NIRS Box DLL Guide v2 - 28/11/17

- **NIRS_OPEN: opens the communication with the NIRS Box.** Must be executed first. Requires as parameter a uint32 pointer to provide the Device Handle, a pointer to an array composed by 128 uint32 and the length of this array (number of cells) in int32 format (provided as value). The NIRS_OPEN function returns a "status bit", equal to 1 if communication opening was OK, otherwise equal to 0. NOTE: Registers_Out_Length = 128.

  PROTOTYPE

  ```
  uint32 NIRS_OPEN(uint32 *Handle, uint32 Registers_Out[], int32
  Registers_Out_Length);
  ```

- **NIRS_CLOSE: closes the communication with the NIRS Box.** Must be executed as the last function. Requires as parameter a pointer to uint32 for the Device Handle. Returns a "status bit", equal to 1 if OK, otherwise equal to 0.

  PROTOTYPE

  ```
  uint32 NIRS_CLOSE(uint32 *Handle);
  ```

- **NIRS_ON: turns-ON or OFF the overall system (power supplies, SiPM module, thermal controls and SYNC generator).** This function must be executed both after the NIRS_OPEN and before the NIRS_CLOSE. When turning ON, the clock starts at 40 MHz and the default measurement integration time is 1 second. Requires as parameters: a pointer to uint32 for the Device Handle, twice the pointer to the same 128 uint32 array used for the NIRS_OPEN, twice the length of this array in int32 format (provided as value) and the ON_nOFF flag, as a uint32 value (1 -> the system is turned ON, 0 -> the system is turned OFF). The function returns 1 if turning ON/OFF was OK, otherwise returns 0. NOTE: Registers_In[] e Registers_Out[] are the same pointer.

  PROTOTYPE

  ```
  uint32_t NIRS_ON(uint32_t *Handle, uint32_t Registers_In[],
  uint32_t Registers_Out[], int32_t Registers_In_Length, int32_t
  Registers_Out_Length, uint32_t ON_nOFF);
  ```

- **NIRS_LASER: turns-ON or OFF the two laser sources.** This function must be executed after the NIRS_ON (and before calling the same function to turn OFF the system). Requires as parameters: a pointer to uint32 for the Device Handle, twice the pointer to the same 128 uint32 array used for the NIRS_OPEN, twice the length of this array in int32 format (provided as value) and the ON_nOFF flag, as a unint32 value (1 -> lasers are turned ON, 0 -> lasers are turned OFF). The function returns 1 if turning ON/OFF was OK, otherwise returns 0.

  PROTOTYPE

  ```
  uint32_t NIRS_LASER(uint32_t *Handle, uint32_t Registers_In[],
  uint32_t Registers_Out[], int32_t Registers_In_Length, int32_t
  Registers_Out_Length, uint32_t ON_nOFF);
  ```

- **NIRS_SET: Allows to change the measurement parameters (SYNC frequency, integration time, active wavelength).** Can be executed at any time after the NIRS_ON, but when a measurement is NOT running. Requires as parameters: a pointer to uint32 for the Device Handle, twice the pointer to the same 128 uint32 array, twice its length (in int32 format), the desired SYNC frequency (expressed in MHz) in uint32 format, the integration time (expressed in milliseconds) in uint32 format and the active wavelength in uint32 format (1 -> Laser1 is always active, 2 -> Laser2 is always active, 3 -> automatic toggling between Laser1 and Laser2 during the measurement). The function returns a "status bit", equal to 1 if OK, otherwise equal to 0.

  PROTOTYPE

  ```
  uint32 NIRS_SET(uint32 *Handle, uint32 Registers_In[], uint32
  Registers_Out[], int32 Registers_In_Length, int32
  Registers_Out_Length, uint32 Frequency, uint32 Time, uint32
  Wavelength);
  ```

- **NIRS_MEASURE: Turns ON or OFF the measurement (TDC and histograms collection).** It can be used to start and stop the data collection (and the laser toggling, if the automatic wavelength selection is used). When the measure is turned ON, it is possible to start polling the system (using the NIRS_ACQ) for valid data. The measurement always starts with Bank 1. Requires as parameters: a pointer to uint32 for the Device Handle, twice the pointer to the same 128 uint32 array used for the NIRS_OPEN, twice the length of this array in int32 format (provided as value) and the ON_nOFF flag, as a uint32 value (1 -> measurement is turned ON, 0 -> measurement is turned OFF). The function returns 1 if turning ON/OFF was OK, otherwise returns 0.

  PROTOTYPE

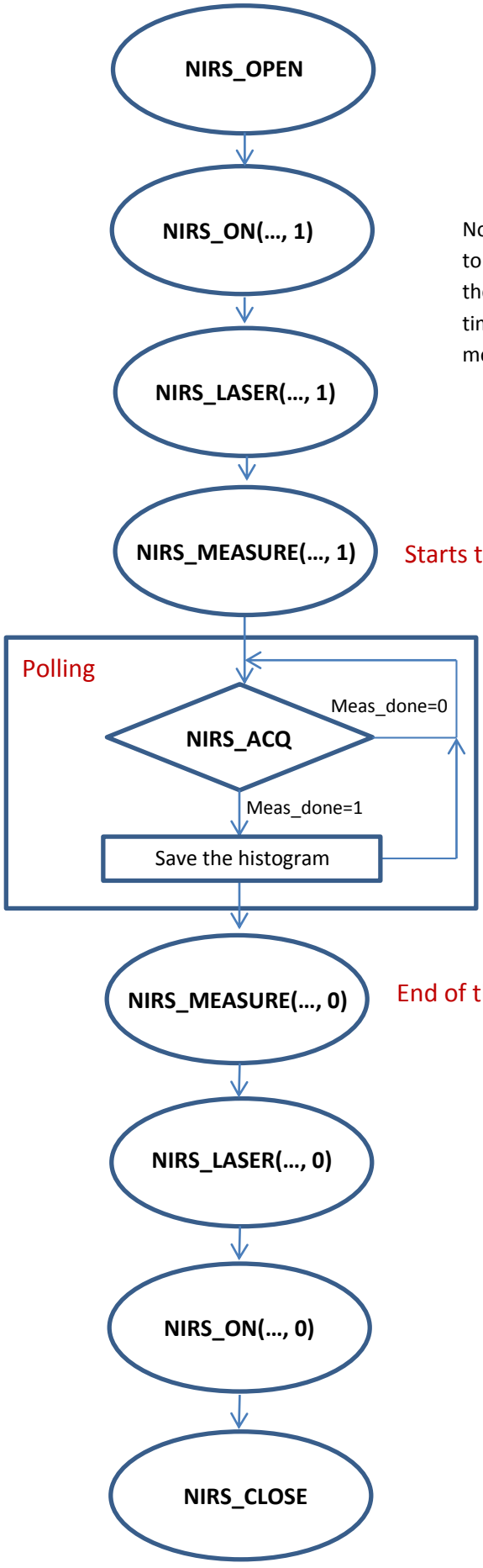
  ```
  uint32_t NIRS_MEASURE(uint32_t *Handle, uint32_t Registers_In[],
  uint32_t Registers_Out[], int32_t Registers_In_Length, int32_t
  Registers_Out_Length, uint32_t ON_nOFF)
  ```

- **NIRS_ACQ: Allows the histogram readout.** Must 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. It is then possible to read the photon arrival times histogram and data from the monitoring counters (SYNC events, SiPM counts, TDC conversions) acquired during the previous integration time interval. Requires as parameters: a pointer to uint32 for the Device Handle, a pointer to an array composed by 8192 uint32 for the histogram, a pointer to an array composed by 3 uint32 for the counters, the lengths of these arrays (8192 and 3 respectively) in int32 format and a pointer to a uint32 for the memory bank readout. The memory bank value corresponds to the laser used during the downloaded measurement (if the system is in the automatic wavelength toggling mode) and can be "1" or "2". The function returns a "status bit" of the measurement, equal to 1 when new data is acquired, otherwise equal to 0. NOTE: Histogram_length = 8192 and Stats_Length=3.

  PROTOTYPE

  ```
  uint32 NIRS_ACQ(uint32 *Handle, uint32 Histogram[], uint32
  Stats[], int32 Histogram_Length, int32 Stats_Length, uint32
  *Bank);
  ```

Application example:



Note:
to change default measurement parameters, the function NIRS_SET can be called at any time after the NIRS_ON (but when the measurement is OFF).