

Feasa ICT SO Library reference guide

1. Introduction

Feasa ICT SO (Shared Library) is a library developed by Feasa Enterprises Ltd and focused in ARM-based embedded systems. This library is intended to ease to the customer the use of the Synchronous Bus of the Feasa ICT LED Analyser through the General Purpose Input Output lines (GPIO) of Linux-based ARM embedded computers. This is not a high-speed library, since its main target is provide compatibility between different ARM platforms, while having high speed routines would mean that the library should be specific to each processor.

2. Previous considerations

To be able to use this Library, you should ensure that the used programming language is able to work with SO libraries.

Remember that the SOs should be generally copied to the `/usr/lib/` folder. Some compilers also accept or need the Library to be placed in the same path of the binary file.

This is a 32bit library and all programming examples included in the CD are targeted to 32bit platforms.

The functions of this Library follow the CDECL calling convention. This should not be an issue since any modern programming language allow to select the calling convention when importing any Library function; others have specific importing functions depending on the calling convention.

Depending on the programming language used, Integer variables does not match in size with the Integer described in the SO Library (32 bit). Please, check your language's reference manual in order to be sure that the proper data types are being used.

This SO Library has been written for the Linux OS and tested on ARM Debian versions, using as hardware platforms the Raspberry PI v1, v2 and v3, BeagleBone Black revC and UDOO neo, so Feasa cannot ensure that this Library will work on different conditions than those described.

The LED Analyser ICT port uses a voltage level of 3.3v, so please, ensure that the ARM computer where you want to connect it uses the same voltage levels, otherwise either the LED Analyser or computer can get damaged. Please, check the ICT manual to get more information about the ICT port / connector.

3. Installation instructions

Before use the SO Library you should previously copy it to the system libraries folder, which can be generally found on the `/usr/lib` or equivalent folder.

Feasa ICT SO · reference guide

by R&D Engineer Carles Martínez Rius, rev 1, © Feasa Enterprises Ltd.



Some programming languages allow to use the SO if you previously copy it to the same folder of the binary file or programming project, where you are calling it from, so in this situations there is no need to copy the SO Library to the system folder.

Sometimes it's required to have certain read permissions for the file, so you should change it according to your system or account policies.

If you are planning to redistribute your application or you are installing it on other computers, you should repeat the installation process in all the target computers.

4. Functions

4.1. Assign signal/line to pin

Before connecting the ICT synchronous bus lines of the LED Analyser ICT port to the ARM host computer pins, we need to define what pins from this computer will be used for each one of the lines of the ICT. For example, CLK line will be connected to pin GPIO16, etc.

```
int FeasaICT_AssignSignal(char * SignalName, int gpio_number);
```

Parameters:

- **SignalName:** this is the name of the line to assign. Available names are: LA_SELECT, CLK, OUT (or LA_IN), IN (or LA_OUT), LA_BUSY (or RY_BY)
- **gpio_number:** is the name of the GPIO pin that will be used to drive the signal specified.
Warning: you should use the GPIO number here and not the PCB pin number or label; be careful with this because you can damage the board. Some lines of the computer can also be shared for other purposes, so the whole setup may not work properly if those pins are not chosen or pre-configured properly. Please check the documentation from the manufacturer.

This function returns 1 if everything gone fine, or 0 on error.

Note: there is a special version of this function for the BeagleBone Black. See below.

4.2 Assign signal/line to pin (BeagleBon Black)

Before connecting the ICT synchronous bus lines of the LED Analyser ICT port to the ARM host computer pins, we need to define what pins from this computer will be used for each one of the lines of the ICT. For example, CLK line will be connected to pin GPIO16, etc.

In the BeagleBone Black, IO lines have names like “gpio1[23]”, where 1 would be the group/block and 23 would be the IO number.

```
int FeasaICT_AssignSignal_BeagleBone(char * SignalName, int group, int number);
```

Parameters:

- **SignalName:** this is the name of the line to assign. Available names are: LA_SELECT, CLK, OUT (or LA_IN), IN (or LA_OUT), LA_BUSY (or RY_BY)
- **group:** number of the group to which the pin/line belongs.
- **gpio_number:** is the name of the GPIO pin that will be used to drive the signal specified.
Warning: you should use the GPIO number here and not the PCB pin number or label; be careful with this because you can damage the board. Some lines of the computer can also be shared for other purposes, so the whole setup may not work properly if those pins are not chosen or pre-configured properly. Please check the documentation from the manufacturer.

This function returns 1 if everything gone fine, or 0 on error.

4.3 Configure

Once all lines have been assigned to the pins, then all the hardware settings need to be saved to start operating normally.

```
int FeasaICT_Configure(void);
```

This function does not accept parameters.

This function returns 1 if the port has been successfully configured, 0 on error.

4.4 Response Timeout

Communicating with the LED Analyser using the synchronous bus could sometimes be problematic due to a bad wiring, a wrong choose of the computer GPIO pins or a de-synchronization generated by any external electrical noise (EMI, glitches, etc.). In order to avoid any issues, a communications Timeout has been provided and it can be modified through the following command:

```
void FeasaICT_SetResponseTimeout(uint32_t Timeout);
```

Parameters:

- Timeout: numeric value given in milliseconds.

This function does not return any value.

If this value is not modified, a default time of 10 seconds will be used.

Note: It is very important to modify this time if long captures are performed.

4.5 Initialize bus

Before establishing any communication with the LED Analyser, first we need to initialize the synchronous bus. This will remove any data from the buffers and ensure that the whole system is ready to transmit and receive commands.

```
int FeasaICT_InitializeBus(void);
```

This function does not accept parameters.

This function returns 1 if the has been successfully configured, 0 on error.

4.6 Capture

The following command is used to trigger a capture through the Synchronous Bus on the ICT port:

```
int FeasaICT_Capture(uint8_t CaptureRange, uint8_t isPWM, uint8_t  
AveragingFactor);
```

Parameters:

- CaptureRange: numerical value to indicate the capture range that will be used. Available ranges: 0=Auto, 1=Low, 2=Medim, 3=High, 4=Super, 5=Ultra.
- isPWM: use 0 for non-PWM captures or 1 when the target LED is modulated in PWM.

- **AveragingFactor**: (for PWM captures only) numerical value that indicates the number of samples that will be taken when performing a PWM capture.

Note: you may want to adjust the Timeout value for long capture times (high exposure values, PWM low intensity measurements, etc).

This function returns 1 if the capture has been successfully done, 0 if any Timeout occurred, or -1 for other errors.

4.7 Download measured data

There are several commands that can be used to download the data obtained as a result of a capture command:

```
FeasaICT_ReadRGBI(uint8_t fiber, uint8_t * Red, uint8_t * Green, uint8_t
                  * Blue, uint32_t * Intensity);

int FeasaICT_ReadHSI(uint8_t fiber, float * Hue, uint8_t * Saturation,
                    uint32_t * Intensity);

int FeasaICT_ReadxyI(uint8_t fiber, float * x, float * y, uint32_t *
                    Intensity);

int FeasaICT_ReadCCT(uint8_t fiber, uint32_t * CCT);

int FeasaICT_ReadDominantWavelength(uint8_t fiber, uint32_t *
                                    Wavelength);

int FeasaICT_ReadWSI(uint8_t fiber, uint32_t * Wavelength, uint8_t *
                    Saturation, uint32_t * Intensity);

int FeasaICT_ReadAbsoluteIntensity(uint8_t fiber, float * AbsInt);
```

Parameters:

All commands have the same structure. The first parameter is the number of the fiber / channel to be measured and the values to be measured are returned through the other parameters, this is why pointers are used.

These function return an integer value of 1 if the values are read successfully, or 0 in case of error or timeout.

4.8 Exposure factor

The exposure factor is used to adjust the sensitivity of the LED Analyser (you can check the LED Analyser manual for more information):

```
int FeasaICT_SetFactor(uint8_t factor);
```

Parameters:

- **factor**: numerical value of the exposure factor to use (typically from 1 to 15).

This function returns an integer value of 1 if no errors are detected, or 0 otherwise.

Note: you may want to adjust the Timeout value for long capture times (high exposure values, PWM low intensity measurements, etc).

4.9 Sequence capture

As explained in the LED Analyser ICT manual, it is possible to perform capture of light sequences or patterns at certain intervals, either for measuring color and intensity data of light patterns or for the measurement of the frequency and duty cycle.

a) First of all, the sequence has to be configured using the following function:

```
int FeasaICT_CaptureSequence_Setup(uint16_t CaptureTime, uint16_t
                                   WaitTime, uint16_t nCaptures);
```

Parameters:

- CaptureTime: numeric value of the desired capture time in milliseconds (time for each capture interval).
- WaitTime: numeric value of the desired interval between captures, in milliseconds.
- Ncaptures: number of captures to perform (Capture + Wait).

It returns an integer value of 1 if the parameters are send successfully, otherwise it returns 0.

b) Optionally, an Intensity threshold can be defined:

```
int FeasaICT_CaptureSequence_SetIntensityThreshold(uint32_t Intensity);
```

Parameters:

- Intensity: numeric value for the intensity threshold.

It returns an integer value of 1 if the parameters are send successfully, otherwise it returns 0.

c) Then, a capture of the sequence is performed:

```
int FeasaICT_CaptureSequence(void);
```

This function does not accept parameters.

It returns an integer value of 1 if the parameters are send successfully, otherwise it returns 0 if a timeout event occurs, or -1 for other types of error.

d) Measured data can be accessed through different commands, depending on the desired parameters: all the Intensity values for every capture/sequence, only the peak values or the pulse times (in case of single pulse measurements).



It is possible to download the measured data for each frame/capture and each specific fiber/channel. It can be done using the following command:

```
int FeasaICT_CaptureSequence_ReadIntensity(uint8_t fiber, uint16_t
Sequence, uint32_t * Intensity);
```

Parameters:

- **fiber**: numerical value of the fiber to read.
- **Sequence**: integer value of the sequence to be accessed.
- **Intensity**: pointer to integer that will be used to store the intensity value read for the given fiber and sequence.

This function returns an integer value of 1 if no errors are detected, or 0 otherwise.

To download only the highest Intensity values for each channel, the following command can be used:

```
int FeasaICT_CaptureSequence_ReadPeakValues(int num_fibers, uint32_t *
PeakValues);
```

Parameters:

- **fibers**: integer with the number of fibers of the LED Analyser.
- **PeakValues**: pointer to integer array that will be used to store the intensity values read for each of the fibers. This variable should be declared before use this function, ensuring that enough memory is reserved to store the Intensity of all the fibers.

This function returns an integer value of 1 if no errors are detected, or 0 otherwise.

If a single pulse of the LED is measured, then, the following command can be used to find out the On and Off times:

```
int FeasaICT_CaptureSequence_ReadPulseTimes(uint8_t fiber, uint16_t *
Toff, uint16_t * Ton, uint16_t * Toff2);
```

Parameters:

- **fiber**: integer with the fiber from which the pulse times will be downloaded.
- **Toff**: pointer to integer that will be used to store the pre-on time in milliseconds of the given LED/fiber (time from the beginning of the capture to the ON-status).
- **Ton**: pointer to integer that will be used to store the on time in milliseconds of the given LED/fiber.
- **Toff2**: pointer to integer that will be used to store the post-on time in milliseconds of the given LED/fiber (time from the end of the ON-status to the end of the capture).

This function returns an integer value of 1 if no errors are detected, or 0 otherwise.