CAN Bus logger with SD-card

0. What is it?

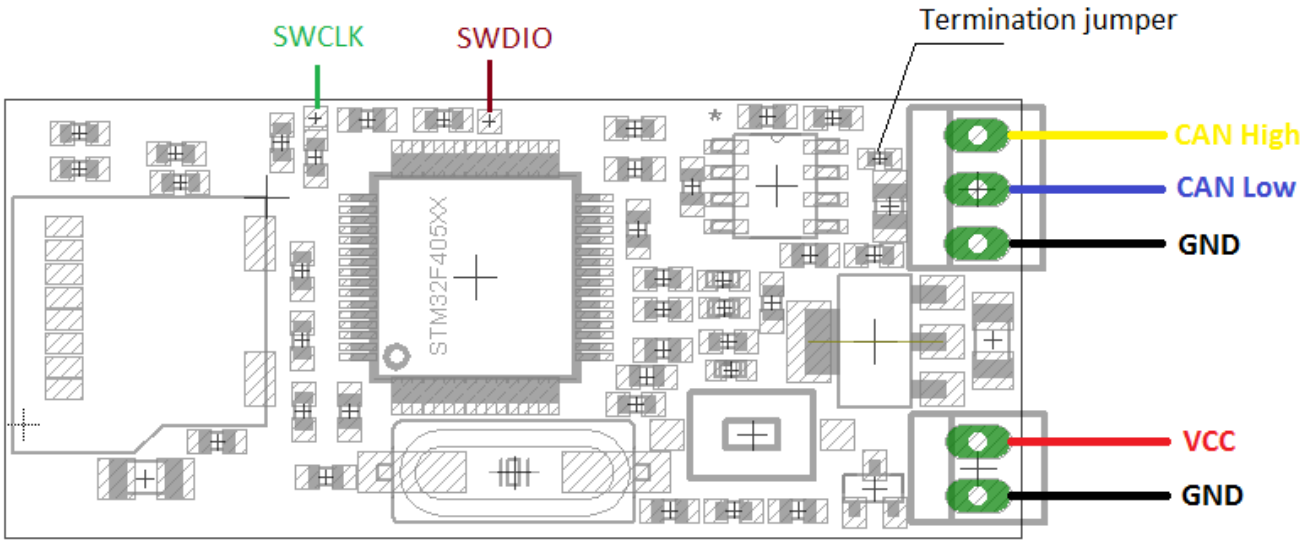
This is just simple logger which writes everything from CAN bus to text file on a micro SD card. It has following features:

- Easy to use: Only one start/stop button and all the settings are stored in configuration text file on the SD card.
- Optional message filtering based on ID mask matching.
- Selectable listen-only mode (without CAN bus acknowledge).
- Three LEDs for indication of logger.

Device specification:

Parameter	Value
Power supply voltage	5.5V-20V
Current consumption	60mA at 5V input
CAN baudrate	up to 1Mbps (any non-standard baudrate supported)
CAN ID mask filters	1
Microcontroller	STM32F405RGT6
CAN transceiver	SN65HVD232DR
PCB size	48.26 mm x 20.85 mm (1.9 in x 0.82 in)

Board connections:



LEDs:

- Green: power on, also toggling every time when the CAN receives and accepts messages.
- Blue: blinks each time when block of data has been written to the SD card.
- Red: fault indication (see below).

1. How to start

- Connect CAN bus.
- Connect power supply (make sure the correct voltage range).
- Place **Config.txt** file to the root folder of SD card (here is example of the file for ODB). It is recommended to choose fastest SD card (UHS Speed Class 1 / U1) and format SD card before use in logger.
- Insert SD card.
- Press "START" button to start log
- The blue LED should blinkperiodically (with speed dependant from writing rate).
- Press "START" button again to stop the log.
- The log file placed to root folder and has name in format: **N.csv**, where N is number between 0 and 99999999. The next time logger will increment a file name number and write to **N+1.csv**, for example if there are files 0.csv, 1.csv and 2.csv, then the next written file will be 3.csv.

2. Configuration file format

Parameter	Parameter format	Meaning
`baud`	Decimal	CAN bus baudrate in kbps
`ack_en`	0 or 1	If 1 then CAN logger is responding with ACK slot on reception of a valid CAN frame. Set to 0 for silent (listen-only) mode.
`id_filter_mask`	Hex	Bit mask for ID filtering
`id_filter_value`	Hex	Expected value for ID
`log_std`	0 or 1	Messages with standard (11 bit) ID are accepted if set to 1
`log_ext`	0 or 1	Messages with extended (24 bit) ID are accepted if set to 1
`timestamp`	0 or 1	If 1 then every record in log file has a timestamp (in milliseconds)
`circular_write`	0 or 1	This will delete oldest file if there is less than 10% of free space on disk (please be aware of possible logging gap up about to 1s)
`start_on_power`	0 or 1	Recording will start immediately on power on (no push button needed)
`start_on_CAN`	0 or 1	This will start logging when a CAN message with specific ID has been received (please give about 50-100ms to prepare file on disk for record)
`stop_on_CAN`	0 or 1	This will stop logging when a CAN message with specific ID has been received
`start_frame_to_name`	0 or 1	The data of the CAN message will be used for file name
`start_id_value`	0 or 1	Bit mask of ID filtering for CAN frame which starts logging
`start_id_mask`	0 or 1	Expected value of ID for CAN frame which starts logging
`stop_id_value`	0 or 1	Bit mask of ID filtering for CAN frame which stops logging
`stop_id_mask`	0 or 1	Expected value of ID for CAN frame which stops logging

Only `baud` is required, all other configuration parameters are optional (and equal to 0 by default).

The ID filter acceptance criterion is:

[CAN message ID] *&* `id_filter_mask` = `id_filter_value` *&* `id_filter_mask`

where & is bitwise logical and operator.

For example if `id_filter_mask` = A = 1010 binary and the `id_filter_value` = 2 = 0010 binary. It means that the bit #1 and bit #3 of CAN identifier will be checked, and the bit #1 is expected to be 1 and bit #3 to be 0. Thus only identifiers with a binary ending of …0X1X will be accepted by CAN1, i.e. in hex 0x?2, 0x?3, 0x?6, 0x?7.
Set `id_filter_mask=` 0 to disable the ID filter.

A message with specific ID will initiate start of logging when `start_on_CAN` defined to 1 and following is true:

[CAN message ID] & `start_id_mask` = `start_id_value` & `start_id_mask`

where & is bitwise logical and operator.

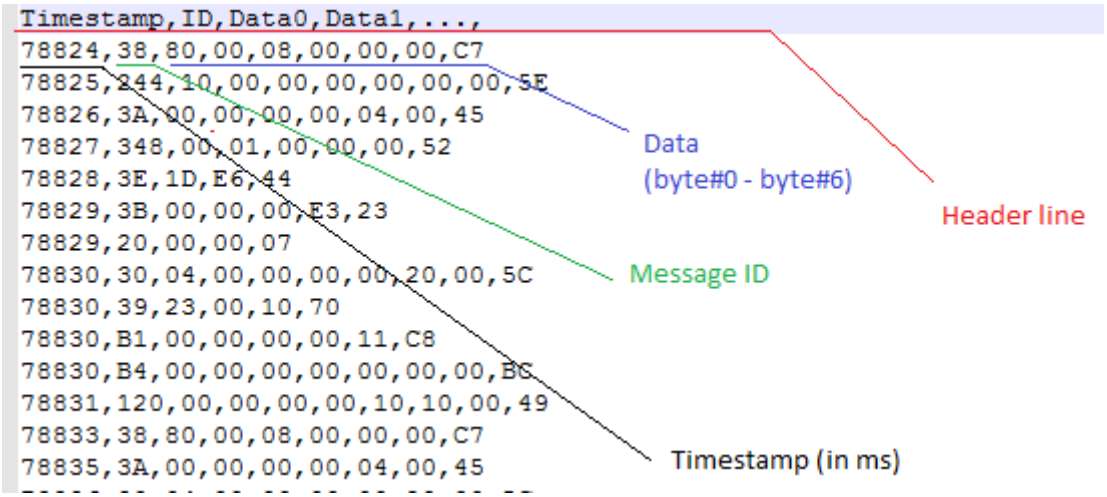
A message with specific ID will stop logging when `stop_on_CAN` defined to 1 and following is true:

[CAN message ID] & `stop_id_mask` = `stop_id_value` & `stop_id_mask`

where & is bitwise logical and operator.

The option `start_frame_to_name` can be used in conjunction with `start_on_CAN`. The lower 4 bits of all data bytes from the starting CAN message will form a new file name of log file. For example if the data for start CAN message defined by ID with `start_id_value/start_id_mask` is = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08}, then the new file name will be 12345678.csv;

3. Log file format



4. Faults and indication

- If red LED and blue LED are both on just after pressing "START" button, then this is a configuration text file problem; check configuration file.
- If red LED is on during logging, then this is either data buffer overflow or SD card problem. Check if there is enough free space on SD card or SD card has acceptable writing speed rate.

5. CAN bus termination

Please note that the 120 ohm termination resistor is permanently placed on board. There is no configurable option for it. However, there is a jumper as 0402 resistor which is possible to unmount for termination disconnection.
If the bus already has termination at the both ends, then you may consider removing on-board termination. However, for most application it's ok to leave it (for example when connecting to vehicle OBD-II diagnostic connector).

6. CAN bus simulator (playback from file)

There is an experimental function implemented to play back recorded data onto a CAN bus, turning device into a CAN bus simulator.

Place recorded log to SD card root folder, rename it to **Play.csv**. Press "START" button, the device will read Play.csv file line by line and transmit all recorded CAN messages back to the bus. After completion of transmission the device will be switched to standard logger mode.

- Playback file will be transmitted with the baudrate from Config.txt file.
- Playback file should be time-stamped (i.e. recorded with timestamp=1 option).
- Playback file should contain a header, i.e. first line of the file will be skipped.
- By default, all messages will be send out using standard (11-bit) identifier, if the identifier value from file fits into 11 bit, otherwise it will be send as 24 bit (if it does require more than 11 bits)
- You can enforce the usage of extended identifiers only by setting log_std=0 in Config.txt
- The green LED will be off during playback, the blue LED will toggle every time when the message has been sent, red LED will be on if any faults occurred.

Limitations:
- All messages should be acknowledged from the CAN bus side.
- It is not exactly time accurate because timestamp has 1 ms granularity.
- The only way to exit playback mode in case if it's stuck (for example due to bus fault), is to power cycle the device.