



## CAN ECU Simulator

### GENERAL DESCRIPTION

The ECU SIMULATOR is a simple system used to provide the function of a vehicle ECUs external interface to a vehicle network or an off-board scan tool. The chip supports the standard interface types and requires minimal external support. To support Scan Tool Development, the ECU SIMULATOR provides a limited set of internal J1979 responses as well as the ability to interface to an external host to provide custom responses to any message type and format.

### APPLICATIONS

- § Scan Tool Development
- § Verification of Scan Tool Operation
- § Vehicle Node Point
- § Multi-Access Network

### USER CONFIGURABLE FEATURES

- § Protocol Selection
- § Tester Address / ISO Address
- § Response Origination
- § Internal Message Validation
- § Network Traffic Simulation
- § Debug Messaging

### SYSTEM COMMUNICATIONS

#### **Vehicle Communication Protocols**

- § CAN 11 Bit Header
  - 500K and 250K baud
  - Full line arbitration
  - Error detection
- § CAN 29 Bit Header
  - 500K and 250K baud
  - Full line arbitration
  - Error detection
- § J1850 PWM
  - 41.6K baud
  - Full line arbitration
  - Error detection
  - In Frame Response - Supported
- § J1850 VPW
  - 10.4K baud
  - Full line arbitration
  - Error detection
  - In Frame Response – Not Supported

- § ISO-9141
  - 10.4K baud
  - No line arbitration
  - Error detection
  - In Frame Response – Not Supported
- § ISO-14230 (KWP-2000)
  - 10.4K baud
  - No line arbitration
  - Error detection
  - In Frame Response – Not Supported

#### **Host Communication Protocol**

- § RS-232 Communication
  - 8 data bits
  - 1 stop bit
  - No parity
  - 57.6K baud
  - CTS – implemented

***Bowser Electronics***

**[www.BowserElectronics.com](http://www.BowserElectronics.com)**

**[Support@BowserElectronics.com](mailto:Support@BowserElectronics.com)**

## **SYSTEM OPERATION**

The ECU SIMULATOR functionality is broken into three parts: host communication, system commands, and request/response communication. All data communicated is in the binary, not ASCII, format (To send data, transmit Chr(&hFF), not “FF”). There is no use of the ASCII character set in this device. Throughout this section, all communication will be described in Hexadecimal values and use the notation 0xXX (i.e. 0x10 = Hex number 10, Decimal number 16).

## **HOST COMMUNICATION**

The ECU SIMULATOR uses an internal Universal Asynchronous Receiver/Transmitter (UART) to provide standard RS-232 communication between the chip and host. Communication parameters are fixed to enable the chip to communicate with the host faster than the vehicle is communicating with the chip. The CTS line is used to indicate when the chip is ready to accept commands from the host. This is an active low (0) signal. Once a complete command is received, the CTS line will go high (1) while the command is executed. During this time, the chip will communicate response messages. When all messages are complete, the CTS line will return to the low (0) state. Used in this manner, the CTS signal serves the dual purpose of indicating ‘system busy’ as well as ‘command cycle complete’. The host system should use the following serial port settings for communication with the ECU SIMULATOR.

Baud Rate	57600
Data Bits	8
Stop Bits	1
Parity	N
Buffer Size	2048
Handshaking	Hardware

## **SYSTEM COMMANDS**

The ECU SIMULATOR uses several system commands to customize the operation of the system. These commands may be issued anytime the CTS signal is low (0). Upon receiving the first byte of a command, the system will wait up to 13ms to receive the rest of a valid command message then set the CTS signal high (1). Invalid commands respond with a single byte 0xFF. The commands are outlined below:

**0x00 Firmware Version Number.** Two-byte response. First byte is the major version; second byte is the minor version.

**0x10 Set Active Protocol.** Single data byte as defined below. Default Value is PWM 0x01. If more than one bit is set, the LSB will prevail. To read current value, send only command 0x10 without databyte.

0x01 -	PWM
0x02 -	VPW
0x04 -	ISO-9141
0x08 -	ISO-14230
0x20 -	CAN 11bit, 500k baud
0x21 -	CAN 11bit, 250k baud
0x40 -	CAN 29bit, 500k baud
0x41 -	CAN 29bit, 250k baud

**0x11 Set ECU Address for non-CAN protocols.** 1 Value byte required. Default is J1979 Valid ECU address 0x10.

**0x12 Internal MIL Auto Set Time after clear for non-CAN protocols.** 1 Value byte required. Valid only when using internal responses. Number of 3.28 second intervals to wait to set MIL again after MODE \$04 Clear. Default value is 0x0A (~32.8seconds).

**0x13 Response Type for non-CAN protocols.** Defines how incoming messages are validated and where responses originate as follows: (note if bit 0 and bit 1 are both set, bit 0 will prevail. If both are clear, no action responses are given (the system will simply monitor the bus).)

**Bit 0 Set.** Responses generated internally

**Bit 1 Set.** Responses come from host

**Bit 4 Set.** Allow only Legislated diagnostics functional header / addresses

**Bit 6 Set.** Prescreen incoming messages for valid CRC/Checksum

**Bit 7 Set.** Prescreen incoming messages for valid 3 Byte Header

Enabling Bit 0 will automatically enable bits 4 and 7. Enabling bit 4 will automatically enable bit 7. Clearing bits 4 and 7 will allow all messages to pass. The following table lists the header byte the ECU SIMULATOR expects for each protocol / addressing mode combination. When prescreening is enabled, only that header will be allowed to pass. Screening will not stop vehicle messages from being sent to the RS-232 port. Screening can still be used when bit 1 is set. This will allow the system to return to listen for messages much quicker than passing all messages.

Header Byte Addressing Mode		
	Physical	Functional
PWM	64	61
VPW	6C	68
ISO-9141	6C	68
ISO-14230	8X	CX

**0x14 Option Flag.** Defines which options are enabled:

**Bit 0 Set.** Enable Debug messages to be transmitted to host. Transmits debug info during initialization to host. Messages will be transmitted from key points in the system to indicate among other things, the status of ISO initialization, Com errors, etc. The Error Code Table lists value and description of the codes that might be returned.

**Bit 1 Set.** Enable Network Traffic. For J1850 only. Transmits a message every 100ms. Use to test arbitration.

**0x15 Set ISO Target Service Address.** 1 Value byte required. Default is J1979 Service address 0x33.

**0x16 Set RS232 Delay.** 1 Value byte required. Sets minimum time to hold CTS Low.  $\text{Time}(\mu\text{S}) = \text{RS232Delay} * 409.6$  Default Value = 0x0A (4096 $\mu\text{S}$ ). Some older PCs may need to increase this time if the PC is missing the CTS transitions.

**0x17 Screening Progress for non-CAN protocols.** Read only command using Command 0x21. Used to indicate message progress through the screening subroutine as defined in command 0x13. This is for diagnostic purposes.

**Bit 0 Set.** Functional Addressing in Header Byte

**Bit 1 Set.** Physical Addressing in Header Byte

**Bit 2 Set.** Enter Target Address Screen

**Bit 3 Set.** Enter OBD Message Screen

**Bit 4 Set.** Enter CRC/Checksum Screen

**Bit 5 Set.** Screening Completed Successfully

**0x20 Reset Commands 0x10 – 0x1F to Default Values.** Single byte reply 0xF0.

**0x21 Read Command Value.** 1 Value byte required. Value is command number to return value from. Valid Range 0x10 – 0x1F. Returns 1 byte value indicating current command value.

## REQUEST/RESPONSE COMMUNICATION

Vehicle bus sourced messages will be retransmitted onto the RS-232 line byte-by-byte as they are received. Once all bytes are received, the message is screened for validity as defined in command 0x13 and processed for a response. If the system is set for external responses, the CTS line is toggled for the duration specified by command 0x16. The system will then wait for 100mS for a response from the host. The first byte returned by the host is the number of response messages to be sent. Next, each of the response messages is sent starting with a message length byte that doesn't include the length byte itself in the count. If multiple messages are to be sent, the ECU SIMULATOR will toggle the CTS line after each message while it is transmitted onto the vehicle bus. Once the CTS line returns, transmit the next message and so on until all messages have been relayed. If the system is set for internal responses, the request is screened through the pre-programmed responses and if a match is found, the response message is set. The response, if any, is then transmitted to the vehicle bus.

## INTERNAL RESPONSES –CAN PROTOCOLS

The responses from the CAN protocols are exactly as specified in J1979-2002 for modes 0x03 to 0x09. Modes 0x01 and 0x02 generate random data except as follows:

PID	RESPONSE
00	BE7FF810
01	8103AA88
12	04
13	03
14	??80
15	??80
1C	09

PIDs 0x14 and 0x15 send a switching signal in the first byte (??) that changes state at a rate of ~200msec.

While in CAN 11 Bit address mode, the ECU SIMULATOR will respond to the following received addresses, OBD-II address 0x7DF and Physical Powertrain Controller address 0x7E0. The ECU SIMULATOR will respond using address 0x7E8.

In CAN 29 Bit address mode, the ECU Simulator will respond to either the functional OBD-II address header 18DB33F1 or the physical Powertrain Controller address header, 18DA10F1. The ECU SIMULATOR will respond using address header 18DAF110.

More information about the CAN headers can be found in ISO 15765-4

## INTERNAL RESPONSES – NON-CAN PROTOCOLS

The following table lists the Modes and PID's the system will respond to when 'Internal Response' is selected using command 0x13. Mode \$01 Sensor data is automatically varied. Two byte responses vary the lower byte while single byte responses vary the lower nibble. Mode \$01, PID \$14 and \$15 generate a square wave with a 2 second period as well as the lower nibble variation. On power-up, the system will report several Mode \$03 and \$07 trouble codes. These can be cleared by issuing a Mode \$04 command. The trouble codes will reset for the time specified by command 0x12. Mode 9 is also supported as specified in J1979-2002

MODE	PID	RESPONSE
01	00	BE7FF811
01	01	8503AA88
01	03	0200
01	04	07
01	05	6F
01	06	76
01	07	85
01	0A	5F
01	0B	4F
01	0C	0C80
01	0D	00
01	0E	94
01	0F	16
01	10	034B
01	11	00
01	12	04
01	13	03
01	14	807A
01	15	80FF
01	1C	09

MODE	PID	RESPONSE
01	20	00180000
01	2C	00
01	2D	80
02	00	70188001
02	02	0301
02	03	0100
02	04	17
02	0C	32C8
02	0D	27
02	11	72
02	20	80180000
02	21	01F2
02	2C	22
02	2D	84
03		030002630473
03		138100000000
04		
05	00	C0400001
05	01	80
05	02	81

MODE	PID	RESPONSE
05	0A	0A051F
05	20	80000000
05	21	807090
06	00	FF0C000000
06	05	014240FFFF
06	05	024AFFFFFFF
06	05	03256FFFFFF
06	05	044ADEFFFF
06	05	0507C01B80
06	05	0604C017C0
06	05	87009A0037
06	05	88009B0028
06	05	0934007000
06	05	8934000A00
06	05	0A01B70352
06	06	41002A00BF
06	06	3500180033
07		155600000000
09		PER J1979

## ERROR CODES

Any time the ECU SIMULATOR is in 'Debug' mode, a response code stream will be sent to indicate what the system is doing and what the result was. Response codes for System Commands are indicated in the description for each command.

### ECU SIMULATOR Response Codes

Response	Description
0x00	No Errors
0x01	Start Receive Timeout
0x02	SOF High Max Time Exceeded
0x03	SOF High Min Time Not Exceeded
0x04	SOF Max Bit Time Exceeded
0x05	SOF Min Bit Time Not Exceeded
0x06	Data High Max Time Exceeded
0x07	Data High Min Time Not Exceeded
0x08	Data Bit Len Min Not Exceeded
0x09	Too many Bytes Received
0x0A	Last Byte Does Not Contain 8 Bits
0x0C	Min EOF Time not exceeded before SOF
0x0E	ISO NO STOP BIT AT END OF BYTE
0x13	ISO ADDRESS NOT VALID
0x14	ISO KEYWORD RECEIVE ERROR
0x16	ISO KEYWORD 2 TRANSMIT ERROR
0x17	ISO ADDRESS RECEIVE ERROR

Response	Description
0x18	ISO WRONG ADDRESS RECEIVED
0x20	RS-232 Byte Receive Timeout
0x21	Invalid Control Byte
0x22	Invalid Message Length
0x23	Transmit Arbitration Timeout
0x24	Line Stuck High
0x25	Line Stuck Low or no feedback
0x26	Expected IFR with no response
0x27	IFR Data High Max Time Exceeded
0x28	IFR Data High Min Time Not Exceeded
0x29	IFR Data Bit Len Min Not Exceeded
0x2A	IFR Too many Bytes Received
0x2C	IFR Started before EOD min
0x2D	ISO Not Initialized before Transmit Call
0xFE	Transmit Block (Next Error Code After Xmit)
0xFF	Receive Block (Next Error Code After Recv)

## APPLICATION EXAMPLES

Programming examples are available on the Bowser Electronics website.