LEO COMMANDS document v0.0.2

1.0 PARSER in MCU

Command structure down from PC:

<GENERAL COMMAND>:<FEATURE SETTINGS> <VALUE>: … :<FEATURE SETTINGS> <VALUE>;

Example: **IDN?;** (request identification)

LOGA:TRGE RISE; (Logic analyzer->trigger event->rising edge)

OSCP:TRIG NORM:DATA 12B_:EDGE RISE:FREQ 1K__; (Scope->trigger normal, data depth 12bit, Trigger edge rising, sampling frequency 1ksms)

OSCP:STRT; (scope start sampling)

Every command always consists of 4 characters and must be terminated by ";"

2.0 PARSER in PC

Command upstream structure:

<GENERAL COMMAND><MODULE DATA><4bytes DELIMITER>

Data are split by delimiter and passed to module defined at the beginning of command. Module must be able to recognize its data.

Example: **SYSTACK_<DELIM>** command acknowledged

OSCPSMPL<DELIM> sampling of scope has started

SYSTEx<DELIM> error #x to be handled from PC – e.g. inform user

OSCPOSC_<4bytes FREQ><1byte RES><3bytes LEN>CH<1byte CHAN><1byte ACTUAL_CH><DATA>S_OK<DELIM>

FREQ – real sampling frequency

RES – resolution of data

LEN – length of the data (number of bytes expected after header)

CHAN – number of channels being transferred. Each channel data starts with its own header

ACTUAL_CH – channel to which belongs following data

DATA – in binary form, number of bytes corresponds to LEN

3.0 Command list from MCU to PC

SYST

ACK_ – acknowledge

Ex – error

<name> send during enumeration

OSCP

DATA – data for scope

To do others

GEN_

CNT_

LAN_

4.0 Command list from PC to MCU

4.1 GENERAL PARSER (Common file -> cmd_parser.c->CmdParserTask())

CMD_IDN (IDN?) - device identification → response: STM32F303-Nucleo

CMD_RESET_DEVICE (RES!) - device reset

CMD_VERSION (VER?) - FW version

CMD_IS_SHIELD (SH_?) - is shield connected?

CMD_SYSTEM (SYST) - system configuration

CMD_COMMS (COMS) - system communicational capabilities (buffer size, speed, pins, usart – usb..)

CMD_SCOPE (OSCP) - oscilloscope command -> handing over to scope‘s parser (DEDICATED)

CMD_GENERATOR (GEN_) -||- (used for both arbitrary and arbitrary PWM generator)

CMD_COUNTER (CNT_) -||-

CMD_SYNC_PWM (SYNP) -||-

CMD_LOG_ANLYS (LOGA) -||-

default: UNSUPORTED_FUNCTION_ERR_STR

4.2 DEDICATED PARSERS (Common -> cmd_parser.c)

4.2.1 SYSTEM PARSER (parseSystemCmd())

CMD_GET_CONFIG (CFG?) - get system configuration

4.2.2 COMMUNICATION PARSER (parseCommsCmd ())

CMD_GET_CONFIG (CFG?) - get communication configuration

4.2.3 Scope (parseCounterCmd())

CMD_SCOPE_TRIG_MODE(TRIG) – set trigger mode

TRIG_MODE(AUTO,F_A_,NORM,SINGLE)

AUTO waits 4 times buffer length, F_A_ fast auto sends data when buffer is filled

CMD_SCOPE_TRIG_EDGE(EDGE) – set trigger event.

TRIG_EDGE(RISE, FALL)

CMD_SCOPE_SAMPLING_FREQ(FREQ) – set sampling frequency xx samples per second

FREQ(1K__,2k__,5k__,10k_,...,10M_,MAX_)

MAX_ – sets the maximum possible frequency

CMD_SCOPE_DATA_LENGTH(LENG) - set number of samples to be acquired

SAMPLES(100_,100k)

CMD_SCOPE_TRIG_LEVEL(LEVL) – set trigger level after this command 4bytes at expected but only 2bytes are read → 65635 = trigger level 100% of ADC value

CMD_SCOPE_TRIG_CHANNEL(TRCH) – select trigger channel

CHANNELS(1CH_,4CH_)

CMD_SCOPE_DATA_DEPTH(DATA)

DATA_DEPTH(12B_,6B__)

CMD_SCOPE_CHANNELS(CHAN) – number of channels to be sampled

CHANNELS(1CH_,4CH_)

CMD_SCOPE_PRETRIGGER(PRET) - set pre-trigger value after this command 4bytes at expected but only 2bytes are read → 65635 = trigger level 100% of buffer length

CMD_SCOPE_START(STRT) – start scope sampling

CMD_SCOPE_STOP(STOP) – stop scope

CMD_SCOPE_NEXT(NEXT) – scope is in idle state after every data transmission – this command starts new sampling

CMD_SCOPE_ADC_CHANNEL_SET(A_CH) – set analog input pin to different channel – 4bytes expected last byte indicated alternative analog input index (defined in mcu_config.h) and second byte indicates which channel should be changed

CMD_SCOPE_ADC_CHANNEL_SET_DEFAULT(ADEF) – set all analog inputs to their default pin

CMD_SCOPE_ADC_CHANNEL_SET_VREF(AREF) – set all analog channels to measure internal Vref

CMD_GET_CONFIG(CFG?) – get scope configuration (max sampling freq, buffer length, number of channels, PIN description strings, Vref value, supported ranges)

CMD_GET_INPUTS(INP?) – get the list of alternative analog inputs

4.2.4 COUTNER PARSER (parseCounterCmd())

CMD_CNT_START (STRT) - start measuring

CMD_CNT_STOP (STOP) - stop measuring

CMD_CNT_DEINIT (DEIN) - de-initialize counter and its resources

CMD_CNT_MODE (MODE) - 4 measurement types

MODE(ETR__,IC__,REF_,TI__)

CMD_CNT_GATE (GATE) - set gate time for ETR mode

GATE(100m,500m,1s__,10s_)

CMD_CNT_EVENT (EVNT) - rising falling edge selection for IC and TI modes

EVENT(RF1_,RF2_,RO1_,RO2_,FO1_,FO2_,SQAB,SQBA)

CMD_CNT_DUTY_CYCLE (DUCY) - measure DC in IC mode (IC mode has to be already initialized)

DUTY_CYCLE_INIT(DCI1,DCI2,DCD1,DCD2,DCE_,DCX_)

CMD_CNT_TI_MODE (TIMD) - time interval meas. mode selection (more info in TIM_TI_Start())

MODE_EVENT_SEQUENCE(SEQD,SEQI)

CMD_CNT_PRESC (PRE1,PRE2) - prescalers for IC mode channel 1 or channel 2

PRESC1(1x__,2x__,4x__8x__)

CMD_CNT_TIMEOUT_TIM (TIMO) - sets timeout for waiting for the incoming edge in Time Interval meas. mode. 32bit number sent after TIMO.

CMD_CNT_SAMPLE_COUNT (BUF1,BUF2) - Defines how many samples should be taken in IC mode ch 2. BUF2 command is followed by a number.

CMD_CNT_MULT_PSC (PSC_) - app sending the value of TIM prescaler – TODO: handle by MCU

CMD_CNT_MULT_ARR (ARR_) - app sending the value of TIM autorel r. – TODO: handle by MCU

CMD_GET_CONFIG(CFG?) - Getting information about counter channels, periph clocks (mcu_config.h) – everything computed in Host – TODO: rewrite to be standalone..

For more information on counters implementation please refer to diploma thesis of very Mgr. Jan Mucala ;-).

4.2.5 LOGIC ANALYZER PARSER (parseLogAnlysCmd()) – TODO: rewrite

CMD_LOG_ANLYS_INIT (INIT) - initialize Logic analyzer

CMD_LOG_ANLYS_DEINIT (DEIN) - deinitialize Logic analyzer

CMD_LOG_ANLYS_START (STRT) - start LA

CMD_LOG_ANLYS_STOP (STOP) - stop LA

CMD_LOG_ANLYS_PRETRIG (PRET) - pretrigger time in milliseconds (32bit num) followed

CMD_LOG_ANLYS_POSTTRIG (POST) - posttriger time, app sends ARR + PSC vals in 32bit: REDO

CMD_LOG_ANLYS_SAMPLING_FREQ (SMPF) - sampling frequency, again ARR + PSC in 32bit: RDO

CMD_LOG_ANLYS_SAMPLES_NUM (SMPN) - data length, 16bit num.

CMD_LOG_ANLYS_TRIGGER_MODE (TRGM) - triggering mode

TRIG_MODE(AUTO,NORM,SING)

CMD_LOG_ANLYS_TRIGGER_CHANNEL (TRGC) - selection of trigger channel 32bit (0 - 7)

CMD_LOG_ANLYS_TRIGGER_EVENT (TRGE) - select trigger edge

TRIG_EDGE(RISE,FALL)

CMD_GET_CONFIG (CFG?) - getting info about the functionality

4.2.6 GENERATOR PARSER (parseGeneratorCmd ()) – TODO: separate both functions? Arbitrary gen. (DAC) + arbitrary PWM gen.

CMD_GEN_MODE (MODE) - choose between arbitrary PWM DC changing mode or arb DAC

CMD_MODE_PWM (PWM_,DAC)

CMD_GEN_DATA (DATA) – incoming data from PC – all signals are handled as arbitrary this commands takes data from PC and put them into channel and right place in buffer

CMD_GEN_SAMPLING_FREQ (FREQ) - set sampling frequency

CMD_GEN_PWM_FREQ_PSC (FPWP) - setting gen. frequency by conf. PSC of TIM (REDO!)

CMD_GEN_PWM_FREQ_ARR (FPWA) - setting gen. frequency by conf. ARR of TIM (REDO!)

CMD_GEN_PWM_DEINIT (GPDI) - generator PWM deinit

CMD_GET_REAL_FREQ (FRQ?) - get recalculated real frequency according to ARR*PSC possibilit.

CMD_GEN_DATA_LENGTH_CH1 (LCH1) - set data length (samples number in one period) in ch 1

CMD_GEN_DATA_LENGTH_CH2 (LCH2) - set data length (samples number in one period) in ch 2

CMD_GEN_CHANNELS (CHAN) - set number of channels available

CHANNELS(1CH_,2CH_,3CH_,4CH_)

CMD_GEN_OUTBUFF_ON (B_ON) - DAC buffer on

CMD_GEN_OUTBUFF_OFF(B_OF) - DAC buffer off

CMD_GEN_DAC_VAL (DAC_) – set generator into static mode to generate DC value (take 4bytes 2bytes per channel – data represents directly DAC value)

CMD_GEN_START (STRT) - start the generator

CMD_GEN_STOP (STOP) - stop the generator

CMD_GEN_RESET (RSET) - reset the generator

CMD_GET_CONFIG (CFG?) - get configuration of the functionality – number of channels, pins, periph clocks (TODO: get rid of it) (mcu_config.h)

CMD_GET_PWM_CONFIG (PCF?) - get configuration of PWM generator

CMD_GENERATOR (CFG?) - get configuration of DAC generator

4.2.7 SYNCHRONIZED PWM PARSER (parseSyncPwmCmd ()) - Will be excluded

CMD_SYNC_PWM_COMMAND (SCOM) - basic commands

SYNC_PWM_INIT (INIT) - initialization

SYNC_PWM_DEINIT (DINIT) - deinitialization of its resources

SYNC_PWM_START (STRT) - start generating

SYNC_PWM_STOP (STOP) - stop generating

CMD_SYNC_PWM_STEP (STEP) - step or continuous generating

CMD_SYNC_PWM_STEP_ENABLE (STEE) - generates only one period

CMD_SYNC_PWM_STEP_DISABLE (STED) - continuous generation

CMD_SYNC_PWM_CHAN_NUM (CNUM) - select the channel number to be configured by CHAN_CONFIG later

CMD_SYNC_PWM_CHAN_CONFIG (CCON) - followed by 32bit num; 1st 16bit number represents capture compare register value on which DMA is triggered to tranfer the second 16bit number into the CC register (in order to change the edge twice in one TIMer period)

CMD_SYNC_PWM_FREQ (SFRQ) - configuring ARR + PSC in one 32bit: REDO

CMD_SYNC_PWM_CHAN_STATE (SSTA) - wow, it’s too messy and difficult..

CMD_GET_CONFIG (CFG?)