LUCA_TRS.dll – User manual

Author: **Mauro Buttafava**
Guide version **0.2**
Firmware version: **5.10**
Date: **13/04/18**

System management functions:	
TRS_INIT	Opens the communication and turns ON the TRS module (slow, needs config. file)
TRS_CLOSE	Turns OFF the TRS module and closes the communication (slow)
TRS_LASER	Turns ON or OFF the TRS laser module (slow, needs config. file)
TRS_MEAS	Starts or Stops a measurement (automatic, manual continuous or single-shot)
TRS_MAP	Sets the automatic measurement map (16 locations: wavelength + integration time)
TRS_SET	Sets the measurement parameters
TRS_WATTEN	Sets the optical attenuation values (fast)
Data acquisition functions:	
TRS_ACQ	Checks for new data and downloads the measurement
TRS_CTRS	Acquires the SiPM count rates (fast)
TRS_RATTEN	Checks the status of optical attenuators (fast)
Diagnostic functions:	
TRS_LSELF	Laser module self-test routine (slow, generates report file)
TRS_SSELF	TRS module self-test routine (slow, generates report file)

TRS_INIT:

Opens the communication with the LUCA TRS module, then turns-ON and initialize the entire system (except the laser section). The function includes a firmware version verification (firmware and library versions must match). This function requires the "system.cfg" configuration file (which has to be placed in the same DLL file folder). **TRS_INIT must be executed as the first function.** TRS_INIT will prompt user messages in case of connection errors, corrupted or missing configuration file or wrong firmware version. **The estimated execution time is 10 seconds.**

NOTE: after system power-ON the FPGA starts to download the configuration bit-stream from the onboard memory. During this operation the connection cannot be established. **Please wait 60 seconds between TRS module turn-ON and TRS_INIT first execution.**

Prototype:

```
uint32_t TRS_INIT(uint64_t *Handle_Out, uint32_t
Total_Registers_Out[], int32_t Total_Registers_Out_Length)
```

Returns:

TRS_INIT returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_Out` is the pointer to the created "Device Handle" (uint64), it must be used for the communication with the system from now on.
- `Total_Registers_Out[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. This array can be initialized with zeros and is then populated by the TRS_INIT function. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_Out_Length` is a int32 which value **must be equal to 128**.

Go to TOP

TRS_CLOSE:

Turns-OFF the entire system (except the laser section, which MUST be already switched OFF) and closes the USB communication. **TRS_CLOSE must be executed as the last function of the program. The estimated execution time is 8 seconds.**

Prototype:

```
uint32_t TRS_CLOSE(uint64_t *Handle_In, uint32_t
Total_Registers_In[], int32_t Total_Registers_In_Length)
```

Returns:

TRS_CLOSE returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Total_Registers_In[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_In_Length` is a int32 which value **must be equal to 128**.

Go to TOP

TRS_LASER:

Turns ON or OFF the laser section of the TRS module. The function enables the thermal control systems, set the laser bias voltages and finally provides the trigger signal. **Laser emission is active after calling NIRS_LASER(…, 1).** This function requires the "laser.cfg" configuration file (which has to be placed in the same DLL file folder). TRS_LASER will prompt user messages in case of corrupted or missing configuration file. The system warm-up time has to be considered starting from the call of NIRS_LASER(…, 1). **The estimated execution time is 80 seconds for turn-ON and 5 seconds for turn-OFF.**

NOTE: Lasers MUST be turned OFF using TRS_LASER(…, 0) before calling the NIRS_CLOSE function.

Prototype:

```
uint32_t TRS_LASER(uint64_t *Handle_In, uint32_t
Total_Registers_In[], uint32_t Total_Registers_Out[], int32_t
Total_Registers_In_Length, int32_t Total_Registers_Out_Length,
uint32_t ON_nOFF)
```

Returns:

TRS_LASER returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Total_Registers_In[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_Out[]` is a pointer to an array composed by 128 uint32 elements, **it MUST be the same pointer of Total_Registers_In[].**
- `Total_Registers_In_Length` is a int32 which value **must be equal to 128**.
- `Total_Registers_Out_Length` is a int32 which value **must be equal to 128**.
- `ON_nOFF` is a uint32 used to turn ON of OFF the laser section of the TRS module. **It must be 1 for turning lasers ON, or 0 for turning lasers OFF.**

[Go to TOP]

TRS_MEAS:

Starts or Stops the execution of TRS measurements. The function also allows to choose between different kinds of measurement operations: automatic (using a pre-defined measurement map), manual continuous (for real-time operation, with a single wavelength) or manual single-shot (the measurement is only started once). **The user must poll the system using TRS_ACQ** to detect new completed measurements and download them. Measurements can be performed also without turning-ON the laser section (i.e. before calling TRS_LASER).

NOTE: Before starting an AUTOMATIC measurement using TRS_MEAS(…, 1), the automatic measurement MAP must be downloaded into the instrument using the TRS_MAP function.

Prototype:

```
uint32_t TRS_MEAS(uint64_t *Handle_In, uint32_t
Total_Registers_In[], uint32_t Total_Registers_Out[], int32_t
Total_Registers_In_Length, int32_t Total_Registers_Out_Length,
uint32_t Meas_Cmd)
```

Returns:

TRS_MEAS returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Total_Registers_In[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_Out[]` is a pointer to an array composed by 128 uint32 elements, **it MUST be the same pointer of Total_Registers_In[].**
- `Total_Registers_In_Length` is a int32 which value **must be equal to 128**.
- `Total_Registers_Out_Length` is a int32 which value **must be equal to 128**.
- `Meas_Cmd` is a uint32, representing the command used to start or stop different kinds of measurement (automatic, manual continuous, manual single-shot). **See table 1 for details.**

Table 1: measurement commands details

Meas_Cmd	Details
0	**STOP** any measurement
1	START an **AUTOMATIC measurement** (see TRS_MAP function)
2	START a **MANUAL measurement**, in CONTINUOUS mode (real-time)
3	TRIGGERS a manual **SINGLE-SHOT measurement**

AUTOMATIC MEASUREMENT:

When an automatic measurement is started, the system continuously executes the scheme previously programmed using TRS_MAP. The laser wavelength is automatically changed, with a switch set-up time of 10 ms. Therefore, the duration of each sub-measurement on the map is the corresponding integration time, plus 10 ms. When a END command is detected (or the 16th sub-measurement has been completed) the system starts again with the firs sub-measurement of the map, until stopped by the user with TRS_MEAS(…, 0). When the automatic measurement is running, the user must continuously poll the system for measurement results, using TRS_ACQ. During the current measurement, the previously completed one is available for download (after the completion of the current measurement, the previous one will be overwritten). When the automatic measurement mode is used, TRS_ACQ function provides also information on the active wavelength and the map index of each downloaded measurement.

MANUAL CONTINUOUS MEASUREMENT:

When the manual continuous measurement mode is used the laser wavelength is always the same and measurements are repeatedly started (in true real-time). The user must continuously poll the system for measurement results, using TRS_ACQ. During the current measurement, the previously completed one is available for download (after the completion of the current measurement, the previous one will be overwritten). The laser wavelength and the integration time for manual-mode measurements can be changed using the TRS_SET function (when no measurement is running). The manual continuous acquisition operation is stopped using TRS_MEAS(…, 0).

MANUAL SINGLE-SHOT MEASUREMENT:

In manual single-shot mode a single measurement is executed when calling TRS_MEAS(…, 3). When the measurement is completed, the system automatically returns in idle condition. After the completion, the user can download the measurement using TRS_ACQ. The laser wavelength and the integration time for manual-mode measurements can be changed using the TRS_SET function (when no measurement is running).

Go to TOP

TRS_MAP:

It is used to program the execution of the automatic measurements. It is composed by a table with 16 locations (called sub-measurements), each of one composed by two fields: the sub-measurement command and the corresponding integration time. The command is used to select the wavelength or to end the sequence in case of less than 16 sub-measurements are needed. After programming the map, the automatic measurement can be started using TRS_MEAS(…, 1). The system goes through the sub-measurements map (with a 10 ms settling time after switching the wavelength) until reaching the 16th location or finding the first END command. Then it restarts from the first location until the measurement is stopped using TRS_MEAS(…, 0). The user must poll the system using TRS_ACQ to, download each completed measurements data.

Prototype:

```
uint32_t TRS_MAP(uint64_t *Handle_In, uint32_t Total_Registers_In[],
uint32_t Total_Registers_Out[], int32_t Total_Registers_In_Length,
int32_t Total_Registers_Out_Length, uint32_t Auto_Cmd[], int32_t
Auto_Cmd_Length, uint32_t Auto_ITime[], int32_t Auto_ITime_Length)
```

Returns:

TRS_MAP returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Total_Registers_In[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_Out[]` is a pointer to an array composed by 128 uint32 elements, **it MUST be the same pointer of Total_Registers_In[].**
- `Total_Registers_In_Length` is a int32 which value **must be equal to 128**.
- `Total_Registers_Out_Length` is a int32 which value **must be equal to 128**.
- `Auto_Cmd[]` is a pointer to an array composed by 16 uint32 elements, for programming the sub-measurement commands (to select the sub-measurement wavelength or the END of the map). See **Table 2** for details.
- `Auto_Cmd_Length` is a int32 which value **must be equal to 16**.

- `Auto_ITime[]` is a pointer to an array composed by 16 uint32 elements, for programming the integration time of each sub-measurement. **The integration time is passed in milliseconds, with a valid range from 100 ms to 10 s.** See **Table 3** for a complete map example.

- `Auto_ITime_Length` is a int32 which value **must be equal to 16**.

Table 2: automatic measurement map commands details

Auto_Cmd	Action
0	END of the map (selects the switch dead channel)
1	Set wavelength: 635
2	Set wavelength: 670
3	Set wavelength: 730
4	Set wavelength: 830
5	Set wavelength: 852
6	Set wavelength: 915
7	Set wavelength: 980
8	Set wavelength: 1040

Table 3: automatic measurement map example

Theoretical measurement scheme:

Sub-meas. index	Wavelength	Sub-meas. Integration time
0	670 nm	1000 ms
1	730 nm	1500 ms
2	635 nm	1000 ms
3	852 nm	2000 ms
…	…	…
12	852 nm	1500 ms
13	END (restart)	Not executed
14	END (restart)	Not executed
15	END (restart)	Not executed

→

Equivalent TRS_MAP parameters:

Auto_Cmd	Auto_ITime
2	1000
3	1500
1	1000
5	2000
…	…
5	1500
0	0
0	0
0	0

Go to TOP

TRS_SET:

It is used to modify TRS system settings and measurement parameters, in particular: i) the SiPM fast counters integration time (see TRS_CTRS); ii) manual measurement wavelength; iii) manual measurement integration time; iv) frequency of the synchronization time-base of the entire system (acting as laser triggering signal and TCSPC stop signal). **Table 4 provides information of TRS_SET parameters valid range and default values.**

NOTE: No measurements have to be running when calling TRS_SET.

NOTE: Laser performance are guaranteed and extensively tested at 40 MHz synchronization rate only! Operation at different frequencies is possible, but not recommended (a frequency change can be useful during system setting-up, to identify light reflections into fibers or optical components).

Prototype:

```
uint32_t TRS_SET(uint64_t *Handle_In, uint32_t Total_Registers_In[],
uint32_t Total_Registers_Out[], int32_t Total_Registers_In_Length,
int32_t Total_Registers_Out_Length, uint32_t Ctrs_ITime, uint32_t
Man_Wavelength, uint32_t Man_ITime, uint32_t Sync_Rate)
```

Returns:

TRS_SET returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Total_Registers_In[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_Out[]` is a pointer to an array composed by 128 uint32 elements, **it MUST be the same pointer of Total_Registers_In[].**
- `Total_Registers_In_Length` is a int32 which value **must be equal to 128**.
- `Total_Registers_Out_Length` is a int32 which value **must be equal to 128**.
- `Ctrs_ITime` is a uint32, representing the integration time duration of the two uncorrelated fast-counters used for continuously monitoring the SiPM count rates. The value is expressed in milliseconds.

- `Man_Wavelength` is a uint32, representing the laser channel used during manual measurements (both continuous and single-shot). Values of this parameter are the same reported in table 2. The value 0 (END) is used to choose the optical switch dead-channel and must be selected to avoid laser emission when no measurements are running. The laser channel selection is immediately executed when calling TRS_SET, thus **laser emission is immediately active when channels from 1 to 8 are selected**.

- `Man_ITime` is a uint32 used to set the integration time duration of manual measurements (both continuous and single-shot). Value is expressed in milliseconds.

- `Sync_Rate` is a uint32 used to change (if needed) the entire TRS module synchronization signal frequency. The recommended value is 40 MHz. The parameter is expressed in MHz.

Table 4: TRS_SET parameters details

Parameter	Unit	Valid value range	Default value
Ctrs_ITime	milliseconds	from 10 ms to 500 ms	100 ms
Man_Wavelength	see Table 2	from 0 to 8	0 (END)
Man_ITime	milliseconds	from 100 ms to 10 s	1 s
Sync_Rate	MHz	from 20 MHz to 50 MHz	40 MHz

Go to TOP

TRS_WATTEN:

Sets the attenuation level of the 8 electrically-actuated optical attenuators, used for the injection power equalization. The function communicates with one attenuator at a time, whose address is specified as a function parameter. The attenuation can be changed using different modalities: (i) specifying the absolute position in steps; (ii) specifying the increment/decrement, in steps, respect to the previous position and (iii) directly providing the absolute attenuation value in dB (see Table 6 for details).

Prototype:

```
uint32_t TRS_WATTEN(uint64_t *Handle_In, uint32_t
Total_Registers_In[], uint32_t Total_Registers_Out[], int32_t
Total_Registers_In_Length, int32_t Total_Registers_Out_Length,
uint32_t Attenuator_Index, uint32_t Motor_Command, uint32_t Value)
```

Returns:

TRS_WATTEN returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Total_Registers_In[]` is a pointer to an array composed by 128 uint32 elements, used to store the temporary system configuration parameters. **Never modify values contained in the Total_Registers array.**
- `Total_Registers_Out[]` is a pointer to an array composed by 128 uint32 elements, **it MUST be the same pointer of Total_Registers_In[].**
- `Total_Registers_In_Length` is a int32 which value **must be equal to 128**.
- `Total_Registers_Out_Length` is a int32 which value **must be equal to 128**.
- `Attenuator_Index` is a uint32, used to address the optical attenuator (**see table 5**).
- `Motor_Command` is a uint32, which define the type of command to send for changing the attenuation value (**see table 6**).
- `Value` is a uint32, representing the new attenuation value, absolute position or position change of the optical attenuator (**see table 6**).

Table 5: optical attenuators index and laser channel wavelengths

Laser channel (wavelength)	Attenuator_Index
635	1
670	2
730	3
830	4
852	5
915	6
980	7
1040	8

Table 6: optical attenuator commands and valid value range

Motor_Command	Short description	Valid value range	Details
0	Set absolute position (steps)	0 to 9999	Sets the optical attenuator **ABSOLUTE position, in STEPS**. Zero is the home position.
1	Increment position (steps)	0 to 999	**Increases** the current attenuator position, in STEPS.
2	Decrement position (steps)	0 to 999	**Decreases** the current attenuator position, in STEPS.
3	Set absolute attenuation (db)	0 to 40	Sets the **ABSOLUTE optical attenuation, in DECIBEL**. Values are pre-calibrated.

Go to TOP

TRS_ACQ:

Checks if a new measurement has been completed and downloads the corresponding TCSPC histograms and monitoring counters data, for both the acquisition channels. **TRS_ACQ has to be executed in a polling process.** Each time a measurement is complete, the function returns 1 and corresponding data are available in the respective variables. The function also provides information on the memory bank used to store the downloaded data. If the automatic measurement mode is used, NIRS_ACQ returns information on the laser channel that was active during the measurement and on the index of the current measurement (in the automatic measurement map).

Prototype:

```
uint32_t TRS_ACQ(uint64_t *Handle_In, uint32_t Histogram_1[],
uint32_t Histogram_2[], int32_t Histogram_1_Length, int32_t
Histogram_2_Length, uint32_t Stats_1[], uint32_t Stats_2[], int32_t
Stats_1_Length, int32_t Stats_2_Length, uint32_t *Mem_Bank, uint32_t
*Laser_Ch, uint32_t *Meas_Index)
```

Returns:

TRS_ACQ returns a new-data flag, equal to 1 if new data have been downloaded successfully, otherwise it returns 0 (and values contained in Histogram_N[] and Stats_N[] are meaningless).

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Histogram_1[]` is a pointer to an array composed by 8192 uint32 elements, used to store the TCSPC histogram from the TDC Unit n° 1, acquired during the previous integration time interval. The bin size is equal to 10 ns/1024 = 9.76 ps.
- `Histogram_2[]` is the same, for the TDC Unit n° 2.
- `Histogram_1_Length` is a int32 which value **must be equal to 8192**.
- `Histogram_2_Length` is a int32 which value **must be equal to 8192**.
- `Stats_1[]` is a pointer to an array composed by 3 uint32 elements, used to store data from the monitoring counters (SYNC events, SiPM counts, TDC conversions) acquired during the previous integration time interval, for the detection channel n° 1.

- `Stats_2[]` is the same, for the detection channel n° 2.
- `Stats_1_Length` is a int32 which value **must be equal to 3**.
- `Stats_2_Length` is a int32 which value **must be equal to 3**.
- `*Mem_Bank` is a pointer to uint32, which value represent the memory bank used to store the downloaded data (can be 0 or 1).
- `*Laser_Ch` is a pointer to uint32, which value represent laser channel that was active during the downloaded measurement (**only if the automatic measurement mode is used**). Values range from 1 to 8, and 0 represents the STOP channel).
- `*Meas_Index` is a pointer to uint32, which value represents the index of the downloaded measurement, with respect to the automatic measurement map (**only if the automatic measurement mode is used**). Values range from 0 to 15.

Go to TOP

TRS_CTRS:

Reads data from the two independently-running fast counters, dedicated to constantly monitor the count-rate of SiPM modules. The integration time of these counters is set using the TRS_SET function (the default value is 100 ms). TRS_CTRS always reads the last valid measurement from the counters, so it **can be called at any time without needing a polling loop**. If the function is called faster than the counters integration time, the downloaded data remain the same until a new integration time interval begins (i.e. the **fast-counters run totally uncorrelated respect to their readout process**).

Prototype:

```
uint32_t TRS_CTRS(uint64_t *Handle_In, uint32_t Counts[], int32_t Counts_Length)
```

Returns:

TRS_CTRS returns a valid-data flag, equal to 1 if counts data have been downloaded successfully, otherwise it returns 0 (and values contained in Counts[] are meaningless).

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Counts[]` is a pointer to an array composed by 2 uint32 elements, used to store the counts of the SiPM detection modules 1 and 2, measured in the integration time interval defined using the function TRS_SET (default is 100 ms).
- `Counts_Length` is a int32 which value **must be equal to 2**.

Go to TOP

TRS_RATTEN:

Reads the status of the 8 electrically-actuated optical attenuators, used for the injection power equalization. The function queries one attenuator at a time, whose address is specified as function parameter. Status details are provided in Table 7. TRS_RATTEN can be repeatedly used after setting a new attenuation (or position value), to quickly know when the attenuator completes the command. This function, together with TRS_WATTEN and TRS_CTRS can be used to interactively and quickly equalize the 8-channels injection light power.

Prototype:

```
uint32_t TRS_RATTEN(uint64_t *Handle_In, uint32_t Attenuator_Index,
uint32_t *Attenuator_Status)
```

Returns:

TRS_RATTEN returns a status word, equal to 1 if executed successfully, otherwise it returns 0.

Parameters:

- `*Handle_In` is the pointer to the "Device Handle" (uint64), created by the function NIRS_INIT at the beginning of the program.
- `Attenuator_Index` is a uint32, used to address the optical attenuator (**see table 5**).
- `*Attenuator_Status` is a pointer to uint32, which value represents the current status of the optical attenuator (**see table 7**).

Table 7: optical attenuators status details

Attenuator_Status	Description
0	IDLE (ready to process commands)
1	DONE (command completed, ready to process next command)
2	MOTOR IS RUNNING (busy)
3	ERROR (command not executed or connection problem)

Go to TOP