ADVANCED VEHICLE TECHNOLOGIES, Inc.

AVT - 423

Multiple Interface

CAN0: 2-wire (high speed) (non-FD)

CAN1: 2-wire (high speed) or Single Wire (SWC) (non-FD)

CAN2: 2-wire (high speed) (FD capable)

CAN3: 2-wire (high speed) (FD capable)

LIN0: LIN communications

LIN1: LIN communications

LIN1: K-Line communications
(not yet functional)

Flexray: Dual or Single channel communications
(not installed, not yet functional)

Software Version 00 05
21 January 2017

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 4

1.1 HARDWARE 4
1.2 SOFTWARE STATUS 4
1.2.1 CAN0 4
1.2.2 CAN1 4
1.2.3 CAN2 4
1.2.4 CAN3 4
1.3 SOFTWARE PLAN 5
1.4 SOFTWARE 5
1.4.1 Determining Software Version 5
1.4.2 Determining Model Number 5
1.4.3 Determining Board Revision Level 5
COMMANDS AND RESPONSES 6

2. GLOSSARY 6

3. AVT-423 OPERATION 7

4. AVT-423 CPU 7

4.1 SUPPORT SOFTWARE 7

5. CLIENT COMPUTER CONNECTION 8

5.1 AVT-423 CONNECTION TO CLIENT COMPUTER 9
5.1.2 Ethernet IP Addressing Modes 9
5.1.3 Changing the IP Address 9
5.2 PACKET COMMUNICATIONS BETWEEN THE AVT-423 AND THE CLIENT COMPUTER 10

6. POWER AND NETWORK CONNECTION 11

6.1 AVT-423 BOARD REVISION “B” & “C” 11
6.2 POWER REQUIREMENTS 12
6.2.1 Ground 12
6.2.2 Input Voltage 12
6.2.3 Power Dissipation 12

7. ADC CONNECTION 13

8. DAC CONNECTION 13

9. OPERATION MODES 13

9.1 SIMULTANEOUS NETWORK OPERATIONS 13

10. NETWORK HARDWARE DESCRIPTIONS 13

10.1 CAN0 - 2-WIRE CAN 13
10.1.1 CAN0 Channel Number 14
10.2 CAN1 - 2-WIRE CAN OR SINGLE WIRE CAN 14
10.2.1 CAN1 Channel Number 14
10.3 CAN2 - 2-WIRE CAN 14
10.3.1 CAN2 Channel Number 15
10.4 CAN3 - 2-WIRE CAN 15
10.4.1 CAN3 Channel Number 15
10.5 LIN0 15
10.5.1 LIN0 Channel Number 16
10.6 LIN1 16

10.6.1 LIN1 Channel Number ... *16*
10.7 KWP ... 16
10.7.1 KWP Channel Number ... *16*
10.8 FLEXRAY ... 16
10.8.1 Flexray Channel Number ... *16*

11. CAN COMMUNICATIONS NOTES ... 16

11.1.1 Disabled ... *17*
11.1.2 Normal ... *17*
11.1.3 Listen Only ... *17*
11.1.4 Transmit Command ... *17*
11.1.5 Receive Response ... *18*
11.1.6 Time Stamps ... *19*
11.2 ACCEPTANCE ID AND MASK ... 19
11.2.1 Configuration ... *19*
11.2.2 Acceptance ID and Mask Operation ... *20*
11.2.3 CAN0 and CAN1 specifics ... *20*
11.2.4 CAN2 and CAN3 specifics ... *20*
11.3 SETTING UP CAN0 OR CAN1 FOR OPERATION ... 21
11.3.1 Communications Example ... *21*
11.4 SETTING UP CAN2 OR CAN3 FOR OPERATION ... 22
11.4.1 Communications Example ... *22*
11.5 PERIODIC MESSAGE SUPPORT ... 23
11.5.1 Type1 Periodic Messages ... *23*
11.5.2 Type2 Periodic Messages ... *23*
11.5.3 Periodic Message Commands ... *24*
11.6 PERIODIC MESSAGE SPECIAL FUNCTIONS ... 24
11.6.1 CAN Frame Data Definition ... *24*
11.6.2 Special Function xxx ... *24*
11.6.3 Special Function xxx ... *25*
11.6.4 Special Function xxx ... *25*
11.7 PERIODIC PAUSE FUNCTION ... 25
11.8 ISO 15765 SUPPORT ... 25

12. LIN0 OPERATIONS ... 25

13. LIN1 OPERATIONS ... 25

14. KWP OPERATIONS – USING LIN1 HARDWARE ... 25

15. COMMANDS ... 26

15.1 RESPONSES ... 37

16. APPENDIX A ... 50

17. APPENDIX B ... 50

18. QUESTIONS ?? ... 51

1. Introduction

This document describes the AVT-423 hardware and software.

The AVT-423 is a multiple network interface for in-vehicle networks. The operation software supports the following networks/protocols on the indicated channels:

- CAN0: 2-wire CAN (non-FD).
- CAN1: 2-wire CAN or Single Wire CAN (SWC) (non-FD).
- CAN2: 2-wire CAN (CAN-FD).
- CAN3: 2-wire CAN (CAN-FD).
- Flexray: Dual or Single channel.
- LIN0: LIN only.
- LIN1; LIN or KWP (K-line).

All operations are simultaneous with the exception of LIN or K-line on channel LIN1.

1.1 Hardware

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

Hardware status: www.AVT-HQ.com/423_hw.htm

1.2 Software Status

At this time, the AVT-423 software supports the following capabilities.

1.2.1 CAN0

Transmit and receive. Sixteen objects (either receive or transmit). Thirty two periodic messages. Periodic messages are Type1 only.

1.2.2 CAN1

Transmit and receive. Sixteen objects (either receive or transmit). Thirty two periodic messages. Periodic messages are Type1 only.

1.2.3 CAN2

Transmit and receive. Sixteen receive objects. Thirty two periodic messages. Periodic messages are Type1 only. CAN-FD communications fully supported. ISO CAN frame CRC (non-ISO is available). Maximum data payload of 64 bytes supported. Maximum baud rate of 8 Mbaud supported.

1.2.4 CAN3

Transmit and receive. Sixteen receive objects. Thirty two periodic messages. Periodic messages are Type1 only. CAN-FD communications fully supported. ISO CAN frame CRC (non-ISO is available). Maximum data payload of 64 bytes supported. Maximum baud rate of 8 Mbaud supported.

1.3 Software Plan

Software for this product is under almost continuous development. This manual will be updated as soon as possible for each new software release.

The following functions are planned for implementation as quickly as is reasonably possible. These tasks are listed in priority order.

- Implement Flexray.
- CAN ISO 15765 support for CAN0 and CAN1.
- KWP using the LIN1 transceiver.
- CAN Type2 periodic messages.

1.4 Software

Refer to our web site for the most up-to-date information about AVT-423 software versions:

www.AVT-HQ.com/432_sw.htm

1.4.1 Determining Software Version

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

- Connect to a client computer running the Hex Terminal or equivalent.
- Power-on the AVT-423 interface unit.
- The power-on notification is:
  $91 $3A indicates AVT-423 operations.
  $93 $04 $xx $yy where 'xx yy' is the software version.
- At any time, send the $B0 command.
- The response will be: $93 $04 $xx $yy where 'xx yy' is the software version.

1.4.2 Determining Model Number

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

- Connect to a client computer running the Hex Terminal or equivalent.
- Power on the AVT-423 interface unit.
- Send the $F0 command.
- The response will be: $93 $28 $xx $yy where xxyy forms the model number.
  (eg. 04 23)

1.4.3 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 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.
0x1234	Indicates hex number 1234.
ADC	Analog to Digital Converter or Conversion.
BRS	A CAN bit. Baud rate switch. Part of CAN-FD. Signals that the data portion of the CAN frame will use a different baud rate.
CAN	Controller Area Network
CAN-FD	CAN with “Flexible Data”. There are two components to CAN-FD. Larger data payload; maximum of 64 bytes and higher baud rate for the data portion of the CAN frame.
CAN0	CAN, channel 0
CAN1	CAN, channel 1
CAN2	CAN, channel 2
CAN3	CAN, channel 3
EDL	A CAN bit. Extended Data Length. Part of CAN-FD. Signals that the data portion of the CAN frame will use the CAN-FD defined data lengths.
IDE	A CAN bit. ID Extended. When this bit = 0 the CAN frame uses an 11-bit ID. When this bit = 1 the CAN frame uses a 29-bit ID; extended ID.
ISO 11898	An ISO specification for CAN and 2-wire CAN physical layer.
ISO 15765	An ISO specification dealing with the formatting of data in the CAN frame data field. Also used in sending blocks of data using CAN. Also known as Multi-Frame Messaging (MFM) or Segmented Messages. This specification also deals with other CAN network issues.
J2411	An SAE specification for Single Wire CAN (SWC).
K-line	Single wire communications protocol. Refer to ISO 9141, ISO 9141-2, and ISO 14230 for more information.
KWP	Key Word Protocol. Several versions exist, the most common being Key Word Protocol 2000. This is a ‘superset’ of ISO 9141 and ISO 9141-2.

LIN	Local Interconnect Network.
LIN0	LIN, channel 7
LIN1	LIN, channel 5
RTR	A CAN bit. Remote Transmission Request. When this bit = 1 it indicates a frame that is requesting a remote node to transmit an answering frame.
SRR	A CAN bit. 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, CAN, each periodic message operates independently.
Type2	Type2 Periodic Message, CAN, messages operate sequentially.
SWC	Single Wire CAN (SAE J2411).
XOR	Bit-wise logical exclusive OR.

3. AVT-423 Operation

The AVT-423 does not have a power switch. The unit powers up and begins operations as soon as external power is applied.

On power-up, the interface will, almost immediately, report to the client computer: $91 $3A which indicates that the interface is an AVT-423.

The interface will then report $93 $04 $xx $yy where ‘xx yy’ is the unit software version number.

Note that the client computer can not establish a TCP/IP connection until the AVT-423 is fully operational. From power-on to full operation is about xxx seconds.

4. AVT-423 CPU

The AVT-423 uses a “Netburner” “Mod 54415-100” CPU module with the following:

- 32-bit, 250 MHz, NXP/Freescale Coldfire processor.
- 64 Mbytes of RAM.
- 32 Mbytes of FLASH.

4.1 Support Software

AVT will make available two PC applications to the user. Both are developed and supplied by Netburner.

Both are available for download from AVT’s web site.
Refer to the last page of this manual for direct link to that web page.

You can also obtain them directly from Netburner: www.Netburner.com

Both are small PC applications (executable) that do not need to be installed.
Both have been tested under Windows XP (32-bit) and Windows 7, both 32-bit and 64-bit versions.

Obtain the executables from AVT's web site or Netburner's web site. Place them in a folder of your choosing, or on the desktop.

When needed, you launch the one you want to use by double clicking on it. They are very easy to use and do not need any explanation or instruction. However, feel free to contact AVT if you have any questions. All contact information is on the last page of this manual.

4.1.1.1 Set IP Address

This PC application is named: "IPSetup.exe"

It is posted on our web site. It is provided by Netburner. It has been verified to work in our lab on laptops running Windows-XP, Windows-7 (both 32 and 64-bit).

http://www.avt-hq.com/download.htm#AVT-423

The 'IPSetup' application will 'try' to find all Netburner hardware, display the IP address and allow you to view and change the IP address and the subnet mask.

NOTE: If your computer (the client) is on a different subnet than the AVT-423, this application will likely not be able to 'find' it. To 'fix' this, temporarily change the subnet mask and/or IP address of your client computer to something in the same domain as the AVT-423.

For example, the factory default IP address of the AVT-423 is 192.168.1.70.

If your computer has an IP address that is NOT of the form 192.168.1.xxx then you likely NOT be able to find the AVT-423. Temporarily change the IP address of your computer to something of the form: 192.168.1.xxx (but not 70) and then run the 'IPSetup' application. When done, return the IP address of your computer to its original setting.

4.1.1.2 Software Update

This PC application is named: "AutoUpdate.exe"

It is posted on our web site. It is provided by Netburner. It has been verified to work in our lab on laptops running Windows-XP, Windows-7 (both 32 and 64-bit).

http://www.avt-hq.com/download.htm#AVT-423

The 'AutoUpdate' will allow you to update the AVT-423 operation software.

You will need to know the IP address of the AVT-423 you want to update.

You will need the new AVT-423 operation software file. This file is posted on AVT's web site along with the software update application.

5. Client Computer Connection

The AVT-423 is an Ethernet TCP/IP server.
The user or test computer is, therefore, a client.

5.1 AVT-423 Connection to Client Computer

5.1.1.1 Ethernet IP Address

The factory default IP address of the AVT-423 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-423 is being used, the setting of the net mask may not be important. Rule of thumb: if connected to a busy network set the net mask to 255.255.255.0.

5.1.1.2 Hardware or MAC Address

Xxx mike – where can this be found ??

5.1.1.3 TCP/IP Port

Communications with the AVT-423 vehicle network interface is via port 10001.

All communications with the AVT-423 vehicle interface is in binary bytes [not ASCII hex]. Refer to Section 5.2 for a description of the 'packetized' communications protocol between the AVT-423 and the client computer. All communications with the AVT-423 follow the exact same rules and formats as that of the AVT-423 and all other AVT interface equipment.

5.1.2 Ethernet IP Addressing Modes

Two IP addressing modes are available for the AVT-423.

- Static
- DHCP

5.1.2.1 Static IP Addressing

The factory default addressing mode for the AVT-423 is static and the address is set to 192.168.1.70. In static mode the Ethernet address of the AVT-423 is always the same and does not change when power is cycled.

5.1.2.2 DHCP Addressing

Setting the AVT-423 IP address to 0.0.0.0 will enable DHCP (Dynamic Client Configuration Protocol) function.

In this mode, the AVT-423 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-423 will then do what xxx ??

5.1.3 Changing the IP Address

To change the IP address of the AVT-423 you should use the Netburner supplied software; described above in Section 4.1.1.1.

5.2 Packet Communications Between the AVT-423 and the Client Computer

Communications between the client computer and the AVT-423, 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 function for CAN3.
  Command: 53 08 03 01.
  Header byte upper nibble 5 indicates a configuration command.
  Header byte lower nibble 3 indicates three bytes follow.
  $08 is the time stamp command.
  $03 indicates channel 3.
  $01 enable time stamps.
- Example #2
  Send a message, to LIN0, as Master, ID = $3C, 8 data bytes.
  Command: 0B 05 01 3C 01 02 03 04 05 06 07 08.
  Header byte = $0B.
  Upper nibble $0 indicates 'to the network'.
  Lower nibble $B indicates 11 bytes follow.
  $01 indicates send as Master.
  $3C is the LIN message ID.
  $01 02 03 … are the 8 data bytes.

- Example #3

  Receive a message from CAN3, 11-bit ID, with 12 data bytes.

Response:	$11 10 03 05 07 77 01 02 03 04 05 06 07 08 09 10 11 12
Header byte:	$11, alternate header format #1.
$10:	16 bytes follow
$03:	channel 03; CAN3.
$05:	receive buffer 05.
$07 77:	CAN frame ID.
01 02 03, …	data bytes.

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

6. Power and Network Connection

The power and network connector (P3) is an industry standard DB-25P connector and requires a DB-25S mate. The pin / signal assignments for the vehicle / network connector are listed here.

Pins with no signal assignment are not connected and should not be used.
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.)

6.1 AVT-423 Board Revision "B" & "C"

Pin #	Description	Notes
1	CAN0_H	
14	CAN0_L	
2	CAN1_H	Transceiver software selected.
15	CAN1_L	Transceiver software selected.
3	CAN2_H	
16	CAN2_L	
4	CAN3_H	
17	CAN3_L	
5	FR-BUS_A_P	
18	FR-BUS_A_M	
6	FR-BUS_B_P	
19	FR_BUS_B_M	
7		

20		
8		
21		
9		
22		
10	CAN1_SWC	Transceiver software selected.
23	LIN0	
11	LIN1	KWP is alternate
24	GND	Same as pin # 12
12	GND	Same as pin # 24
25	RAW_VIN	Same as pin # 13
13	RAW_VIN	Same as pin # 25

P3 (the DB-25P connector on the AVT-423 board)
Table 1

6.2 Power Requirements

The AVT-423 board requires a suitable external power supply. Fairly clean +8 to +18 VDC.

6.2.1 Ground

Common ground is required between the AVT-423 board and all connected devices. On P3 there are two 'ground' pins, #12 and #24. Both are connected directly to the ground plane of the AVT-423 board. Only one is needed for normal operations.

6.2.2 Input Voltage

The external power supply is connected to P3 pin #13 or #25. The two pins are connected together, internally, on the AVT-423 board. Only one is needed for normal operations.

6.2.3 Power Dissipation

Power dissipation of the AVT-423 is listed here.

Current draw, minimum, maximum, and average - were measured using a 100 msec sample window. (Fluke Digital Multimeter model 87.) Measurements taken with board connected to client (logical connection) but no network activity and no activity between the board and the client.

Input Voltage	Min / Max/ Ave Measured Current	Power
+8 VDC	380 / 488 / 419 mA	3.4 W
+10 VDC	304 / 392 / 337 mA	3.4 W

+12 VDC	270 / 338 / 298 mA	3.6 W
+15 VDC	213 / 263 / 231 mA	3.5 W
+18 VDC	218 / 179 / 195 mA	3.5 W

7. ADC Connection

None at this time. There is a planned future hardware upgrade to add several channels of Analog to Digital Converter capability.

8. DAC Connection

None at this time. There is a planned future hardware upgrade to add one or two channels of Digital to Analog Converter capability.

9. Operation Modes

Unlike many other AVT interfaces, the AVT-423 does not have "Operation Modes".

This interface comes 'alive' with all networks initialized, operational, but disabled.

9.1 Simultaneous Network Operations

With two exceptions, all networks can be operated simultaneously.

Exception #1.
CAN1 is software selected to be either a 2-wire or a Single Wire CAN channel.
They are separate transceivers. You can not use both at the same time.

Exception #2.
KWP operations use the LIN1 transceiver. Thus, LIN1 can be operated as LIN (channel 5) or KWP (channel 6) but not both at the same time.

10. Network Hardware Descriptions

Technical details of each network channel is described in the following sections.

10.1 CAN0 - 2-wire CAN

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

It uses the Microchip MCP2562FD-E/SN transceiver.

CAN0 is not CAN-FD capable.

Refer to Table 1 for pin and signal definitions.

Termination can be software selected to be 'in' or 'out'. The default is 'in'.
Refer to the '$7x 62' command.

The AVT-423 board has been designed to support several different network termination schemes for CAN0. The factory default is the split termination consisting of two 52.3 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. The two opto-relays used to switch the termination add 7 ohms each.

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

10.1.1 CAN0 Channel Number

For the user, CAN0 is designated channel 0.

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

10.2 CAN1 - 2-wire CAN or Single Wire CAN

CAN1 is either a high speed 2-wire CAN channel that is ISO 11898 compliant or a low speed Single Wire CAN (SWC) channel that is J2411 compliant.

For 2-wire operations it uses the Microchip MCP2562FD-E/SN transceiver.

For single wire operations it uses the On-Semi NCV7356D2G transceiver.

CAN1 is not CAN-FD capable.

Refer to Table 1 for pin and signal definitions.

For 2-wire operations, termination can be software selected to be ‘in’ or ‘out’. The default is ‘in’. Refer to the ‘$7x 62’ command.

The AVT-423 board has been designed to support several different network termination schemes for CAN1 2-wire operations. The factory default is the split termination consisting of two 52.3 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. The two opto-relays used to switch the termination add 7 ohms each.

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

10.2.1 CAN1 Channel Number

For the user, CAN1 is designated channel 1.

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

10.3 CAN2 - 2-wire CAN

CAN2 is a high speed 2-wire CAN channel that is ISO 11898 compliant.

It uses the Microchip MCP2562FD-E/SN transceiver.

CAN2 is CAN-FD capable.

Refer to Table 1 for pin and signal definitions.

Termination can be software selected to be 'in' or 'out'. The default is 'in'.
Refer to the '$7x 62' command.

The AVT-423 board has been designed to support several different network termination schemes for CAN2. The factory default is the split termination consisting of two 52.3 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. The two opto-relays used to switch the termination add 7 ohms each.

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

10.3.1 CAN2 Channel Number

For the user, CAN2 is designated channel 2.

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

10.4 CAN3 - 2-wire CAN

CAN3 is a high speed 2-wire CAN channel that is ISO 11898 compliant.

It uses the Microchip MCP2562FD-E/SN transceiver.

CAN3 is CAN-FD capable.

Refer to Table 1 for pin and signal definitions.

Termination can be software selected to be 'in' or 'out'. The default is 'in'.
Refer to the '$7x 62' command.

The AVT-423 board has been designed to support several different network termination schemes for CAN3. The factory default is the split termination consisting of two 52.3 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. The two opto-relays used to switch the termination add 7 ohms each.

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

10.4.1 CAN3 Channel Number

For the user, CAN3 is designated channel 3.

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

10.5 LIN0

The LIN0 bus is a low speed, single wire, multi-drop, ground referenced network.

The AVT-423 uses the NXP (Freescale) MC33660 transceiver. Maximum baud rate is inferred to be 150 kbps. A 1 Kohm resistor is used as the LIN bus (K-line) pull-up to the same supply voltage for the AVT-423 board.

10.5.1 LIN0 Channel Number

LIN0 is designated channel 7.

10.6 LIN1

The LIN1 bus is a low speed, single wire, multi-drop, ground referenced network.

The AVT-423 uses the NXP (Freescale) MC33660 transceiver. Maximum baud rate is inferred to be 150 kbps. A 1 Kohm resistor is used as the LIN bus (K-line) pull-up to the same supply voltage for the AVT-423 board.

10.6.1 LIN1 Channel Number

LIN1 is designated channel 5.

10.7 KWP

Key Word Protocol communications uses the LIN1 transceiver and associated K-line. Refer to Section 10.6, above, for information about the physical layer.

10.7.1 KWP Channel Number

KWP is designated channel 6.

10.8 Flexray

The Flexray bus is implemented using the NXP (Freescale) MC9S12XF512MLM device.

A dual-bus Flexray interface is implemented. The plans are for it to be available as a dual-bus or single-bus network interface.

The AVT-423 uses the ON Semiconductor NCV7383DB0R2G device with a common mode choke.

10.8.1 Flexray Channel Number

Flexray is designated channel 9.

11. CAN Communications Notes

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

Each CAN channel of the AVT-423 is independent of all other channels.
    This applies to all channel parameters.

Each CAN channel of the AVT-423 has three operating modes:
    Disabled
    Normal
    Listen only. Only available for CAN0 and CAN1; not yet implemented.

Refer to Sections 15 and 16 for detailed information about CAN messages and packets to and from the network.

11.1.1 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

11.1.2 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 an acceptance ID according to the mask and associated rules, are passed to the client. Refer to Section 11.2 for a discussion about Acceptance ID and Mask.

The CAN channel is enabled for receive and transmit.

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

11.1.3 Listen Only

This feature/function has not yet been implemented in software.

11.1.4 Transmit Command

The fields and construction of a transmit command are shown here. The transmit command is also explained in the Commands and Responses – Sections 15 and 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.

All three formats are acceptable in ascending order. In other words a 0x yy … command can be expressed as '$0x yy' or as '$11 0x yy' or as '$12 00 0x yy'. Likewise, an '$11 xx' command can be expressed as '$12 00 xx'.

11.1.4.1 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 15 for a complete description of the transmit command.

11.1.4.2 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 15 for a complete description of the transmit command.

11.1.4.3 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 ISO 15765). Refer to the beginning of Section 15 for a complete description of the transmit command.

Note: ISO 15765 is not yet implemented.

11.1.4.4 CAN0 and CAN1 Byte Count Limits

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

Note that ISO 15765 has not yet been implemented for CAN0 or CAN1.

The current maximum is 8 data bytes.

11.1.4.5 CAN2 and CAN3 Byte Count Limits

CAN2 and CAN3 support CAN-FD and the maximum number of data bytes in a transmit command is 64.

In CAN-FD there are fixed field lengths, listed below. If a transmit command does not contain the proper number of bytes in the data field, the transmit command will be declared to be in error and the user will be notified with the following error responses: '$22 7F 10' and '$32 yy FF'.

However, there is an automatic padding function for CAN2 and CAN3. If enabled, and if the number of data bytes is less than 64, the AVT-423 will automatically pad the data field to the next higher data byte count. For example: You specify a transmit command to CAN2 with 10 data bytes and the pad function is enabled. Then the AVT-423 will add two pad bytes to the data field, raising it to 12 and then queue up that CAN frame for transmission.

The '$73 60 0x 0y' command disables and enables the pad function.

The '$73 61 0x yy' command specifies the pad byte value.

CAN-FD data field lengths:

- 0 to 8 data bytes.
- 12 data bytes.
- 16 data bytes.
- 20 data bytes.
- 24 data bytes.
- 32 data bytes.
- 48 data bytes.
- 64 data bytes.

11.1.5 Receive Response

There are two possible 'from the network' responses that the AVT-423 can send to the client.

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.

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-423 interface.

11.1.5.1 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 for a description of all bytes in the packet.

11.1.5.2 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 for a description of all bytes in the packet.

11.1.5.3 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 for a description of all bytes in the packet.

11.1.6 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 four byte value immediately after the packet header byte but before the CAN

11.1.6.1 CAN0 and CAN1 Time Stamp Clock

For CAN0 and CAN1, 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 increment is the inverse of the CAN channel baud rate. For example, if the baud rate is 500 kbaud, then the time stamp interval is 2 microseconds.

The time stamp clock and counter are separate for CAN0 and CAN1.

The time stamp rolls over at $0000FFFF.

11.1.6.2 CAN2 and CAN3 Time Stamp Clock

For CAN2 and CAN3, the time stamp is a 32-bit free running counter that is driven by a 2 kHz clock. As a result, the time stamp increment is 0.5 msec.

It appears that the time stamp clock and counter are shared for CAN2 and CAN3 (they both read the same counter. xxx Mike - this needs to be verified.

The time stamp rolls over at $FFFFFFFF.

11.1.6.3 Transmit Acknowledgment Description

Refer to Section 16 for a complete description of the transmit ack response to the client with and without time stamps.

11.1.6.4 Receive Message Description

Refer to Section 16 for a complete description of the receive message response to the client with and without time stamps.

11.2 Acceptance ID and Mask

11.2.1 Configuration

Each CAN channel of the AVT-423 is independent of all other channels.

Each CAN channel of the AVT-423 has 16 ($0 to $F) message objects.

For CAN0 and CAN1 each message object can be configured as either transmit or receive.

For CAN2 and CAN3 each message object is receive only.

Each message object (when or if) configured for receive can be set for 11 or 29-bit acceptance IDs.

Each message object (when or if) configured for receive has an associated acceptance ID mask.

Each bit of the mask can be set for "must match" or "don't care". The default is all bits are "must match". A '1' in a bit position means "must match".

The combination of the acceptance ID and associated mask give the user flexibility as to what messages are received by designated objects.

Note that there are a few differences between CAN0 / CAN1 and CAN2 / CAN3.

11.2.2 Acceptance ID and Mask Operation

Acceptance IDs and Masks are associated as pairs.

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

Using the '$75 18 …' form of the acceptance ID command implies that you are specifying an 11-bit ID and the condition of the IDE bit is '0'.

Using the '$77 18 …' form of the acceptance ID command implies that you are specifying a 29-bit ID and the condition of the IDE bit is '1'.

A one in a bit position in a mask is a Must Match condition for that bit in the acceptance ID.

A zero in a bit position in a mask is a Don't Care condition for that bit in the acceptance ID.

How the acceptance IDs and masks operate.

- A message is received from the network.
- The message ID is passed through the first enabled receive object.
  (Mask and acceptance ID.)
- If there is a match, the message is passed to the client.
- If no match, the process is repeated for the next enabled receive object.
- This continues until either a match is made or there are no more enabled receive objects.

11.2.3 CAN0 and CAN1 specifics

The user can specify the RTR (Remote Transmission Request) bit as part of the Acceptance ID. It is bit 6 of the channel byte of the '$7x 2A' command.

The user can not specify the IDE bit in the acceptance ID.

The user can not specify the IDE bit in the mask.

11.2.4 CAN2 and CAN3 specifics

The user can (but does not need to) specify the EDL (Extended Data Length) bit as part of the Acceptance ID. It is bit 5 of the channel byte of the '$7x 18' byte.

For the mask, the user can specify if the IDE bit is "don't care" or "must match". It is bit 7 of the channel byte of the '$7x 2C' command.

For the mask, the user can specify the EDL bit is “don’t care” or “must match”. It is bit 5 of the channel byte of the ‘$7x 2C’ command.

11.2.4.1 ID/Mask Example

Channel CAN0. Use object $04. Desired message is a 29-bit ID = 12 34 56 78, all bits are “must match”. RTR bit is 0 (do not receive RTR frames).

The following commands are used.

```
; set CAN0 ID# 04
77 2A 00 04 12 34 56 78

; set CAN0 Mask# 04
77 2C 00 04 1F FF FF FF
```

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.)

11.3 Setting up CAN0 or CAN1 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. Leave the CAN channel disabled during setup.
3. Set the CAN channel baud rate.
4. Set up the object(s) you want to use.
5. For each object:
   set the mode (receive or transmit),
   if receive, set the acceptance ID,
   if receive, set the Mask,
   enable that object.
6. Enable the CAN channel.

11.3.1 Communications Example

Set up CAN1, use object #0 for receive, object #5 for transmit.

```
; set CAN1 to 500 kbaud
73 0A 01 02

; set CAN1 ID0 = 07 E0
75 2A 01 00 07 E0

; set CAN1 Mask0 low order 4-bits are don’t care.
75 2C 01 00 07 F0

; enable CAN1 object #0 for receive
74 04 01 00 01

; enable CAN1 object #5 for transmit
74 04 01 05 02
```

; enable CAN1 for normal operations
73 11 01 01

; send a message with 5 bytes in the data field to ID = 07 80
09 01 05 07 80 04 11 22 33 44

; receive the transmit ack = 02 01 A5

; receive a message from the network = 0C 01 00 07 E3 05 AA BB CC DD EE 00 00

11.4 Setting up CAN2 or CAN3 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. Leave the CAN channel disabled during setup.
3. Set the CAN channel baud rate.
4. Set up the object(s) you want to use.
5. For each receive object:
   set the acceptance ID,
   set the Mask,
   enable that object.
6. Enable the CAN channel.

11.4.1 Communications Example

Set up CAN3, use object #A receive 11-bit ID messages of the form 07 Ex, transmit ID = 07 80, send one message, receive one message.

; set CAN3 to 1 Mbaud
73 0A 03 01

; set CAN3 IDA = 07 E0
75 2A 03 0A 07 E0

; set CAN3 MaskA low order 4-bits are don't care.
75 2C 03 0A 07 F0

; enable object $A for receive
74 04 03 0A 01

; enable CAN3 for normal operations
73 11 03 01

; send a message with 5 bytes in the data field to ID = 07 80
09 03 00 07 80 04 11 22 33 44

; receive the transmit ack = 02 03 A0

; receive a message from the network = 0C 03 0A 07 E3 05 AA BB CC DD EE 00 00

11.5 Periodic Message Support

At this time only Type1 periodic messages are supported. Type1 operations is the default for all periodic messages.

11.5.1 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.

11.5.1.1 Type1 Example

Want to send two messages on CAN2 at 500 kbaud. One message every 500 msec. The other message every 750 msec. Using CAN2 Type1 operations, here is a sequence of commands to do this.

1. ; Set CAN2 baud rate to 500 kbps
   73 0A 02 02
2. ; Enable CAN2 normal operations
   73 11 02 01
3. ; Define periodic message #01, ID = 0246, data = 03 A3 B4 C5
   79 18 02 01 02 46 03 A3 B4 C5
4. ; Set periodic message #01 for an interval count of 1000 = 1000 msec.
   75 1B 02 01 03 E8
5. ; Enable periodic message #01
   74 1A 02 01 01
6. CAN2, periodic message $01 is now active.
7. ; Define periodic message #06, ID = 0498, data = 04 1A 2B 3C 4D
   7A 18 02 06 04 98 04 1A 2B 3C 4D
8. ; Set periodic message #06 for an interval count of 500 = 500 msec.
   75 1B 02 06 01 F4.
9. ; Enable periodic message #06
   74 1A 02 06 01
10. CAN2, periodic message $06 is now active.

11.5.2 Type2 Periodic Messages

This feature/function has not yet been implemented in software.

Any periodic messages that are designated as Type2 are transmitted in sequence.

If any messages for a CAN channel are to be set up for Type2 operations, the first periodic message must be used and designated for Type2. The timer of the first periodic message is used as the timer for all Type2 messages on that channel.

For example, set-up periodic message $00 (the first message) ID and data field. Set PM # $00 timer increment for 100 msec. ($64). Set-up PM # $05 and $18 ID and data. Their timers are not used. Each time the timer for PM # $00 expires, the next periodic message will be queued for transmission. Thus, PM # $00, $05 $18 will transmitted in sequence. At the end of the sequence, the function loops back to the top.

11.5.3 Periodic Message Commands

All commands are listed in Section 15. A brief summary is provided here.

- 7x 18 Define a periodic message.
- 7x 1C Specify the object for the periodic message
  CAN0 and CAN1 only.
- 7x 1A Periodic message disable/enable.
- 7x 1B Periodic message timer increment.

11.6 Periodic Message Special Functions

These features/functions have not yet been implemented in software.

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.

11.6.1 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).

11.6.2 Special Function xxx

xxx

11.6.3 Special Function xxx

xxx

11.6.4 Special Function xxx

xxx

11.7 Periodic Pause Function

This function has not yet been implemented in software.

The Periodic Pause function, when enabled for a specific CAN channel, will inhibit all CAN periodic messages whenever an ISO 15765 transaction is in-progress. Note that this only applies to ISO 15765 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.

11.7.1.1 Periodic Pause Function Command

This function has not yet been implemented in software.

Refer to the 7x 1F command in Section 15 for detailed information regarding the command format.

11.8 ISO 15765 Support

This function has not yet been implemented in software.

12. LIN0 Operations

This function has not yet been implemented in software.

13. LIN1 operations

This function has not yet been implemented in software.

14. KWP operations – using LIN1 Hardware

This function has not yet been implemented in software.

15. 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, CAN1

0x qr 0s 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, 1, 2, 3.
- s: buffer number $0 to $F.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data.

0: CAN packet for transmission to the network.

CAN2, CAN3

0x qr 00 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. (only valid for FD frame)
    - 0: normal frame.
    - 1: RTR true, remote transmit request.
  - b5: 0
  - b4: 0
- r: channel: 0, 1, 2, 3.
- tt vv: 11-bit ID, right justified.
- tt vv ww zz: 29-bit ID, right justified.
- mm nn ...: data.

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

CAN0, CAN1

11 xx qr 0s 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
r: channel: 0, 1.
s: buffer number $0 to $F.
tt vv: 11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data.

CAN0, CAN1

12 xx yy qr 0s 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
r: channel: 0, 1.
s: buffer number $0 to $F.
tt vv: 11-bit ID, right justified.
tt vv ww zz: 29-bit ID, right justified.
mm nn ...: data.

Data byte count limitations CAN0 and CAN1

Maximum is 8 data bytes.

CAN2, CAN3

11 xx qr 00 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. (only valid for FD frame)
0: normal frame.

		1:	RTR true, remote transmit request.
	b5:	0	
	b4:	0	
r:	channel: 2, 3.		
00:	bits not used.		
tt vv:	11-bit ID, right justified.		
tt vv ww zz:	29-bit ID, right justified.		
mm nn ...:	data.		

CAN2, CAN3

12 xx yy qr 00 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.
	b6:	RTR. (only valid for FD frame)	
	b5:	0	
	b4:	0	
r:	channel: 2, 3.		
00:	bits not used.		
tt vv:	11-bit ID, right justified.		
tt vv ww zz:	29-bit ID, right justified.		
mm nn ...:	data.		

Data byte count limitations CAN2 and CAN3

Maximum is 64 bytes using EDL.

2: Reset.

21 10:	Reset CAN0.
21 11:	Reset CAN1.
21 12:	Reset CAN2.
21 13:	Reset CAN3.
21 20:	Reset LIN0.
21 21:	Reset LIN1.
21 30:	Reset Flexray.

3:

4:

5: Configuration.

CAN0, CAN1, CAN2, CAN3, LIN0, LIN1, KWP

52 8 0r Time stamp status query for channel.
  r channel: 0, 1, 2, 3, 7, 5, 6.
53 08 0x 0y: Disable / Enable time stamp for specified channel.
  r channel: 0, 1, 2, 3, 5, 6, 7.
  y: 0: Disable. [Default.]
    1: Enable.

CAN0, CAN1, CAN2, CAN3, LIN0, LIN1, KWP

52 40 0r Transmit ack status query for channel ‘r’.
  r channel: 0, 1, 2, 3, 7, 5, 6.
53 40 0r 0y: Disable / Enable transmit acks for specified channel.
  r channel: 0, 1, 2, 3, 7, 5, 6.
  y: 0 disable.
    1 enable. [Default.]

----
51 6A: Query for CPU heart beat LED blink rate.
53 6A yy zz: Set CPU heart beat LED blink rate.
  yy zz: LED half period time, msec.

6: Initialization

7: CAN configuration.

CAN0, CAN1, CAN2, CAN3

72 04 0r: Object query for CAN channel.
  r: channel: 0, 1, 2, 3.
74 04 0r 0y 0z Configure object for CAN channel.
  r: channel: 0, 1, 2, 3.
  y: buffer number 0 to F.
  z: 0: buffer disabled. [Default.]
    1: buffer enabled for receive.
    2: buffer enabled for transmit. (CAN0/1 only.)

CAN0, CAN1, CAN2, CAN3

72 0A 0r: Baud rate query for CAN channel.
  r channel: 0, 1, 2, 3.
73 0A 0r yy: Set baud rate for CAN channel.
  r channel: 0, 1, 2, 3.

	yy:	00:	user specified using 74 0B 0x rr ss command
		01:	1 Mbps.
		02:	500 Kbps. [Default.]
		03:	250 Kbps.
		04:	125 Kbps.
		0A:	33.333 Kbps.
		0B:	83.333 Kbps.
		0C:	2 Mbps. (CAN2 and CAN3 only)
		0D:	4 Mbps. (CAN2 and CAN3 only)
		0F:	8 Mbps. (CAN2 and CAN3 only)
74 0A 0r yy zz:	Set baud rate for CAN channel.		
	r:		channel: 2, 3.
	yy:	00:	user specified using 74 0B 0x rr ss command
		01:	1 Mbps.
		02:	500 Kbps. [Default.]
		03:	250 Kbps.
		04:	125 Kbps.
		0A:	33.333 Kbps.
		0B:	83.333 Kbps.
		0C:	2 Mbps.
		0D:	4 Mbps.
		0F:	8 Mbps.
	zz:	00:	user specified using 74 0B 0x rr ss command
		01:	1 Mbps.
		02:	500 Kbps. [Default.]
		03:	250 Kbps.
		04:	125 Kbps.
		0A:	33.333 Kbps.
		0B:	83.333 Kbps.
		0C:	2 Mbps.
		0D:	4 Mbps.
		0F:	8 Mbps.

CAN0, CAN1, CAN2, CAN3

72 0B 0r:	Query for Bit Timing Registers (BTR) of CAN channel.	
	r:	channel: 0, 1, 2, 3.
76 0B 0r ss tt vv ww:	Set Bit Timing Registers (BTR) of CAN channel.	
	r:	channel: 0, 1, 2, 3.
	rr ss:	Bit Timing Register 0.
	vv ww:	Bit Timing Register 1.

Contact the factory for a complete definition of the bit timing registers for CAN0/1 and CAN2/3.

CAN0, CAN1, CAN2, CAN3

72 11 0r:	Query for operational status query for CAN channel.			
	r:		channel: 0, 1, 2, 3.	
73 11 0r 0z:	Set operational mode for CAN channel.			
	r:		channel: 0, 1, 2, 3.	
	z:	0:	disabled. [Default for all CAN channels.]	
		1:	enabled for normal operations.	

CAN1

71 12:	Query for Single Wire CAN (SWC) transceiver status. CAN1 only.		
72 12 0y:	Set SWC transceiver mode. CAN1 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) format is different from the AVT-853 command set. Channel and message numbers are swapped.

CAN0, CAN1, CAN2, CAN3

73 18 0r pp:	Periodic message setup query.			
	r:	channel: 0, 1, 2, 3.		
	pp:	message number, $00 to $1F.		
7x 18 yr pp tt vv ww zz mm nn ...				Periodic message setup command.
	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	
	r:	channel: 0, 1, 2, 3.		
	pp:	message number, $00 to $1F.		
	tt vv:		11-bit ID, right justified.	
	tt vv ww zz:		29-bit ID, right justified.	
	mm nn ...:		data field.	

CAN0, CAN1

73 19 0r pp:	Query for object assignment for CAN periodic message.	
	r:	Channel: 0, 1.
	pp:	Message number, $00 to $1F.
74 19 0r pp 0y	Assign object to CAN periodic message.	
	r:	Channel: 0, 1.

pp: Message number, $00 to $1F.
y: Object number $0 to $F.

CAN0, CAN1, CAN2, CAN3

73 1A 0r zz: Periodic message disable/enable status query.
r: channel: 0, 1, 2, 3.
zz: message number, $00 to $1F.

74 1A 0r zz 0v: Periodic message disable/enable command.
r: channel: 0, 1, 2, 3.
zz: message number, $00 to $1F.
v: 0 disabled.
1 enabled.

CAN0, CAN1, CAN2, CAN3

73 1B 0r zz: Periodic message interval count status query.
r: channel: 0, 1, 2, 3.
zz: message number, $00 to $1F.

74 1B 0r zz vv ww: Periodic message interval count command.
r: channel: 0, 1, 2, 3.
zz: message number, $00 to $1F.
ww vv: interval count.

CAN0, CAN1, CAN2, CAN3, LIN0, LIN1, KWP

72 1C 0r Disable all periodic message for channel ‘r’.
r channel: 0, 1, 2, 3, 7, 5, 6.
r: ‘F’ – disable all messages, all channels.

CAN0, CAN1

73 2A 0r 0z: Report specified acceptance ID.
r: channel: 0, 1.
z: Acceptance ID number; $0 to $F.

75 2A yr 0z ss tt: Set acceptance ID.
y: b7: 0.
b6: RTR.
0: normal frame.
1: RTR true, remote transmit request.
b5: 0
b4: 0
r: channel: 0, 1.
z: Acceptance ID number; $0 to $F.
ss tt: Acceptance ID, 11-bit.

77 2A yr 0z ss tt vv ww: Set acceptance ID.
y: b7: 0.

b6: RTR.
0: normal frame.
1: RTR true, remote transmit request.
b5: 0
b4: 0
r: channel: 0, 1.
z: Acceptance ID number; $0 to $F.
ss tt vv ww: Acceptance ID, 29-bit.

CAN2, CAN3

73 2A 0r 0z: Report specified acceptance ID.
r: channel: 2, 3.
z: Acceptance ID number; $0 to $F.

75 2A yr 0z ss tt: Set acceptance ID.
y: b7: 0.
b6: 0.
b5: EDL bit (CAN-FD).
0: not CAN-FD frame.
1: is CAN-FD frame.
b4: 0
r: channel: 2, 3.
z: Acceptance ID number; $0 to $F.
ss tt: Acceptance ID, 11-bit.

77 2A yr 0z ss tt vv ww: Set acceptance ID.
y: b7: 0.
b6: 0.
b5: EDL bit (CAN-FD).
0: not CAN-FD frame.
1: is CAN-FD frame.
b4: 0
r: channel: 2, 3.
z: Acceptance ID number; $0 to $F.
ss tt vv ww: Acceptance ID, 29-bit.

CAN0, CAN1

73 2C 0r 0z: Report acceptance mask.
r: channel: 0, 1.
z: Mask number, $0 to $F.

75 2C 0r 0z ss tt: Specify 11-bit acceptance mask.
x: b7: 0.
b6: 0.
b5: 0.
b4: 0.
r: channel: 0, 1.
z: Mask number, $0 to $F.

ss tt: Mask value, 11-bit.
- 1: Bit must match.
- 0: Bit is don't care.

77 2C 0r 0z ss tt vv ww: Specify 29-bit acceptance mask.
- x:
  - b7: 0.
  - b6: 0.
  - b5: 0.
  - b4: 0.
- r: channel: 0, 1.
- z: Mask number, $0 to $F.
- ss tt vv ww: Mask value, 29-bit.
  - 1: Bit must match.
  - 0: Bit is don't care.

CAN2, CAN3

73 2C 0r 0z: Report specified mask.
- r: channel: 2, 3.
- z: Mask number, $0 to $F.

75 2C yr 0z ss tt: Specify 11-bit acceptance mask.
- x:
  - b7: IDE bit.
    - 1: bit must match.
    - 0: bit is don't care.
  - b6: 0.
  - b5: EDL bit.
    - 1: bit must match.
    - 0: bit is don't care.
  - b4: 0
- r: channel: 2, 3.
- z: Mask number, $0 to $F.
- ss tt: Mask value, 11-bit.
  - 1: Bit must match.
  - 0: Bit is don't care.

77 2C yr 0z ss tt vv ww: Specify 29-bit acceptance mask.
- x:
  - b7: IDE bit.
    - 1: bit must match.
    - 0: bit is don't care.
  - b6: 0.
  - b5: EDL bit.
    - 1: bit must match.
    - 0: bit is don't care.
  - b4: 0
- r: channel: 2, 3.
- z: Mask number, $0 to $F.
- ss tt vv ww: Mask value, 29-bit.
  - 1: Bit must match.

0: Bit is don't care.

CAN1

71 45:	Query for selected CAN1 physical layer.
72 45 01:	Set CAN1 physical layer to single wire CAN.
72 45 02:	Set CAN1 physical layer to 2-wire CAN. [Default.]

CAN2, CAN3

72 60 0r Query for status of extended data length padding.
r: channel: 2, 3.
73 60 0r 0y: Set status of extended data length padding.
r: channel: 2, 3.
y: 0: disabled. [Default.]
1: enabled.

CAN2, CAN3

72 61 0r Query for pad byte value.
r: channel: 2, 3.
73 61 0r yy: Set pad byte values.
r: channel: 2, 3.
yy: pad byte value. [Default = $EE.]

CAN0, CAN1, CAN2, CAN3

72 62 0r: Query for status of CAN bus termination.
r: channel: 0, 1, 2, 3.
73 62 0r 0y: Set CAN bus termination state.
r: channel: 0, 1, 2, 3.
y: 0: disabled.
1 enabled. [Default.]

8:

9:

A:

B: Software version.

B0:	Request software version number. (Same as B1 01.)
B1 01:	Request software version number. (Same as B0.)

B1 02: Request FPGA version number.

C:

D: Operational mode.

D0: Request operational mode report.

E: xxx.

F: Model Query and Reset

F0: Query for model number.

F1 A5: Restart the AVT-423 (a form of software reset).

15.1 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: <u>Transmit acknowledgements (if enabled).</u>

CAN0, CAN1
02 0r Az: Transmit ack.
r: channel number: 0, 1.
z: transmit buffer number.

CAN2, CAN3
02 0r A0: Transmit ack.
r: channel number: 2, 3.

CAN0, CAN1
06 00 00 jj kk 0r Az: Transmit ack.
00 00 jj kk: time stamp
r: channel number: 0, 1.
z: transmit buffer number.

CAN2, CAN3
06 jj kk ll mm 0r A0: Transmit ack.
jj kk ll mm: time stamp
r: channel number: 2, 3.

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

LIN0, LIN1, KWP
04 jj kk ll mm 0r pp: Transmit ack.
jj kk ll mm: time stamp
r: channel number: 7, 5, 6.
pp: receive status byte (defined below).

0: <u>CAN packet received from the network.</u>

CAN0, CAN1, CAN2, CAN3
0x jj kk ll mm qr tt vv ww zz nn pp ... :
x: count of bytes to follow.

jj kk ll mm:	time stamp [optional]		
q:	b7:	IDE.	
		0:	11-bit ID.
		1:	29-bit ID.
	b6:	RTR.	
		0:	normal frame.
		1:	RTR frame.
	b5:	0	
	b4:	0	
r:	channel: 0, 1, 2, 3.		
tt vv:	11-bit ID, right justified.		
tt vv ww zz:	29-bit ID, right justified.		
nn pp...:	data.		

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

CAN0, CAN1, CAN2, CAN3

11 xx jj kk ll mm qr tt vv ww zz nn pp... :

xx:	count of bytes to follow.		
jj kk ll mm:	time stamp [optional]		
q:	b7:	IDE.	
		0:	11-bit ID.
		1:	29-bit ID.
	b6:	RTR.	
		0:	normal frame.
		1:	RTR frame.
	b5:	0	
	b4:	0	
r:	channel: 0, 1, 2, 3.		
tt vv:	11-bit ID, right justified.		
tt vv ww zz:	29-bit ID, right justified.		
nn pp ...:	data.		

CAN0, CAN1, CAN2, CAN3

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

xx yy :	count of bytes to follow.		
jj kk ll mm:	time stamp [optional]		
q:	b7:	IDE.	
		0:	11-bit ID.
		1:	29-bit ID.
	b6:	RTR.	
		0:	normal frame.
		1:	RTR frame.
	b5:	0	
	b4:		
r:	channel: 0, 1, 2, 3.		
tt vv:	11-bit ID, right justified.		

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

2: Error Responses.

----
21 01 Inbound command too long, flushed.

----
21 02 FIFO2 too full, flushed and reset.

----
21 10 CAN0 initialization error.

----
21 11 CAN1 initialization error.

----
21 12 CAN2 initialization error.

----
21 13 CAN3 initialization error.

----
22 18 xx CAN dlc length error in 7x 18 periodic message command response.

----
23 1A yy zz UART0 initialization verification error.
yy: umr0.
zz: umr1.

----
22 20 yy Command processing time out.
yy: header of offending command.

----
22 34 yy Read command time out.
yy: header of offending command.

----
22 7F 03 Transmit command, invalid channel number.

----
22 7F 06 '0x' transmit command, 11-bit, command too short.

----

22 7F 07 '0x' transmit command, 11-bit, command too long.

----

22 7F 08 '0x' transmit command, 29-bit, command too short.

----

22 7F 09 '0x' transmit command, 29-bit, command too long.

----

22 7F 0A '12' transmit command, 11-bit ID, data length too long.
CAN2 or CAN3 only.

----

22 7F 0B '12' transmit command, 29-bit ID, data length too long.
CAN2 or CAN3 only.

----

22 7F 0C '12' transmit command, 11-bit ID, incorrect data length, padding disabled.
CAN2 or CAN3 only.

----

22 7F 0D '12' transmit command, 29-bit ID, incorrect data length, padding disabled.
CAN2 or CAN3 only.

----

22 7F 0E '12' transmit command; adl > dlc_lng.
CAN2 or CAN3 only.

----

22 7F 0F Transmit command, invalid buffer number.

----

22 7F 10 Transmit command, data length too short for edl = 9 (12 data bytes)
0x transmit command, 11-bit ID.
CAN2 or CAN3 only.

----

22 7F 11 Transmit command, data length too short for edl = 9 (12 data bytes)
0x transmit command, 29-bit ID.
CAN2 or CAN3 only.

----

21 84 Command buffer mode fault.

----

28 90 0z aa bb cc dd ee ff CANz error
z: CAN channel 0, 1.

aa bb: '1E' error register.
cc dd: '20' error register.
ee ff: '22' error register.

----

24 91 0z aa bb — CANz error
z: CAN channel 2, 3.
aa bb: error flag bit map.
b15:
b14:
b13:
b12:
b11:
b10:
b9:
b8:
b7: bus off.
b6: bus warning.
b5: error counter ii overrun.
b4: FD protocol exception.
b3: RM protocol exception.
b2: transmit fifo overflow.
b1: receive fifo overflow.
b0: edl is clear, dlc > 8.

----

22 92 0z — CANz bus off warning.
z: CAN channel 0, 1, 2, 3.

----

23 93 0z aa — CANz lost frame counter.
z: CAN channel 0, 1, 2, 3.
aa: lost frame counter.

----

22 E5 01 — LIN0 transmit command too short.

----

22 E6 01 — KWP transmit command too short.

----

22 E7 01 — LIN1 transmit command too short.

----

22 E9 01 — Flexray transmit command too short.

3: Command error.

----

31 yy — Command error.
yy: header of offending command.

----

32 yy FF — Command not processed.
yy: header of offending command.

----

32 xx yy — No such transmit channel number.
xx: header byte of offending command.
yy: channel number.

4:

5:

6: Configuration reports.

CAN0, CAN1, CAN2, CAN3, LIN0, LIN1, KWP

63 08 0r 0y: Time stamp status.
r channel: 0, 1, 2, 3, 7, 5, 6.
y: 0 disabled. [Default.]
1 enabled.

CAN0, CAN1, CAN2, CAN3, LIN0, LIN1, KWP

63 40 0r 0y: Send transmit acknowledgements to client.
r channel: 0, 1, 2, 3, 7, 5, 6.
y: 0 disabled.
1 enabled. [Default.]

----

63 6A yy zz: CPU heart beat LED blink rate.
yy zz: LED half period time, msec.

7: Initialization attempt response.

8: CAN configuration reports.

CAN0, CAN1, CAN2, CAN3

83 04 0r 0y — Configuration of CAN channel object.

- r — channel: 0, 1, 2, 3.
- y:
  - 0: object disabled. [Default.]
  - 1: object enabled for receive.
  - 2: object enabled for transmit. (CAN0/1 only.)

CAN0, CAN1, CAN2, CAN3

83 0A 0r yy: — Baud rate for CAN channel.

- r — channel: 0, 1, 2, 3.
- yy:
  - 00: user specified using 74 0B 0x rr ss command
  - 01: 1 Mbps.
  - 02: 500 Kbps. [Default.]
  - 03: 250 Kbps.
  - 04: 125 Kbps.
  - 0A: 33.333 Kbps.
  - 0B: 83.333 Kbps.
  - 0C: 2 Mbps. (CAN2 and CAN3 only)
  - 0D: 4 Mbps. (CAN2 and CAN3 only)
  - 0F: 8 Mbps. (CAN2 and CAN3 only)

CAN2, CAN3

84 0A 0r yy zz: — Baud rate for CAN channel.

- r: channel: 2, 3.
- yy:
  - 00: user specified using 74 0B 0x rr ss command
  - 01: 1 Mbps.
  - 02: 500 Kbps. [Default.]
  - 03: 250 Kbps.
  - 04: 125 Kbps.
  - 0A: 33.333 Kbps.
  - 0B: 83.333 Kbps.
  - 0C: 2 Mbps.
  - 0D: 4 Mbps.
  - 0F: 8 Mbps.
- zz:
  - 00: user specified using 74 0B 0x rr ss command
  - 01: 1 Mbps.
  - 02: 500 Kbps. [Default.]
  - 03: 250 Kbps.
  - 04: 125 Kbps.
  - 0A: 33.333 Kbps.
  - 0B: 83.333 Kbps.
  - 0C: 2 Mbps.
  - 0D: 4 Mbps.

0F: 8 Mbps.

CAN0, CAN1, CAN2, CAN3

86 0B 0r ss tt vv ww: Bit Timing Registers (BTR) of CAN channel.

r: channel: 0, 1, 2, 3.

rr ss: Bit Timing Register 0.

vv ww: Bit Timing Register 1.

Contact the factory for a complete definition of the bit timing registers for CAN0/1 and CAN2/3.

CAN0, CAN1, CAN2, CAN3

83 11 0r 0z: Operational mode for CAN channel.

r: channel: 0, 1, 2, 3.

z: 0: disabled. [Default for all CAN channels.]

1: enabled for normal operations.

CAN1

82 12 0y: SWC transceiver mode. CAN1 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) format is different from the AVT-853 command set.

CAN0, CAN1, CAN2, CAN3

8x 18 yr pp tt vv ww zz mm nn ... Periodic message setup.

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

r: channel: 0, 1, 2, 3.

pp: message number, $00 to $1F.

tt vv: 11-bit ID, right justified.

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

mm nn ...: data field.

CAN0, CAN1

84 19 0r pp 0y — Object assigned to CAN periodic message.
- r: Channel: 0, 1.
- pp: Message number, $00 to $1F.
- y: Object number $0 to $F.

CAN0, CAN1, CAN2, CAN3

84 1A 0r zz 0v: — Periodic message disable/enable status.
- r: channel: 0, 1, 2, 3.
- zz: message number, $00 to $1F.
- v: 0 disabled.
  1 enabled.

CAN0, CAN1, CAN2, CAN3

84 1B 0r zz vv ww: — Periodic message interval count.
- r: channel: 0, 1, 2, 3.
- zz: message number, $00 to $1F.
- ww vv: interval count.

CAN0, CAN1, CAN2, CAN3, LIN0, LIN1, KWP

62 1C 0r — All periodic message for channel ‘r’ disabled.
- r: channel: 0, 1, 2, 3, 7, 5, 6.
- r: ‘F’ – all messages, all channels disabled.

CAN0, CAN1, CAN2, CAN3

85 2A yr 0z ss tt: — Acceptance ID.
- y:
  - b7: 0.
  - b6: RTR.
    - 0: normal frame.
    - 1: RTR true, remote transmit request.
  - b5: 0
  - b4: 0
- r: channel: 0, 1.
- z: Acceptance ID number; $0 to $F.
- ss tt: Acceptance ID, 11-bit.

87 2A yr 0z ss tt vv ww: — Acceptance ID.
- y:
  - b7: 0.
  - b6: RTR.
    - 0: normal frame.
    - 1: RTR true, remote transmit request.
  - b5: 0

b4: 0
r: channel: 0, 1.
z: Acceptance ID number; $0 to $F.
ss tt vv ww: Acceptance ID, 29-bit.

CAN2, CAN3

85 2A yr 0z ss tt: Acceptance ID.
y: b7: 0.
b6: 0.
b5: EDL bit (CAN-FD).
0: not CAN-FD frame.
1: is CAN-FD frame.
b4: 0
r: channel: 2, 3.
z: Acceptance ID number; $0 to $F.
ss tt: Acceptance ID, 11-bit.

87 2A yr 0z ss tt vv ww: Acceptance ID.
y: b7: 0.
b6: 0.
b5: EDL bit (CAN-FD).
0: not CAN-FD frame.
1: is CAN-FD frame.
b4: 0
r: channel: 2, 3.
z: Acceptance ID number; $0 to $F.
ss tt vv ww: Acceptance ID, 29-bit.

CAN0, CAN1

85 2C 0r 0z ss tt: 11-bit acceptance mask.
x: b7: 0.
b6: 0.
b5: 0.
b4: 0.
r: channel: 0, 1.
z: Mask number, $0 to $F.
ss tt: Mask value, 11-bit.
1: Bit must match.
0: Bit is don't care.

87 2C 0r 0z ss tt vv ww: 29-bit acceptance mask.
x: b7: 0.
b6: 0.
b5: 0.
b4: 0.
r: channel: 0, 1.
z: Mask number, $0 to $F.

ss tt vv ww: Mask value, 29-bit.
1: Bit must match.
0: Bit is don't care.

CAN2, CAN3

85 2C yr 0z ss tt: 11-bit acceptance mask.
x: b7: IDE bit.
1: bit must match.
0: bit is don't care.
b6: 0.
b5: EDL bit.
1: bit must match.
0: bit is don't care.
b4: 0
r: channel: 2, 3.
z: Mask number, $0 to $F.
ss tt: Mask value, 11-bit.
1: Bit must match.
0: Bit is don't care.

87 2C yr 0z ss tt vv ww: 29-bit acceptance mask.
x: b7: IDE bit.
1: bit must match.
0: bit is don't care.
b6: 0.
b5: EDL bit.
1: bit must match.
0: bit is don't care.
b4: 0
r: channel: 2, 3.
z: Mask number, $0 to $F.
ss tt vv ww: Mask value, 29-bit.
1: Bit must match.
0: Bit is don't care.

CAN1

82 45 01: Set CAN1 physical layer to single wire CAN.
82 45 02: Set CAN1 physical layer to 2-wire CAN. [Default.]

CAN2, CAN3

83 60 0r 0y: Status of extended data length padding.
r: channel: 2, 3.
y: 0: disabled. [Default.]
1: enabled.

CAN2, CAN3

83 61 0r yy: Pad byte value.
r: channel: 2, 3.
yy: pad byte value. [Default = $EE.]

CAN0, CAN1, CAN2, CAN3

83 62 0r 0y: CAN bus termination state.
r: channel: 0, 1, 2, 3.
y: 0: disabled.
1 enabled. [Default.]

9: Board status information.

91 01 10: CAN0 reset complete.

91 01 11: CAN1 reset complete.

91 01 12: CAN2 reset complete.

91 01 13: CAN3 reset complete.

92 01 20: LIN0 reset complete.

92 01 21: LIN1 reset complete.

92 01 30: FLEXRAY reset complete.

93 04 xx yy: Software version report. Version is xx yy.

93 28 0x yz: Model number report. xyz is the model number.

91 3A: AVT-423 Ethernet connect and operating report.

94 3B xx yy zz: FPGA version number report.

A:

B:

C:

D:

E:

F:

16. Appendix A

xxx

17. Appendix B

Xxx

18. 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

AVT-423 specific web pages:

Set IP address
Update Operation Software
Operation Software file
http://www.AVT-HQ.com/dowload.htm#AVT-423

Operation Software Version Descriptions
http://www.AVT-HQ.com/423_sw.htm

Change / Version Notes

0003: Initial release.

0004: Changed LIN0 to channel 7 and LIN1 to channel 5. This puts LIN0, LIN1, and KWP into the same associations as the AVT-853. Other minor corrections in the command and responses sections.

0005: LIN0 and LIN1 are now operational. Corrected and updated various sections. Reviewed, updated, and corrected some commands and responses.