

CAN Mode Periodic Messages

Advanced Vehicle Technologies interface unit model numbers: AVT-418, 512, 717, and 718 all support a single channel of CAN (Controller Area Network) protocol operations.

Models 418, 512, 717, and 718 support Type0 and Type1 periodic messages.

Models 418 and 718 also support Type2 periodic messages (firmware version 6.6 and above).

Periodic Messages

The periodic message functions provide a way to set up an AVT interface unit so that the interface unit will automatically transmit one or more CAN messages without additional host intervention.

There are three different types of periodic message functions. They are called: Type0; Type1; and Type2.

Caution

The three different types of periodic messages are very different in operation. Each method has advantages and disadvantages. The user should select the appropriate type to use for the particular application at hand.

Conflicts can arise, depending on the situation. You should be aware of these possible conflicts and avoid them.

Special note: At no time should more than one object be enabled to transmit with same ID.

Definitions

A message object is set of registers (or a mailbox) within the CAN controller device.

For transmit operations there are 14 (decimal) message objects available.

They are numbered \$01 to \$0E (in hex).

[All AVT interfaces that support CAN use the Intel 82527 CAN controller device.]

Type0 Periodic Messages

Type0 periodic messages use the CAN controller message objects directly. The complete message is stored in the CAN controller device. When the timer for the enabled message object expires, the object is trigger to transmit and the timer is reset.

To use the Type0 periodic message function, perform the following steps:

- Define each message object that you want to use. (Valid objects are \$01 to \$0E.)
- Set up the object with all parameters (message ID or arbitration field, standard or extended, receive or transmit, and data length. (7x 05 command)
- Load the data into the object. (7x 06 command)
- Enable the object. (73 04 command)
- Set the periodic rate. (7x 15 command)
- Enable the Type0 function for the object. (7x 14 command)

No transmission report (8x 09) is generated when a Type0 periodic message is transmitted.

RTR's are permitted. (RTR - Remote Transmission Request.)

If you transmit a message using the short form method using an object enabled for Type0 operations you will overwrite the settings for that object. The settings will remain in that condition until (and if) you write to that object again. It is also possible that a corrupted message will be transmitted onto the CAN network.

Type1 Periodic Messages - Independent

Type1 periodic messages are stored in memory. They are designated by message number. When the timer for that message number expires; the message is loaded into the designated object in the CAN controller; the object is triggered to transmit; and the timer for that message number is reset.

Message numbers \$01 to \$0A are assigned to object #1 and known as Group1.

Message numbers \$0B to \$14 are assigned to object #2 and known as Group2.

The timers for all messages (\$01 to \$14) are all independent. Thus, the periodic rate for every defined and enabled message is independent.

If you want to transmit a message using an object that is also being used to support Type1 operations (objects #1 and/or #2), you must use the "short form" transmit command. Your command will be processed as soon as that object become available. No messages are destroyed.

To use the Type1 periodic message function, perform the following steps:

- Define the message number you want to use. Valid message numbers are \$01 thru \$14.
- Initialize the message number with all parameters (message ID or arbitration field, standard or extended, receive or transmit, and data length. (7x 18 command)
- Load the data into the message number. (7x 19 command)
- Set the periodic rate. (7x 1B command)
- Set the Group for independent operations. (7x 0C command)
- Enable the message number for operations. (7x 1A command)

No transmission report (8x 09) is generated when a Type1 periodic message is transmitted.

RTR's are permitted. (RTR - Remote Transmission Request.)

Possible conflict: Type1 periodic messages use message object #1 and #2. Do not program message object #1 or #2 for Type0 operations when Type1 operations are also enabled.

More information about the commands mentioned can be found later in this document and in the "Master Commands and Responses" document. The most current revision is available from our web site at: www.AVT-HQ.com/download.htm#Notes

Type2 Periodic Messages - Sequential

Type2 periodic messages use the same structure as Type1; they are stored in memory and assigned to a specific CAN object.

Type2 messages are only transmitted in sequence.

Message numbers \$01 to \$0A are assigned to object #1 and known as Group1.

For Group1 messages:

- Timer01 sets the time interval between the transmission of each message, in sequence.
- Timer02 sets the time interval from the last message in the group to the first message in the group.

Message numbers \$0B to \$14 are assigned to object #2 and known as Group2.

For Group2 messages:

- Timer0B sets the time interval between the transmission of each message, in sequence.
- Timer0C sets the time interval from the last message in the group to the first message in the group.

The Group1 sequence is independent of Group2.

If you want to transmit a message using an object that is also being used to support Type1 operations (objects #1 and/or #2), you must use the "short form" transmit command. Your command will be processed as soon as that object become available. No messages are destroyed.

A Group can be enabled for Type1 or Type2 operations, but not both.

To use the Type2 periodic message function, perform the following steps:

- Define the message numbers you want to use.
 - Group1 valid message numbers are \$01 thru \$0A.
 - Group2 valid message numbers are \$0B thru \$14.
- Initialize each message number with all parameters (message ID or arbitration field, standard or extended, receive or transmit, and data length. (7x 18 command)
- Load the data into the message number. (7x 19 command)
- Set the sequential interval; Timer01 if Group1 or Timer0B if Group2. (7x 1B command)
- Set the repeat interval; Timer02 if Group1 or Timer0C if Group2. (7x 1B command)
- Set the Group for sequential operations. (7x 0C command)
- Enable or disable the time delay for messages that are disabled. (7x 0D command)
- Enable the message number for operations. (7x 1A command)

No transmission report (8x 09) is generated when a Type1 periodic message is transmitted.

RTR's are permitted. (RTR - Remote Transmission Request.)

Possible conflict: Type1 periodic messages use message object #1 and #2. Do not program message object #1 or #2 for Type0 operations when Type1 operations are also enabled.

More information about the commands mentioned can be found later in this document and in the "Master Commands and Responses" document. The most current revision is available from our web site at: www.AVT-HQ.com/download.htm#Notes

Timer Interval

There is one global periodic timer interval. The default interval is 10 milliseconds.

The 7x 1E command permits selecting the timer interval to either 5 or 10 milliseconds.

The 7x 15 command sets the count for each message object (Type0).

The 7x 1B command sets the count for each message number (Type1 and Type2).

The message periodic rate is dependent on the count value and the timer interval setting. A count of \$64 (100 decimal) with the timer set to 10 milliseconds results in message transmission every 1.00 seconds. With that same count (\$64) and the timer set to 5 milliseconds the message transmission interval is 0.50 seconds.

Performance

Type0 periodic messages are generally faster in operation. The CAN controller message object is pre-loaded and is simply triggered when the timer for that object expires.

The timer for Type0 periodic messages is one byte with values from \$01 to \$FF resulting in timing ranges of 10 milliseconds to 2.55 seconds or 5 milliseconds to 1.275 seconds; depending on the setting of the timer interval.

Type1 and Type2 periodic messages are a little bit slower since for each transmission the message must be loaded into the CAN controller and then triggered for transmission.

[You will probably never be able to tell. Maybe on the order of tens to, at most, hundreds of microseconds.]

The timer for Type1 and Type2 periodic messages is two bytes with values from \$0001 to \$FFFF resulting in timing ranges of 10 milliseconds to 655.35 seconds or 5 milliseconds to 327.675 seconds; depending on the setting of the timer interval.

7x 0C Command

This command sets the groups for Type1 or Type2 operations.

- 71 0C: status query.
- 72 0C 00: Group1 - Type1 [default]
Group2 - Type1 [default]
- 72 0C 01: Group1 - Type2
Group2 - Type1
- 72 0C 02: Group1 - Type1
Group2 - Type2.
- 72 0C 03: Group1 - Type2
Group2 - Type2

7x 0D Command

During Type2 operations, sequential periodic, there are two ways to handle message numbers that are not enabled. Either skip that message or wait.

If a message number is enabled, the time delay before and after it is transmitted is set by Timer01 (if Group1) or Timer0B.

If the message number is disabled the interface can either skip that message or wait (as if it were enabled).

- 71 0D: status query
- 72 0D 00: Do not wait for disabled message numbers. (default)
- 72 0D 01: Wait for every message, regardless if it is enabled or not.

7x 14 Command

- 72 14 xx: Type0 periodic message status query. xx - object number.
 - 73 14 xx 00: Type0 periodic function disabled for object xx.
 - 73 14 xx 01: Type0 periodic function enabled for object xx.
- Note: Valid object numbers: \$01 to \$0E.

7x 15 Command

- 72 15 xx: Type0 periodic message interval query. xx - object number.
- 73 15 xx yy: Type0 periodic message interval command.
xx - object number, \$01 to \$0E.
yy - time interval in 5 or 10 millisecond increments.
(Time increment set by 7x 1E command.)

7x 16 Command

- 71 16: Disable all Type0 periodic messages.

7x 18 Command

7x 18 xx yy zz rr aa bb cc ...

Type1 and 2 periodic message setup.

xx - message number (\$1 to \$14).

yy - 01 = receive; 10 = transmit.

zz - 01 = 11-bit; 10 = 29-bit.

rr - data length, updated when data field is written

(7x 19 command).

Arbitration field is right justified.

aa bb: 11-bit arbitration field.

aa bb cc dd: 29-bit arbitration field.

7x 19 Command

72 19 xx: Type1 and 2 periodic message data query.

xx - message number (\$1 to \$14).

7x 19 xx aa bb cc ... : Type1 and 2 periodic message data.

xx - message number (\$1 to \$14).

aa bb cc ... - data field, variable length.

7x 1A Command

72 1A xx: Type1 and 2 periodic message status query.

xx - message number \$01 to \$14.

There are two forms of the 7x 1A enable / disable command.

73 1A xx yy Enable or disable individual message number.

xx = message number to enable or disable.

yy = 00 to disable, 01 to enable.

74 1A gg xx yy Access the mask directly.

gg = 01 for Group 1; 02 for Group2.

xx yy is the enable / disable mask.

bit 0 is always zero.

bits 1 thru 10 are used to enable disable message numbers \$01 thru \$0A; Group1.

bits 1 thru 10 are used to enable disable message numbers \$0B thru \$14; Group2.

bits 11 thru 15 are always zero.

Examples:

Disable all Group1 messages (operating as Type1 or 2): 74 1A 01 00 00.

You have already set up message numbers \$1, \$3, \$4, and \$9.

Enable those Group1 messages (operating as Type 1 or 2): 74 1A 01 02 1A.

7x 1B Command

72 1B xx: Type1 or 2 periodic message interval query.
xx - message number \$01 to \$14.

74 1B xx rr ss: Type1 or 2 periodic message interval command.
xx - message number, \$01 to \$14.
rr ss -time interval in 5 or 10 millisecond increments.
(Time increment set by 7x 1E command.)

7x 1C Command

71 1C: Disable all Type1 and 2 periodic messages; both Group1 and Group2.

7x 1E Command

71 1E: Periodic message timer query.

72 1E 00: Timer interval = 5 milliseconds.

72 1E 01: Timer interval = 10 milliseconds. {Default}

A Keep Alive Example

You need to send the module under test a “Keep Alive” message every second. You also need to send the module specific messages during the testing. All of these messages use the same message ID (or arbitration field). How to do this ??

Wrong Method

- Declare object #1 as a transmit object and use it to transmit your test messages.
- Declare object #2 as a transmit object. Load it with the keep alive message. Set object #2 for Type0 periodic message.

This won't work, because:

- You must never set up and have enabled more than one transmit object with the same message ID (arbitration field) at the same time. It is possible and likely that you will turn the AVT interface unit into a “babbling idiot”. [I won't go into the details here of why. If you don't understand, write or call me and I'll explain.]

Solution #1

- Handle both the keep alive and test messages in your application software.
- This will work but could be cumbersome and/or difficult to implement.

Solution #2 - A better method

- Set up Type1 periodic message #1 (which uses object #1) as your keep alive message.
- Send your test messages out object #1 using the “short form” transmit command.
- The AVT interface will prevent either message from being lost or corrupted.

Solution #2 - Example

1. F1 A5 ; reset the interface unit
2. E1 99 ; switch to CAN mode
3. 72 11 02 ; two wire CAN-C
4. 72 0A 04 ; 125 kbps
5. 77 18 01 10 01 00 07 44 ; Type1 periodic message #1 setup
6. 76 19 01 68 6A F1 3F ; Type1 periodic message #1 data
7. 72 1E 01 ; Type1 timer interval 10 msec.
8. 74 1B 01 00 64 ; Type1 periodic message #1 to 100 = 1000 msec.
9. 73 1A 01 01 ; Type1 periodic message #1 enabled

At this point the AVT interface will send the programmed message using object #1 once per second until reset or disabled.

Send your test messages using object #1 and the short form transmit command.

10. 0B 01 07 44 01 02 03 04 05 06 07 08 ; test message with dummy data of: 01 - 08

Caution

[Do not use the 7x 05, 7x 06, 7x 04, 7x 07 command sequence to send your test messages. It will not work properly.]