset into the data field, zero based.
yy: mask value.

LIN1

51 8B: LIN1 Special Function #3 (LSF3) disable/enable status query.
52 8B 00: Disable the function.
52 8B 01: Enable the function.

LIN1

51 8C: LIN1 Special Function #3 (LSF3) byte sequence query.
5x 8C rr ss ... : rr ss ... the specified byte sequence.

LIN1

51 8D: LIN1 Special Function #3 (LSF3) rcv/xmt map query.
53 8D rr ss: Bit map specifying if each byte is receive or transmit.
Bit0 is the first byte on the network (after the synch byte).

LIN1

51 8E: LIN1 Special Function #3 (LSF3) watchdog reset query.
52 8E rr: Set the watchdog reset value to 'rr' milliseconds.
(default = 0x14, 20 milliseconds).

LIN1

51 90: Short LIN1 (SLIN) function status query.
52 90 00: Disable Short LIN1 function.
52 90 01: Enable Short LIN1 function.

LIN1

51 91: Query for periodic message $0A data field.
5x 91 rr ss tt …: Set data field of periodic message $0A.

LIN1, LIN0

53 95 0y 0z: LIN Special Function #4 (LSF4, Counter X) status query.
54 95 0y 0z 0w: Disable/enable command.
57 95 0y 0z 0w 0r 0s 0t: LIN Special Function #4 (LSF4, Counter X) set-up command.
y: channel: 5, 7.
z: 0 = disable.
1 = enable.
r: counter size; 1 to 4 bits.
s: byte location; 0 to 7.
t: bit location; 0 to 7 (of counter lsb).

6: Initialization

KWP

---

61 11: CARB mode 5-baud initialization.

KWP

6x 13 yy zz ...: FAST initialization.
x: count of bytes to follow.
yy zz ... : start communications message.

7: CAN configuration.

CAN0, CAN4

71 0A: Request baud rate settings for both CAN channels.
72 0A 0y: Request baud rate setting for channel CANy.
y: channel: 0, 4.
73 0A 0y zz: Set baud rate for CAN channel.
y: channel: 0, 4.

zz:	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

CAN0, CAN4

71 0B: Request Bit Timing Register (BTR) settings for both CAN channels.

72 0B 0y: Request BTR settings for channel CANy.
- y: channel: 0, 4.

74 0B 0y rr ss: Set Bit Timing Registers (BTR) for channel CANy.
- y: channel: 0, 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; which is 8.0000 MHz.

CAN0, CAN4, LIN1, KWP, LIN0

71 0C: Periodic message group operational control, status query.
All channels, all groups, are reported.

72 0C 0y: Query for periodic message status of all groups.
- y: channel: 0, 4, 5, 6, 7.

All groups are reported.

73 0C 0y 0z: Query for status of one periodic message group.
- y: channel: 0, 4, 5, 6, 7.
- z: group, 1 or 2

74 0C 0y 0v 0w: Periodic message group operational control command.
- y: channel: 0, 4, 5, 6, 7.
- v: group, 1 or 2
- w: mode:
  - 0: Disabled.
  - 1: Type1 enabled.
  - 2: Type2 enabled.

CAN0, CAN4

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

72 0E 0y: Outbound flow control separation time query.
- y: channel: 0, 4.

73 0E 0y zz: Set outbound flow control separation time.
- y: channel: 0, 4.
- zz: separation time [Default = $0A = 10 msec.]

(There are rules in ISO 15765 for setting this parameter.)

CAN0, CAN4

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

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

74 0F rs tt uu: Set outbound flow control ID, 11-bit; no AE byte.
- r: 0 = 11-bit ID.
- s: channel: 0, 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: channel: 0, 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: channel: 0, 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: channel: 0, 4.
- tt uu vv ww: 29-bit ID.
- ae: AE byte.

CAN0, CAN4

71 11: Operational mode status query for all CAN channels.

72 11 0y: Operational mode status query for CAN channel.
- y: channel: 0, 4.

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

CAN4

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

72 12 0y: Set SWC transceiver mode. CAN4 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 channel number and message number fields reversed as compared to all other commands.

CAN0, CAN4

73 18 vv 0y: Periodic message setup query.
- vv: Message number, $01 to $58. (when CAN0 is specified)
- y: channel: 0, 4.

7x 18 vv yz tt vv ww zz mm nn ... Periodic message setup command.
- vv: Message number, $01 to $58. (when CAN0 is specified)
- y:
  - 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
- z: channel: 0, 4.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data field.

LIN1, LIN0

73 18 vv 0y: Periodic message setup query.
- vv: Message number, $01 to $0A.
- y: channel: 5, 7.

7x 18 vv 0y 0z ww pp qq rr ... Periodic message setup command.
- vv: Message number, $01 to $0A.
- y: channel: 5, 7.
- z: 0: slave message.
  - 1: master message.
- ww: message ID.
- pp qq rr ...: data field.

KWP

73 18 vv 06: Periodic message setup query.
- vv: Message number, $01 to $0A.
- 6: channel: 6.

7x 18 vv 06 pp qq rr ... Periodic message setup command.
- vv: Message number, $01 to $0A.
- 6: channel: 6.
- pp qq rr ...: data field.

CAN0, CAN4, LIN1, KWP, LIN0

73 1A 0y zz: Periodic message disable/enable status query.
- y: channel: 0, 4, 5, 6, 7.
- zz: Message number, $01 to $58 or ($0A).

74 1A 0y zz 0v: Periodic message disable/enable command.
- y: channel: 0, 4, 5, 6, 7.
- zz: Message number, $01 to $58 or ($0A).
- v: 0 disabled.
  1 normal mode enabled.
  2 slave mode enabled.
  3 both modes enabled.

CAN0, CAN4, LIN1, KWP, LIN0

73 1B 0y zz: Periodic message interval count status query.
- y: channel: 0, 4, 5, 6, 7.
- zz: Message number, $01 to $58 or ($0A).

74 1B 0y zz vv: Periodic message interval count command.
- y: channel: 0, 4, 5, 6, 7.
- zz: Message number, $01 to $58 or ($0A).
- vv: interval count.

CAN0, CAN4, LIN1, KWP, LIN0

72 1C 0y: Disable all periodic messages of one channel.
- y: channel: 0, 4, 5, 6, 7.

72 1C EE: Disable all periodic messages, all channels.
(Note: the setup for each periodic message is not affected.)

CAN0, CAN4

71 1F: Periodic Message Pause status query; both CAN channels.

72 1F 0x: Periodic Message Pause status query.
- x: channel: 0, 4

73 1F 0x 0y: Periodic Message Pause command.
- x: channel: 0, 4
- y: 0: disabled.
  1: enabled.

CAN0, CAN4

71 26: ISO 15765 operations status query, both CAN channels.

72 26 0x: ISO 15765 operations status query.
- x: channel: 0, 4

73 26 0x 0y: ISO 15765 operations.
- x: channel: 0, 4
- y: 0: Mode0 (disabled)

1: Mode1
2: Mode2

CAN0, CAN4

71 27: Outbound message padding status query, both CAN channels.

72 27 0x: Outbound message padding status query.
- x: channel: 0, 4

73 27 0x 0y: Outbound message padding.
- x: channel: 0, 4
- y: 0: disabled.
  1: enabled.

74 27 0x 0y vv: Outbound message padding.
- x: channel: 0, 4
- y: 0: disabled.
  1: enabled.
- vv: pad byte.

CAN0, CAN4

71 29: ISO 15765 receive ID, query both CAN channels.

72 29 0y: ISO 15765 receive ID query for CANy.
- y: channel: 0, 4

7x 29 xy rr ss …: ISO 15765 receive ID set command
- 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: 0, 4
- rr ss: 11-bit receive ID
- rr ss tt vv: 29-bit receive ID

CAN0, CAN4

71 2A: Acceptance ID query for all CAN channels.

72 2A 0y: Acceptance ID query for CAN channel.
- y: channel: 0, 4

73 2A 0y 0z: Report specified acceptance ID.
- y: channel: 0, 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: channel: 0, 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).

CAN0, CAN4

71 2B: ID/Mask mode configuration query for all CAN channels.

72 2B 0y: ID/Mask mode query for CAN channel.
y: channel: 0, 4

73 2B 0y 0z: Set ID/Mask mode for CAN channel.
y: channel: 0, 4
z: 2: Two 32-bit IDs and masks.
4: Four 16-bit IDs and masks.
8: Eight 8-bit IDs and masks.

CAN0, CAN4

71 2C: Report all masks for all CAN channels.

72 2C 0x: Report all masks for CAN channel.
y: channel: 0, 4

73 2C 0x 0z: Report specified mask.
y: channel: 0, 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: channel: 0, 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).

CAN0, CAN4

73 2D 0x 0y: Acceptance ID register direct read.
x: channel: 0, 4
y: Acceptance ID register number, 0 to 7.

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

CAN0, CAN4

73 2E 0x 0y: Mask register direct read.
x: channel: 0, 4
y: Mask register number, 0 to 7.

74 2E 0x 0y zz: Mask register direct write.
x: channel: 0, 4
y: Mask register number, 0 to 7.
zz: Register value to write.

CAN0, CAN4

71 30: AE byte; disable / enable status query, both CAN channels.

72 30 0y: AE byte; disable / enable status query.
y: channel: 0, 4

73 30 0y 0z: AE byte; disable / enable command.
y: channel: 0, 4
z: 0 disable.
1 enable.

CAN0, CAN4

71 33: Query for CAN ISO 15765 receive buffer time-out value.

72 33 xx: Set CAN ISO 15765 receive buffer time-out value.
Time is in 174.8 msec increments.
Both CAN0 and CAN4 channels use this value.

CAN0, CAN4

71 34: Query for CAN ISO 15765 inbound flow control separation time default value.

72 34 xx: Set CAN ISO 15765 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.

CAN0, CAN4

71 35: ISO 15765 pacing timer, query both CAN channels.

72 35 0y: ISO 15765 pacing timer, query one CAN channel.
- y: channel: 0, 4

73 35 0y zz: Set ISO 15765 pacing timer.
- y: channel: 0, 4
- zz: pacing timer count. (Timer interval is loops.) [Default = 00]

Refer to Section 7.13.8.1 for information about the 7x 35 command.

CAN0, CAN4

71 36: ABX separation time query.

74 36 0r xx yy: ABX separation time command.
- r: 1 = millisecond count
- r: 2 = loop count. [Default]
  (loop time is about 45 microseconds)
- xx yy = count. [Default = $00 0A]

CAN0, CAN4

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).

CAN0, CAN4

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).

CAN0, CAN4

71 39: ABX data count query.
73 39 xx yy: ABX data count command.
xx yy: count of bytes to be transmitted.

CAN0, CAN4

71 3A ABX control status query, both channels.
72 3A 0y: ABX control status query for specified CAN channel.
73 3A 0y 0z: ABX control command.
r: channel: 0, 4.
s: 0 = disable / terminate operations.
s: 1 = enable / start operations.
75 3A 0y 0z mm nn: ABX control command with start address.
r: channel: 0, 4.
s: 0 = disable / terminate operations.
s: 1 = enable / start operations.
mm nn: start address; $0000 to $3FFF
77 3A 0y 0z mm nn pp qq: ABX control command with start address and byte count.
r: channel: 0, 4.
s: 0 = disable / terminate operations.
s: 1 = enable / start operations.
mm nn: start address; $0000 to $3FFF
pp qq: byte count; $0001 to $8000

CAN0, CAN4

71 3B: CAN channel activity status query, both CAN channels.
72 3B 0y: CAN channel activity status query for channel CANy.
73 3B 0y 0z: CAN channel activity command.
x: channel: 0, 4
y: 0 = disable. [Default]
y: 1 = enable.

CAN0, CAN4

71 3C: CAN channel activity query, both CAN channels.
72 3C 0y: CAN channel activity query for specified CAN channel.
y: channel: 0, 4

CAN0, CAN4

71 3F: Query for CAN pacing timer, both channels.

72 3F 0x: Query for CAN pacing timer, channel CANx.
- x: channel: 0, 4.

73 3F 0x yy: Set CAN pacing timer for CANx to value yy.
- x: channel: 0, 4.
- yy: pacing timer value. (Timer interval is 1 msec.) [Default = 00]

Refer to Section 7.8.4.5 for information about the 7x 3F command.

CAN0, CAN4

73 40 0x yy: Query for ARC function status.

74 40 0x yy 0z: Set ARC status for periodic message.
- x: channel: 0, 4.
- yy: Periodic message number, $01 to $58.
- z: 0 = disable
  1 = enable

Note: ARC = Asynchronous Rolling Counter.
Refer to Section 7.12.2

CAN0, CAN4

71 41: ATD monitor status, query both CAN channels.

72 41 0y: ATD monitor status query one CAN channel.
- y: channel: 0, 4.

73 41 0y 0z: Disable / Enable ATD monitor function for channel CANy.
- y: channel: 0, 4.
- z: 0 = disable. [Default]
  1 = enable.

CAN0, CAN4

71 42: Report and clear ATD monitor values, both CAN channels, all ATD channels.

72 42 0y: Report and clear ATD monitor values, channel CANy, all ATD channels.
- y: channel: 0, 4.

73 42 0y 0z: Report and clear ATD monitor value, channel CANy, ATD channel y.
- y: channel: 0, 4.
- Z: ATD channel: 0 to F

CAN0, CAN4

71 43: ATD monitor expected ID, query both CAN channels.
72 43 0y: ATD monitor expected ID, query one CAN channel.
y: channel: 0, 4.
7x 43 xy rr ss …: Set ATD monitor expected ID, channel CANy.
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: 0, 4.
rr ss: 11-bit receive ID
rr ss tt vv: 29-bit receive ID

CAN0, CAN4

71 44: ATD monitor expected data bytes, query both CAN channels.
72 44 0y: ATD monitor expected data bytes, query one CAN channel.
75 44 0y mm nn pp: Set ATD monitor function, expected bytes, channel CANy.
y: CAN channel: 0, 4.
mm nn pp: the expected first three bytes of ATD message.

---

71 45: Query for selected CAN4 physical layer.
72 45 01: Set CAN4 physical layer to single wire CAN.
72 45 02: Set CAN4 physical layer to 2-wire CAN.

CAN0, CAN4

~~71 46: Query for status of CAN flow control error responses, both CAN channels.~~
~~72 46 0x: Query for status of CAN flow control error responses, CAN channel x.~~
~~73 46 0x 0y: Set status of CAN flow control error responses.~~
~~x: channel: 0, 4.~~
~~y: 0 = disabled.~~
~~1 = enabled. [Default]~~
~~When ISO15765 processing is enabled, a non-zero value in the block size field of a flow control frame will result in the 22 5F 3B or 22 5F 7B error response from the AVT-85x to the host. This command will suppress those error responses.~~

CAN0, CAN4

73 47 0x yy: Query for RUP function status.
74 47 0x yy 0v: RUP command for periodic message - Mode1.
x: channel: 0, 4.

yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable
counter is byte6, CRC is byte7.

75 47 0x yy 0v wz: RUP command for periodic message - Mode2.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
2 = enable
w: 0 = do not include the CRC byte.
1 = include the CRC byte.
z: location of the counter
0 to 7 if the CRC byte is omitted.
0 to 6 if the CRC byte is included.

Note: RUP = Rolling UPper nibble.
Refer to Section 7.12.3

7x 48 mm nn … Test the J1850 CRC function.
This function is for testing only. It otherwise does nothing.

LIN1, LIN0

73 49 0y zz: LIN Special Function #1 (LSF1, Counter0) status query.
y: channel: 5, 7.
zz: Message number, $01 to $0A.

74 49 0y zz 0v: LIN Special Function #1 (LSF1, Counter0) command.
y: channel: 5, 7.
zz: Message number, $01 to $0A.
v: 0 disabled.
1 enabled.

CAN0, CAN4

73 4A 0x yy: Query for CAC function status.

74 4A 0x yy 0z: Set CAC status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
z: 0 = disable
1 = enable

75 4A 0x yy 0z rr: Set CAC status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
z: 0 = disable
1 = enable
rr: constant value used in checksum computation.

Note: CAC = Counter and Checksum.

Refer to Section 7.12.4

CAN0, CAN4

71 4C: Query for first periodic message assigned to CAN4 group 1.

72 4C yy: Set 'yy' to be the first periodic message assigned to CAN4 group 1.
[Default = $21]
Minimum = $03
Maximum = $63

CAN0, CAN4

71 4D: Query for first periodic messages assigned to CAN0 and CAN4 group 2.

72 4D 0x: Query for first periodic message assigned to CANx group 1.

73 4D 0x yy: Set first periodic message assigned to CANx group 1 to be message 'yy'.
[Default CAN0 = $11]
[Default CAN4 = $31]
CAN0 Minimum = $02
CAN0 Maximum = $62
CAN4 Minimum = $04
CAN4 Maximum = $58

CAN0, CAN4

73 4E 0x yy: Query for CIB function status.

74 4E 0x yy 0v: Set CIB status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable

76 4E 0x yy 0v 0w rr: Set CIB status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable
w: location of counter, 0 to 7.
rr: increment value.

Note: CIB = CAN Increment Byte.
Refer to Section 7.12.5

CAN0, CAN4

73 4F 0x yy: Query for RC2 function status.

74 4F 0x yy 0v: Set RC2 status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable

1 = enable
Note: RC2 = Rolling Counter 2.
Refer to Section 7.12.6

CAN0, CAN4

71 50: Query for Lost Frame counter / timer function status

73 50 xx yy: Set the Lost Frame counter and timer functions.
xx is for the counter threshold function
xx = 00: count threshold function disabled
xx = 01 to FF: count threshold set point
yy is for the timer function
yy = 00: timer function disabled
yy = 01 to FF: wait time, in milliseconds, to report lost frame count.

CAN0, CAN4

73 51 0x yy: Query for RC3 function status.
74 51 0x yy 0v: Set RC3 status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable
Note: RC3 = Rolling Counter 3.
Refer to Section 7.12.7

CAN0, CAN4

73 52 0x yy Query for RC4 function status.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
74 52 0x yy 0v Disable / Enable the RC4 function.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable
75 52 0x yy 0v rt Define RC4 function.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable.
1 = enable.
r: 0 = counter is lower nibble.
1 = counter is upper nibble.
t: location of counter, 0 to 6.
Note: RC4 = Rolling Counter 4.
Refer to Section 7.12.8

CAN0, CAN4

72 5A 0x: CAN Digital Output function set-up query
- x: channel: 0, 4

76 5A 0x 0r rr st vv: CAN Digital Output function setup
- x: channel: 0, 4
- rrr: 11-bit ID
- s: 0 = if bit-wise mask and match
  1 = if byte wise match only
- t: byte offset into data field, 0 to 7
- vv: mask and match value

78 5A 0x 0r rr rr rr st vv: CAN Digital Output function setup
- x: channel: 0, 4
- rrrrrrr: 29-bit ID
- s: 0 = if bit-wise mask and match
  1 = if byte wise match only
- t: byte offset into data field, 0 to 7
- vv: mask and match value

CAN0, CAN4

72 5B 0x: CAN Digital Output function status query.
- x: channel: 0, 4.

73 5B 0x 0y: CAN Digital Output function status command.
- x: channel: 0, 4.
- y: 0 = disable.
  1 = enable.

8:

9:

A:

B: Firmware version.

B0: Request firmware version number.

B1 01: Request firmware version and sub-version numbers.

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 (stand alone) mode.

F: Model Query and Reset

F0:	Query for model number.
F1 A5:	Restart the AVT-85x (a form of software reset).

16.1 CAN Mode - Responses

High nibble, shown in left column, bits b7 - b4 indicates the Response type.
Low nibble, bits b3 – b0 indicates how many bytes are to follow.

0: Transmit acknowledgements (if enabled).

CAN0, CAN4
02 0y 0z: Transmit ack.
- y: Channel number: 0, 4.
- z: Transmit buffer number.

CAN0, CAN4
04 jj kk 0y 0z: Transmit ack.
- jj kk: time stamp
- y: Channel number: 0, 4.
- z: Transmit buffer number.

LIN1, KWP, LIN0
02 0y pp: Transmit ack.
- y: Channel number: 5, 6, 7.
- pp: receive status byte (defined below).

LIN1, KWP, LIN0
04 jj kk 0y pp: Transmit ack.
- jj kk: time stamp
- y: Channel number: 5, 6, 7.
- pp: receive status byte (defined below).

0: CAN packet received from the network.

CAN0, CAN4
0x jj kk qr tt vv ww zz mm nn ... :
- x: count of bytes to follow.
- jj kk: time stamp [optional]
- q: 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
- r: channel: 0, 4.
- tt vv: 11-bit ID, right justified.

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

0: LIN packet received from the network.

LIN1, LIN0

0x jj kk 0y pp qq rr ss ...

x: count of bytes to follow.
jj kk: time stamp [optional]
y: channel: 5, 7.
pp: receive status byte (defined below).
qq message ID.
rr ss ... message data.

pp: received status byte
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

0: ABIC packet received from the network.

LIN1

0x jj kk 15 pp qq rr ss tt ...

x: count of bytes to follow.
jj kk: time stamp [optional]
15: ABIC; channel: 5.
pp: receive status byte (defined just below).
qq PID (protected ID)
rr: CMD (ABIC module command)
ss tt ... message bytes.

pp: received status byte
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

0: KWP packet received from the network.

KWP

0x jj kk 06 pp qq rr ss ...

x: count of bytes to follow.
jj kk: time stamp [optional]
6: channel: 6.
pp: receive status byte (defined below).
qq rr ss ... message data.

pp: received status byte
b7:
b6: From this device.
b5: Transmission successful.
b4: Lost arbitration.
b3:
b2:
b1:
b0: Checksum error.

1: CAN packet received from the network; alternate header formats.

CAN0, CAN4

11 xx jj kk xy tt vv ww zz mm nn ... :

xx: count of bytes to follow.
jj kk: 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 = transmit message 'as-is'.
1 = format message for ISO 15765.
y: channel: 0, 4.
tt vv: 11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data.

CAN0, CAN4

12 xx yy jj kk qr tt vv ww zz mm nn ... :

xx yy: count of bytes to follow.
jj kk: time stamp [optional]
q: 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 = transmit message 'as-is'.
1 = reformat message for ISO 15765.

r: channel: 0, 4.
tt vv: 11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data.

1: LIN packet received from the network; alternate header formats.

LIN1, LIN0

11 xx jj kk 0y pp qq rr ss ...

xx: count of bytes to follow.
jj kk: time stamp [optional]
y: channel: 5, 7.
pp: receive status byte (defined below).
qq message ID.
rr ss ... message data [optional].

LIN1, LIN0

12 xx yy jj kk 0z pp qq rr ss ...

xx yy: count of bytes to follow.
jj kk: time stamp [optional]
z: channel: 5, 7.
pp: receive status byte (defined below).
qq message ID.
rr ss ... message data [optional].

pp: received status byte
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

1: ABIC packet received from the network; alternate header formats.

LIN1

11 xx jj kk 15 0p qq rr ss tt ...

xx: count of bytes to follow.
jj kk: time stamp [optional]
15: ABIC; channel: 5.

pp: receive status byte (defined below).
qq PID (protected ID)
rr: CMD (ABIC module command)
ss tt ... message bytes.

LIN1

12 xx yy jj kk 15 pp qq rr ss tt ...

xx yy: count of bytes to follow.
jj kk: time stamp [optional]
15: ABIC; channel: 5.
pp: receive status byte (defined below).
qq PID (protected ID)
rr: CMD (ABIC module command)
ss tt ... message bytes.

pp: received status byte
- 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

1: KWP packet received from the network; alternate header formats.

KWP

11 xx jj kk 06 pp qq rr ss ...

xx: count of bytes to follow.
jj kk: time stamp [optional]
6: channel: 6.
pp: receive status byte (defined below).
qq rr ss ... message data.

KWP

12 xx yy jj kk 06 qq rr ss ...

xx yy: count of bytes to follow.
jj kk: time stamp [optional]
6: channel: 6.
pp: receive status byte (defined below).
qq rr ss ... message data.

pp: received status byte
- b7:
- b6: From this device.

b5: Transmission successful.
b4: Lost arbitration.
b3:
b2:
b1:
b0: Checksum error.

2: Error Responses.

21 0E: Transmit command too long.

---

22 34 xx: Command time-out.
xx: header byte of offending command.
Caused by an incomplete command or a command that
did not complete execution in allotted time. [0.5 seconds.]

---

21 35: Time out reading bytes from 12 xx yy command (less than 12 bytes).
[3 seconds.]

LIN1 (ABIC)

24 40 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

LIN1, KWP, LIN0

22 4A xx: Message of length 1 received.
xx: the one byte that was received.

KWP

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:

CAN0, CAN4, KWP

21 5B: Time out trying to send received block to host.
[3 seconds.]

LIN0

22 5C yy: LIN0 lost frame counter. Messages (frames) received from the LIN bus but thrown out because they could not be forwarded to the client.
'yy' is the count of lost frames. 'yy' is allowed to roll-over.

LIN1

22 5D yy: LIN1 lost frame counter. Messages (frames) received from the LIN bus but thrown out because they could not be forwarded to the client.
'yy' is the count of lost frames. 'yy' is allowed to roll-over.

CAN0, CAN4

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.

CAN0, CAN4

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-ISO 15765 transmit command too short, 11-bit.
07: CAN0; non-ISO 15765 transmit command too long, 11-bit.
08: CAN0; non-ISO 15765 transmit command too short, 29-bit.
09: CAN0; non-ISO 15765 transmit command too long, 29-bit.
0A: CAN0; non-ISO 15765 11/12 transmit command too long.
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-ISO 15765 transmit command too short, 11-bit.
47: CAN4; non-ISO 15765 transmit command too long, 11-bit.
48: CAN4; non-ISO 15765 transmit command too short, 29-bit.
49: CAN4; non-ISO 15765 transmit command too long, 29-bit.
4A: CAN4; non-ISO 15765 11/12 transmit command too long.
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:

CAN0

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

CAN4

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.

KWP

24 85 xx yy zz: KWP error.

xx:

b7: both lin1 and kwp active.
b6: short to ground detected.
b5: periodic message error.
b4: error decoding sci1cr1.
b3: periodic message too long.
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: transmit command processing return code error.

zz:
b7: transmit date register empty.
b6: transmit complete.
b5: receive data register full.
b4: idle.
b3: overrun.
b2: noise.
b1: framing error.
b0: parity error.

LIN1

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.
b4: no receive buffer available (break received).
b3: receive byte not equal transmit byte (master).
b2: receive buffer mode unknown 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 or too high.
b3: transmit command control byte error, not master or slave
b2: transmit command slave with no data.
b1: 0
b0: received a zero length message.

tt:
b7: 0
b6: 0
b5: periodic message too long.
b4: periodic message mode not 1 or 2.
b3: master mode error (state is unknown or invalid).
b2: receive byte not equal transmit byte (slave).
b1: slave mode error (state is unknown or invalid).
b0: master state transmit 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.

CAN0, CAN4

22 8C xx: CAN transmit timeout.

LIN1

21 92: LIN Special Function #3 (LSF3), sequence mis-match.
Byte not as expected. Function aborted.

LIN0

25 96 rr ss tt vv

rr:
- b7: short to ground detected.
- b6: transmit command processing return code error.
- b5: error decoding length bits, receive manager.
- b4: no receive buffer available (break received).
- b3: receive byte not equal transmit byte (master).
- b2: receive buffer mode unknown 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 or too high.
- b3: transmit command control byte error, not master or slave
- b2: transmit command slave with no data.
- b1: 0
- b0: received a zero length message.

tt:
- b7: 0
- b6: 0
- b5: periodic message too long.

b4: periodic message mode not 1 or 2.
b3: master mode error (state is unknown or invalid).
b2: receive byte not equal transmit byte (slave).
b1: slave mode error (state is unknown or invalid).
b0: master state transmit 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: Command error.

31 xx: xx = Header byte of message in error.

3:

4:

5:

6: Configuration reports.

LIN1, KWP, LIN0

63 01 0x 0y: Send received checksum to host.
x: channel: 5, 6, 7.
y: 0 disabled. [Default.]
1 enabled.

LIN1

62 02 xx: Receive buffer timeout measured from most recent received byte.
xx milliseconds

KWP

63 03 xx yy: K-line baud rate divisor is xx yy.

LIN1, KWP, LIN0

63 06 0x 0y: Echo transmitted message to host.
- x: channel: 5, 6, 7.
- y: 0 disabled [Default.]
  1 enabled.

CAN0, CAN4, LIN1, KWP, LIN0

63 08 0x 0y: Time stamp status.
- x: channel: 0, 4, 5, 6, 7.
- y: 0 disabled. [Default.]
  1 enabled.
  CAN: inverse of baud clock.
  LIN1, KWP, LIN0: 1 msec.
  2 enabled.
  all channels: 1 msec.

For all channels the time stamp is 1 millisecond resolution.

KWP

62 13 xx: 5-baud address is xx.

LIN1, KWP, LIN0

63 19 0x 0y: Transmit checksum.
- x: channel: 5, 6, 7.
- y: 0 disabled.
  1 enabled. [Default.]

LIN1, KWP, LIN0

63 24 0x 0y: Receive network messages.
- x: channel: 5, 6, 7.
- y: 0 disabled.
  1 enabled. [Default.]

LIN1, KWP

62 27 xx: P4 time, transmit message inter-byte time, is xx milliseconds.

LIN1, LIN0

63 28 0x 0y: Receive ID byte processing.
- x: channel: 5, 7.
- y: 0 disabled.
  1 enabled. [Default.]

KWP
62 2A xx: P3 time, end of receive to start of transmit time, is xx milliseconds.

KWP
62 2B xx: Receive buffer expiration time is xx milliseconds.

KWP
63 2C xx yy: Key bytes from 5-baud initialization attempt; xx and yy.

KWP
62 30 yy: A break is being transmitted onto the K-line for 'yy' milliseconds.
62 30 00: The break has ended.

CAN0, CAN4, LIN1, KWP, LIN0
63 40 0x 0y: Send transmit acknowledgements to host.
x: channel: 0, 4, 5, 6, 7.
y: 0 disabled.
1 enabled. [Default.]

KWP
63 46 xx yy: Bus idle time (W5) prior to initialization attempt = xx yy milliseconds.

KWP
62 47 xx: FAST initialization low time = xx milliseconds.

KWP
62 48 xx: FAST initialization high time = xx milliseconds.

KWP
62 4B 00: Transmit checksum is normal. [Default]
62 4B 01: Transmit checksum is 2's complement.
62 4B 02: Transmit checksum is XOR.

---
62 4C xx: Command processing delay.
Delay is xx timer ticks (5x 63 command).

LIN1

62 50 01: LIN1 bus baud rate is 2400 baud
62 50 02: LIN1 bus baud rate is 9600 baud. [Default]
62 50 03: LIN1 bus baud rate is 19200 baud.
64 50 04 xx yy: LIN1 bus baud rate is equal to:
25 000 000 / (16 * $xxyy)

LIN1

62 52 xx: Maximum frame time is $xx milliseconds.

KWP

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 timer ticks (5x 63 command).

LIN1, LIN0

63 5A 0x 0y: LIN checksum select.
x: channel: 5, 7.
y: 0 LIN 1.3 Classic checksum. [Default.]
1 LIN 2.0 Enhanced checksum.

---

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.

LIN1, LIN0

63 66 0x 0y: ID byte only messages.
x: channel: 5, 7.
y: 0 Suppress (discard). [Default.]
1 Send to host.

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.

---

62 67 01: Internal baud rate is set for 19.2 kbaud.
62 67 02: Internal baud rate is set for 38.4 kbaud.
62 67 04: Internal baud rate is set for 57.6 kbaud.
62 67 08: Internal baud rate is set for 115.2 kbaud.
62 67 FF: Internal baud rate is set for 230.4 kbaud.
62 67 20: Internal baud rate is set for 460.8 kbaud.
62 67 40: Internal baud rate is set for 921.6 kbaud.

LIN1, KWP, LIN0

62 69 00: KWP, LIN1, LIN0 secondary modes disabled.
62 69 01: LIN1 secondary mode enabled.
62 69 02: KWP secondary mode enabled.
62 69 04: LIN0 secondary mode enabled.
62 69 05: LIN1 and LIN0 secondary modes enabled.
62 69 06: KWP and LIN0 secondary modes 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.

LIN1

63 6B xx yy: Break of duration xx yy microseconds was transmitted to the network.

KWP

62 6C 00: Fast transmit disabled.
62 6C 01: Fast transmit enabled.

LIN1, LIN0

63 6F 0x 0y: Slave response message ack or echo.
- x: channel: 5, 7.
- y: 0 disable.
  1 enable and ack with ID.
  2 enable and echo complete transmitted message.

KWP

6x 75 yy zz … MMR function mask definition.
x – count of bytes to follow
yy zz … mask bytes

KWP

6x 76 yy zz … MMR function match definition.
x – count of bytes to follow
yy zz … match bytes

KWP

6x 77 yy zz … MMR function response definition.
x – count of bytes to follow
yy zz … command bytes

KWP

62 78 00: MMR function disabled..
62 78 01: MMR function enabled.

LIN1

62 7D 00: Suppress ‘noise’ only error responses (25 86 00 00 00 x4).
62 7D 01: Do not suppress any ’25 86 rr ss tt vv’ error responses.

LIN1, LIN0

64 7F 0x 0y 0v: LIN Special Function #2 (LSF2) function status.
- x: channel: 5, 7.
- y: periodic message number $0 to $A.
- v: 0 = disabled.
  1 = enabled.

LIN0

62 80 xx: Receive buffer timeout measured from most recent received byte.
xx milliseconds

LIN0
62 81 01: LIN0 bus baud rate is 2400 baud
62 81 02: LIN0 bus baud rate is 9600 baud. [Default]
62 81 03: LIN0 bus baud rate is 19200 baud.
64 81 04 xx yy: LIN0 bus baud rate is equal to:
25 000 000 / (16 * $xxyy)

LIN0
62 82 xx: Maximum frame time is $xx milliseconds.

LIN0
63 83 xx yy: Break of duration xx yy microseconds was transmitted to the network.

LIN0
62 84 xx: P4 time, transmit message inter-byte time, is xx milliseconds.

LIN1
62 89 00: LIN1 (channel 5) Digital Output is disabled.
62 89 01: LIN1 (channel 5) Digital Output is enabled, send message to user.
62 89 02: LIN1 (channel 5) Digital Output is enabled, discard the message.

LIN1
64 8A rr 0x yy: LIN1 (channel 5) Digital Output function setup report.
rr: LIN ID of message to 'look' for.
x: Offset into the data field, zero based.
yy: Mask value.

LIN1
62 8B 00: LIN1 Special Function #3 (LSF3) is disabled.
62 8B 01: LSF is enabled.

LIN1
6x 8C rr ss ... : rr ss ... LIN1 Special Function #3 (LSF3) specified byte sequence.

LIN1
63 8D rr ss: LIN1 Special Function #3 (LSF3) rcv/xmt bit map.
Bit0 is the first byte on the network (after the synch byte).

LIN1
62 8E rr: LIN1 Special Function #3 (LSF3) watchdog reset.
Value in milliseconds.

LIN1
62 90 00: Short LIN1 function disabled.
62 90 01: Short LIN1 function enabled.

LIN1
6x 91 rr ss tt …: Data field of periodic message $0A.

LIN1, LIN0
67 95 0y 0z 0w 0r 0s 0t: LIN Special Function #4 (LSF4, Counter X) status.
- y: channel: 5, 7.
- z: 0 = disable.
  1 = enable.
- r: counter size; 1 to 4 bits.
- s: byte location; 0 to 7.
- t: bit location; 0 to 7 (of counter lsb).

7: Initialization attempt response.

KWP
71 00: Initialization attempt failure.

KWP
71 11: Initialization attempt success.

8: CAN configuration reports.

CAN0, CAN4
83 0A 0x yy: Baud rate for CAN channel.
- x: channel: 0, 4.
- yy:
  - 00: User specified using 74 0B 0x rr ss command.
  - 01: 1 Mbps.
  - 02: 500 kbps.
  - 03: 250 kbps.
  - 04: 125 kbps.
  - 0A: 33.333 kbps.

0B: 83.333 kbps.

[CAN0: Default = 500 kbps.]
[CAN4: Default = 500 kbps (usually) or 33.3333 kbaud]

CAN0, CAN4

84 0B 0x rr ss: Bit Timing Registers (BTR) for CAN channel.
- x: channel: 0, 4.
- rr: Bit Timing Register 0 setting.
- ss: Bit Timing Register 1 setting.

CAN0, CAN4, LIN1, KWP, LIN0

84 0C 0y 0v 0w: Periodic message group operation status.
- y: channel: 0, 4, 5, 6, 7.
- v: Group, 1 or 2.
- w: Mode.
  - 0: Disabled.
  - 1: Type1 enabled.
  - 2: Type2 enabled.

CAN0, CAN4

83 0E 0r ss: Outbound flow control separation time.
- r: channel: 0, 4.
- ss: Separation time (msec.).

CAN0, CAN4

84 0F rs tt uu: Outbound flow control ID, 11-bit; no AE byte.
- r: 0 = 11-bit ID.
- s: channel: 0, 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: channel: 0, 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: channel: 0, 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: channel: 0, 4.
- tt uu vv ww: 29-bit ID.

ae: AE byte.

CAN0, CAN4

83 11 0x 0y: Operational mode for CAN channel.
- x: channel: 0, 4.
- y: 0: Disabled. [Default for CAN0 and CAN4.]
  1: Enabled for normal operations.
  2: Enabled for listen only operations.

CAN4

82 12 0x: SWC transceiver mode.
- x: 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.

CAN0, CAN4

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: channel: 0, 4.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data field.

LIN1, LIN0

7x 18 vv 0y 0z ww pp qq rr ... Periodic message setup.
- vv: Message number, $01 to $0A.
- y: channel: 5, 7.
- z: 0: slave message.
  1: master message.
- ww: message ID.

pp qq rr ...: data field.

KWP

7x 18 vv 06 pp qq rr ... Periodic message setup.
- vv: Message number, $01 to $0A.
- 6: channel: 6.
- pp qq rr ...: data field.

CAN0, CAN4, LIN1, KWP, LIN0

84 1A 0x yy 0v: Periodic message disable/enable status.
- x: channel: 0, 4, 5, 6, 7.
- yy: Message number, $01 to $20 or ($0A).
- v: 0 disabled.
  1 normal mode enabled.
  2 slave mode enabled.
  3 both modes enabled.

CAN0, CAN4, LIN1, KWP, LIN0

84 1B 0x yy vv: Periodic message interval count.
- x: channel: 0, 4, 5, 6, 7.
- yy: Message number, $01 to $20 or ($0A).
- vv: interval count.

CAN0, CAN4, LIN1, KWP, LIN0

82 1C 0x: All periodic messages of channel x disabled.
- x: channel: 0, 4, 5, 6, 7.

82 1C EE: All periodic messages, all channels, disabled.

CAN0, CAN4

83 1F 0x 0y: Periodic Message Pause status.
- x: channel: 0, 4
- y: 0: disabled.
  1: enabled.

CAN0, CAN4

83 26 0x 0y: ISO 15765 operations.
- x: channel: 0, 4.
- y: 0: Mode0 (disabled)
  1: Mode1
  2: Mode2

CAN0, CAN4

83 27 0x 0y: Outbound message padding.
- x: channel: 0, 4.
- y: 0: disabled.
  1: enabled.

84 27 0x 0y vv: Outbound message padding.
- x: channel: 0, 4.
- y: 0: disabled.
  1: enabled.
- vv: pad byte (only if enabled).

CAN0, CAN4

8x 29 xy rr ss …: Report ISO 15765 receive 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: channel: 0, 4.
- rr ss: 11-bit receive ID
- rr ss tt vv: 29-bit receive ID

CAN0, CAN4

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: channel: 0, 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).

CAN0, CAN4

83 2B 0x 0y: Report ID/Mask mode for CAN channel.

- x: channel: 0, 4.
- y: 2: Two 32-bit IDs and masks.
  4: Four 16-bit IDs and masks.
  8: Eight 8-bit IDs and masks.

CAN0, CAN4

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: channel: 0, 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).

CAN0, CAN4

84 2D 0x 0y zz: Report acceptance ID register.

- x: channel: 0, 4.
- y: Acceptance ID register number, 0 to 7.
- zz: Register value read.

CAN0, CAN4

84 2E 0x 0y zz: Report mask register.

- x: channel: 0, 4.
- y: Mask register number, 0 to 7.
- zz: Register value read.

CAN0, CAN4

83 30 0s 0t: AE byte; disable / enable status report.

s: channel: 0, 4.
t: 0: disabled.
1: enabled.

CAN0, CAN4

82 33 xx: CAN ISO 15765 receive buffer time-out value.
Time is in 174.8 msec increments.
Both CAN0 and CAN4 channels use this value.

CAN0, CAN4

82 34 xx: CAN ISO 15765 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 (according to ISO 15765).
Both CAN0 and CAN4 channels use this value.

CAN0, CAN4

83 35 0y zz: CANx ISO 15765 pacing timer.
y: channel: 0, 4.
zz: pacing timer count. Refer to Section 7.13.8.1.

---

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.

CAN0, CAN4

83 3A 0r 0s: ABX control status.
r: channel: 0, 4.
s: 0 = operations are disabled.
s: 1 = operations are enabled.

85 3A 0y 0z mm nn: ABX control response, with start address.
r: channel: 0, 4.
s: 0 = disable / terminate operations.
s: 1 = enable / start operations.
mm nn: start address; $0000 to $3FFF

87 3A 0y 0z mm nn pp qq: ABX control response, with start address and byte count.
r: channel: 0, 4.
s: 0 = disable / terminate operations.
s: 1 = enable / start operations.
mm nn: start address; $0000 to $3FFF
pp qq: byte count; $0001 to $8000

CAN0, CAN4

83 3B 0x 0y: CAN channel activity status.
x: channel: 0, 4.
y: 0 disabled.
1 enabled.

CAN0, CAN4

83 3C 0x yy: CAN channel activity status.
x: channel: 0, 4.
yy: frame count since last query.

---

83 3F 0x yy: Pacing timer
x: channel: 0, 4.
yy: timer count; time tick is 1 msec.

CAN0, CAN4

84 40 0x yy 0z: ARC function status for periodic message.
x: channel: 0, 4.

yy: Periodic message number, $01 to $20.
z: 0 = disabled
1 = enabled

CAN0, CAN4

83 41 0y 0z: ATD monitor function status.
y: channel: 0, 4.
z: 0 = disable. [Default]
1 = enable.

CAN0, CAN4

89 42 0y 0z mm nn pp qq tt vv ATD monitor report.
y: channel: 0, 4.
z: ATD channel: 0 to F
mm nn: count
pp qq: minimum value read
tt vv: maximum value read

CAN0, CAN4

8x 43 xy rr ss …: ATD monitor expected ID report.
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: 0, 4.
rr ss: 11-bit receive ID
rr ss tt vv: 29-bit receive ID

CAN0, CAN4

85 44 0y mm nn pp: ATD monitor function, expected bytes, channel CANy.
y: channel: 0, 4.
mm nn pp: the expected first three bytes of ATD message.

CAN4

82 45 00: Hardware does not support this command.
82 45 01: CAN4 physical layer is set to single wire CAN.
82 45 02: CAN4 physical layer is set to 2-wire CAN.

CAN0, CAN4

~~83 46 0x 0y: Status of CAN flow control error responses.~~
~~x: channel: 0, 4.~~
~~y: 0 = disabled.~~
~~1 = enabled.~~

CAN0, CAN4

84 47 0x yy 0z: RUP function status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $20.
z: 0 = disabled
1 = enabled

LIN1, LIN0

84 49 0y zz 0v: LIN Special Function #1 (LSF1, Counter0) status.
y: channel: 5, 7.
zz: Message number, $01 to $0A.
v: 0 disabled.
1 enabled.

---

82 48 mm … Test the J1850 CRC function.
mm: computed J1850 CRC byte.

CAN0, CAN4

85 4A 0x yy 0z rr: CAC function status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $20.
z: 0 = disabled
1 = enabled
rr: constant value used in checksum computation.

CAN0, CAN4

82 4C yy: Message ‘yy’ is first periodic message assigned to CAN4 group 1.

CAN0, CAN4

83 4D 0x yy: Message ‘xx’ is first periodic message assigned to CANx group 2.

CAN0, CAN4

86 4E 0x yy 0v 0w rr: CIB status for periodic message.

x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable
w: location of counter, 0 to 7.
rr: increment value.

CAN0, CAN4

84 4F 0x yy 0v: RC2 status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable

CAN0, CAN4

83 50 xx yy: Lost Frame counter and timer functions.
xx is for the counter threshold function
xx = 00: count threshold function disabled
xx = 01 to FF: count threshold set point
yy is for the timer function
yy = 00: timer function disabled
yy = 01 to FF: wait time, in milliseconds, to report lost frame count.

CAN0, CAN4

84 51 0x yy 0v: RC3 status for periodic message.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable
1 = enable

CAN0, CAN4

85 52 0x yy 0v rt RC4 function definition.
x: channel: 0, 4.
yy: Periodic message number, $01 to $58.
v: 0 = disable.
1 = enable.
r: 0 = counter is lower nibble.
1 = counter is upper nibble.
t: location of counter, 0 to 6.

CAN0, CAN4

86 5A 0x 0r rr st vv: CAN Digital Output function setup

x: channel: 0, 4
rrr: 11-bit ID
s: 0 = if bit-wise mask and match
1 = if byte wise match only
t: byte offset into data field, 0 to 7
vv: mask and match value

88 5A 0x 0r rr rr rr st vv: CAN Digital Output function setup
x: channel: 0, 4
rrrrrrr: 29-bit ID
s: 0 = if bit-wise mask and match
1 = if byte wise match only
t: byte offset into data field, 0 to 7
vv: mask and match value

CAN0, CAN4

83 5B 0x 0y: CAN Digital Output function status.
x: channel: 0, 4.
y: 0 = disable.
1 = enable.

9: Board status information.

92 04 xx: Firmware version report. Version is xx.
93 04 xx yy: Firmware version report. Version is xx yy.
91 07: VPW operations.
91 08: DLC initialization complete.
91 0F: KWP operations.
91 10: CAN operations.
91 19: LIN1 operations.
91 24: CAN0 reset.
91 25: CAN4 reset.
91 27: Idle state.
91 29: LIN0 operations.
93 28 0x yz: Model number report. xyz is the model number.

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.

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.
	06	2.56 msec.

---

51 67: Query for internal baud rate setting. (AVT-853 only)
52 67 01: Set internal baud rate to 19.2 kbaud.

52 67 02: Set internal baud rate to 38.4 kbaud.
52 67 04: Set internal baud rate to 57.6 kbaud.
52 67 08: Set internal baud rate to 115.2 kbaud.
52 67 FF: Set internal baud rate to 230.4 kbaud.
52 67 20: Set internal baud rate to 460.8 kbaud.
52 67 40: Set internal baud rate to 921.6 kbaud.
New setting does not take affect until reset;
either power-on reset or software reset (F1 A5).
XPort baud rate must be changed to match.

---

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.

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.

---

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.
B1 01: Request firmware version and sub-version numbers.

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-85x (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.

- 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 xx: Internal baud rate.

xx:

01:	19.2 kbaud
02:	38.4 kbaud
04:	57.6 kbaud
08:	115.2 kbaud
FF:	230.4 kbaud
20:	460.8 kbaud
40:	921.6 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.

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. Version is xx.
93 04 xx yy: Firmware version report. Version is xx yy.
91 07: VPW operations.
91 08: DLC initialization complete.
91 0F: KWP operations.
91 10: CAN operations.
91 19: LIN operations.
91 27: Idle state.
93 28 0x yz: Model number report. xyz is the model number.

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 (Stand Alone) 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:

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:
25 000 000 / (16 * xxyy) [all values shown are decimal]
Example: for K-line bus baud rate = 10400;
xxyy = $00 96 (hex) = 150 (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).

KWP
52 30 yy: Send a break onto the K-line for 'yy' milliseconds.

---

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.
06	2.56 msec.

---

51 67:	Query for internal baud rate setting. (AVT-853 only)
52 67 01:	Set internal baud rate to 19.2 kbaud.
52 67 02:	Set internal baud rate to 38.4 kbaud.
52 67 04:	Set internal baud rate to 57.6 kbaud.
52 67 08:	Set internal baud rate to 115.2 kbaud.
52 67 FF:	Set internal baud rate to 230.4 kbaud.
52 67 20:	Set internal baud rate to 460.8 kbaud.
52 67 40:	Set internal baud rate to 921.6 kbaud.
	New setting does not take affect until reset;
	either power-on reset or software reset (F1 A5).
	XPort baud rate must be changed to match.

---

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.

	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: 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.
B1 01: Request firmware version and sub-version numbers.

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-85x (a form of software reset).

18.1 KWP (Stand Alone) 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.
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:

b7: transmit date 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 30 yy: A break is being transmitted onto the K-line for 'yy' milliseconds.
62 30 00: The break has ended.

---

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: Internal baud rate is set for 19.2 kbaud.
62 67 02: Internal baud rate is set for 38.4 kbaud.
62 67 04: Internal baud rate is set for 57.6 kbaud.
62 67 08: Internal baud rate is set for 115.2 kbaud.
62 67 FF: Internal baud rate is set for 230.4 kbaud.
62 67 20: Internal baud rate is set for 460.8 kbaud.
62 67 40: Internal baud rate is set for 921.6 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.

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: 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. Version is xx.
93 04 xx yy: Firmware version report. Version is xx yy.
91 07: VPW operations.
91 08: DLC initialization complete.
91 0F: KWP operations.

91 10:	CAN operations.
91 19:	LIN operations.
91 27:	Idle state.
93 28 0x yz:	Model number report. xyz is the model number.

A:

B:

C:

D:

E:

F:

19. Appendix A

A Telnet listing of the factory default settings for an AVT-853 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-853 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-853 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-853 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 (230400) ?
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-853 XPort will save the configuration and reboot - that will terminate the telnet session.

Wait at least 20 seconds for the AVT-853 XPort unit to finish saving and rebooting before trying to establish a connection to the AVT-853 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 230400, I/F Mode 4C, Flow 02
Port 10001
Remote IP Adr: --- 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-853 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-853 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)

NOTE: XPort web pages reside in the XPort device and may be different than as described here. That fact is not a problem.

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-853 XPort will save the new parameters and then reboot. It takes about 1 minute for the AVT-853 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-853 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 *(may be different / higher)*
Hardware Address:	This is the so-called MAC address. It is unique to each AVT-853 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-853 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-853 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:	230400
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]

21. Appendix C

Raising the Internal Baud rate of the AVT-853.

On the AVT-853 board, there is a microcontroller that controls vehicle communications. There is also the Lantronix XPort embedded serial server device that controls Ethernet TCP/IP communications with the host computer. The microcontroller and the XPort communicate with one another using an embedded high speed serial link. The speed of that link affects total message or data throughput.

The default baud rate of the internal serial link is 230.4 kbaud. That is usually sufficient for most applications. If necessary that baud rate can be increased to a maximum of 921.6 kbaud. Below is a procedure to change the AVT-853 internal baud rate.

Here is a procedure on how to raise the internal baud rate of the AVT-853 interface above the factory default value of 230.4 kbaud.

Remember, the internal baud rate has nothing to do with any of the vehicle networks.
It only affects how the microcontroller on the AVT-853 board communicates with the XPort device on the board. Both have to set to the same baud rate.

1. Using the Hex Terminal, or similar software, enter an operating mode such as CAN.
   The switch to CAN mode command is: E1 99.

2. Change the internal baud rate of the microcontroller to what you desire using the 52 67 xx command.

3. Wait 2 seconds.

4. Send the reset command F1 A5. There will not be any response.

5. In order to set the XPort baud rate to a rate above 230.4 kbaud, you have to first set the XPort CPU to High Performance mode.

6. Use a browser, go to the XPort page.
   (For example, using the default IP address of the AVT-853, enter this on the address line of your browser: http://192.168.1.70).

7. There is no User Name or Password for the XPort. Do not enter anything.
   Select OK on the login page.

8. On the left, select Server.

9. Set the CPU Performance Mode to High.

10. On the left, select Apply Settings.

11. Wait for the XPort to reboot.

12. Clear the browser cache.
    This is very important on some browsers.

13. Go to the XPort page.

14. On the left, select Channel 1 Serial Settings.

15. In the Channel 1 section, select the Baud Rate to be the same as the rate you set the microcontroller to in step 2, above.

16. Select the OK button on the bottom of the page.

17. On the left, select Apply Settings.

18. Close the browser.

19. Wait for the XPort to reboot. (Usually about 15 seconds.)

20. Using the Hex Terminal, or similar software, send the B0 software version request to the AVT-853 and verify proper communications.
    The response should be 92 04 xx, the software version number. If you get the correct response, all is working well.

22. Questions ??

Contact the factory by e-mail, phone, or fax.
Contact information is provided here and on the bottom of page 1.

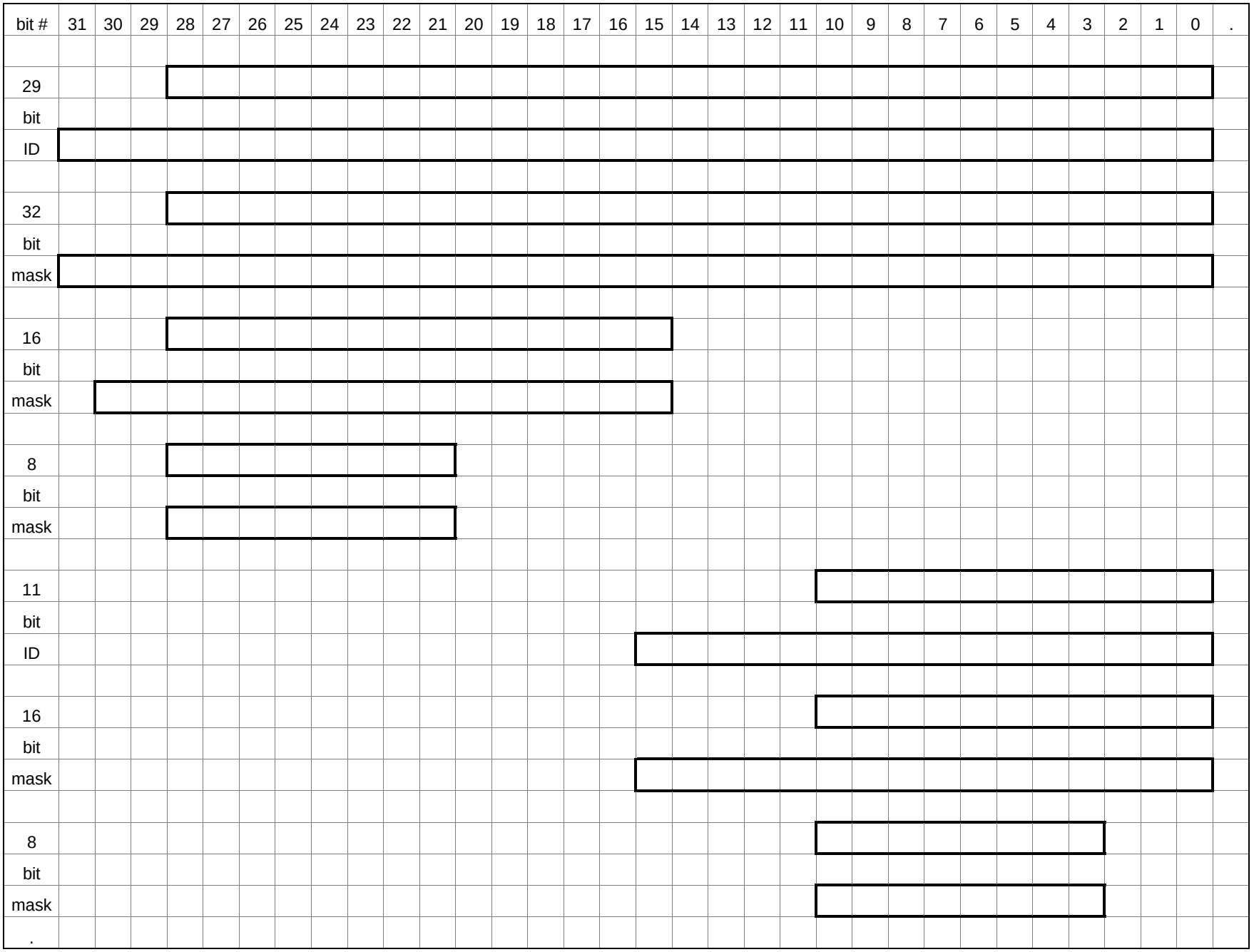
Post: 1509 Manor View Road
Davidsonville, MD 21035 USA

Phone: +1-410-798-4038
Fax: +1-410-798-4308

E-mail: Support@AVT-HQ.com
Web site: www.AVT-HQ.com

23. 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