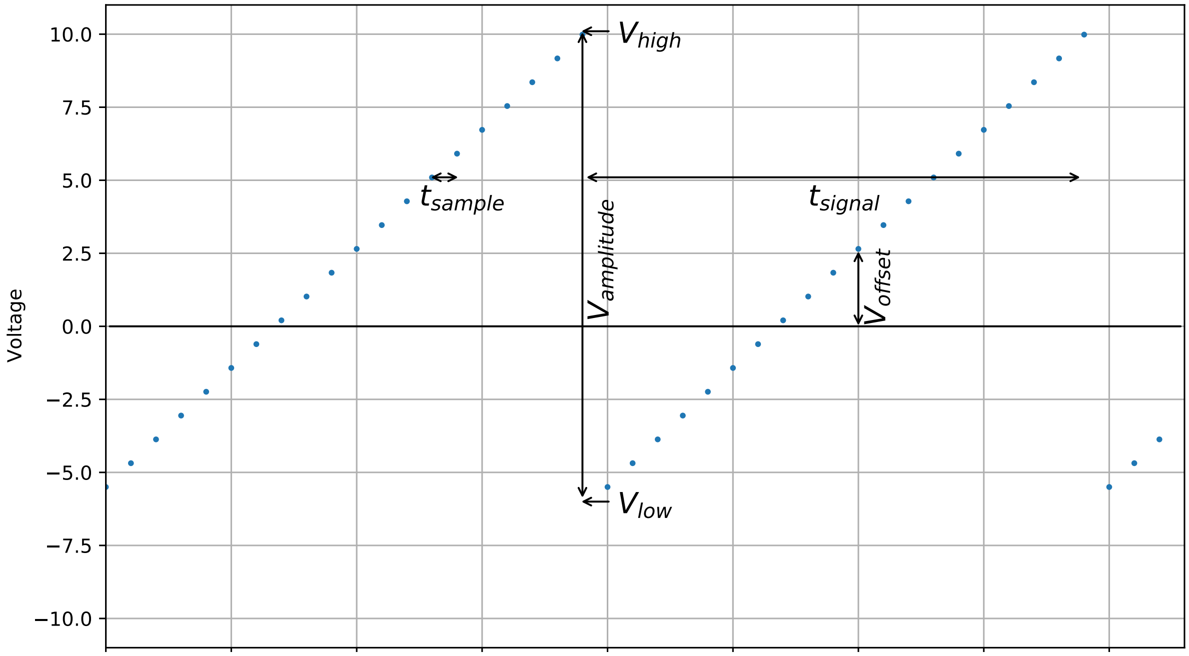
# 1 User Requirements

| Tag RU-1 | Basic Functionality | Module | General |
|---|---|---|---|
| Description | The FirmWare has to access available hardware, to generate two-channel signals in ramp-, constant or arbitrary form, with<br>• sample-rates ( $\neq$ signal-frequency) up to 250kSPS<br>• a resolution of 16bit<br>• resulting in ±10 volts of output voltage | | |



$t_{sample}$ ... sampling-time or -period, alias: trigger-rate

$t_{signal}$ ... signal-time or -period

$f_{sample}$ ... sampling-frequency or -rate

$f_{signal}$ ... signal-frequency or -rate

$N$ ... sample-count, length of the signal-vector

$$f_{sample} = \frac{1}{t_{sample}} = N \cdot f_{signal}$$

$$f_{signal} = \frac{1}{t_{signal}} = \frac{1}{N \cdot t_{sample}}$$

$V_{amplitude}$ ... difference between maximum and minimum voltage of a signal.

$V_{offset}$ ... deviation of a signal from 0 volts.

$V_{high}$ ... maximum voltage of a signal

$V_{low}$ ... minimum voltage of a signal

$$V_{amplitude} = V_{high} - V_{low}$$

$$V_{offset} = \frac{V_{high} + V_{low}}{2}$$

$$\rightarrow V_{high} = V_{offset} + \frac{V_{amplitude}}{2} \quad V_{low} = V_{offset} - \frac{V_{amplitude}}{2}$$

| Tag RI-5 | Priorities | Module | General |
|---|---|---|---|
| Description | In case of temporal overlapping tasks, first priority lays with analogue signal generation, second prio with USB-connectivity, third prio with Miscellaneous functions. | | |

| Tag RU-2 | Last Command Counts | Module | General |
|---|---|---|---|
| Description | The last submitted and accepted value for each parameter is the valid one. | | |

| Tag RU-3 | Parameters | Module | General |
|---|---|---|---|
| Description | The FirmWare has to implement user-adjustable parameters according to Tab. 1. | | |

| Parameter | Values | reset value | Dim. | Type |
|---|---|---|---|---|
| TriggerA State | off\|idle\|arm\|run | idle | | enum |
| TrigA Input | USB\|ext\|TrigB\|butt0 | TrigB | | enum |
| TrigA Signal-Rate | 100m ... 125k | 30.00e3 | Hz | float |
| TrigA Signal-Period | 8u ... 10 | 3.33e-5 | s | float |
| TrigA Size | 0 ... 250000 | 1000 | samples | int |
| TriggerB State | off\|idle\|arm\|run | idle | | enum |
| TrigB Signal-Rate | 100m ... 125k | 30 | Hz | float |
| TrigB Input | USB\|ext\|TrigC\|butt1 | TrigC | | enum |
| TrigB Signal-Period | 8u ... 10 | 3.33e-2 | s | float |
| TrigB Size | 0 ... 250000 | 1000 | samples | int |
| TriggerC State | off\|idle\|arm\|run | idle | | enum |
| TrigC Input | USB\|ext\|butt2 | USB | | enum |
| TrigC Signal-Rate | 20m ... 125k | 3e-2 | Hz | float |
| TrigC Signal-Period | 8u ... 50 | 33.33 | s | float |
| TrigC Size | 0 ... 250000 | 1 | samples | int |
| SourceA Mode | triggered\|detached\|singleshot | triggered | - | enum |
| SourceA Function | ramp\|arbitrary | ramp | - | enum |
| SourceA Symmetry | 0 ... 100 | 0 | percent | float |
| SourceA Amplitude | 0 ... 20 | 20 | volts | float |
| SourceA Offset | -10 ... +10 | 0 | volts | float |
| SourceA High-Volt | -10 ... +10 | +10 | volts | float |
| SourceA Low-Volt | -10 ... +10 | -10 | volts | float |
| SourceA Const-Volt | -10 ... +10 | 0 | volts | float |
| SourceA timeout | 0 ... 1000 | 0 | ms | float |
| SourceB Mode | triggered\|detached\|singleshot | triggered | - | enum |
| SourceB Function | ramp\|arbitrary | ramp | - | enum |
| SourceB Symmetry | 0 ... 100 | 0 | percent | float |
| SourceB Amplitude | 0 ... 20 | 20 | volts | float |
| SourceB Offset | -10 ... +10 | 0 | volts | float |
| SourceB High-Volt | -10 ... +10 | +10 | volts | float |
| SourceB Low-Volt | -10 ... +10 | -10 | volts | float |
| SourceB Const-Volt | -10 ... +10 | 0 | volts | float |
| SourceB timeout | 0 ... 1000 | 0 | ms | float |
| I2C mode | off\|USB\|slave | off | - | enum |
| UART mode | off\|USB\|slave | off | - | enum |
| Galvo-Relay | off\|on | off | - | bool |
| SLD-Relay | off\|on | off | - | bool |
| AIM-Relay | off\|on | off | - | bool |
| CAM-Relay | off\|on | off | - | bool |
| Relay5 | off\|on | off | - | bool |
| Relay6 | off\|on | off | - | bool |
| Watchdog | off\|reset\|powerdown\|keepalive | | - | enum |
| WDGTimeout | 0 ... 1000 | 1000 | ms | int |
| CRCmode | off\|on | off | - | bool |
| VerboseMode | off\|on | on | - | bool |
| A-in mode | off\|USB\|trig'd | - | - | enum |
| A-in value | 0 ... $2^{12}$ | - | LSB | int |
| D-IO mode | off\|in\|out | - | - | enum |
| D-IO value | 0 ... $2^{16}$ | - | bin-vect | int |

Table 1: user-adjustable parameters

| Tag RU-4 | USB-Protocol | Module | USB-Stack |
|---|---|---|---|
| Description | The device has to provide the user with a USB-Interface. It has to be in the form of a VCP, text-based and SCPI-oriented. Messages in either direction may be up to 100 characters long and have to be delimited by the linefeed symbol '\n'. | | |

| Tag RU-5 | USB-Actions | Module | USB-Stack |
|---|---|---|---|
| Description | The FirmWare has to perform actions and state transitions as requested by USB-messages. | | |

| Tag RU-6 | Verbose | Module | USB-Stack |
|---|---|---|---|
| Description | The FW has to reply to every USB-command with a meaningful answer. This is called a 'verbose'-mode, has to be active on startup, but detachable by SCPI-command. Opposite is called $laconic$ - mode | | |

| Tag RU-7 | USB-Timing | Module | USB-Stack |
|---|---|---|---|
| Description | USB-messages sent from the device to the host must be sent with a minimum interval of 1ms. The device must receive USB-messages in intervals up to 1ms. | | |

| Tag RU-8 | Case-Insensitivity | Module | USB-Stack |
|---|---|---|---|
| Description | The SCPI-detection has to be case-insensitive, and respond to the long form as well as the short form of SCPI commands. | | |

| Tag RU-9 | USB-turnoff | Module | USB-Stack |
|---|---|---|---|
| Description | The FirmWare has to deactivate USB-reactivity during A-, B- or C-scans, unless in freerun-mode. On startup, this functionality is active. | | |

| Tag RU-10 | SCPI | Module | USB-Stack |
|---|---|---|---|
| Description | The FirmWare has to parse USB-messages in a SCPI-fashion as defined in document "USB-Protocol.pdf", into FW-internal data structures. | | |

| Tag RU-11 | Restart | Module | USB-Stack |
|---|---|---|---|
| Description | The FirmWare has to perform a complete System-restart, when requested by USB-command. | | |

| Tag RU-12 | Standard-SCPIs | Module | USB-Stack |
|---|---|---|---|
| Description | The FirmWare has to implement mandatory SCPI-command according to IEEE 488.2 | | |

| Tag RU-13 | Arbitrary Signal Vectors | Module | Signals |
|---|---|---|---|
| Description | The FirmWare must provide functionality to load user-defined arbitrary signal vectors, individually for both channels. In $verbose$ - mode, every single transmitted value will be replied with a meaningful message, in $laconic$ - mode, only the average value of the final vector will be replied Values will be transmitted one value per USB-command. optional: Transmit-mode to submit values chunk-wise. | | |

| Tag RU-14 | Vector Length | Module | Signals |
|---|---|---|---|
| Description | The FirmWare must provide functionality to set a user-defined signal vector length, either for ramp- and arbitrary signal, individually for both channels. | | |

| Tag RU-15 | source states | Module | Signals |
|---|---|---|---|
| Description | Signal generation must contain the following operational modes: $triggered$, $detached$, $single-shot$<br>• $triggered$ : each pulse of the corresponding trigger causes the next vector value to be represented at the analogue output (default)<br>• $detached$ : analogue output holds a certain constant level, regardless of trigger and vector values ( alias: $ref-pos$ - mode )<br>• $single-shot$ : analogue output holds a certain constant level, and returns to 0 volt after a specified timeout. | | |

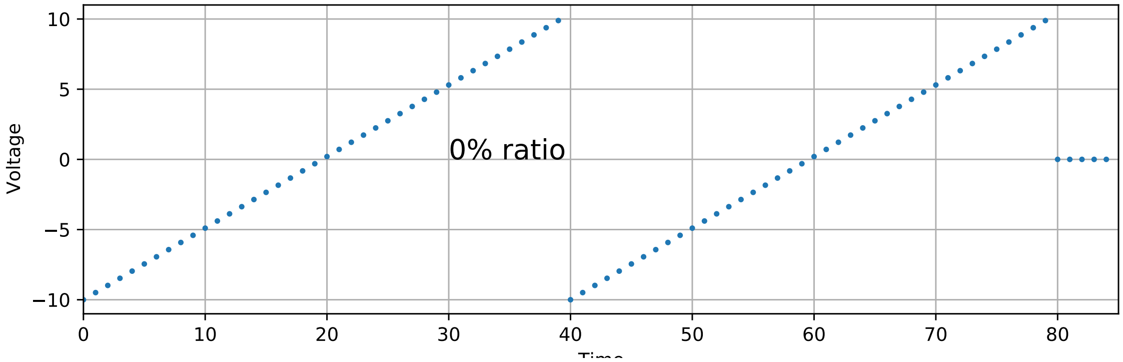
| Tag RU-16 | Default Ramp Signals | Module | Signals |
|---|---|---|---|
| Description | By default, signal vectors are to be loaded with ramp signals. | | |

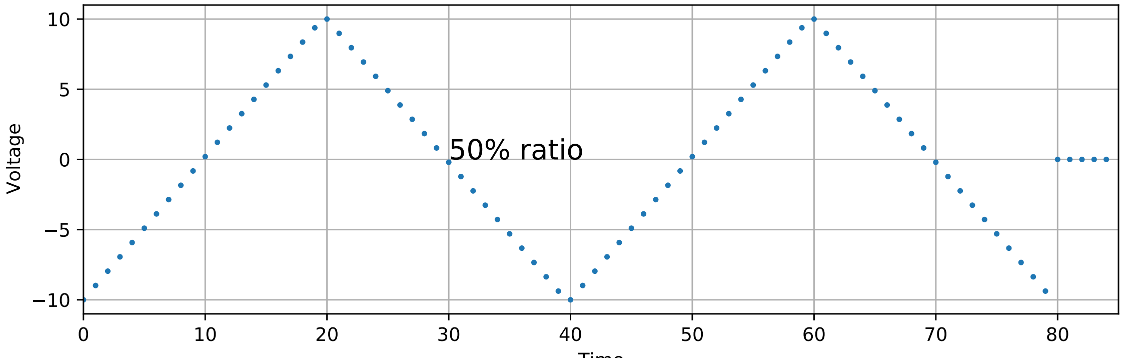
| Tag RU-17 | signal-end | Module | Signals |
|---|---|---|---|
| Description | The FirmWare has to stop signal generation upon completion of all vector lengths and reset analogue outputs to 0V. | | |

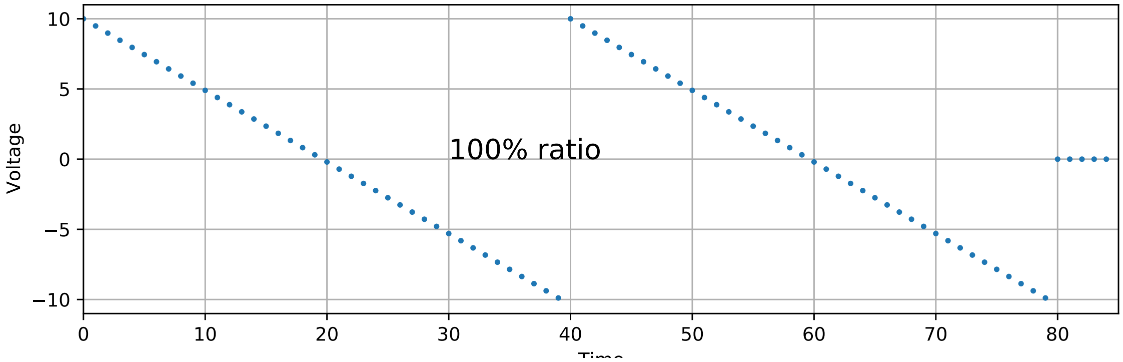
| Tag RU-18 | free-run | Module | Signals |
|---|---|---|---|
| Description | The FirmWare has to provide a freerun mode. This mode continues signal generation, until a specific stop command is submitted via USB. | | |

| Tag RU-19 | Adjustable Signal Parameters | Module | Signals |
|---|---|---|---|
| Description | Signal generation has to be adjustable in amplitude and offset **or** high and low-voltage, signal-freq, **or** -period ). This values apply to ramp- as well as arbitrary signals and will be applied to the signal vectors in a overwriting manner. | | |

| Tag RU-20 | Ramp symmetry | Module | Signals |
|---|---|---|---|
| Description | Ramp signals must have adjustable symmetry/asymmetry between 0$\%$ and 100$\%$. The according meaning is depicted in the following graphics. | | |







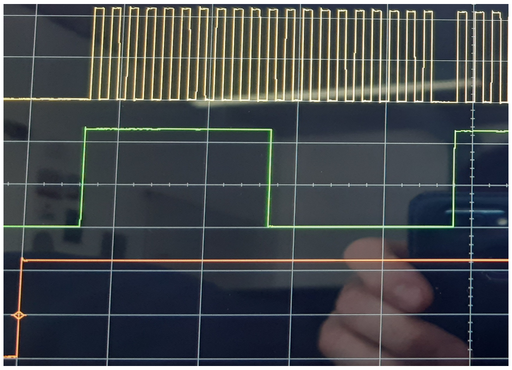
| Tag RU-21 | Trigger-IO | Module | Triggers |
|---|---|---|---|
| Description | Internal Trigger-Pulses must be put out via corresponding Trigger-outputs | | |

| Tag RU-22 | Trigger-Source | Module | Triggers |
|---|---|---|---|
| Description | Trigger-Modules must be implemented to handle timing of the signal-generation, comprising following input-sources: $USB$, $Trigger - Input$, $Superior\text{-}Trigger$, $Push - button$ | | |

| Tag RU-23 | Timing-Parameters | Module | Triggers |
|---|---|---|---|
| Description | The FirmWare has to accept signal frequency or signal period and signal vector length as parameters. It has to reply with the actual frequency/period or an error message. | | |

| Tag RU-24 | Timing-Calc | Module | Triggers |
|---|---|---|---|
| Description | The FirmWare has to derive necessary sample-rates and trigger-periods from signal period and vector length, either by calculation or by selection from a look-up-table. | | |

| Tag RU-25 | Sequences | Module | Triggers |
|---|---|---|---|
| Description | The FirmWare has to generate sequences of A, B and C-Triggers. A-Trigger pulses have a duty-cycle of 50%, B and C-Trigger have falling edges upon completion. | | |



| Tag RU-26 | Buttons,LEDs | Module | Miscellaneous |
|---|---|---|---|
| Description | The FirmWare must access the available push-buttons and state-LEDs. | | |

| Tag RU-27 | Button-Function | Module | Miscellaneous |
|---|---|---|---|
| Description | Push-buttons must be programmed to cause transitions to the devices internal state, in a debounced manner. | | |

| Tag RU-28 | LED-Function | Module | Miscellaneous |
|---|---|---|---|
| Description | State-LEDs have to represent the current internal state of the device: $idle$, $armed$, $running$ or $error$. | | |

| Tag RU-29 | Relays | Module | Miscellaneous |
|---|---|---|---|
| Description | The FirmWare has to provide access to the available relays. Access must consist of $close$, $open$ and $read$-functions | | |

| Tag RU-30 | Additional IOs | Module | Miscellaneous |
|---|---|---|---|
| Description | The FirmWare has to provide access for available UART-, $I^2C$-, SPI-modules, as well as digital IOs and analogue inputs. | | |

| Tag RU-31 | Additional IO-Modes | Module | Miscellaneous |
|---|---|---|---|
| Description | Functionality for USART-, $I^2C$-, SPI-modules, the digital IOs and analogue inputs must consist of $activation$, $de-activation$, $write$ and $read$. | | |

| Tag RU-32 | IO Read | Module | Miscellaneous |
|---|---|---|---|
| Description | $read$-Function must send received information to the host via USB. $read$-Function must be performed upon USB-command, or slave-action. | | |

| Tag RU-33 | CRC | Module | Miscellaneous |
|---|---|---|---|
| Description | The FirmWare has to implement functions to perform cyclic-redundancy-check calculations and apply it on verification of incoming strings and adaption of outgoing strings | | |

| Tag RU-34 | Watchdog Functionality | Module | Miscellaneous |
|---|---|---|---|
| Description | The FirmWare has to implement functions to enable the processors built-in watchdog and set its parameters. Available modes have to be $reset$, $powerdown$, $keepalive$ | | |

# 2 Specifications

## 2.1 Calculations

### 2.1.1 Resolution and LSB

mapping 20Vpp Voltage space to a resolution of 16bit

- 0 ... 30000 ... 60000
- 1000 ... 31000 ... 61000
- 0 ... 32767 ... 65535
- ???

→ LSB ≙ ...mV

### 2.1.2 Trigger-Lines and Timers

utilisation of the output compare - timers

- TrigA ≙ $TRIG\_2$ ≙ PB3 $\leftarrow TIM2_CH2$
- TrigB ≙ $EN\_3$ ≙ PC6 $\leftarrow TIM8_CH1$
- TrigC ≙ $EN\_4$ ≙ PC7 $\leftarrow TIM3_CH2$

### 2.1.3 Triggers and Voltage - Outputs

association of Triggers and their analogue outputs

- TriggerB → SourceB → Vout1
- TriggerA → SourceA → Vout2

### 2.2 FSM, FW-Struct

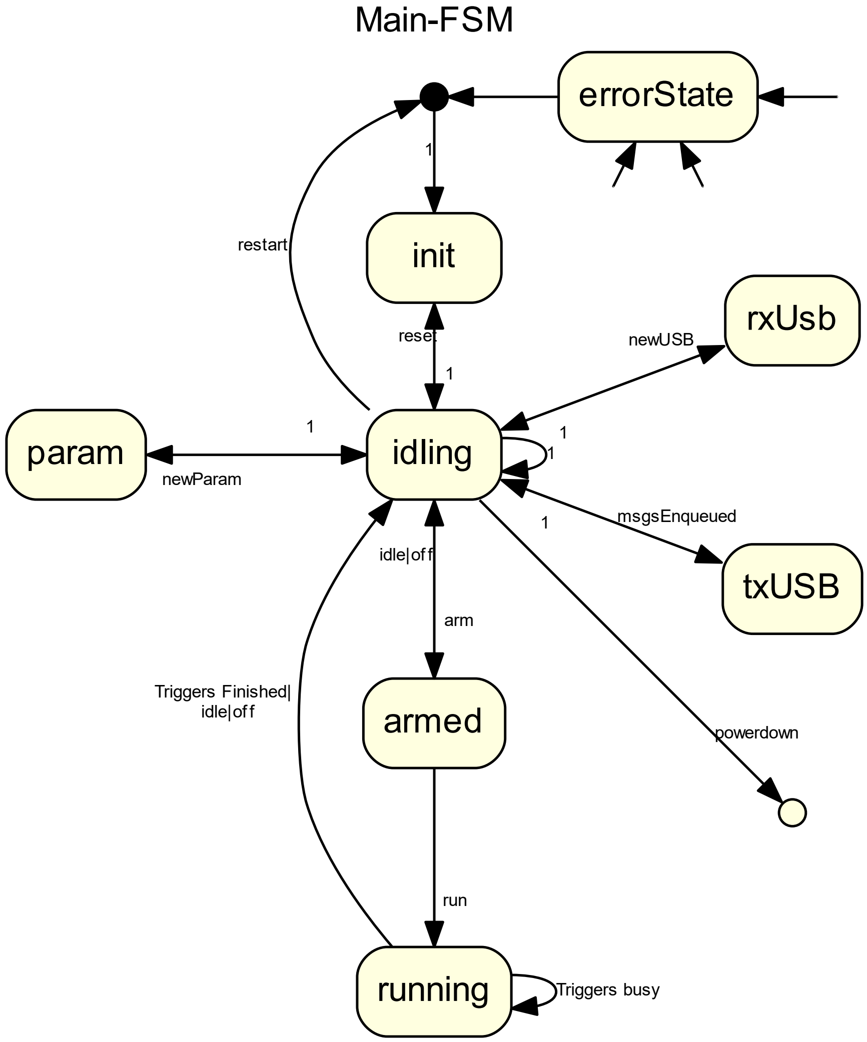


Figure 1: overarching Finite state machine

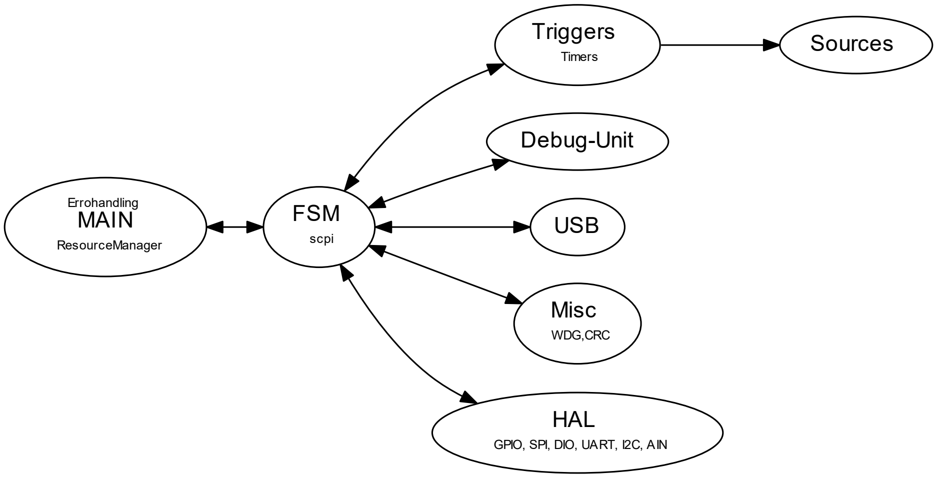


Figure 2: Modular structure

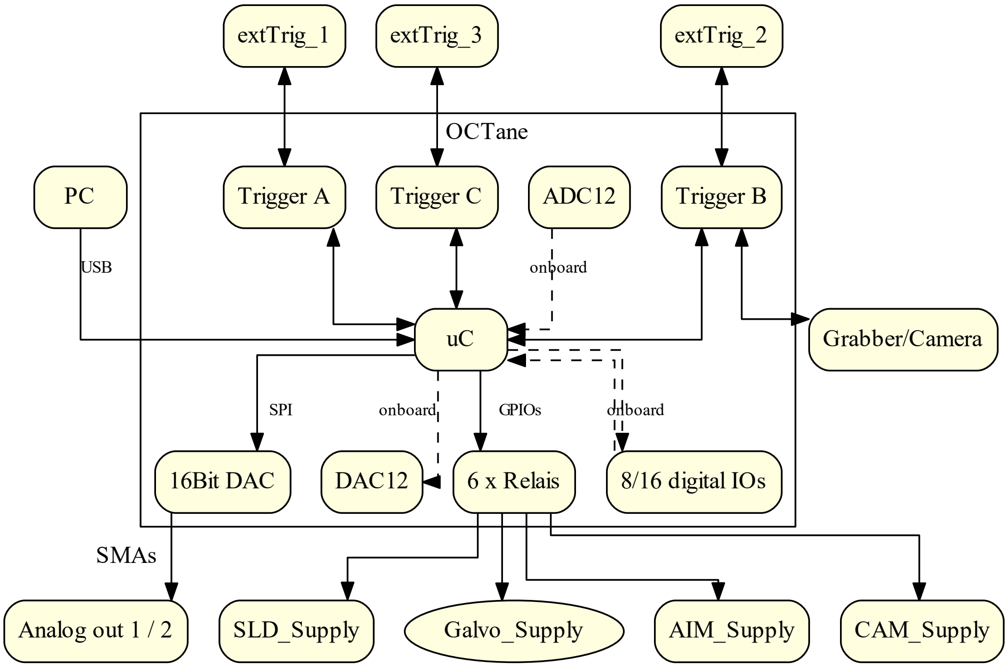


Figure 3: HardWare

| Tag RI- | Modules | Module | General |
|---|---|---|---|
| Description | The Firmware has to be partitioned into these Modules:<br>• Main (Errorhandling)<br>• Triggers (Timers)<br>• Sources (SPI)<br>• Debug-Unit<br>• obs? System (WDG,CRC)<br>• FSM (SCPI)<br>• USB<br>• ResourceManager<br>• HAL (GPIO,WDG,CRC,DIO,UART,I2C,AIN)<br>• obs? Misc (DIO,UART,I2C,AIN) | | |

| Tag RI- | FSM-States | Module | Main |
|---|---|---|---|
| Description | the main-FSM has to implement the following states<br>• init<br>• running<br>• armed<br>• idling<br>• parametrizing<br>• txUSB<br>• rxUsb<br>• errorState | | |

| Tag RI- | ResMan | Module | General |
|---|---|---|---|
| Description | All init()-Functions must probe the ResMan and only take and use a Resource when free. All deinit() - Functions must release Resources. | | |

| Tag RI- | Source-Modules | Module | Signals |
|---|---|---|---|
| Description | 16Bit-Voltage-Sources have to be accessible by following functions:<br>• init() *// not sure if necessary*<br>• sendWord(**bool** source, **uint16_t** word)*//send 16Bit value over SPI to analog output*<br>• loadArb(**bool** source, **uint16_t** size)<br>• scaleArb(**bool** source, **uint16_t** high, **uint16_t** low) // rescale signal-vector vertically<br>• loadRamp(**bool** source, **uint16_t** size, **uint16_t** high, **uint16_t** low)<br>• enab/disab analogue outputs A or B<br>• deinit()*//not sure if necessary*<br>and following helper functions:<br>• **float** word2volt(**uint16_t** word)<br>• **uint16_t** volt2word(**float** voltage)*//–10V –> 0x00, 0 –> 30000, +10V –> 60000*<br>•<br>and structures holding following data:<br>• SPIx<br>• signalVector<br>• mode (triggered, detached, sinlgeshot)<br>• word min // upper vert. limit of ramp/arb<br>• word max // lower vert. limit of ramp/arb<br>• uint32 pulseTime<br>• | | |

| Tag RI- | scaling signals | Module | Signals |
|---|---|---|---|
| Description | Vertically scaling of signal vectors will be applied irreversibly to signal-vectors in place. Amplitude, offset, high- and low-voltages are to be converted into min- and max-word and these again calculated onto existing vector values. | | |

| Tag RI- | writing signals | Module | Signals |
|---|---|---|---|
| Description | Writing an arbitrary signal vector is only permitted in 'arbitrary'-mode. Writing an arbitrary signal vector is complete if sufficient values were submitted and accepted. Writing an arbitrary signal vector can be aborted by setting the Sources mode to 'ramp'. | | |

| Tag RI- | ResMan | Module | General |
|---|---|---|---|
| Description | The FirmWare has to perform resource-management. This denotes to sanity-check managed resources being used by functionalities and deny functions if usage of resources would overlap. | | |

| Tag RI- | ResourceList | Module | General |
|---|---|---|---|
| Description | List of Resources to be managed:<br>• Analogue Outputs, including SPIs, enable-Pins<br>• Analogue Inputs<br>• Digital Inputs/Outputs<br>• UART-Port<br>• I2C-Port<br>• Debug-Unit<br>• all processor-pins | | |

| Tag RI- | Debug-Unit | Module | Main |
|---|---|---|---|
| Description | The Firmware has to implement a Debug-Unit, 8 digital outputs, that can be used to signalize certain events by setting/resetting/toggling them. Required functions are<br>• initDbgUnit(**void**)<br>• setDbgPinX(**void**)<br>• clrDbgPinX(**void**)<br>• tglDbgPinX(**void**)<br>• deinitDbgUnit(**void**) | | |

| Tag RI- | USB-Transceiver | Module | USB |
|---|---|---|---|
| Description | The Firmware has to implement a USB-Transceiver for string-messages via VCP/CDC. Endpoints, to send and receive data have to established, as well as functions to access these endpoints. Required functions are<br>• **uint8_t** CDC_Transmit_FS(**uint8_t*** Buf, **uint16_t** Len);<br>• *// static int8_t CDC_Init_FS(void);*<br>• *// static int8_t CDC_DeInit_FS(void);*<br>• *// static int8_t CDC_Control_FS(uint8_t cmd, uint8_t* pbuf, uint16_t length);*<br>• **static** int8_t CDC_Receive_FS(**uint8_t*** pbuf, **uint32_t** *Len);<br>• **uint8_t** newRxUSB(**void**);<br>• **uint32_t** lenRxUSB(**void**);<br>• **void** clrRxUSB(**void**);<br>• **uint8_t** * getRxUSB(**void**);<br>• **bool** txUsb()<br>• initUsb()<br>• suspendUsb()<br>• resumeUsb()<br>• deinitUsb() | | |

| Tag RI- | HAL-Module | Module | |
|---|---|---|---|
| Description | A hardware abstraction layer (HAL) has to be implemented, providing access to all necessary IO-Lines, serial-peripherals, the watchdog timer, cyclic-redundancy-check | | |

| Tag RI- | HAL-Module | Module | |
|---|---|---|---|
| Description | must provide following interfacing functions:<br>• initGPIOS()<br>• setPin()<br>• rstPin()<br>• getPin()<br>• deinitGPIOS()<br>•<br>• initWDG(mode)<br>• setWDGtimeout()<br>• deinitWDG()<br>• initCRC()<br>• bool rxCRC(char * )<br>• txCRC(char * )<br>• deinitCRC()<br>• AIN, I2C, UART, DIO,<br>and following helper functions:<br>•<br>and structures holding following data:<br>• | | |

| Tag RI- | scpi detection | Module | |
|---|---|---|---|
| Description | Implement USB-protocol in rSCPI.h, separately in short/longform, as well as an enum, representing the index of every command in the LUT. In the USB-ISR only mapping of the recieved string to a global variable and signaling to the FSM in main, that new data is to be processed happens, as well as sending out eventual Strings via USB. SCPI parsing in the FSM: looping over the SCPI-LUT and strncmp it to the input, until positive. Then either execute command immediately, or sscanf in the data. | | |

| Tag | scpi case-insensitive | Module | USB |
|---|---|---|---|
| Description | strncascmp() ensures, that scpi-commands are detected case-insensitive | | |

| Tag RI- | thread-safe vriables | Module | main |
|---|---|---|---|
| Description | Signalling between the FSM and the ISRs have to be thread-safe, and are therefore done via atomic operations, for example flags, semaphores or mutexes. Larger quantities of shared data, e.g. the signal vectors, are to be written, when the according ISR is deactivated, and may not be written on, while ISRs might read them. | | |

| Tag RI- | Naming Conventions | Module | General |
|---|---|---|---|
| Description | The following conventions shall be applied:<br>• camelCase<br>• acting-Functions: 'verbNoun()'<br>• binary queries: 'is...()'<br>• | | |

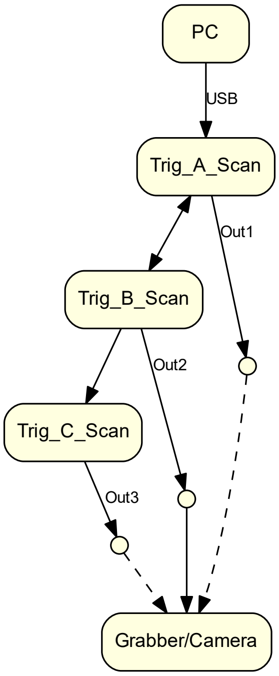


Figure 4: Trigger-Structure, basic concept

| Tag RI- | naming convention | Module | General |
|---|---|---|---|
| Description | naming scheme for digital IOs:<br>• void set<PinXY>();<br>• void clr<PinXY>();<br>• bool get<PinXY>();<br>e.g.:<br>• void setEN3();<br>• void clrLED3();<br>• bool getGPIO7(); | | |

| Tag RI- | third party libs | Module | |
|---|---|---|---|
| Description | • CMSIS - ARM CoreM4 - Libraries<br>• stdbool.h<br>• usbdcdcif.h<br>• STM32F4 Pin- and Register-Defines | | |

K:\FH\MA\wim.txt durchforsten

enums

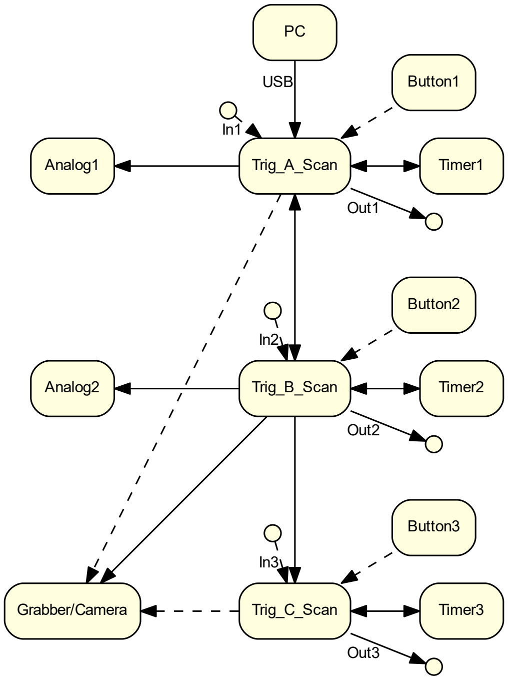


Figure 5: Trigger-Structure, extended concept

```
SourceA|B
TrigA|B|C
vars

* Module:
main - Superloop
FSM
uint8_t mainFSM() return 0 ... powerdown, -1 error, 1 ... reset, 2 ... restart
bool parametrize(SCPI-ID, char * scpiString)
TestCases: mirror all the functionalities requested by User-requirements
HAL Buttons, GPIOs, Relays, LEDs
Miscellaneous - AnalogIN: Burst-mode wo ADCs die DAC-vektoren missbrauchen?
- DIO
- UART
- I2C
System CRC, Wdg, Pwd/Rest/Rese/...
```

| Tag RI- | | Module | |
|---|---|---|---|
| Description | | | |

| Tag RI- | opt pausing of Triggers | Module | |
|---|---|---|---|
| Description | Optional functionality: a falling Edge on a Trigger-In or pushing a dedicated Button (at red LED) during the running state, pauses or stops a whle Trigger-Sequence | | |

| Tag RI- | rxUSB and parametrizing | Module | USB-Stack |
|---|---|---|---|
| Description | upon recieving a valid USB-packages, a parametrizazion-procedure has to be executed: validieren, applizieren, module updaten (Timers, Vektoren, Miscs ....) parametrize() - functions sits within the FSM Module and holds the long list of scpiID -> functions-calls | | |

| Tag RI- | ISRs | Module | |
|---|---|---|---|
| Description | Necessary ISRs:<br>- Buttons<br>- Timers for Triggers<br>- Trigger-Inputs<br>- UART, I2C, SPI<br>- USB<br>- Timer for Debounce<br>- Timer for Timeouts | | |

| Tag RI- | USB - safety | Module | |
|---|---|---|---|
| Description | USB-Inputs have to be sanity-checked regarding frequency, length and meaningful messages, as well as parameters within specified ranges | | |

| Tag RI- | avoid reach-through | Module | General |
|---|---|---|---|
| Description | Timers, purposed for Triggers, are only to be accessed by their corresponding Trigger-unit. SPI-Ports are only to be accessed via their corresponding Source-Units. | | |

| Tag RI- | ISR names | Module | IRQs |
|---|---|---|---|
| Description | Are defined in 'startup.....s'-assembler-file. | | |

| Tag RI- | Timer units | Module | Timer |
|---|---|---|---|
| Description | must provide following interfacing functions:<br>- **bool** initTimer(TIM_TypeDef * TIMx)*//generic init to PWM–mode, no parameters*<br>- **bool** setTimer(TIM_TypeDef * TIMx, **uint32_t** PSC, **uint32_t** ARR, **uint32_t** Pulse, **uint32_t** count)<br>- **void** startTimer(TIM_TypeDef * TIMx)enable IRQ<br>- **void** pauseTimer(TIM_TypeDef * TIMx)disableIRQ<br>- **void** stopTimer(TIM_TypeDef * TIMx)?= reset(**void**)*//pause(), reset counters, clearPin, zeroDAC()*<br>- **void** deinitTimer(TIM_TypeDef * TIMx)*//generic deactivation of module*<br>- **void** ISRs(**void**)<br>and following helper functions:<br>- timeStruct timerLUT(period)<br>- timeStruct timerHybrid(period) ... get PSC from LUT, calc ARR and pulse<br>- $t_{samp} = \frac{((PSC+1)*(ARR+1))}{TCLK}$ -> $PSC = \frac{t_{samp}*TCLK}{(ARR+1)} - 1$<br>- PSC-LUT: Zeitbereiche innerhalb derer ein PSC gilt, tmin $= ((PSC+1)*(0+1))/CLK$, tmax = $((PSC+1)*(2^16+1))/CLK$<br>- jeder Eintrag in PSC-LUT ist eine union aus iTmin, iTmax, oPSC. timerHybrid() schleiferlt da drueber, bis passender<br>- PSC gefunden und rechnet daraus ARR und pulse<br>- timeStruct timerCALC(period) // timerFreq = Fclk/((PSC + 1)(ARR+1))<br>- $uint32\_t$ getTimerPSC(period) // LUT calculating PSC from given Timer-period: 0 for 4us ...<br>and structures holding following data:<br>- timeStruct: ARR, PSC, CCRx, ?length? | | |

| Tag RI- | Trigger units | Module |  |
|---|---|---|---|
| Description | must provide following interfacing functions<br>• bool init(TRIG-ID, length, period, mode/input, , ...)<br>• arm(TRIG-ID)<br>• run(TRIG-ID) = start(TRIG-ID)<br>• pause(TRIG-ID)<br>• stop(TRIG-ID)<br>• reset(TRIG-ID)<br>• deinit(TRIG-ID)<br>and following helper functions:<br>• nix, weil Timing in den Timer-Units berechnet wird<br>and structures holding following data:<br>• TRIG-ID, TIM-ID<br>• state<br>• length, signal-period = time = 1/rate = 1/freq<br>• input/event-source |  |  |

| Tag RI- | ResourceManager | Module | Main |
|---|---|---|---|
| Description | must provide following interfacing functions:<br>• bool isTaken( <ResENUM> )<br>• bool take( <ResENUM> )<br>• void release( <ResENUM> )<br>and following helper functions:<br>•<br>and structures holding following data:<br>• ResENUM RES_ RES_TRIGA RES_TRIGB RES_TRIGC RES_SOURCEA RES_SOURCEB RES_GPIO_E RES_UART RES_I2C RES_SPI RES_ADC RES_Relay RES_TIMx RES_TIMx RES_TIMx RES_TIMx RES_<br>• Ausschlusstabelle zw Pins |  |  |

| Tag RI- | SPI units | Module | |
|---|---|---|---|
| Description | must provide following interfacing functions:<br>• bool initSPI(SPIx) (in 16Bit Mode)<br>• txSpi(SPIx, word )<br>• word rxSpi(SPIx)<br>• deinitSPI(SPIx)<br>and following helper functions:<br>•<br>and structures holding following data:<br>• | | |

| Tag RI- | SCPI Module | Module | |
|---|---|---|---|
| Description | must provide following interfacing functions:<br>• word getIDfromSCPIstring(char * scpiString, uint16 strLen)<br>and structures holding following data:<br>• Stringlists containing recendt-shorts and -longs and norm-commands<br>• enum with exact same order as stringlists<br>• OR: **struct** longform, shortform, ID and for every ID a **#define** 0 TRIGA_STATE_OFF | | |

| Tag RI- | Main module | Module | |
|---|---|---|---|
| Description | The firmware has to implement a main-Module, initializing all permanent modules (System-Clock, USB), activate the main-FSM and execute error-handling. It must provide following functions:<br>• main()<br>• ErrorHandler()<br>and structures holding following data:<br>• scpi-identification string<br>• Processor Clock<br>•<br>• | | |

| Tag RI- | SysTick | Module | main |
|---|---|---|---|
| Description | A SysTick has to be established with a period of 10ms $\mu$s . | | |

| Tag RI- | Clock | Module | main |
|---|---|---|---|
| Description | The Processor has to run at a frequency of 96MHz. | | |

| Tag RI- | ms-Delay | Module | main |
|---|---|---|---|
| Description | The FW has to implement a loosely timed Delay function uint32 start=systickcount; while (systickcount-start<ms); **void** delay_ms(**int** ms) | | |

| Tag RI- | Finite State Machine | Module | main |
|---|---|---|---|
| Description | The FW has to implement an event-driven FSM, that handles the necessary states, transitions and events according to figure 1 | | |

| Tag RI- | LUT Signals | Module | todo |
|---|---|---|---|
| Description | The FW shall implement look-up-tables, 2 vectors, one for each Generator-channel, each at least of $2^{16}$ words(16bit) length, to contain user-defined waveforms for the Analog Outputs | | |

| Tag RI- | Relais abstractions | Module | todo |
|---|---|---|---|
| Description | The FW shall implement functions to access relais, providing ’turn on’, ’turn off’ and ’retrieve status’ | | |

| Tag RI- | Abstraction of SignalGenerator-HW | Module | todo |
|---|---|---|---|
| Description | The FW shall implement functions to access the analog outputs, relying on the SPI-abstraction provided by REQ2.2 Enabling/disabling specified channels, setting specified voltage values. | | |

| Tag RI- | Abstraction of onboard-analogs | Module | todo |
|---|---|---|---|
| Description | desc | | |

| Tag RI- | Abstractions of highspeed digital IOs | Module | todo |
|---|---|---|---|
| Description | The FW shall implement functions to manipulate and read the states of the Trigger IOs and SPI-channels and also to define them as outputs (initially), or inputs. | | |

| Tag RI- | Abstractions of lowspeed digital IOs | Module | todo |
|---|---|---|---|
| Description | The FW shall implement functions to manipulate and read the states of the enable-lines of analog-outputs, and relay-lines and also to define them as outputs | | |

| Tag RI- | Abstractions of misc. digital IOs | Module | todo |
|---|---|---|---|
| Description | The FW shall implement functions to manipulate and read the states of Buttons, Status-LEDs, Port3(8IOs), Port5(16IOs) and also to define them as outputs (initially), or inputs. | | |

| Tag RI- | ISR | Module | General |
|---|---|---|---|
| Description | The FW has to implement ISR-callbacks, to notice trigger-events on Button/Ext TriggerA...D, ISRs for TimersA,B and C, USBrx, ?USBtx? | | |

| Tag RI- | Encoder/Stepper | Module | todo |
|---|---|---|---|
| Description | optional: The Firmware has to access available hardware to read encoder signals and send stepper commands. | | |

| Tag RI- | how to Trigger | Module | Triggers |
|---|---|---|---|
| Description | • A Trigger Event is either a SCPI-Command "TRIGx:STAT:RUN", an external logic Signal on signal-line TRIGx, or by pressing the button BUTTx, or, in linked mode, issued by the superior Trigger.<br>• A Trigger Event causes the according Timer-Interrupt to be enabled. The affected Trigger-unit then, runs for a number of steps specified by "TRIGx:COU count", with a speed defined by "TRIG0:TIME time" or "TRIGx:FREQ freq", and after that disables its own Interrupt.<br>• Activation of at least one Trigger causes the USB-Interrupt to be deactivated, to ensure no interference with the critical timing of Triggering and Signal Generation. Unless Trigger a or B have a duration of at least 2 seconds, or at least one Trigger has a Step-Count of ’0’,in which case USB-Interrupt needs to be active, otherwise, the whole OCTane would be frozen for too long.<br>• Every step, the Trigger pulses its own Trigger-Output line, reads the recent value from its Generators LUT and sends that value to the Gen via SPI, increases the step count and potentially triggers its inferior Trigger.<br>• In linked mode, the superior unit enables the IRQ of the inferior. In independent mode, a button, an ext Trigger, or the SCPI-Command triggers an IRQ. | | |

| Module | Prio | Type | Size | Purpose | initial value |
|---|---|---|---|---|---|
| FSM | H | struct | | | |
| | H | enum | - | state | INIT |
| | M | enum | - | stateNext | INIT |
| | H | flag | - | inUSBnew | low |
| | H | flag | - | outUSBnew | low |
| USB-Stack | H | struct | | | |
| | H | string | | inUSB | empty |
| | H | string | | outUSB | empty |
| | H | | | max. String lenght | |
| SCPI | H | | | | |
| | H | str-list | | SCPI commands lon | |
| | H | str-list | | SCPI responses | |
| | H | str-list | | error codes | |
| | H | enum | | command coding | |
| Trigger A | H | struct | | | |
| Trigger B | H | struct | | | |
| Trigger C | H | struct | | | |
| Generator 1 | H | struct | | | |
| | H | 16 bit | $> 2^{16}$ | Signal-Vector | 0s |
| Generator 2 | H | struct | | | |
| | H | 16 bit | $> 2^{16}$ | Signal-Vector | zeroes |
| Relais 1..8 | H | structs | | | |
| Watchdog | H | struct | | | |
| CRC | H | struct | | | |
| IO-Lines | H | structs | | | |

Table 2: required data inside FW

| Function | Prio | Port | HW-Identifier | Type | initial value |
|---|---|---|---|---|---|
| Trigger 1 | high | A | $TRIG\_1$ | digital IO, HighSpeed | low |
| Trigger 2 | high | B | $TRIG\_2$ | digital IO, HighSpeed | low |
| Trigger 3 | high | B | $TRIG\_3$ | digital IO, HighSpeed | low |
| Trigger 4 | high | B | $TRIG\_4$ | digital IO, HighSpeed | low |
| SPI 1 | high | A | $SCLK\_1, NSS\_1, MISO\_1, MOSI\_1$ | Serial Peripheral IF, HighSpeed | 0x0000 |
| SPI 2 | high | B | $SCLK\_2, NSS\_2, MISO\_2, MOSI\_2$ | Serial Peripheral IF, HighSpeed | 0x0000 |
| Relais, SLD | high | D | $GPIO\_8$ | digital out, LowSpeed | low |
| Relais, AIM | high | D | $GPIO\_7$ | digital out, LowSpeed | low |
| Relais, CAM | high | D | $GPIO\_6$ | digital out, LowSpeed | low |
| Relais, Galvo | high | D | $GPIO\_5$ | digital out, LowSpeed | low |
| State LED, 1 | low | D | $STATE\_1$ | digital out, LowSpeed | low |
| State LED, 2 | low | D | $STATE\_2$ | digital out, LowSpeed | low |
| State LED, 3 | low | D | $STATE\_3$ | digital out, LowSpeed | low |
| State LED, 4 | low | D | $STATE\_4$ | digital out, LowSpeed | low |
| PushButton, 1 | mid | D | $BUTT\_1$ | digital in, LowSpeed | n.a. |
| PushButton, 2 | mid | D | $BUTT\_2$ | digital in, LowSpeed | n.a. |
| PushButton, 3 | mid | D | $BUTT\_3$ | digital in, LowSpeed | n.a. |
| PushButton, 4 | mid | D | $BUTT\_4$ | digital in, LowSpeed | n.a. |
| enable Analog1 | high | C | $EN\_1$ | digital out, LowSpeed | low |
| enable Analog2 | high | C | $EN\_2$ | digital out, LowSpeed | low |
| Analog in 1 | low | C | $ADC\_1$ | analog In | n.a. |
| Analog in 2 | low | C | $ADC\_2$ | analog In | n.a. |
| Analog in 3 | low | C | $ADC\_3$ | analog In | n.a. |
| Analog in 4 | low | C | $ADC\_4$ | analog In | n.a. |
| USB | high | - | $USB\_FS\_DM$ | USB Data- | n.a. |
| USB | high | - | $USB\_FS\_DP$ | USB Data+ | n.a. |
| USB | high | - | $USB\_FS\_ID$ | USB ident. | n.a. |

Table 3: Mapping of IO-Lines

# 3 Implementation

## 3.1 Modules

- Main.h/.c (Errorhandling)
- ResourceMan.h/.c
- Triggers.h/.c
- Timers.h/.c
- Sources.h/.c
- DebugUnit.h/.c
- FSM.h/.c
- SCPI.h/.c
- USB
- HAL (GPIO, SPI)
- Misc.h/.c (WDG,CRC,DIO,UART,IRC,AIN)
- or: HAL (GPIO, SPI, WDG,CRC,DIO,UART,IRC,AIN)

# 4 USB-Protocol for OCTane (SCPI)

| Sub-sys | Parameter | Value | Command | Response |
|---|---|---|---|---|
| Trigger A | State | off\|idle\|arm\|run | TRIGgerA:STATe OFF | <state>\|<error> |
| | State | | TRIGgerA:STATe IDLE | <state>\|<error> |
| | State | | TRIGgerA:STATe ARM | <state>\|<error> |
| | State | | TRIGgerA:STATe RUN | <state>\|<error> |
| | Mode (freerun) | finite | TRIGgerA:MODE FINite | <mode> \|<error> |
| | Mode | infinite | TRIGgerA:MODE INFinite | <mode> \|<error> |
| | Input | USB | TRIGgerA:INput USB | <input>\|<error> |
| | Input | external input | TRIGgerA:INput EXTernal | <input>\|<error> |
| | Input | Trigger B | TRIGgerA:INput TRIGgerB | <input>\|<error> |
| | Input | Trigger C | TRIGgerA:INput TRIGgerC | <input>\|<error> |
| | Input | Button | TRIGgerA:INput BUTTon | <input>\|<error> |
| | Signal-Rate | 1.0e-1 ... 125e3 | TRIGgerA:RATE <freq> | <time>\|<error> |
| | Signal-Period | 8e-6 ... 10 | TRIGgerA:PERIod <time> | <time>\|<error> |
| | Vector-Size | 1...250000 | TRIGgerA:SIZE <size> | <size>\|<error> |
| Trigger B | State | off\|idle\|arm\|run | TRIGgerB:STATe OFF | <state>\|<error> |
| | State | | TRIGgerB:STATe IDLE | <state>\|<error> |
| | State | | TRIGgerB:STATe ARM | <state>\|<error> |
| | State | | TRIGgerB:STATe RUN | <state>\|<error> |
| | Mode (freerun) | finite | TRIGgerB:MODE FINite | <mode> \|<error> |
| | Mode | infinite | TRIGgerB:MODE INFinite | <mode> \|<error> |
| | Input | USB | TRIGgerB:INput USB | <input>\|<error> |
| | Input | External | TRIGgerB:INput EXTernal | <input>\|<error> |
| | Input | Trigger C | TRIGgerB:INput TRIGgerC | <input>\|<error> |
| | Input | Button | TRIGgerB:INput BUTTon | <input>\|<error> |
| | Signal-Rate | 1.0e-1 ... 125e3 | TRIGgerB:RATE <freq> | <time>\|<error> |
| | Signal-Period | 8e-6 ... 10 | TRIGgerB:PERIod <time> | <time>\|<error> |
| | Vector-Size | 1...250000 | TRIGgerB:SIZE <size> | <size>\|<error> |
| Trigger C | State | off\|idle\|arm\|run | TRIGgerC:STATe OFF | <state>\|<error> |
| | State | | TRIGgerC:STATe IDLE | <state>\|<error> |
| | State | | TRIGgerC:STATe ARM | <state>\|<error> |
| | State | | TRIGgerC:STATe RUN | <state>\|<error> |
| | Mode (freerun) | finite | TRIGgerC:MODE FINite | <mode> \|<error> |
| | Mode | infinite | TRIGgerC:MODE INFinite | <mode> \|<error> |
| | Input | USB | TRIGgerC:INput USB | <input>\|<error> |
| | Input | External | TRIGgerC:INput EXTernal | <input>\|<error> |
| | Input | Button | TRIGgerC:INput BUTTon | <input>\|<error> |
| | Signal-Rate | 1.0e-1 ... 125e3 | TRIGgerC:RATE <freq> | <time>\|<error> |
| | Signal-Period | 8e-6 ... 10 | TRIGgerC:PERIod <time> | <time>\|<error> |
| | Vector-Size | 1...250000 | TRIGgerC:SIZE <size> | <size>\|<error> |
| Source-A | Mode | triggered | SOURceA:MODE TRIGgered | <mode>\|<error> |
| | Mode | detached | SOURceA:MODE DETached | <mode>\|<error> |
| | Mode | singleshot | SOURceA:MODE SINGleshot | <mode>\|<error> |
| | Function | Ramp | SOURceA:FUNCtion:SHAPe RAMP | <func>\|<error> |
| | Function | Arbitrary | SOURceA:FUNCtion:SHAPe ARBitrary | <func>\|<error> |
| | Symmetry | 0 ... 100 | SOURceA:RAMP:RATIO <ratio> | <ratio>\|<error> |
| | Arb load | - | SOURceA:ARBitrary:LOAD | <count>\|<error> |
| | Arb val | ±10.000 | SOURceA:ARBitrary:VALUe <idx, val> | <idx, val>\|<error> |
| | Amplitude | 0.000...20.000 | SOURceA:FUNCtion:AMPlitude <ampl> | <ampl>\|<error> |
| | Offset | ±10.000 | SOURceA:FUNCtion:OFFset <offs> | <offs>\|<error> |
| | High | ±10.000 | SOURceA:FUNCtion:HIgh <high> | <high>\|<error> |
| | Low | ±10.000 | SOURceA:FUNCtion:LOw <low> | <low>\|<error> |

<table>
<tr><td></td><td>Constant</td><td>±10.000</td><td>SOURceA:VOLTage:LEVel <volts></td><td><volts>|<error></td></tr>
<tr><td></td><td>Timeout</td><td>1...1000ms</td><td>SOURceA:PULSe:WIDth <time></td><td><time>|<error></td></tr>
<tr><td>Source-B</td><td>Mode</td><td>trig|det|single</td><td>SOURceB:MODE TRIGgered</td><td><mode>|<error></td></tr>
<tr><td></td><td>Mode</td><td>trig|det|single</td><td>SOURceB:MODE DETached</td><td><mode>|<error></td></tr>
<tr><td></td><td>Mode</td><td>trig|det|single</td><td>SOURceB:MODE SINGleshot</td><td><mode>|<error></td></tr>
<tr><td></td><td>Function</td><td>Ramp</td><td>SOURceB:FUNCtion:SHAPe RAMP</td><td><func>|<error></td></tr>
<tr><td></td><td>Function</td><td>Arbitrary</td><td>SOURceB:FUNCtion:SHAPe ARBitrary</td><td><func>|<error></td></tr>
<tr><td></td><td>Symmetry</td><td>0 ... 100</td><td>SOURceB:RAMP:RATIO <ratio></td><td><ratio>|<error></td></tr>
<tr><td></td><td>Arb load</td><td>-</td><td>SOURceB:ARBitrary:LOAD</td><td><count>|<error></td></tr>
<tr><td></td><td>Arb val</td><td>±10.000</td><td>SOURceB:ARBitrary:VALUe <idx, val></td><td><idx, val>|<error></td></tr>
<tr><td></td><td>Amplitude</td><td>0.000...20.000</td><td>SOURceB:FUNCtion:AMPlitude <ampl></td><td><ampl>|<error></td></tr>
<tr><td></td><td>Offset</td><td>±10.000</td><td>SOURceB:FUNCtion:OFFset <offs></td><td><offs>|<error></td></tr>
<tr><td></td><td>High</td><td>±10.000</td><td>SOURceB:FUNCtion:HIgh <high></td><td><high>|<error></td></tr>
<tr><td></td><td>Low</td><td>±10.000</td><td>SOURceB:FUNCtion:LOw <low></td><td><low>|<error></td></tr>
<tr><td></td><td>Constant</td><td>±10.000</td><td>SOURceB:VOLTage:LEVel <volts></td><td><volts>|<error></td></tr>
<tr><td></td><td>Timeout</td><td>1...1000ms</td><td>SOURceB:PULSe:WIDth <time></td><td><time>|<error></td></tr>
<tr><td>Relays</td><td>Galvo</td><td>close|open|read</td><td>ROUTe:<CLOSe|OPEN|STATE?> GAL</td><td><state>|<error></td></tr>
<tr><td></td><td>SLD</td><td>close|open|read</td><td>ROUTe:<CLOSe|OPEN|STATE?> SLD</td><td><state>|<error></td></tr>
<tr><td></td><td>AIM</td><td>close|open|read</td><td>ROUTe:<CLOSe|OPEN|STATE?> AIM</td><td><state>|<error></td></tr>
<tr><td></td><td>CAM</td><td>close|open|read</td><td>ROUTe:<CLOSe|OPEN|STATE?> CAM</td><td><state>|<error></td></tr>
<tr><td>I2C</td><td>mode</td><td>OFF</td><td>I2C::MODE OFF</td><td><mode>|<error></td></tr>
<tr><td></td><td>mode</td><td>USB</td><td>I2C::MODE USB</td><td><mode>|<error></td></tr>
<tr><td></td><td>mode</td><td>slave-action</td><td>I2C::MODE SLAVeaction</td><td><mode>|<error></td></tr>
<tr><td></td><td>write</td><td>0 ... 255</td><td>I2C::WRITe <val></td><td><val>|<error></td></tr>
<tr><td></td><td>read</td><td>0 ... 255</td><td>I2C::READ</td><td><val>|<error></td></tr>
<tr><td>UART</td><td>mode</td><td>OFF</td><td>UART:MODE OFF</td><td><mode>|<error></td></tr>
<tr><td></td><td>mode</td><td>USB</td><td>UART:MODE USB</td><td><mode>|<error></td></tr>
<tr><td></td><td>mode</td><td>slave-IRQ</td><td>UART:MODE SLAVeaction</td><td><mode>|<error></td></tr>
<tr><td></td><td>write</td><td>0 ... 255</td><td>UART:WRITe <val></td><td><val>|<error></td></tr>
<tr><td></td><td>read</td><td>0 ... 255</td><td>UART:READ</td><td><val>|<error></td></tr>
<tr><td>DIO</td><td>mode</td><td>OFF</td><td>DIGIO:MODE OFF</td><td><val>|<error></td></tr>
<tr><td></td><td>mode</td><td>input</td><td>DIGIO:MODE IN</td><td><val>|<error></td></tr>
<tr><td></td><td>mode</td><td>output</td><td>DIGIO:MODE OUT</td><td><val>|<error></td></tr>
<tr><td></td><td>write</td><td>0 .. 65535</td><td>DIGIO:WRIte <val></td><td><val>|<error></td></tr>
<tr><td></td><td>read</td><td>0 .. 65535</td><td>DIGIO:READ</td><td><val>|<error></td></tr>
<tr><td>AnalogIN</td><td>mode</td><td>OFF</td><td>ANAlog0|1|2|3:MODE OFF</td><td><val>|<error></td></tr>
<tr><td></td><td>mode</td><td>USB</td><td>ANAlog0|1|2|3:MODE USB</td><td><val>|<error></td></tr>
<tr><td></td><td>mode</td><td>triggered</td><td>ANAlog0|1|2|3:MODE TRIGA</td><td><val>|<error></td></tr>
<tr><td></td><td>mode</td><td>triggered</td><td>ANAlog0|1|2|3:MODE TRIGB</td><td><val>|<error></td></tr>
<tr><td></td><td>mode</td><td>triggered</td><td>ANAlog0|1|2|3:MODE TRIGC</td><td><val>|<error></td></tr>
<tr><td></td><td>read</td><td>0 ... 4095</td><td>ANAlog0|1|2|3:READ</td><td><val>|<error></td></tr>
<tr><td>System</td><td>CRCmode</td><td>OFF</td><td>SYStem:CRC16 OFF</td><td><state>|<error></td></tr>
<tr><td></td><td>CRCmode</td><td>on</td><td>SYStem:CRC16 ON</td><td><state>|<error></td></tr>
<tr><td></td><td>ShutDown</td><td>-</td><td>SYStem:POWerdown</td><td>POWD|<error></td></tr>
<tr><td></td><td>ListSCPI</td><td>-</td><td>SYStem:LISt</td><td><list>|<error></td></tr>
<tr><td></td><td>RESEt</td><td>-</td><td>SYStem:RESEt</td><td>RESE|<error></td></tr>
<tr><td></td><td>RESTart</td><td>-</td><td>SYStem:RESTart</td><td>REST|<error></td></tr>
<tr><td></td><td>Verbosity</td><td>OFF</td><td>SYStem:VERBose OFF</td><td><mode>|<error></td></tr>
<tr><td></td><td>Verbosity</td><td>on</td><td>SYStem:VERBose ON</td><td><mode>|<error></td></tr>
<tr><td></td><td>Watchdog</td><td>OFF</td><td>SYStem:WATchdog OFF</td><td><mode>|<error></td></tr>
<tr><td></td><td>Watchdog</td><td>on</td><td>SYStem:WATchdog ON</td><td><mode>|<error></td></tr>
<tr><td></td><td>Time</td><td>1...1000ms</td><td>SYStem:WATchdog <time></td><td><time>|<error></td></tr>
</table>

Table 4: OCTane USB-Protocol, commands

| Sub-sys | Parameter | possible messages | occurence | | |
|---|---|---|---|---|---|
| Trigger A\|B\|C | State | TrigX idling\|armed\|running | sent on every state change | | |
| Trigger A\|B\|C | Input | -200 | error, if button in use | | |
| Trigger A\|B\|C | Signal-Rate | -200 | error, if out-of-range | | |
| Trigger A\|B\|C | Signal-Period | -200 | error, if out-of-range | | |
| Trigger A\|B\|C | Vector-Size | -200 | error, if out-of-range | | |
| Source A\|B | Arb load | -200 | error, if not in Arb-mode | | |
| Source A\|B | Arb val | VectorX complete | if sufficient amount of values was sent | | |
| Source A\|B | Arb val | -200 | error, if out-of-range | | |
| Source A\|B | Arb val | -200 | error, if exeeds vector-size | | |
| Source A\|B | Symmetry | -200 | error, if out-of-range | | |
| Source A\|B | Amplitude | -200 | error, if out-of-range | | |
| Source A\|B | Offset | -200 | error, if out-of-range | | |
| Source A\|B | High | -200 | error, if out-of-range | | |
| Source A\|B | Low | -200 | error, if out-of-range | | |
| Source A\|B | Constant | -200 | error, if out-of-range | | |
| Source A\|B | Timeout | -200 | error, if out-of-range | | |
| AIN | input value | AINx: <value> | sent on every corresp. Trigger | | |
| DIN | input value | DIN: <value> | sent on every DIO:READ-Command | | |
| UART | input value | UART: <value> | sent on every corresp. Trigger | | |
| I2C | input value | I2C: <value> | sent on every corresp. Trigger | | |

Table 5: OCTane USB-Protocol, responses

| Command | Description | Action | Return |
|---|---|---|---|
| *CLS | Clear Status Command | | |
| *ESE | Standard Event Status Enable Command | | |
| *ESE? | Standard Event Status Enable Query | - | |
| *ESR? | Standard Event Status Register Query | - | |
| *IDN? | Identification Query | - | ID-String |
| *OPC | Operation Complete Command | | |
| *OPC? | Operation Complete Query | - | |
| *RST | Reset Command | | |
| *SRE | Service Request Enable Command | | |
| *SRE? | Service Request Enable Query | - | |
| *STB? | Read Status Byte Query | - | Status Byte |
| *TST? | Self-Test Query | - | |
| *WAI | Wait-to-Continue Command | | |

Table 6: IEEE 488.2 mandatory commands

| Command | Description | Action | Return |
|---|---|---|---|
| *AAD | Accept Address Command | | |
| *CAL? | Calibration Query | | |
| *DDT | Define Device Trigger Command | | |
| *DDT? | Define Device Trigger Query | | |
| *DLF | Disable Listener Function Command | | |
| *DMC | Define Macro Command | not imp'd | |
| *EMC | Enable Macro Command | not imp'd | |
| *EMC? | Enable Macro Query | not imp'd | |
| *GMC? | Get Macro Contents Query | | |
| *IST? | Individual Status Query | | |
| *LMC? | Learn Macro Query | not imp'd | |

| | | | |
|---|---|---|---|
| *LRN? | Learn Device Setup Query | | |
| *OPT? | Option Identification Query | | |
| *PCB | Pass Control Back | | |
| *PMC | Purge Macros Command | not imp'd | |
| *PRE | Parallel Poll Enable Register Command | | |
| *PRE? | Parallel Poll Enable Register Query | | |
| *PSC | Power-On Status Clear Command | | |
| *PSC? | Power-On Status Clear Query | | |
| *PUD | Protected User Data Command | | |
| *PUD? | Protected User Data Query | | |
| *RCL | Recall Command | | |
| *RDT | Resource Description Transfer Command | | |
| *RDT? | Resource Description Transfer Query | | |
| *SAV | Save Command | | |
| *TRG | Trigger Command | | |
| *RMC | Remove Individual Macro Command | not imp'd | |
| *SDS | Save Default Device Settings Command | | |

Table 7: IEEE 488.2 optional commands

# 5 Standard operating procedures

**ToDo:** update to new protocol

| | |
|---|---|
| SOURce1:FUNCtion:Amplitude 6 | |
| SOURce1:FUNCtion:Offset 3 | |
| SOURce2:FUNCtion:Amplitude 4 | |
| SOURce2:FUNCtion:Offset -4 | |
| TRIGgerC:STATe RUN | start scan sequence |

Table 8: one Volume-Scan

| | |
|---|---|
| SOUR2:VOLT:LEV 4.5 | both Galvos in fixed positions |
| SOUR1:VOLT:LEV -2.95 | no Triggers |
| ... | |
| SOUR2:VOLT:LEV 0 | Send galvos home afterwards |
| SOUR1:VOLT:LEV 0 | |

Table 9: A-Scan in one position

| | |
|---|---|
| TRIGgerB:STATe stop | deactivate |
| TRIGgerA:STATe stop | in exactly this order |
| SOUR1:VOLT:LEV 0 | send Galvo home |
| SOUR1:mode:trig | reattach Galvo to TriggerB |
| TRIGgerB:MODE trigC | reattach TriggerB to TriggerC |

Table 10: B-Scan in one position, continuous A-Scans, 'A-Freerun' Mode-'infinite'

| | |
|---|---|
| SOUR1:MODE free | detach Galvo from its Trigger |
| SOURce2:FUNCtion:Amplitude 3.5 | |
| SOURce2:FUNCtion:Offset 1.95 | |
| TRIGgerB:Mode CONTinuous | ...Trigger will run forever |
| TRIGA:PRE 4 | |
| TRIGA:tcou 74 | ...40kHz A-Scans |
| TRIGB:pre 64 | ....10Hz B-Scans |
| TRIGA:cou 1550 | 1550 samples |
| TRIGB:tcou 36500 | 10Hz |
| TRIGgerB:STATe RUN | activate |
| | |
| TRIGgerB:STATe stop | activate |
| TRIGA:cou 1250 | 1250 samples |
| TRIGB:tcou 14600 | 25Hz |
| TRIGgerB:STATe RUN | activate |
| | |
| TRIGgerB:STATe stop | deactivate |
| TRIGA:cou 620 | 620 samples |
| TRIGB:tcou 7300 | 50Hz |
| TRIGgerB:STATe run | activate |
| | |
| TRIGgerB:STATe stop | deactivate in exactly |
| TRIGgerA:STATe stop | this order |
| SOUR1:VOLT:LEV 0 | send Galvo home |
| SOUR1:mode:trig | reattach Galvo to TriggerB |
| TRIGgerB:MODE trigC | reattach TriggerB to TriggerC |

Table 11: Ivan Patch

# 6 Constraints, Assumptions

Usage of the Processor STM32F407VGT6 imposes following relevant constraints:

- max. clock speed 72MHz
- 1MB program memory
- max. 82 IO-channels

## 6.1 Reference Documents

## 6.2 Abbreviations and Acronyms

| | |
|---|---|
| uC | MicroController |
| FW | Firmware, the Software, running on the uC |
| OCT | Optical Coherence Tomography |
| SW | Software, the Software, running on the OCT-System |
| FSM | Finite State Machine |
| CRC | Cyclic Redundancy Check |
| IO | Input-Output, bidirectional Communcation Lines |
| USB | Universal Serial Bus |
| VCP | Virtual Com Port, a serial connection via USB |
| USB | Universal Serial Bus |
| SCPI | Standard Commands for Programmable Instruments, as defined by IEEE 488.2 |
| LUT | Look-up-table |
| IRQ | Interrupt request |
| ISR | Interrupt-service-routine, a function within the FW, that is called by an IRQ |
| HW | Hardware, the entirety of uC, the PCB and peripherals |
| SLD | Super luminiscence Diode |
| AIM | Aiming Laser |
| CAM | Camera |
| LED | Light emitting diode |
| LSB | Least significant bit |

Table 12: Abbreviations