The OpenOtto Project: Implementation Notes

$Date: 2004/07/08 02:34:14 $

Abstract

Implementation notes for the Otto suite of software.

Copyright

`$Id: implementation.lyx,v 1.7 2004/07/08 02:34:14 alpha Exp $`

1 Foreward

match sections with arch spec

finish API definitions, data types, milestones

2 Hardware

2.1 Software Data Link Devices

2.1.1 Serial

todo: interface circuit

one per interface, VPW, PWM, ISO (K&L), CAN

- interface similarities
  - VPW half duplex EIA-232
  - PWM half duplex RS-485
  - ISO half duplex EIA-232, 2 lines (MAX3323E)

  - CAN (MAX3050?)

- circuits

  - serial to logic - MAX3235E 20/PDIP.300
    - signal protection - MAX367 18/PDIP.300
  - logic to VPW/ISO (half duplex RS-232) - MAX3323E 16/PDIP.300
  - logic to PWM (half duplex RS-485) - MAX3441E 8/PDIP.300
  - logic to CAN - MAX3050 8/SO.150
  - regulators
    - 5V - MAX883 8/PDIP.300
    - 3.3V - MAX882 8/PDIP.300

2.1.2 Parallel

todo: interface circuit

2.2 Hardware Data Link Devices

2.2.1 Microcontroller

2.2.2 Programmable Hardware

2.2.3 Development Board

todo: dos based firmware

3 Software

3.1 Drivers

todo: generic interface for sw data link, common set of methods (irq, read bit, write bit, others?)

share code, where possible

- per interface type: irq, read bit, write bit
- per architecture: get time, set timer callback or wait/sleep
- per bus type: encode/decode routines

3.1.1 kernel drivers

based on lirc_serial (possibly lirc_parallel as well)

see obd_serial from old obd in archive for starting point

3.1.2 ottod

written in C

3.2 Libraries

todo: bindings for ?

3.2.1 libvin

- 17 characters, of [A-HJ-NPR-Z0-9]
- section one, 3 chars
  - manufacturer, make, type
- section two, 5 chars
  - line, series, body type, engine type, restraint system (passenger car), GVW (multipurpose passenger vehicle)
- section three, 1 char
  - check digit, as spec in VIN
- section four, 8 chars
  - first char is year, as spec in VIN
  - second char is plant of manufacturer
  - char 3-8 is sequential manufacture number

3.2.2 libobd2

- Request current powertrain diagnostic data (mode 0x01)

```
struct {
      char mode = 0x01;
      char pid;
    } request;
    struct {
      char mode = 0x41;
      char pid;
      int len; /* 1-4, number of bytes in data */
      char data[4];
    } response;
  • todo:  special method for determining support for
    other methods (pid 0x00)?
```

    - `tables`
        - `pids (same as mode 0x02)`
        - `bitmapped data for pid 0x01, 0x03, 0x12, 0x13, 0x1e`
        - `constants for pid 0x1C`

- request powertrain freeze frame data (mode 0x02)

  ```
  struct {
    char mode = 0x02;
    char pid;
    char frame;
  } request;
  struct {
    char mode = 0x41;
    char pid;
    char frame;
    int len;  /* 1-4, number of bytes in data */
    char data[4];
  } response;
  ```

  - tables shared with mode 0x01

- request emission-related powertrain diagnostic trouble codes (mode 0x03)

  ```
  struct {
    char mode = 0x03;
  } request;
  struct {
    char mode = 0x43;
    char data[6];
  } response;
  ```

  - message is fixed length, dtcs are in data two bytes each
  - multiple response may be received
  - otto_getdtc(int *num, int *dtc) returns 1-3 DTCs
    - this method calls mode 0x01, pid 0x01 first to determine total DTCs to expect
    - dtc is array of length num of all DTCs
  - tables
    - in dtcdb

- clear/reset emission-related diagnostic information (mode 0x04)

```
struct {
  char mode = 0x04;
} request;
struct {
  char mode = 0x44;
} response;
```

  - otto_cleardtc() returns ok/fail

- request oxygen sensor monitoring test results (mode 0x05)

```
struct {
  char mode = 0x05;
  char testid;
  char sensornum;
} request;
struct {
  char mode = 0x45;
  char testid;
  char sensornum;
  char data[4];
} response;
```

  - test id 0x00, 0x20, 0x40, 0x60, 0x80, 0xA0, 0xC0, 0xE0
  - data is bitmapped support for next 20 testids

```
struct {
  char mode = 0x45;
  char testid;
  char sensornum;
  char value;
  char min;  /* optional */
  char max;  /* optional */
} response;
```

  - otto_getsensortestresult(int test, int sensornum)
  - todo: get support function similar to mode 0x01?
  - table
    - testids
    - min/max/scaling for tests (in unified SLOT definitions?)

- request on-board monitoring test results for non-continuously monitored systems (mode 0x06)

```
struct {
  char mode = 0x06;
  char testid;
} request;
struct {
  char mode = 0x46;
  char testid;
  char _x = 0xff;
  char data[4];
} response;
```

  - testids multiple of 0x20, bitmapped support for next 0x20 testids

```
struct {
  char mode = 0x46;
  char testid;
  char type;  /* test limit type and component ID */
  char value[2];
  char limit[2];
} response;
```

  - only valid if mode 0x01 pid 0x01 indicates test is complete
  - otto_gettestreult(int test)
  - tables
    * testids

- request on-board monitoring test results for continuously monitored systems (mode 0x07)

```
struct {
  char mode = 0x07;
} request;
struct {
  char mode = 0x47;
  char data[6];
} response;
```

  - table: in dtc db

- request control of on-board system, test, or component (mode 0x08)

```
struct {
  char mode = 0x08;
  char testid;
  char data[5];
} request;
```

```
struct {
  char mode = 0x48;
  char testid;
  char data[5];
} response;
```

  - tables
    - testids

- request vehicle information (mode 0x09)

```
struct {
  char mode = 0x09;
  char info;
} request;
struct {
  char mode = 0x49;
  char info;
  char count;
  char data[4];
} response;
```

  - tables
    - info type

3.3 Applications

3.3.1 ottoconfig

perl? C?

3.3.2 ottodump

written in perl

3.3.3 ottocat

written in perl

3.3.4 scantool/xscantool

possibly written in perl/gtk; sections optimized in C

3.3.5 ottomann

possibly written in perl/gtk; sections optimized in C

4 OSI Model

- data link devices (send/recv raw packet) (better called LLC devices?)
  - iso uart (iso9141 interface)
  - sw bit banging devices (serial) (saej1850 interface)
  - (IP encapsulation)
  - (proprietary stuff)
- network driver (addressing) (data link MAC?)
  - obd2 over data link (j1850, j2178)
  - iso9141
  - iso14230
  - obd2 over iso9141 (iso9141, j1850)
  - obd2 over iso14230
- transport (or network, if above is data link MAC?)
  - diag modes
  - enhanced diag modes
  - message formats
- presentation
  - security
  - prn/slot
- application
  - scan tool

A Schedule

- "implementation plan"
  - make skeletal files from arch spec
  - move notes from implementation file to skeletal files
- serial port interface hardware
- bring up devboard
- libobd2-link

  - header
  - encode/decode for use with devboard

- devboard firmware (dos)
- ottod
- ottoconfig
- ottodump
- ottocat
- libobd2
  - header
- scantool
- xscantool
- libotto
- ottomann
- libvin
- otto_serial kernel driver