30 July 2014

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 types of periodic messages operate very differently. 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).

[AVT interfaces AVT-418, 512, 717, 718 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.) This is done by setting an object to receive when setting up the periodic message. When the object is triggered to transmit and RTR will go out.

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 periodic 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.

Periodic message numbers $01 to $10 are assigned to object #1 and known as Group1.

Periodic message numbers $11 to $20 are assigned to object #2 and known as Group2.

The timers for all messages ($01 to $20) are 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 lost or 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 $20.
- 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.

Periodic message numbers $01 to $10 are assigned to object #1 and known as Group1.
For Group1 messages:

- Timer $01 sets the time interval between the transmission of each message, in sequence.
- Timer $02 sets the time interval from the last message in the group to the first message in the group.

Periodic message numbers $11 to $20 are assigned to object #2 and known as Group2.
For Group2 messages:

- Timer $11 sets the time interval between the transmission of each message, in sequence.
- Timer $12 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 lost or 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 $10.
  Group2 valid message numbers are $11 thru $20.
- 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; Timer $01 if Group1 or Timer $11 if Group2. (7x 1B command)
- Set the repeat interval; Timer $02 if Group1 or Timer $12 if Group2. (7x 1B command)
- Set the Group for sequential operations. (7x 0C command)

- Enable or disable (as desired) 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 2, 5, or 10 milliseconds.

The 7x 15 command sets the count for each Type0 periodic message.

The 7x 1B command sets the count for each Type1 or Type2 periodic message.

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 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. The additional processing time is on the order of tens to, at most, a few hundred 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 Timer $01 (if Group1) or Timer $11 (if Group2).

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 set up.

xx - message number ($01 to $20).
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 ($01 to $20).

7x 19 xx aa bb cc ... : Type1 and 2 periodic message data.
xx - message number ($01 to $20).
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 $20.

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
gg = 02 for Group2
xx yy is the enable / disable bit mask
the 16 bits are used to enable disable message numbers $01 thru $10; Group1
the 16 bits are used to enable disable message numbers $11 thru $20; Group2

Examples:

Disable all Group1 messages (operating as Type1 or 2): 74 1A 01 00 00.
You have already set up message numbers $01, $03, $04, and $09.
Enable those Group1 messages (operating as Type 1 or 2): 74 1A 01 01 0D.

7x 1B Command

72 1B xx:	Type1 or 2 periodic message interval query. xx - message number $01 to $20.
74 1B xx rr ss:	Type1 or 2 periodic message interval command. xx - message number, $01 to $20. 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}
72 1E 02:	Timer interval = 2 milliseconds.

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 set up
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.]

Appendix A - Counter and Checksum Function

This section only applies to the AVT-718 (and AVT-418) firmware.

On special request, AVT-718 firmware version 7.1 (0A) was developed with a new function for CAN periodic messages. It is named the ‘Counter and Checksum’ function.

Every CAN Type 1 or Type 2 periodic message can be independently set up with a programmable rolling counter and checksum in the data field.

For reference:		The bytes in the CAN data field are labeled as follows:

ID (11 or 29 bit)	data 0	data 1	data 2	data 3	data 4	data 5	data 6	data 7

There are two new commands; one to set up the function and the other to enable or disable the function. These are described in the following sections. Two examples follow at the end of this Appendix.

To use this new function, the user specifies:

- Periodic message number; $01 to $20.
- Size of the counter; 2, 3, or 4 bits.
- Location of the counter; byte number and bit number within that byte (of the least significant bit of the counter).
- Type of checksum to apply; 8-bit sum, 4-bit sum, 4-bit XOR.
- Location of the checksum; byte number and upper or lower nibble.

Counter Notes

Each periodic message has its own programmable counter. The counter is incremented each time that periodic message is transmitted. The counter rolls over when it reaches the terminal count.

The 4-bit counter sequence is: $0 to $F.
The 3-bit counter sequence is: $0 to $7.
The 2-bit counter sequence is: $0 to $3.

Checksum Notes

The user can choose from three different checksum methods.

The checksum algorithms were developed based on examples provided by the customer that referenced: “25953NCS14/42”.

Each checksum method is described here.

8-Bit Sum

Right justify the 11-bit ID.

Sum the following:
byte containing the upper 3-bits of the ID
byte containing the lower 8-bits of the ID
all data bytes up to the location of the checksum

Sum is modulo 256 (no carry).

The checksum is the lower 8-bits of that result.

4-Bit Sum

Right justify the 11-bit ID.

Sum the following:
- byte containing the upper 3-bits of the ID
- byte containing the lower 8-bits of the ID
- all data bytes up to the location of the checksum

Sum is modulo 256 (no carry).

Then sum the upper nibble of the result with the lower nibble of the result.
This sum is modulo 16 (no carry).

If the checksum is to be placed into the lower nibble of a data byte, then the upper nibble of that location is added to the sum computed just above.

The checksum is that 4-bit result.

4-Bit XOR (bit-wise exclusive OR)

Using XOR as the summation operator (not arithmetic sum),
sum the following:
- all the data bytes up to the location of the checksum

Then, XOR the upper and lower nibble of that result.

```
If
    The checksum is located in the upper nibble of a byte,
    XOR the lower nibble of that byte with the result above.
Else
    The checksum is located in the lower nibble of a byte,
    XOR the upper nibble of that byte with the result above.
```

The checksum is that 4-bit result.

Configuration Notes

In general, it would seem logical that the user would specify the location of the counter to be before the location of the checksum. Regardless, the user has complete freedom to place the counter and checksum anywhere in the CAN data field.

When the data field of the periodic message is defined (using the 7x 19 command), be sure to put dummy bytes in the message so that the data field is the correct length. The ‘Counter and Checksum’ function will overwrite the bytes as specified in the function set up command (7x 4B command). The ‘Counter and Checksum’ function does not examine or change the length of the data field of the periodic message.

7x 4B Set Up Command - 'Counter and Checksum'

72 4B xx:	Query for the set up of periodic message 'xx'.	
75 4B xx rs yz tv:	Set up the function.	
	xx:	periodic message number; $01 to $20
	r:	counter size 2 = 2-bit counter 3 = 3-bit counter 4 = 4-bit counter
	s:	checksum type 1 = 8-bit sum 2 = 4-bit sum 3 = 4-bit XOR
	y:	counter location, the least significant bit of the counter 0 to 7 (eg. 0 places the counter in the low nibble).
	z:	counter location, byte byte number in CAN data field; $0 to $7.
	t:	checksum location, nibble 0 = lower nibble 1 = upper nibble
	v:	checksum location, byte byte number in CAN data field; $0 to $7.

7x 4A Enable / Disable Command - 'Counter and Checksum'

72 4A xx:	Query for the 'Counter and Checksum' function status for periodic message 'xx'. xx = $01 to $20 xx = $FF, then all bit flags are reported. (See the 75 4A command below)
73 4A xx 00:	Disable the 'Counter and Checksum' function for periodic message 'xx'.
73 4A xx 01:	Enable the 'Counter and Checksum' function for periodic message 'xx'.
75 4A FF rr ss tt vv:	Writes all bit flags for the 'Counter and Checksum' function. The 32-bits correspond to the 32 (decimal, $20 hex) periodic messages. Lowest order bit is the 'Counter and Checksum' enable bit for periodic message $01. Likewise, the highest order bit is for periodic message $20.

Example #1

Want to use periodic message #13 ($0D), with 4-bit counter in bits 3:0 (lower nibble) of byte 6 with the checksum in byte 7 using the 8-bit sum method.

75 4B 0D 41 06 07

Note that the nibble location of the checksum does not matter.

Example #2

Want to use periodic message #23 ($17), with 2-bit counter in bits 5:4 of byte 7 with the checksum in the lower nibble of byte 7 using the 4-bit XOR method.

75 4B 17 23 47 07

Error Responses

There are two new error responses that the AVT-718 will issue to the host if a problem is encountered during operations.

Group1 error response

```
23 8D 0x 0y
        x = 1 if the counter size is invalid
        x = 2 if the checksum type is invalid

        y = the periodic message number in Group1 that had the problem
                This periodic message is then disabled;  both as a periodic message
                and the 'Counter and Checksum' function.
```

Group2 error response

```
23 8E 0x 0y
        x = 1 if the counter size is invalid
        x = 2 if the checksum type is invalid

        y = the periodic message number in Group2 that had the problem
                This periodic message is then disabled;  both as a periodic message
                and the 'Counter and Checksum' function.
```

Appendix B

Rolling Nibble Function

AVT-718 firmware version 6.9 (B) was never released for distribution. It was a test version given only to one customer.

In that version was a function called ‘Rolling Nibble’. In that function the counter and checksum were fixed and only available for periodic messages $0A and $14.

In AVT-718 firmware version 7.0 (0E), the ‘Rolling Nibble’ function is still available. This was done to be backward compatible for existing software.

There is was one command associated with this function; described below.

‘Rolling Nibble’ Function Description

A rolling 4-bit counter is inserted in the lower nibble of data byte 5 of the CAN data field.

The checksum is an 8-bit sum, modulo 256 (no carry), of data bytes 0 to 5. The checksum byte is inserted into data byte 6 of the CAN data field.

7x 1D Enable / Disable Command - ‘Rolling Nibble’

71 1D:	Query for the ‘Rolling Nibble’ status for periodic messages $0A and $14.
72 1D xx:	Query for the ‘Rolling Nibble’ status for periodic message ‘xx’. xx = $0A or $14 (only)
73 1D xx 00:	Disable the ‘Rolling Nibble’ function for periodic message ‘xx’. xx = $0A or $14 (only)
73 1D xx 01:	Enable the ‘Rolling Nibble’ function for periodic message ‘xx’. xx = $0A or $14 (only)