SWT21 lab kit
User manual

Binäs Teknik AB

March 27, 2021

Contents

1 General

The SWT21 lab kit allows communication and measurements using a single microcontroller with a carrier board. The lab kit runs a firmware allowing a computer to be connected as host which can then control the lab kit using the host interface. This allows us to inspect network communication, interact with hardware, etc. from the host computer.

The general connectivity of the lab kit is

test	test	Comment
WiFi	spec?	AP mode and STA mode
Bluetooth	4.2?	
CAN	2.0	Only standard ID:s, 120 Ω termination via jumper configuration
LIN	2.0	Master or slave via jumper configuration
UART		3.3V maximum 2 Mbit/s, configurable 1-2 stop bits and parity bit
ADC	2x	0-3 V or 0-30 V via jumper configuration
DAC	2x	0-3 V or 0-12 V via jumper configuration (0-12 V) requires external power supply)

1.1 Syntax rules

The syntax used for commands in this manual is as follows

- <text> - indicates option to be replaced by other text
- [text] - optional argument to be replaced by other text if used
- [text]... or <text>... - indicates multiple arguments may be used
- {a|b} - means either a or b may be used

2 Host interface

The device is connected to a computer via USB. This connection provides a USB-UART interface which presents itself as a serial port on the computer.

The communication parameters are:

- Baudrate: 2 Mbit/s
- Stopbits: 1
- Parity: None

2.1 System requirements

Windows driver, linux kernel, etc...

2.2 Protocol

The communication interface uses text-based communication where each line is a command or a response. Each line may be up to 255 bytes and ends with a new line control character (\n). Other control characters will be ignored. The format of the commands is as follows:

```
<module> <command> [arguments]...
e.g.: CAN config filter1 1f0 f70
```

Responses can either be acknowledged with an

```
OK [data]
```

, where data is any optional data returned by the command, or an error:

```
ERR <reason>
```

Unsolicited commands are events sent from the lab kit to the host without a preceding command asking directly for it. It is used for events like incomming communication packets and periodic measuremnt data. The syntax is the same as for commands but no response is expected.

3 Firmware update

To update the firmware download and install the official flash download tool from `https://www.espressif.com/en/support/download/other-tools`

Steps:

1.
2.

4 CAN

The CAN bus can be used to send and recive 11-bit ID CAN frames. When sending frames they can either be sent as single frames or as periodic frames. Single frames are enqueued on the transmit buffer immediately after a ”CAN send” command has been sent. Periodic frames stores the frame data and configuration in one of eight periodic frame buffers. When enabled the periodic frame buffers will send its payload periodically according to its configuration.

4.1 CAN commands

4.1.1 CAN status

Syntax

CAN status

Description

This command returns the current status of the CAN subsystem.

Return values

OK <rx on> <err frame count>

rx on - the state of receiving CAN frames.
err frame count - the total number of CAN errors received.

4.1.2 CAN rx

Syntax

CAN rx {on|off}

Description

This command enables or disables receiving CAN frames

Return values

OK

4.1.3 CAN send

Syntax

CAN send <can_id>#{R|data}

Description

This command sends a single CAN frame on the CAN bus. On insufficient CAN memory it will return an overflow error.

can_id - the CAN ID in hexadecimal
R - represents a remote frame
data - is the frame data in hexadecimal

Return values

OK
ERR Invalid argument
ERR Overflow

4.1.4 CAN config

Syntax
CAN config <config key> <arg>...

Description
This command sets the configuration values of the CAN subsystem. Leaving out an argument returns the current value(s) with the same argument format.

Return values
OK
OK <value>...
ERR Invalid argument

4.1.4.1 CAN config baudrate

Config key
baudrate

Arguments
<baudrate> - CAN baudrate in bits/s (e.g. 125000)

Description
This configuration sets the CAN baudrate.

4.1.4.2 CAN config filter<n>

Config key
filter0
filter1

Arguments
<filter> <mask>
<filter> - Filter value in hexadecimal (e.g. 3de) <mask> - Filter mask in hexadecimal (e.g. 7ff)

Description
This configuration sets the values used by the acceptance filters.

periodic send with some storage buffers? add can tx on|off for periodic

4.2 Unsolicited CAN commands

4.2.1 CAN frame

Syntax

CAN frame <can_id>#{R|data}

Description

This command is sent when the system receives a CAN frame including its data

can_id - the CAN ID in hexadecimal
R - represents a remote frame
data - is the frame data in hexadecimal

4.2.2 CAN error frame

Syntax

CAN errframe <flag>

Description

This command is sent when the system receives a CAN error frame

flag - the error flag

5 Command overview

```
All command and responses are '\\n'-terminated.
<module> <command> [arguments]
Max length

Common commands:
config set <parameter> <value> ...
config get <parameter>

Responses
---------
OK [data]
ERR <reason>

ERR Invalid command

ADC commands
------------
```

```
ADC<n> off
ADC<n> single
ADC<n> periodic <period (us)> [offset (us)]
ADC<n> status
    - OK ADC<n> status: <off/single/periodic <period> <offset>>

ADC<n> config set raw
ADC<n> config get raw              - OK ADC<n> raw: [on/off]
ADC<n> config set range <low> <high>
ADC<n> config get range            - OK ADC<n> range: <low> <high>
ADC<n> config timestamp <off/on>

Unsolicited ADC commands
------------------------
ADC<n> value [timestamp] <value>

DAC commands
------------
DAC<n> voltage
DAC<n> raw
DAC<n> config set range <low> <high>
DAC<n> config get range <low> <high>
pwm?

UART commands
-------------
UART<n> send <length>
UART<n>
UART<n> config set baudrate <baudrate>
UART<n> config get baudrate
UART<n> config set format <format>     - e.g. 8n1
UART<n> config get format

UART config parameters
----------------------
baudrate - <baudrate int32>
format - <format string> <bits><n|o|p><1|2>
    e.g. 8n1 for 8 bits, no parity and 1 stop bit

Unsolicited UART commands
-------------------------
UART<n> overrun error <n>
UART<n> underrun error <n>
UART<n> framing error <n>
UART<n> parity error <n>
```