SWT21 lab kit
User manual

Binäs Teknik AB

May 26, 2021

Contents

1 General

The SWT21 lab kit allows communication and measurements using a single microcontroller with a carrier board. The lab kit comes with a pre-installed firmware described in this manual 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 lab-kit may also be used as part of a model based environment where it may act either as an HIL executor or as a device under test. This functionallity is not described in this manual.

It is also possible to upload custom firmwares either developed independently or based on the original firmware described in this manual. For more information see the repository at `https://github.com/jlublin/swt21_fw`

The general connectivity of the lab kit is

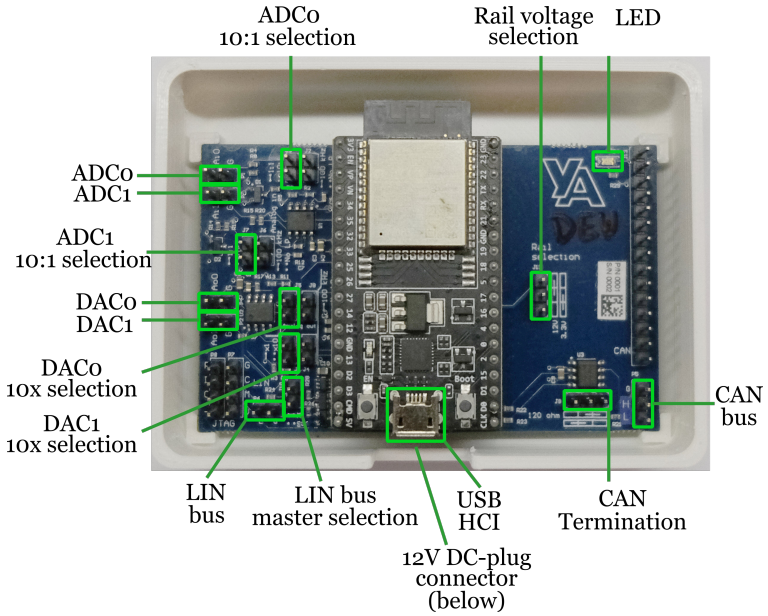
test	test	Comment
WiFi 802.11 b/g/n 2.4 GHz		AP mode and STA mode
Bluetooth 4.2		Classic & Low energy mode
CAN 2.0	1x	Only standard ID:s, 120 Ω termination available via jumper configuration
LIN 2.2	1x	Master or slave via jumper configuration
UART	1x	3.3V maximum 2 Mbit/s, configurable 1-2 stop bits 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

This manual uses the following syntax for describing commands

<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 Hardware layout



3 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, (COM3, /dev/ttyUSBx, or similar).

The HCI on the lab kit is preconfigured to

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

To access the HCI you may use any serial port monitor of your choice, one recommended monitor is miniterm which is part of the pyserial Python package. To use it follow these steps in the command prompt/terminal:

1. Run `pip3 install pyserial`
2. Run `python -m serial.tools.miniterm <COM-port> 2000000` where <COM-port> is replaced with the actual COM port, e.g. COM3.

3.1 System requirements

The lab kit requires a computer with an USB port and is compatible with Windows, MacOS and Linux.

Drivers for the USB-UART bridge are available from `https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers`

3.2 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>
```

General errors:

- ERR Invalid command
- ERR Invalid arguments
- ERR Command too long

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 incoming communication packets and periodic measurment data. The syntax is the same as for commands but no response is expected.

3.2.1 Boot

On boot an informational string will be written on the host interface with the following syntax

```
SWT21 lab kit
Booting...
Firmware version: <fw version>
Boot reason: <reason>
```

<reason> may be due to several reasons:

- Power on
- SW reset
- WDT reset
- Brownout
- OS panic
- Unknown

4 Firmware update

To update the firmware download and install esptool from the official Python repository (Python is required to be installed).

Steps:

1. Start a command prompt/terminal.
2. Run "pip3 install esptool" to install esptool.
3. Run `esptool.py --chep esp32 --port <port> --baud 921600 write_flash 0x10000 <firmware.bin>` (where <port> is the port in use, e.g. COM3, and <firmware.bin> is your firmware file, e.g. v1.1.bin)
4. Press the EN and the Boot buttons on the lab kit, then release first the EN button and then the Boot button.
5. If the flash counter does not start retry the last step again.

5 LED

The LED can be used to control the LED on the board which may be used to identify which board is currently connected.

5.1 LED commands

5.1.1 led help

Syntax

led help

Description

This command prints out a summary of the LED commands.

Return values

OK
<help text>

5.1.2 led {on|off}

Syntax
led {on|off}

Description
This command turns the LED on or off. Default state is off.

Return values
OK

5.1.3 led blink [half period]

Syntax
led blink [half period]

Description
This command starts blinking the LED staying on and off for half period milliseconds. If ommited the default time of 500 ms is used.

half period - the time spent in on or off state respectively in milliseconds

Return values
OK

6 CAN

The CAN bus can be used to send and recive 11-bit ID CAN frames. When sending frames they can only be sent as single frames. Single frames are enqueued on the transmit buffer immediately after a ”can send” command has been sent.

6.1 CAN commands

6.1.1 can help

Syntax
can help

Description
This command prints out a summary of the CAN commands.

Return values
OK
<help text>

6.1.2 can rx

Syntax
can rx {on|off}

Description
This command enables or disables receiving CAN frames. When off no unsolicited CAN frame commands will be sent. Default state is off.

Return values
OK

6.1.3 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

Example: `can send 13f#02e8`

Return values

OK
ERR Invalid argument
ERR Overflow

6.1.4 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

6.1.4.1 can config brp

Config key
baudrate

Arguments
<brp> - CAN clock baudrate prescaler

Description
This configuration sets the CAN baudrate prescaler. The final baudrate can be calculated from

$$\text{can bitrate} = \frac{80000000}{brp*(1+tseg\_1+tseg\_2)}$$

6.1.4.2 can config tseg_1

Config key
tseg_1

Arguments
<tseg_1> - CAN time segment 1

Description
This configuration sets the number of time quantas for CAN tseg_1. The final baudrate can be calculated from

$$\text{can bitrate} = \frac{80000000}{brp*(1+tseg\_1+tseg\_2)}$$

6.1.4.3 can config tseg_2

Config key
tseg_2

Arguments
<tseg_2> - CAN time segment 2

Description
This configuration sets the number of time quantas for CAN tseg_2. The final baudrate can be calculated from

$$\text{can bitrate} = \frac{80000000}{brp*(1+tseg\_1+tseg\_2)}$$

6.1.4.4 can config sjw

Config key
sjw

Arguments
<sjw> - CAN synchronization jump width

Description
This configuration sets the CAN synchronization jump width. This is the maximum number of time quantas that the synchronization may be synchronized.

6.2 Unsolicited CAN commands

6.2.1 CAN RX

Syntax
CAN RX: <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

Example: `CAN RX: 74e#8e98`

7 LIN

The LIN bus can be used to send and recive LIN 2.2 frames. When sending frames they can only be sent as single frames. Single frames are enqueued on the transmit buffer immediately after a ”lin single” command has been sent.

7.1 LIN commands

7.1.1 lin help

Syntax
lin help

Description
This command prints out a summary of the LIN commands.

Return values
OK
<help text>

7.1.2 lin txbuf

Syntax
lin txbuf <id>#<data>

Description
This command loads <data> into the send buffer for <id>.

id - the LIN ID in decimal
data - is the frame data in hexadecimal

Example: `lin txbuf 17#02e8`

Return values
OK
ERR Invalid argument

7.1.3 lin single

Syntax
lin single <id>

Description
This command enques sending out a frame header for <id>. This command should only be run on the LIN master.

id - the LIN ID in decimal

Return values
OK

7.1.4 lin config

Syntax
lin config <config key> [<arg>]

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

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

7.1.4.1 lin config rx [<id> <len> <chks-type>] | off]

Config key
rx [{<id> <len> <chks-type>] | off}]

Arguments
<id> - LIN frame id
<len> - LIN frame len
<chks-type> - LIN frame checksum type (0: classic, 1: enhanced)

Description
This configuration sets the configuration for a LIN frame to receive by this node. Using the off key will disable receiving this LIN frame.

7.1.4.2 lin config tx [<id> <len> <chks-type>] | off]

Config key
tx [{<id> <len> <chks-type>] | off}]

Arguments
<id> - LIN frame id
<len> - LIN frame len
<chks-type> - LIN frame checksum type (0: classic, 1: enhanced)

Description
This configuration sets the configuration for a LIN frame to transmitt by this node. Using the off key will disable transmitting this LIN frame.

7.2 Unsolicited LIN commands

7.2.1 LIN RX

Syntax
LIN RX: <id>#<data>

Description
This command is sent when the system receives a LIN frame.

id - the LIN ID in decimal
data - is the frame data in hexadecimal

Example: `LIN RX: 13#0e12`

8 UART

The UART can be used to send and recive UART characters.

8.1 UART commands

8.1.1 uart help

Syntax
uart help

Description
This command prints out a summary of the UART commands.

Return values
OK
<help text>

8.1.2 uart sendline

Syntax
uart sendline <text>

Description
This command sends <text> over UART and ends with a newline character '\n'.

text - the line of text to send

Example: `uart sendline Hello world!`

Return values
OK
ERR Invalid argument

8.1.3 uart config

Syntax
uart config <config key> [<arg>]

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

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

8.1.3.1 uart config baudrate [<baudrate>]

Config key
config baudrate [<baudrate>]

Arguments
<baudrate> - UART baudrate

Description
This configuration sets the baudrate for the UART for both transmitting and receiving.

8.1.3.2 uart config parity [<parity>]

Config key
config parity [<parity>]

Arguments
<parity> - UART parity, (n: none, e: even, o: odd)

Description
This configuration sets the parity for the UART for both transmitting and receiving.

Example: `uart config parity e` will set even parity

8.1.3.3 uart config stopbits [<stopbits>]

Config key
config stopbits [<stopbits>]

Arguments
<stopbits> - UART stopbits

Description
This configuration sets the number of stopbits for the UART for both transmitting and receiving.

8.2 Unsolicited UART commands

8.2.1 UART

Syntax
UART: <byte>

Description
This command is sent when the system receives a byte on UART.

byte - the byte received, hexadecimal if < 0x20, otherwise printed as ASCII.

Example: `UART: 0x0a`
Example: `UART: H`

9 ADC

The ADC can be used to measure voltages, both static and dynamic. It can take either single measures or using triggers to take many measuremnts around a trigger condition. Two channels are available which can be accessed using `adc0` or `adc1`.

Each channel may be configured in hardware using jumpers to either use 1:1 voltage scaling with an input voltage range of 0-3 V or 10:1 scaling with a voltage range of 0-30 V.

Values from the ADC may either be a floating point value in volts or a raw ADC conversion value from the ADC in hexadecimal.

The submodule uses the following calibration values for converting raw ADC values to voltages using a linear mapping.

Calibration variable	Description
adc<n>_v1x_0_2v	the measured raw ADC value when applying 0.2 V to the ADC configured in 1:1 voltage division.
adc<n>_v1x_2v	the measured raw ADC value when applying 2 V to the ADC configured in 1:1 voltage division.
adc<n>_v10x_2v	the measured raw ADC value when applying 2 V to the ADC configured in 10:1 voltage division.
adc<n>_v10x_20v	the measured raw ADC value when applying 20 V to the ADC configured in 10:1 voltage division.

9.1 ADC commands

9.1.1 adc<n> help

Syntax

adc<n> help

Description

This command prints out a summary of the ADC commands.

Return values

OK
<help text>

9.1.2 adc<n> off

Syntax

adc<n> off

Description

This command disables all periodic readings.
n - the ADC channel number, 0 or 1

Example: `acd0 off`

Return values

OK

9.1.3 adc<n> single

Syntax

adc<n> single

Description

This command performs a single ADC measurement and returns the value immediately.
n - the ADC channel number, 0 or 1

Return values

ADC<n> <value>
value - the measured vaule, either in floating point or hexadecimal raw value

9.1.4 adc0 trig <trig value> <sample rate> <m> <n>

Syntax

adc0 trig <trig value> <sample rate> <m> <n>

Description

This command is only available on ADC channel 0.
This command performs a trigged measurement, it will continuosly monitor the ADC input for the trigger value and when it finds it it will store <m> number of samples before it and <n> numbers of samples after it and send it to the host.

trig value - the raw trigger value in hexadecimal
sample rate - the rate at which to sample the ADC in samples per second
m - the number of values to output before the trigger value
n - the number of values to output after the trigger value

Return values

ADC0 clk: <clk>
<clk> - The actual sample rate used by the ADC

9.1.5 adc0 trig off

Syntax
adc0 trig off

Description
This command is only available on ADC channel 0.
This command disables any currently running trigger condition.

Return values
OK

9.1.6 adc<n> config

Syntax
adc<n> config <config key> [<arg>]

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

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

9.1.6.1 adc<n> config raw [on/off]

Config key
config raw [on/off]

Arguments
on/off - whether to enable or disable raw mode

Description
When raw mode is enabled then ADC values will be printed in hexadecimal raw mode, when disabled floating point voltage values will be used.

9.1.6.2 adc<n> config 10x [on/off]

Config key
config 10x [on/off]

Arguments
on/off - whether to enable or disable 10x mode

Description
When 10x mode is enabled voltage calculations will be based on 10:1 scaling being enabled, otherwise 1:1 scaling will be used.

9.2 Unsolicited ADC commands

9.2.1 ADC trig <start>+<len>

Syntax
ADC trig <start>+<len>
<value> ...

Description
This command is sent after an ADC trigger has been found. If all samples requested before trigger was stored then the first command will start at 0, otherwise it will start at a higher number. The trigger position is found at index n from the "ADC trig" command.

start - the start index of values in this command *len* - the number of values in this command *value* - the raw value in hexadecimal

Example:
```
ADC trig 312+3
    23 78 23
```

10 DAC

The DAC can be used to output voltages. It can take either voltages or raw values as arguments. Two channels are available which can be accessed using `dac0` or `dac1`.

Each channel may be configured in hardware using jumpers to either use 1x voltage scaling with an output voltage range of 0-3 V or 10x scaling with a voltage range of 0-12 V. Note: >2 V requires 12 V rail to work correctly.

The submodule uses the following calibration values for converting raw DAC values to voltages using a linear mapping.

Calibration variable	Description
dac<n>_min_1x	the measured output voltage at raw value 0 with 1x multiplier configuration.
dac<n>_max_1x	the measured output voltage at raw value 255 with 1x multiplier configuration.
dac<n>_min_10x	the measured output voltage at raw value 0 with 10x multiplier configuration.
dac<n>_max_10x	the measured output voltage at raw value 255 with 10x multiplier configuration.

10.1 DAC commands

10.1.1 dac<n> help

Syntax

dac<n> help

Description

This command prints out a summary of the DAC commands.

Return values

OK
<help text>

10.1.2 dac<n> voltage <voltage>

Syntax

dac<n> voltage <voltage>

Description

This command sets DAC channel n to the specified voltage.
n - the ADC channel number, 0 or 1

voltage - the DAC voltage requested
Example: `dac0 1.4`

Return values OK

OK

10.1.3 dac<n> raw <value>

Syntax
dac<n> raw <value>

Description
This command sets DAC channel n to the specified raw value.
n - the ADC channel number, 0 or 1

value - the DAC raw value requested in decimal (0-255)
Example: `dac0 12`

Return values OK
OK

10.1.4 dac<n> config

Syntax
dac<n> config <config key> [<arg>]

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

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

10.1.4.1 dac<n> config 10x [on/off]

Config key
config 10x [on/off]

Arguments
on/off - whether to enable or disable 10x mode

Description
When 10x mode is enabled the code will assume that the DAC channel jumper is configured in 10x mode.

11 Calibration

The calibration subsystem is used to store and retreive calibration values. The values are retained after a firmware upgrade.

All values are stored as unsigned 32 bit integers. Floating point values are used by converting them to the corresponding 32 bit unsigned integer on a little-endian system.

11.1 Calibration commands

11.1.1 calibration help

Syntax
calibration help

Description
This command prints out a summary of the calibration commands.

Return values
OK
<help text>

11.1.2 calibration list

Syntax
calibration list

Description
This command prints out all the calibration values and their current values.

Return values
OK

11.1.3 calibration write <parameter> <value>

Syntax
calibration write <parameter> <value>

Description
This command writes a new value to a parameter. *parameter* - the parameter to update
value - the value to set

Return values
OK

11.1.4 calibration read <parameter>

Syntax
calibration read <parameter>

Description
This command returns the current value of a parameter.
parameter - the parameter to read

Return values
OK <value>
ERR Could not read parameter