Диагностика по CAN

Запросы, посылаемые по шине CAN к блокам управления

Стандарт ISO 15765-3

Команда инициализации диагностики

N_PDU type	29 bit CAN Identifier								CAN frame data field							
	28 26	25	24	23 16	15 8	7 0	1	2	3	4	5	6	7	8
SingleFrame	110_2	0	0	218_{10}	TA	SA	PCI	Data						

Короткое сообщение (менее 8 байт)

TA (Target Address) – адрес блока управления, к которому посылается запрос

SA (Source Address) – адрес блока управления, посылающего сообщение

PCI – количество байтов информации в посылаемом запросе

DATA – собственно команда (смотри описания команд далее)

Пример:

18DA0BFA 3 23 255 255 0 0 0 0

Идентификатор

Здесь:

18DA – Адресация к блоку управления

0B – Блок управления тормозной системы, к которому адресуется тестер FA – тестер (F1 –стандартный тестер, FA – программа SDP3, F9 – ECOM)

3 – (PCI) Количество байтов в сообщении

23_{10} (=17_{16}) – Команда на считывание кодов неисправности

255 255 – Старший и младший байты группы кодов неисправности (если требуется считать все коды = FF)

Обработка больших объемов (содержащих более 8 байт информации)

N_PDU type	29 bit CAN Identifier								CAN frame data field							
	28 26	25	24	23 16	15 8	7 0			1	2	3	4	5	6	7	8
FirstFrame	110_2	0	0	218_{10}	TA	SA			PCI				Data			
ConsecutiveFrame	110_2	0	0	218_{10}	TA	SA			PCI				Data			

Если передается большой объем информации, первое сообщение (First Frame) включает информацию (Protocol Control Information - PCI) о количестве байт в виде (0001 xxxx xxxx xxxx), где xxxx xxxx xxxx - количество байтов информации

Последующие сообщения (Consecutive Frame) содержат информацию о месте расположения сообщения (PCI) в виде 0010 xxxx, где xxxx = SN (Sequence number) пробегает значения от 0 до F_{16} Таким образом, первый байт принимает значения 32, 33, 34 ... 47, 32, 33 ...

First Frame - Первое сообщение

Если первый байт в сообщении 16, второй байт содержит информацию о полном количестве байт передаваемой информации

16	163	98	252	160	49	55	57	18DAF900
33	54	48	57	50	0	0	0	18DAF900
34	0	0	0	0	0	0	0	18DAF900
35	0	0	0	0	0	0	0	18DAF900
36	0	0	0	0	0	0	0	18DAF900
37	0	0	0	0	0	0	0	18DAF900
38	0	0	0	0	0	0	0	18DAF900
39	0	0	0	0	0	0	0	18DAF900
40	0	0	0	0	0	0	0	18DAF900
41	0	0	0	0	0	0	0	18DAF900
42	0	0	0	0	0	0	0	18DAF900
43	0	0	0	0	0	0	0	18DAF900
44	0	0	0	0	0	0	0	18DAF900
45	0	0	0	0	0	0	0	18DAF900
46	0	0	0	0	0	0	0	18DAF900
47	0	0	0	0	0	0	0	18DAF900
32	0	0	0	0	0	0	0	18DAF900
33	0	0	0	0	0	0	0	18DAF900

Обработка больших объемов (содержащих более 8 байт информации)

N_PDU type	29 bit CAN Identifier						CAN frame data field							
	28 26	25	24	23 16	15 8	7 0	1	2	3	4	5	6	7	8
FlowControl	110_2	0	0	218_{10}	TA	SA	PCI						N/A	

После передачи первого сообщения (First Frame), блок управления, принимающий поток информации, должен передать сообщение (Flow Control) – Управление потоком информации.

Первый байт PCI сообщения Flow Control может принимать 2 значения:
0011 0000 (48_{10}) - разрешение передавать информацию
0011 0001 (49_{10}) - WAIT

Второй байт PCI – параметр BS = Block Size – означает количество сообщений, которое может быть передано без повторной передачи сообщения Flow Control (параметр 0 означает, что сообщение FC более передаваться не будет)

Третий байт – параметр ST = Separation Time – означает минимальное время между передачей сообщений

Запрос от программы ECOM (F9) блоку управления двигателя (00) на считывание данных по общему идентификатору

3 34 252 160 3 255 255 251 7.217 18DA00F9

После получения в ответ первого сообщения, передается сообщение Flow Control

48 0 0 104 241 226 144 124 7.233 18DA00F9

Наиболее полезные команды

Перезапуск (Reset) блока управления

Команда – байт 1: **11_{16} = 22_{10}**

Команда – байт 2 **01_{16} (Power on – имитация полного выключения питания)**

03_{16} (Key on – имитация выключения зажигания)

Пример:

18DA0BFA	2	22	01	0	0	0	0	0

Ответ блока управления

Положительный: 51_{16} = 81_{10} Пример: **18DAFA0B 2 81 255 255 255 255 255 255**

Отрицательный: 10_{16} = 16_{10} Общий отказ

12_{16} = 18_{10} Подфункция не поддерживается – неверный формат

22_{16} = 34_{10} Неверные условия, ошибка последовательности запроса 33_{16} = 51_{10} Нет доступа по соображениям безопасности 78_{16} = 120_{10} Необходим правильный Rcvd-rsp 80_{16} = 128_{10} Сервис не поддерживается в активной сессии диагностики

Наиболее полезные команды

Считывание данных по локальному идентификатору

(При диагностике не рекомендуется адресовать сообщения для считывания данных по адресу элемента памяти – только по локальному идентификатору)

Команда – байт 1: **21_{16} = 33_{10}**

Команда – байт 2 **Локальный идентификатор**

Пример:

18DA00F9	2	33	240	53	3	255	255	251

(диагностика блока EMS программой ECOM)

Первый байт для ответа, состоящего из неск сообщений

Байтов в сообщении

Положительный ответ

Локальный идентификатор

Ответ блока управления

Положительный: 61_{16} = 97_{10} Пример: **18DAF900 16 14 97 240 8 240 0 0**

Отрицательный: 10_{16} = 16_{10} Общий отказ

12_{16} = 18_{10} Подфункция не поддерживается – неверный формат

22_{16} = 34_{10} Неверные условия, ошибка последовательности запроса 33_{16} = 51_{10} Нет доступа по соображениям безопасности 78_{16} = 120_{10} Необходим правильный Rcvd-rsp 80_{16} = 128_{10} Сервис не поддерживается в активной сессии диагностики

Наиболее полезные команды

Считывание данных по общему идентификатору

(Такой запрос позволяет получить ответ о схеме адресации, о возможных локальных идентификаторах или о серверах, подключенных к дополнительной шине CAN)

Команда – байт 1: **$22_{16} = 34_{10}$**

Команда – байт 2 **F0,**

Байт 3: 10 – в ответе сервера должна содержаться информация о схеме адресации, которая должна использоваться при взаимодействии

11 – В ответе должна содержаться информация о локальных идентификаторах, которые содержатся в сервере

12 – 14 – В ответе должна содержаться информация о локальном идентификаторе, по которому следует обращаться к данному серверу

15 – В ответе должна содержаться информация об идентификаторах серверов, подключенных к другой сети CAN данного мастер-сервера

16 – В ответе должен содержаться адрес, используемый для диагностики

Ответ блока управления

Положительный: $62_{16} = 98_{10,}$ Далее должны следовать повторение байтов 2 – 3 и адреса

Наиболее полезные команды

Считывание данных по адресу

(Такой запрос позволяет получить ответ о схеме адресации, о возможных локальных идентификаторах или о серверах, подключенных к дополнительной шине CAN)

Команда – байт 1: **23_{16} = 35_{10}**

Команда – байт 2 - 4 **Адрес (байт 2 = Старший байт адреса**
байт 3 = средний байт адреса **байт 4 = младший байт адреса)**

Команда – байт 5 **Объем памяти**

Ответ блока управления

Положительный: 63_{16} = 99_{10},

Наиболее полезные команды

Считывание всех кодов неисправности

(Позволяет считать все возможные коды неисправности, в том числе незаписанные в память)

Команда – **байт 1**: **18_{16} = 24_{10}**

Команда – **байт 2** статус запроса кода неисправности 02_{16} = 02_{10} Считывание записанных в память кодов $03_{16} = 03_{10}$ Считывание всех возможных кодов

Команда – **байт 3, 4** Адрес кода неисправности. В случае необходимости считать все коды, передается сообщение FF_{16} $FF_{16} = 255_{10}$ 255_{10}

Ответ блока управления

Байт 1: Положительный: 58_{16} = $88_{10,}$

Байт 2: Количество кодов в памяти

Байты 3, 4, 5: Информация по 1 коду – Байт 3 = старший байт кода
Байт 4 = младший байт кода
Байт 5 = статус кода

Байты 6, 7,8: Информация по 2 коду (и т.д.)

Статус кода: Бит 0 Неактивный код Бит 1 Код был активен за последний цикл Бит 2 Проводится тестирование Бит 3 Тест не проходит из-за других кодов Бит 4 Тест не завершен Бит 5 Код активен и занесен в память Бит 6 Код активен Бит 7 Код был активен за последний цикл

Наиболее полезные команды

Считывание занесенных в память кодов неисправности

Команда – **байт 1**: $17_{16} = 23_{10}$

Команда – **байт 2, 3** Адрес кода неисправности. В случае необходимости считать все коды, передается сообщение FF_{16} $FF_{16} = 255_{10}$ 255_{10}

Ответ блока управления

Байт 1: Положительный: $57_{16} = 87_{10,}$

Байт 2: Количество кодов в памяти

Байты 3 -12: Инфо по 1 коду – Байт 3 = старший байт кода Байт 4 = младший байт кода Байт 5 = статус кода (см описание на предыдущем сл) Байт 6 = Количество переходов в активное состояние Байты 7 – 12 Дата (в стандарте Секунды – минуты – часы – месяц – день – год

Наиболее полезные команды

Считывание Freeze Frames (информации, заносимой в память при возникновении кодов неисправности)

Команда – **байт 1**: **12_{16} = 18_{10}**

Команда – **байт 2** Адрес записи. В случае необходимости считать все записи, передается сообщение **FF_{16} = 255_{10}**

Команда – байт 3 может принимать следующие значения:

00 – запрашиваются все данные

04 – запрашиваются данные по соответствующему коду (определяемому байтами 4 – 5)

При этом байт 2 должен иметь значение **FF_{16} = 255_{10}** 83_{16} = 131_{10} – запрашивается таблица данных, в которой каждая серия Freeze Frames озаглавлена кодом неисправности, к которому она относится.

Команда – байты 4-5 (0001_{16} – $FFFF_{16}$) код неисправности (используется в случае, если должны быть считаны параметры только для одного кода неисправности)

Ответ блока управления

Описание содержится в отдельной презентации

Наиболее полезные команды

Стирание кодов неисправности

Команда – **байт 1**: $\mathbf{14_{16} = 20_{10}}$

Команда – **байт 2, 3** Адрес кода. В случае необходимости стереть все коды, передается сообщение $\mathbf{FFFF_{16} = 255\ 255_{10}}$

Примечание: Данная команда не стирает активные коды неисправности. Для стирания активного кода используйте рутину (см. Следующий слайд)

Наиболее полезные команды

Активация рутин

Команда – **байт 1**: **31_{16} = 49_{10}**

Команда – **байт 2** **Локальный идентификатор рутины**

Пример рутины для блока управления EBS

22_{16} = 34_{10} **Стирание всех (в том числе активных) кодов неисправности**

Наиболее полезные команды

Проверка/управление сигналами на входах по локальному идентификатору

Команда – **байт 1**: **$30_{16} = 48_{10}$**

Команда – **байт 2** **Локальный идентификатор**

Команда – байт 3 **Байт 3 может принимать следующие значения:** 00 – возврат к нормальному управлению 01 – выдача информации о входных/выходных сигналах, параметрах 03 – выдача информации о масштабировании сигналов 06 – режим управления выходными сигналами

Пример локальных идентификаторов для данного режима для системы EBS 5 см на следующем слайде

Локальные идентификаторы для режима «Проверка/управление сигналами на входах по локальному идентификатору» - пример для системы EBS 5

$11_{16} = 17_{10}$ Крутящий момент двигателя в режиме считывания – 01 – фактический момент в соответствии с сообщением EEC1 (Offset 125, диапазон +/- 125%) в режиме управления – 06 – требуемый момент – сообщение TSC1_AE

$12_{16} = 18_{10}$ Момент ретардера в режиме считывания – 01 – фактический момент в соответствии с сообщением ERC1_RD (Offset 125, диапазон +/- 125%) в режиме управления – 06 – требуемый момент – сообщение TSC1_ARD

$13_{16} = 19_{10}$ Момент моторного замедлителя в режиме считывания – 01 – фактический момент в соответствии с сообщением ERC1_REX (Offset 125, диапазон +/- 125%) в режиме управления – 06 – требуемый момент – сообщение TSC1_AREX

$28_{16} = 40_{10}$ Датчики ESP В режиме считывания – 01:

Байт 4 – 5 Угол поворота руля. Разрешение 1/512 рад, offset 16363 Байт 6 – 7 Показания датчика угла рыскания. Разрешение 1/2048 рад/с, offset 4095 Байт 8 – 9 Боковое ускорение. Разрешение 1/256 м/с^2, offset 4095 Байт 10 – 11 Поправка угла поворота руля. Разрешение 1/16384 рад, offset 25728 Байт 12 – 13 Поправка бокового ускорения. Разрешение 1/4096 м/с^2, offset 25152 Байт 14 – 15 Поправка датчика угла рыскания. Разрешение 1/65536 рад/с, offset 8590 Байт 16 – 17 Чувствительность датчика угла рыскания. Разрешение 1/8192 1/бит

Локальные идентификаторы для режима «Проверка/управление сигналами на входах по локальному идентификатору» - пример для системы EBS 5

$39_{16} = 57_{10}$ Цифровые входы/выходы в режиме считывания – 01 : **Байт 4**

– информация по клапанам ABS. (00 – выкл, 01 – вкл, 10 – ошибка, 11 – не опр)

Левый клапан: подача – биты 0, 1; вентиляция – биты 2, 3

Правый клапан: подача – биты 4, 5; вентиляция – биты 6, 7

$40_{16} = 64_{10}$ Требуемое замедление
в режиме считывания – 01
Байты 4 - 5 Разрешение 1/64 м/с^2 offset 32000

$43_{16} = 67_{10}$ Параметры ISC
в режиме считывания – 01 Байты 4 - 5 Параметр Каппа управления замедлением Разрешение 5 кг/м^2 /g = 0,0005 атм/g Байты 6 – 7 Параметр Phi_a (статич) = $P_{передн}/P_{задн}$ Разрешение 1%, диапазон 0 – 510% Байты 8 – 9 Параметр Phi_в (динамический) Байты 10 – 11 Параметр Phi_ist (для поддерживающей оси) Байт 12. Параметр Abbrband (согл торм усилий) Разрешение 1%, диапазон 0 – 100% Байт 13 Масса. Разрешение 0,4 тонны, Offset 0. Байт 14 Количество реальных торможений. Разрешение 1, максимум 255.

$45_{16} = 69_{10}$ Токи питания модуляторов. Разрешение 1/1024 А, диапазон 5 Ампер

Байт 4 – 5 Модулятор добавочный 1 Байт 6 – 7 Модулятор добавочный 2

Байт 8 – 9 Модулятор передний Байт 10 – 11 Модулятор задний

Байт 12 – 13 Датчики ESP Байт 14 – 15 ТСМ

Data Byte	Parameter Name	Cvt	$Value_{16}$
#1	**readDataByLocalIdentifier Request SId**	**M**	**21**
#2	recordLocalIdentifier	M	

Data Byte	Parameter Name	Cvt	$Value_{16}$
#1	**readDataByLocalIdentifier Positive Response Service Id**	**M**	**61**
#2	recordLocalIdentifer	M	
#3 : #n+2	recordValue#1 recordValue#n	M U	

$21_{16} = 33_{10}$ $61_{16} = 97_{10}$

Запрос тестера на считывание данных по локальному идентификатору $240_{10} = F0_{16}$

2 байта информации **Команда считать данные** **Считываемый идентификатор**

Разрешение продолжить передачу данных

2	33	240	53	3	255	255	251	24.470	18DA00F9
48	0	0	104	241	226	144	124	24.490	18DA00F9

Ответ блока управления двигателя

Информация – более 1 сообщения **Количество байтов** **Да, я отвечу**

16	14	97	240	8	0	0	0	25.839	18DAF900
33	8	59	0	0	8	240	0	25.859	18DAF900
34	0	0	0	0	0	0	0	25.869	18DAF900

Data Byte	Parameter Name	Cvt	$Value_{16}$
#1	**readDataByCommonIdentifer Request SId**	**M**	**22**
#2 - #3	recordCommonIdentifier	M	xxxx

Data Byte	Parameter Name	Cvt	$Value_{16}$
#1	**readDataByCommonIdentifer Positive Response Service Id**	**M**	**62**
#2 - #3	recordCommonIdentifier	M	xxxx
#3 : #n+3	recordValue#1 recordValue#n	M U	

Запрос тестера на считывание данных по общему идентификатору

3	**34**	**251**	**48**	**2**	**0**	**255**	**251**	**17.701**	**18DA00F9**

5	**98**	**251**	**48**	**0**	**231**	**0**	**0**	**17.708**	**18DAF900**

9.5.1 InputOutputControlByIdentifier (2F hex) service

In the case where the first byte of the controlOptionRecord is used as an InputOutputControlParameter, then Table 47 defines the data parameters applicable for the implementation of this service on CAN.

Table 47 — Data parameter definition — inputOutputControlParameter

Hex	Description	Cvt	Mnemonic
00	returnControlToECU	U	RCTECU
01	resetToDefault	U	RTD
02	freezeCurrentState	U	FCS
03	shortTermAdjustment	U	STA

4	**47**	**244**	**30**	**3**	**255**	**255**	**251**	**24.391**	**18DA00F9**
48	**0**	**0**	**104**	**241**	**226**	**144**	**124**	**24.410**	**18DA00F9**

16	**13**	**111**	**244**	**30**	**3**	**18**	**144**	**24.408**	**18DAF900**
33	**3**	**32**	**1**	**0**	**0**	**160**	**14**	**24.428**	**18DAF900**

ISO 15765-3 :2004(E)

Более общая информация

в ISO 14229-1

Diagnostic service name (ISO 14229-1)	Service Id value (hex)		Subfunction supported	suppressPosRspMsgIndicationBit = TRUE (1) (No response) supported [a]	Subclause
Diagnostic and Communication Management Functional Unit					
DiagnosticSessionControl	10	16	Yes	Yes	9.2.1
ECUReset	11	17	Yes	Yes	9.2.2
SecurityAccess	27	39	Yes	Yes	9.2.3
CommunicationControl	28	40	Yes	Yes	9.2.4
TesterPresent	3E	62	Yes	Yes	9.2.5
SecuredDataTransmission	84	132	—	N/A	9.2.6
ControlDTCSetting	85	133	Yes	Yes	9.2.7
ResponseOnEvent	86	134	Yes	Yes	9.2.8
LinkControl	87	135	Yes	Yes	9.2.9
Data Transmission Functional Unit					
ReadDataByIdentifier	22	34		N/A	9.3.1
ReadMemoryByAddress	23	35	—	N/A	9.3.2
ReadScalingDataByIdentifier	24	36	—	N/A	9.3.3
ReadDataByPeriodicIdentifier	2A	42	—	N/A	9.3.4
DynamicallyDefineDataIdentifier	2C	44	Yes	Yes	9.3.5
WriteDataByIdentifier	2E	46	—	N/A	9.3.6
WriteMemoryByAddress	3D	45	—	N/A	9.3.7
Stored Data Transmission Functional Unit					
ReadDTCInformation	19	25	Yes	Yes	9.4.1
ClearDiagnosticInformation	14	20	—	N/A	9.4.2
Input/Output Control Functional Unit					
InputOutputControlByIdentifier	2F	47	—	N/A	9.5.1
Remote Activation Of Routine Functional Unit					
RoutineControl	31	49	Yes	yes	9.6.1
Upload/Download Functional Unit					
RequestDownload	34	52	—	N/A	9.7.1
RequestUpload	35	53	—	N/A	9.7.2
TransferData	36	54	—	N/A	9.7.3
RequestTransferExit	37	55	—	N/A	9.7.4

Y 486 R10 136-3

Диагн адресация 3

Diagnostic Service Name	Section	SId	SS	AS	PS	DS	ES	SSS	note
Network Management									
disableNormalMsgTransmission	2.1	28	40						
enableNormalMsgTransmission	2.2	29	41						
controlDTCSetting	2.3	85	133						
Diagnostic Sessions									
testerPresent	3.1	3E	62	✓	✓	✓	✓	✓	
Diagnostic Management									
startDiagnosticSession	4.1	10	16	✓	✓	✓	(✓)	(✓)	
securityAccess	4.2	27	39		✓	✓	(✓)	(✓)	
ecuReset	4.3	11	17	✓	✓	✓	✓	✓	
readEcuIdentification	4.4	1A	26	✓	✓	✓	(✓)	(✓)	
Data Transmission									
readDataByLocalIdentifier	5.1	21	33	✓	✓	✓	(✓)	(✓)	
readDatabyCommonIdentifier	5.2	22	34	✓	✓	✓			
readMemoryByAddress	5.3	23	35			✓		✓	
writeDatabyLocalIdentifier	5.4	3B	59	✓	✓	✓	(✓)	(✓)	F
writeDataByCommonIdentifier	5.5	2E	46	✓	✓	✓			F
writeMemoryByAddress	5.6	3D	61			✓		✓	F
Stored Data Transmission									
readDTCByStatus	6.1	18	24	✓		✓			
readStatusOfDTC	6.2	17	23	✓		✓			
readFreezeFrameData	6.3	12	18	✓					
clearDiagnosticInformation	6.4	14	20	✓		✓			
Input Output Control									
inputOutputControlBy LocalIdentifer	7.1	30	48	✓		✓		(✓)	
Remote Activation of Routine									
startRoutineByLocalIdentifier	8.1	31	49	✓	✓	✓	(✓)	(✓)	F
stopRoutineByLocalIdentifier	8.2	32	50	✓	✓	✓	(✓)	(✓)	
requestRoutineResultsBy LocalIdentifer	8.3	33	51	✓	✓	✓	(✓)	(✓)	
startRoutineByAddress	8.4	38	56				✓	✓	
Upload / Download									
requestDownload	9.1	34	52				✓		F
requestUpload	9.2	35	53				✓		F
transferData	9.3	36	54				✓		
requestTransferExit	9.4	37	55				✓		

16	Diagnostic Session Control	4.1	9.2.1
17	ECU Reset	4.3	9.2.2
18	Read Freeze Frame Data	6.3	
20	Clear Diagnostic Information	6.4	9.4.2
23	Read Status of DTC	6.2	
24	Read DTC by Status	6.1	
25	Read DTC Information		9.4.1
26	Read ECU Identification	4.4	
33	Read Data by Local Identifier	5.1	
34	Read Data by Common Identifier	5.2	9.3.1
35	Read Memory by Address	5.3	9.3.2
36	Read Scaling Data by Identifier		9.3.3
39	Security Assess	4.2	
40	Disable Normal Messages Transmission	2.1	9.2.3
41	Enable Normal Messages Transmission	2.2	9.2.4
42	Read Data by Periodic Identifier		9.3.4
44	Dynamically Define Data Identifier		9.3.5

45	Write Memory By Address		9.3.7
46	Write Data by Common Identifier	5.5	9.3.6
47	Input/Output Control by Identifier		9.5.1
48	Input/Output Control by Local Identifier	7.1	
49	Start Routine by Local Identifier	8.1	9.6.1
50	Start Routine by (Common) Identifier	8.2	
51	Request Routine Result	8.3	
52	Request Download	9.1	9.7.1
53	Request Upload	9.2	9.7.2
54	Transfer Data	9.3	9.7.3
55	Request Transfer Exit	9.4	9.7.4
56	Start Routine by Address	8.4	
59	Write Data by Local Identifier	5.4	
61	Write Memory by Address	6.1	
62	Tester Present	3.1	9.2.5