SCAB: serial to CAN bridge

IGREBOT team

Contents

Abstract

This document describes the SCAB api library and the DSPIC33F firmware.

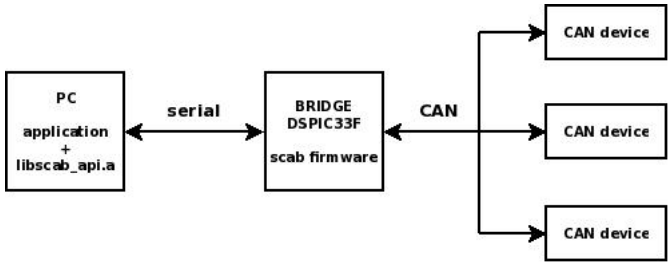

Figure 1: serial to CAN bridging

1 Introduction

1.1 Overview

Usually, PCs do not have the hardware interfaces required to communicate on a CAN network. SCAB is an opensource project to add CAN connectivity to PCs by bridging a serial port. To do so, SCAB provides the following:

- a firmware to be flashed on a DSPIC33F board,
- a programming interface implemented in a library used by host applications.

1.2 Availability

The project is maintained in a GIT repository:

`https://github.com/texane/scab`

In the remaining of this document, the expression:

$SCAB_REPO_DIR

denotes the directory where the repository was cloned.

1.3 Dependencies

The project depends on the following softwares:

- a working LINUX system with standard GNU tools,
- MPLABX version 1.0 .

Note that WINDOWS and MACOSX are not yet supported.

2 Host application programming interface

2.1 Overview

The source code is located in:

$SCAB_REPO_DIR/src/api

The API is shipped as a standalone static library and can be built using:

```
cd $SCAB_REPO_DIR/build/api ;
make ;
```

It produces the file:

$SCAB_REPO_DIR/build/api/libscab_api.a

If a GNU toolchain is used, the library is linked in a client application by adding the following flags to command line:

```
-L$SCAB_REPO_DIR/build/api -lscab_api
```

2.2 Interface documentation

```
int scab_open(scab_handle_t**, const char*);
int scab_close(scab_handle_t*);
int scab_sync_serial(scab_handle_t*);
int scab_read_frame(scab_handle_t*, uint16_t*, uint8_t*);
int scab_write_frame(scab_handle_t*, uint16_t, const uint8_t*);
int scab_enable_bridge(scab_handle_t*);
int scab_disable_bridge(scab_handle_t*);
int scab_set_can_filter(scab_handle_t*, uint16_t, uint16_t);
int scab_clear_can_filter(scab_handle_t*);
int scab_get_handle_fd(scab_handle_t*);
```

2.3 Example program

TODO

2.4 Limitations

TODO

3 Device firmware

3.1 Overview

The firmware is a piece of software put on a DSPIC33F board to perform the forwarding of frame to and from the host PC an the CAN network.

The source code is located in:

$SCAB_REPO_DIR/src/device

Assuming that MPLABX version 1.0 has been installed with the default paths, the firmware can be compiled using:

```
cd \$SCAB_REPO_DIR/build/device.X ;
make ;
```

It produces the file:

$SCAB_REPO_DIR/build/device.X/dist/default/production/device.X.production.hex

which is used to program the DSPIC33F flash.

3.2 Limitations

TODO

4 Protocol between host and device

4.1 Overview

All the symbolic constants used in this section can be found in the file:

$SCAB_REPO_DIR/src/common/scab_common.h

All the packets share the same basic format and are of the same fixed length SCAB_CMD_SIZE (currently 11 bytes). Thus, even for packets with smaller payloads, SCAB_CMD_SIZE must be sent over the serial link.

The packets can be sorted in 2 groups:

- bridge management commands: set parameters such as link speeds, CAN filters ...
- frame forwarding: actual data frame forwarding.

A management command is always sent by the host to the bridge device. It always requires a SCAB_CMD_STATUS to be sent, containing the completion status of the command.

SCAB_CMD_FRAME is the only non management packet. It can be sent either by the host to send a frame on the CAN network, or by the device to forward a frame from the CAN network to the host. There is no SCAB_CMD_STATUS packet involved in those transfers.

Note that the little endian byte ordering is used between the host and the bridge.

4.2 Packet formats

4.2.1 Basic packet format

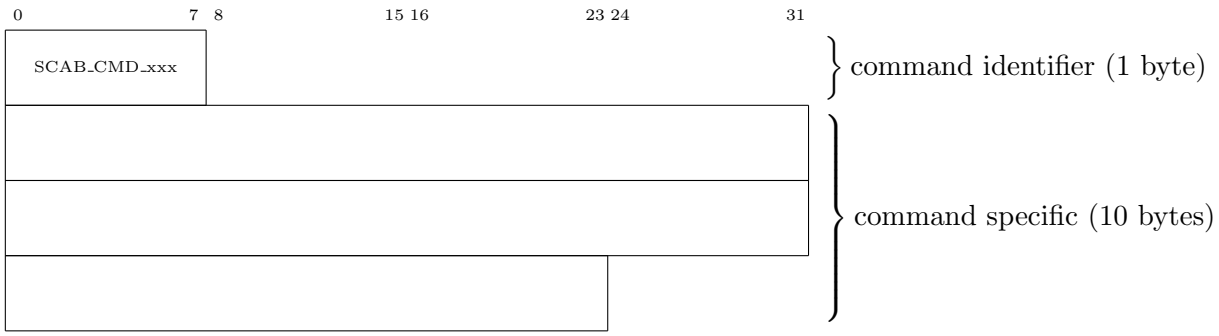

Figure 2: Basic packet format

All the packets follow the same basic format. This format is then specialized using the command identifier. The data field is then used according to the command as described in the next sections.

4.2.2 SCAB_CMD_FRAME format

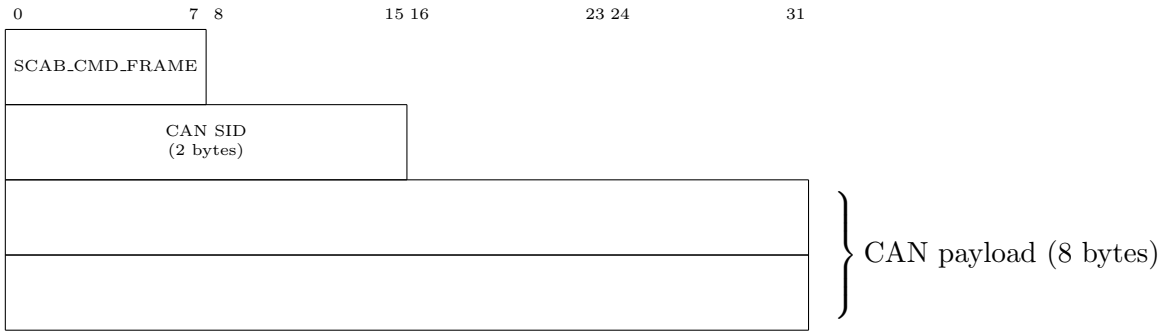

Figure 3: SCAB_CMD_FRAME format

4.2.3 SCAB_CMD_SYNC format

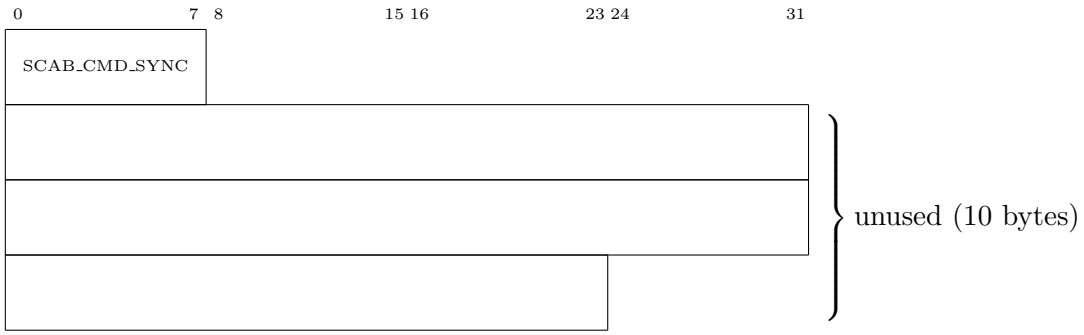

Figure 4: SCAB_CMD_SYNC format

4.2.4 SCAB_CMD_ENABLE format

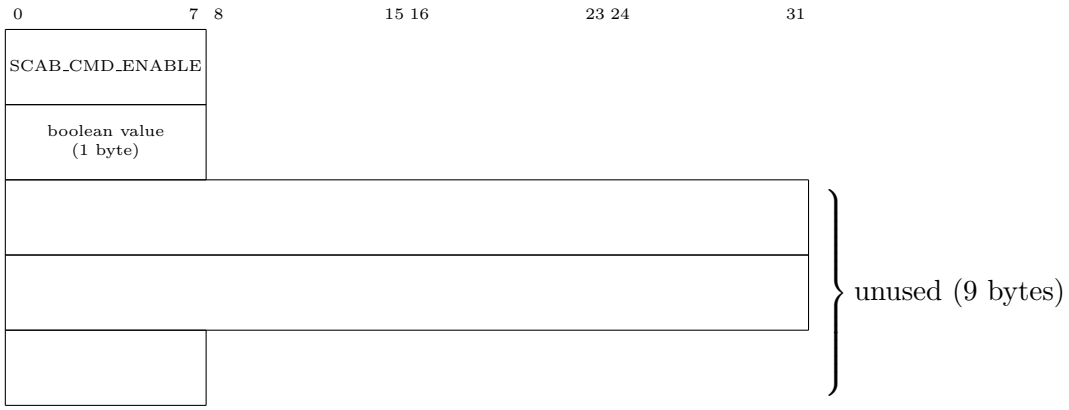

Figure 5: SCAB_CMD_ENABLE format

4.2.5 SCAB_CMD_SET_CAN_TIMES format

TODO, not yet implemented

4.2.6 SCAB_CMD_SET_CAN_FILTER format

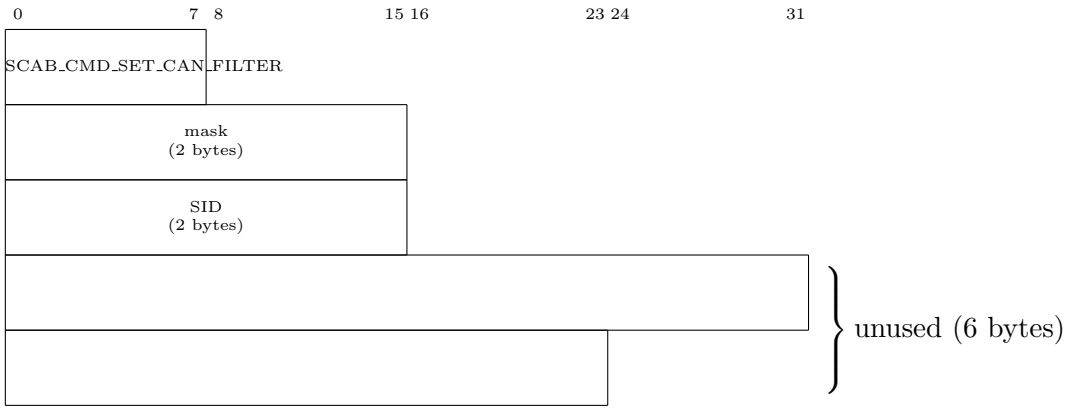

Figure 6: SCAB_CMD_SET_CAN_FILTER format

4.2.7 SCAB_CMD_CLEAR_CAN_FILTER format

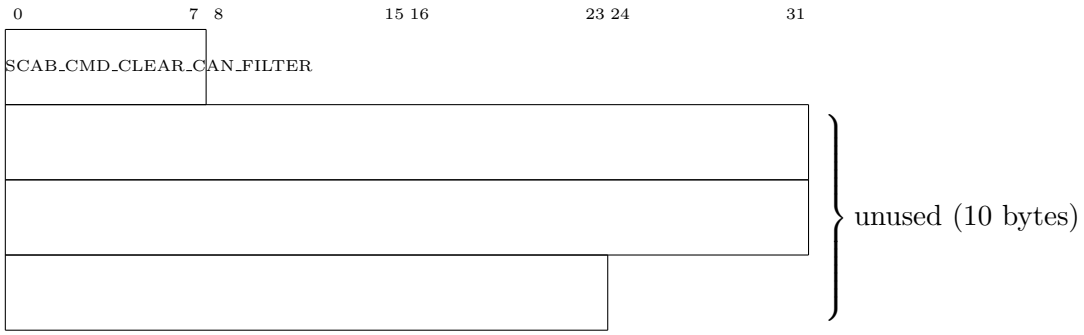

Figure 7: SCAB_CMD_CLEAR_CAN_FILTER format

4.2.8 SCAB_CMD_STATUS format

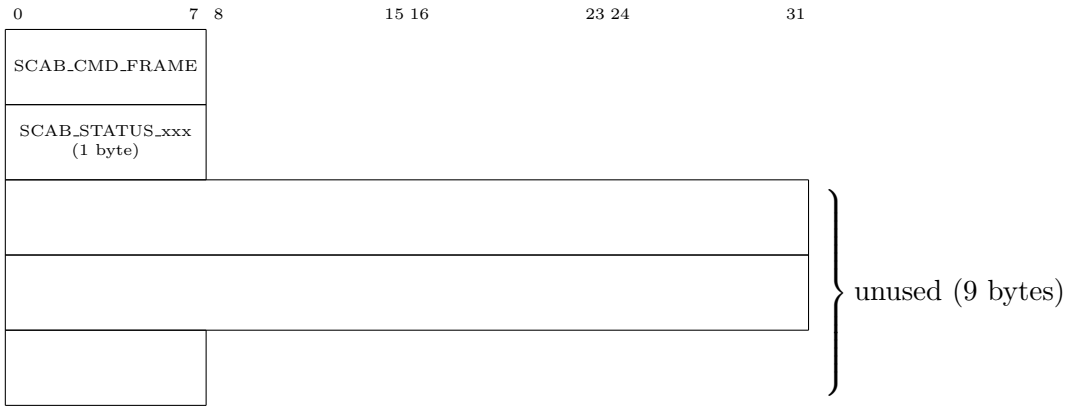

Figure 8: SCAB_CMD_STATUS format