Consult protocol, Nissan Technical egroup, Issue 7

1. Electrical and Signaling protocol
   1.1. Consult terminal or PC communications is via three wire bus. TX, RX and Async Clock.

   1.2. TX data to ECU level is 12V, idle low (see interface schematic). ECU output is open collector and can be sourced from the interface logic level (5 V). The ECU data output is idle high.
   Clock to the ECU seems fine with a 5v source from the interface.

   1.3. Communications baud rate is 9600 baud (default). Clock speed for this baud rate is 153.6 Khz, plus or minus 5 khz (as tested by PL& MS). Other baud rates and matched clocks may be possible as well. Clock GENERATED by the Host terminal (Consult or PC Interface). Synchronizing the clock to the data lines is NOT necessary. E.g. the clock free runs at the required speed. (16x the data rate).

   1.4. The Data Format is 1 start bit (always low), 8 data bits, no parity, 1 stop bit (always high).
   The data is actually processed by the ECU's micro LSB first. Each serial string is made up of blocks of : <Start bit (0)> <8 data bits LSB-MSB(xxxxxxxx)><Stop bit (1)> Data is sent as raw Hex,8N1, no spaces or carriage return characters etc are used.

   1.5. Initialization String. The ECU will not communicate until a valid hex initialization string message is detected.. (0xff)(0xff)(0xef) must be sent at the correct baud rate with the clock present on the ECU clock pin. Once Initialized the ECU responds with (0x10), it is then ready for commands. Idle timeout period seems to be longer than 30 minutes (as tested by PL & MS) or may only be required after an ECU is depowered again. In testing sometimes the Initialization string needed to be sent twice.

2. Command format and sequence (as tested with a laptop and terminal program in HEX mode).

   2.1. Consult terminal (or PC) sends a query command whose value determines function to be performed.

2.2. ECU responds with the same command byte but having inverted all bits as an error check. Data send after a command is echoed same as sent.

E.g. Send (0xd1)(hex), ECU responds with (0x2e). For valid commands only.

2.3. Consult terminal sends a go ahead/terminate message byte (0xf0)

2.4. ECU responds with result of query starting with a frame start leader character (0xff), then a byte it indicate the number of data bytes to follow, then the data. ECU WILL STREAM THE REQUESTED DATA until forced to stop.

2.5. Consult terminal then sends a stop byte, Command 0x30

e.g. Send (0xd1)(0xf0) , Ecu replies (0x2e)(0xff)(0x02)(0x55)(0x00). Where 2e is the inverted command, FF is the frame start byte, 02 is the number of data bytes to follow, 55 is the fault code (55 = no fault), 00 is fault code MSB.

3. Query/Reply commands list

3.1. Read DTC (fault codes) (0xd1)
terminate command with (0xf0) to start data stream, stop with (0x30). Multiple fault codes are returned in the same string, each code has two bytes. First byte is the code number, following byte is the number of ECU (engine starts) since the code last occurred. It clears automatically after 50 start cycles if not re-occurring. (0x2e)(0xff)(0x04)(0x13)(0x00)(0x8)(03) = two faults. 13 & 8, three starts since fault 8 was last recorded.

3.2. Clear DTC (fault codes) (0xc1) terminate command with (0xf0) to start data stream, stop with (0x30). This is the Normal way to Clear Fault Codes from Memory.

3.2a Clear DTC (Set#3) (0x51) terminate command with (0xf0) to start data stream, stop with (0x30).
0x51 reads and clears a set of DTC (Diagnostic Trouble Code) registers that are intended to register transient/difficult to diagnose problems.
When you give a **0x51** command, the ECU clears these registers. Then, whenever a problem pops up, the matching DTC is sent out to the host. You would possibly use it by, for example, sending **0x51** and then wriggling connectors until a DTC appears. These DTCs are set by slightly different places in the ECU code - for example, most sensors are checked in a separate subroutine. The MAF sensor DTC is checked there. But the voltage is also checked in the MAF reading routine itself. If it fails, the special DTC is set, but not the regular one.

There are at least three sets of DTC registers
- a set that are used to check for repeat conditions (set every time it occurs)
- a set used by the CEL OBD (for ECUs with the switch on the back) and consult (set after a predetermined number of occurences of the first set)
- a set used only by Consult for transient conditions (the set cleared and read by **0x51**)
- A subset used to trigger the CEL (Check Engine Light)

So the DTC (diagnostic trouble code) commands:

0xC1 clears the second set of DTC registers, but not the first or third, and then reads them back.

0xD1 gets the second set of DTC registers, but not the first or third, and then reads them back without changing them.

0x51 clears the third set of registers and then returns whatever DTC re-appears.

3.3. Read ECU Part Number (0xd0)
terminate command with (0xf0) to start data stream, stop with (0x30). Part Number format is XXXX 23710-YYYYYY. ECU sends frame with 22 data bytes (0x16 bytes). Bytes 2 & 3 = XXXX. Bytes 19 to 22 = YYYYYY. 23710 is a constant not included in the data. Eg. 0488 23710-50F00 where 50F00 is the ECU SW version.

3.4. Stop Acknowledgment response (0xCF, from ECU). Sending byte (0x30) is used to stop a current query reply, but ECU will continue to send the data till the end of a complete frame, where it then sends a Stop Acknowledgment byte (0xCF). Data should be read until this byte is received to keep data integrity or the last frame dumped.

3.5. Read any Register Parameter (Live sensor data stream)(0x5a)<parameter code to read>(0xf0) for single byte stream
(0x5a)<parameter>(0x5a)<next parameter>(0x5a)<next parameter>(0xf0) for a multi-byte response stream. Maximum of 20 bytes. Terminate command with (0xf0) to start data stream, stop with (0x30).

Be careful to ensure the byte you are attempting to read is valid for the ECU type. e.g. sending command 5A 00 (tach), ecu responds with A5 00 (valid command). But sending command 5A 06 (Maf #2), ecu responds with A5 FE (invalid command for a single MAF (non GTR) ecu).

Sample Sensor Data Register Table

(See Generic register document for a complete listing)

Data Name	Byte	Register No	Scaling
Tachometer	MSB	0x00	See LSB
Tachometer	LSB	0x01	Value * 12.5 (RPM)
MAF voltage	MSB	0x04	Value * 5 (mV)
MAF voltage	LSB	0x05	Value * 5 (mV)
MAF #2 voltage	MSB	0x06	Value * 5 (mV)
MAF #2 voltage	LSB	0x07	Value * 5 (mV)
Coolant temp	Single byte	0x08	Value – 50 (degC)
Vehicle speed	Single byte	0x0b	Value * 2 (kph)
Injector pulse time	MSB	0x14	See LSB
Injector pulse time	LSB	0x15	Value / 100 (mS)

3.6. Read any ROM BYTE (Internal ROM, program code) (0xc9) (0xc9)<High byte><low byte>(0xf0) for single byte (0xc9)<High byte><low byte>(0xc9)<high byte><low byte>(0xf0) for a multi-byte response. Maximum of 8 bytes.
terminate command with (0xf0) to start data stream, stop with (0x30).ROM space starts at 8000h and ends at ffffh. RAM addresses can be read by thismethod also.

For example, to read 3 bytes at addresses 0x8000...0x8002, send:C9 80 00 C9 80 01 C9 80 02 C0   ECU response:36 80 00 36 80 01 36 80 02 FF 03 05 03 0836 is inverted C9 etc, FF 03 is a header and the record's length, then the threedata bytes.
All the ECU eprom data can be read out by reading 8 bytes at at time and appending to a file. This is used to create ROM binaries for use in expansion boards.

4. Activation Commands – To alter ECU output

4.1.1. Commands to temporarily alter ECU outputs for Fuel, Ignition timing and relays etc can be done by sending a Activation byte (0x0A) then a activation type register to set followed by the new register data. Registers and results are in the table below. Eg 0x0A XX YY where XX is the test type, YY is the data. Terminate the command with F0.

Multiple commands are also allowed. E.g. 0A 81 67 0A 82 02 FE sets fuel injection +3% and Timing +2 deg.

Note that the stop command (0x30) not only stops the ECU data stream but also cancels all Active Commands.

4.1.2. Set ECU to use different temperature value (shift temp correction point, start/stop fan).

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x80	XX+50	Set Coolant temp. (Temp+50) in hex

4.1.3.
Adjust Fuel Injection Time- Rich or Lean the mixture

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x81	0x64	Fuel injection normal (100%)
0x0A	0x81	0x67	Fuel injection +3% (103% of normal)
0x0A	0x81	0x6A	Fuel injection +6% (106% of normal)

Normal (100%) = 0x64. Add or subtract from this value to alter up or down.

4.1.4. Adjust Static Ignition timing - Advance or Retard by 1 degree steps.

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x82	0x00	Timing normal.
0x0A	0x82	0x02	Timing +2 degree
0x0A	0x82	0xFE	Timing -2 degree

Normal = 0x00. Add 1 per degree increase. Subtract 1 (roll over to 0xFF, 0xFE etc) to decrease timing.

4.1.5. Adjust IAAC Valve Opening %- Open or Close by 0.5 % per step. (Duty Cycle)

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x84	0x00	Opening DC normal.
0x0A	0x84	0x02	Opening +1 %
0x0A	0x84	0xFC	Opening -2 %

Normal = 0x00. Add 1 per 0.5% increase. Subtract 1 (roll over to 0xFF, 0xFE etc) to decrease by 0.5%.

4.1.6. Perform Power Balance Test – Stop individual cylinder firing (uses bit logic).

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x88	0x00	Firing (Fuel & Ignition) Normal
0x0A	0x88	0x01	1st Cylinder OFF
0x0A	0x88	0x02	2nd Cylinder OFF
0x0A	0x88	0x04	3rd Cylinder OFF
0x0A	0x88	0x08	4th Cylinder OFF
0x0A	0x88	0x10	5th Cylinder OFF
0x0A	0x88	0x20	6th Cylinder OFF
0x0A	0x88	0x40	7th Cylinder OFF
0x0A	0x88	0x80	8th Cylinder OFF

4.1.7.
Fuel Pump Relay Control test – Stop fuel pump.

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x89	0x00	Pump Relay ON
0x0A	0x89	0x01	Pump Relay OFF

4.1.8.
P/Reg Control Solenoid

Activate	Type	Data	Meaning

Command	Byte	Byte	
0x0A	0x8A	0x00	Unknown

4.1.9.
Clear Self-Learning Value – Reset learned trim values (closed loop).

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x8B	0x00	Clear Learned data

4.1.A.
IAAC Control

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x8C	0x00	Unknown

4.1.B.

Activate Command	Type Byte	Data Byte	Meaning
0x0A	0x8D	0x00	Unknown

Value in Register 1C reset.

There is now separate documents listed for the known register output for SR20 ECU type, 300ZX type and KA24E type, also a Generic document that lists all known registers and scaling plus any comments about possible differences between ECU’s. Some registers above may not be correct and intended as samples only.

4.1.9. Nissan Technical group members please forward new registers or commands to the group and I will add them Or alternately add them here yourself but move the original file to Old Documents. Also report errors in this document to Martin.

24-May-2006