ANXXXX

STLUX385A – STEVAL-ILL066V1 – FIRMWARE

Firmware Application Manual

1 Introduction

The STEVAL-ILL066V1 LED driver board is a complete and configurable solution that manages a single high brightness LED string using the STLUX385A digital controller to drive two power conversion stages as well as handling the DALI protocol and the 0-10V bus.

The topology implemented is a PFC + LC with primary side regulation. This topology doesn't require a secondary side feedback and hence doesn't use any opto-couplers, increasing the LED driver lifetime.

This manual introduces the use of the STLUX385A firmware which manages the board's hardware. All the features regarding the functionality of the demo bard are managed by the firmware described here. The firmware is taking full advantage of the SMEDs to guarantee real time MOSFETs protection.

This manual is based on the FW revision **V3R35**.

The firmware used for this board uses two libraries to manage the power conversion stages.

- The first library manages the PFC hardware, regulate the PFC output voltage changing the PFC ON time or the PFC OFF time, make some protection and monitor the PFC activity.
- The second library manages the Half Bridge hardware, regulate the output current using the ZVS principle, and monitor the output load.

For complete information on the demo board please refer to the Application Note AN4461.

For more information about STLUX385A and the complete STLUX product family, please refer to the STLUX Datasheet and STLUX Reference Manual.

Contents

List of tables

List of figures

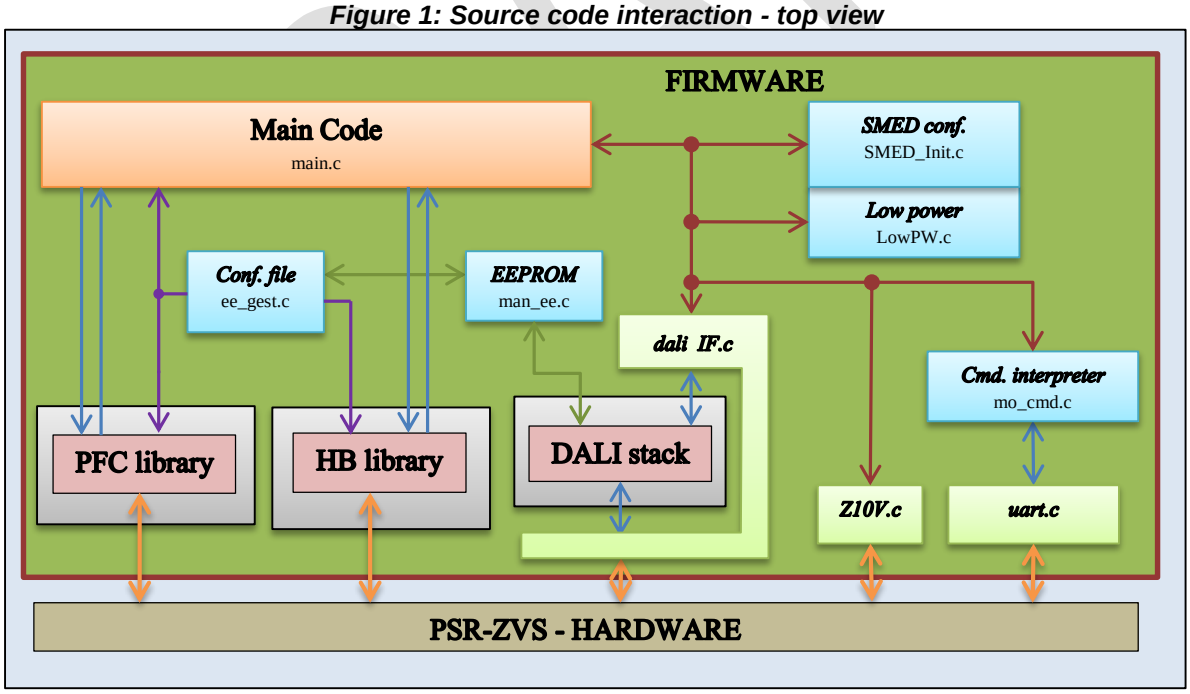
2 Source code description – top view

The ILL066V1 firmware codes manage the two stage digital converter ad some interfaces. The two stages drive the PFC MOSFET and the Half Bridge (HB) MOSFETs using the SMED. To manage the PFC and the ZVS hardware two libraries are provided. The main code manages the firmware startup phases, call others enabled modules like DALI or 0-10V interface and survey the board. The list of all available source file (module) is:

- main.c The main source code.
- ee_gest.c The EEPROM area handler (DALI stack manage directly the EEPROM).
- man_ee.c The EEPROM driver, write and read the EEPROM area using polling.
- LowPW.c The low power routines when DALI stack witch off the output current.
- mo_cmd.c The serial command interpreter.
- SMED_Init.c The SMED configurator output file used to initialize the SMED behaviour.
- stlux_itc.c Subroutines to handle the interruption (not the interrupt handlers).
- stlux_optonbyte.c Restore the default STM8 option byte (used only for Raisonance compiler).
- dali_IF.c The DALI driver and the interfaces from DALI stack and main code.
- Dali_Stack\ The DALI stack directory.
- uart.c The serial driver.
- Z10V.c The 0-10V interface driver.
- HB_LIB.LIB The Half Bridge library for the ZVS stage.
- PFC_LIB.LIB The PFC library for the PFC stage.

The interaction from all modules are schematized on the *Figure 1*

Figure 1: Source code interaction - top view



2.1 main.c

On this file there is the entry for the “c” program after the “c” initialization. The entire main task is executed into this code, all library and subroutines is called by this code.

The “main.c” file initializes the input and output signal, call the hardware initialization, call the “ee_gest.c” module and call the necessary routine to initialize the firmware routines. During the initialization phases the

ADC is configured to continuously convert the entire eight input channel. When, during the normal execution, one ADC channel value is necessary simply read the ADC output register and get the last ADC conversion. Using the 6MHz ADC clock, one channel is converted every 3uS and a new conversion on the same channel is available every 24uS.

On the main.c file there is referenced the STMR interrupt routine. The STMR interrupt routine is the based timer to check and control all the hardware. The STMR interrupt routine call also the PFC and HB algorithms. On this interrupt some flags are used to inform other routines when 100uS or 10mS time is elapsed. On this STMR interrupt routine a 32 bit counter is updated every second.

The main.c file calls the ee_gest.c module and restores the ram area from all values memorized on the EEPROM data area. Some variables define also which interfaces are active and if the PFC and HB library are enabled. The main.c file also controls which type of STLUX is used. If the STLUX type is out of a predefined value (STLUX385A), the execution is disabled and forces the “debug mode”. On debug mode only the serial line is active and all others ILL066V1 board functions are disabled. When a new code is loaded into the STLUX, the debug mode is automatically enabled. To remove the debug mode the user needs to use “in 7 0” serial command.

The main.c checks also the ILL066V1 internal power. If the internal power voltage is out of a predefined level, one “warning” message shown on the serial line but the execution continues. On the ILL066V1 demo board this voltage is the “VD_14V” (14V) and “VCC” (3.3V) level. During the start-up phases the IWDG peripherals is enable to reset the STLUX if an infinite loop is executed. The predefined IWDG reset time is 10mS.

When all the initial setup and test are executed and validated, the main module enters into the main loop. During the infinite loop the STLUX code checks and calls some routines of PFC and of HB library. Into this main loop, the main module checks if new data is available from external interfaces and, when new data is available, calls the proper subroutines to do the necessary action. When the main loop arrives at the end without pending action, the STM8 core enters into “wait for interrupt” instruction to reduce its power.

If the user wants to check the STLUX core load simple check, using an oscilloscope, the ON time of the GREEN LED. When the GREEN LED is on, the STLUX core is active and execute code, otherwise, when the GREEN LED is off, the STLUX CPU is into “*wfi*” instruction.

2.2 ee_gest.c

On this file there is all the necessary routine to manage the EEPROM file system used to modify the ILL066V1 demo board behavior. Inside this file there is a structure to identify all the modifiable parameters.

The main call the “*ck_ee()*” routine to check if the EEPROM data area is initialized, if is not initialized the default value is loaded before continuing. After this first check the “*ck_ee()*” routine initialize all the ram variable using the value stored into the EEPORM data area. After this phase, the routine checks the parameter #10 (ROP) value to initialize, if necessary, the ROP register. When the ROP register is set TRUE the parameter file is read only and no modification is possible.

When the user modify one parameter value, the “*ee_set()*” routine is called to modify the EEPROM content. The new value is applied only during the startup and not on the fly.

To show all the available parameters and the actual value the “*ee_show_help()”* routine is available. This routine is called when the user type the “in 0 0” serial command.

This module uses the man_ee.c module to physically write and read the EEPROM.

2.3 man_ee.c

This file provides the low level routine to manage the EEPROM contents. Provide the write and the read routines used by the DALI stack and by the ee_gest.c files. It is possible to write and read a byte (u8) and a word

(u16). The EEPROM read or EEPROM write operation wait the previous EEPROM write end's before really read/write the data contents. This allow the interrupt routine to start otherwise, if the second write operation start before end the previous, the CPU is stalled (stop the CPU clock).

2.4 LowPW.c

This file provides the low power module used to entering into low power mode and to exiting from low power mode. It's called by the main.c file when the DALI stack set the LED off.

The low power is obtained switching OFF all the not necessary peripherals, call the "*wfi*" instruction and modify the internal watchdog time to 500mS. When DALI RX pin receive a high to low transition, the CPU is active to the maximum clock (16MHz) and check the related DALI RX frame. If the RX frame active the output power, the low power module exit and restore all the STLUX functionality, otherwise, after a two second timeout, return back into low power mode. This functionality permit a very low power consumption when the output current is switch OFF.

2.5 mo_cmd.c

This file provides the entire command line interpreter. Analyze all the character received by the serial line and, when the command is finish and validated, execute the corresponding subroutines. This module working in connection to the uart.c file which provide the UART driver.

2.6 SMED_Init.c

This file is the output of the SMD configurator program. It is called by the main.c to initialize the SMED registers during the start-up phases. This file does not start the SMED functionality but only load a "predefined" configuration on the SMED register.

2.7 stlux_itc.c

Provide the "*ITC_SetSoftwarePriority()*" function to change the default interrupt priority level. Provide also the TRAP and NMI interrupt entry routines. The TRAP and the NMI interrupt on the ILL066V1 demo board are not used.

2.8 stlux_optonbyte.c

This file provides the correct Option Byte setup.

Only the Raisonance environment uses this file. This fie provide the default option byte value. This file is not referenced by the main but the option byte is writing during the code downloaded using the RLINK-SWIM interface.

The IAR environment, when this document is writing, do not initialize the Option Byte structures.

2.9 dali_IF.c

This module is the interfaces from main to DALI stack.

This module manage the DALI peripherals interrupt and inform the DALI stack when a good frame is receive. Manage also the DALI byte transmission using the same interrupt vector.

The "*Dali_Init()"* function manage the startup of the DALI interfaces, set the baud-rate, set the 500ms timer to detect the disconnection of DALI bus, initialize some DALI stack structures/variable.

The "*DALI_Enalbe()*" function enable the DALI communication and the DALI interrupt.

The "*DALI_CheckAndExecuteReceivedCommand()*" function is called by the main when a new frame is received by the DALI bus and call the DALI stack to take the necessary action.

The "*Get_DALI_Random()*" function is called by the DALI stack to get an u8 random value.

The "*Send_DALI_Frame()*" function is called by the DALI stack to schedule the transmission of the u8 byte. The physical transmission start after 3mS.

The "*set_dimm()*" function manage the output current request from DALI stack. This subroutine activate/deactivate the low power module when necessary.

2.10 Dali_Stack\

The entire DALI stack is putted into a separated directory. The DALI stack require only the CCO interrupt functionality for the internal timing (1mS) and the DALI tx-rx interrupt to receive and transmit the DALI frame. The DALI interfaces is mutually exclusive with 0-10V interface. The DALI stack is compliant with the IEC 62386-102 rev1.0 and to IEC 62386-207 rev1.0 request.

2.11 uart.c

This module take all the necessary action to manage the serial line. The default baud rate is 115200 bps but, modify the source code, any others baud rate is available. This module implement an input/output drive using the UART interrupt to reading or writing every bytes on the serial line. The buffer length is modifying changing the TXSIZE (default is 160 byte) and RXSIZE (default is 16 bytes) constant used to send out or receive in the byte. On this module is also implemented the interfaces to the standard "*printf*" subroutine.

NOTE: User never insert a "*printf*" function into the interrupt routine code, this cause a program fault.

2.12 Z10V.c

This module manage the 0-10V interfaces using the voltage on ADCIN[3] input pin and CCO output pin. The CCO output pin create the frequency used for this interface. The 0-10V interfaces is mutually exclusive with DALI interfaces, only one interfaces working at the same time. When 0-10V interfaces is used also the serial command take some limitation. This limitation regarding only the output current. The UART interfaces is not able to change the output current using the "ll" command because the 0-10V interfaces take precedence and overwrite the "ll" command level.

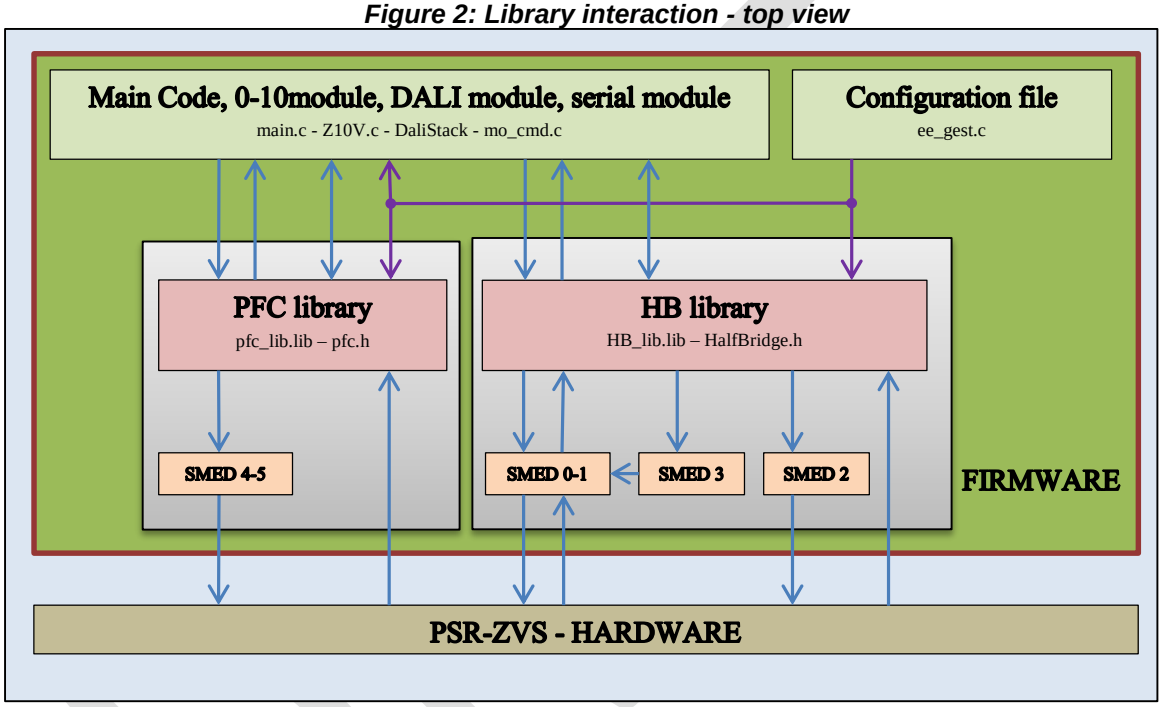
3 PFC and HB library description – top view

There is two library present on the firmware of ILL066V1 demo board. The first library module is to handle the PFC HW circuit; the second library module is to handle the Half Bridge HW circuit. All libraries is referenced using a include file. The two include file export the all variable and all subroutines called from other sources code. Into this libraries are also present some interrupt vectors.

All the modifiable parameters used into these two libraries are loaded during start-up before enable the PFC and HB function. To modify the start-up parameters use the "*in*" or "*ip*" serial command.

Next is showing the interaction from the two libraries (described into this document) and the other part of source code (main, command interfaces, configuration files - not described here).

Figure 2: Library interaction - top view



The entire two libraries receive all parameter from the "configuration file" during the power-on (after the reset). The value of this "configuration file" is stored into EEPROM area. The "configuration file" defines the parameters that are related to the SMED activity, the external hardware or modify some functionality of the library. The "configuration file" controls also the "board" functionality. This "configuration file" is modifiable using serial command ("*in*" and "*ip*" command) but the new data is applied only during the next power-on.

Some others parameters are modifiable on the fly by dedicated serial command and are "real time" (applied immediately after modification). That direct commands do not change the startup parameters. To understand all serial commands please refer to the AN4461 and the "SW release note" associated to this software version.

Every value described into this manual are define as the typical value. Some consideration is necessary to identify, case by case, the variation from temperature, production variation and STLUX tolerance. Please consider this entire element when changed the variable parameters.

The "configuration file" is specifying on *chapter 5*.

3.1 PFC algorithm detail

To the aim of finding a compensation network able to achieve the PFC control goal, it is necessary to get an insight into the control loop of such systems. This can be synthesized as shown in the block diagram of *Figure 3*. The $G3_{(S)}$, $G4_{(S)}$ and H is implemented into hardware, Vref, and the "PI" algorithm is implemented on the STLUX firmware. The STLUX create the necessary ON-OFF time and the SMED event taking in account the ZCD event (not designed on the *Figure 3*).

Figure 3: Control loop of PFC stage: block diagram

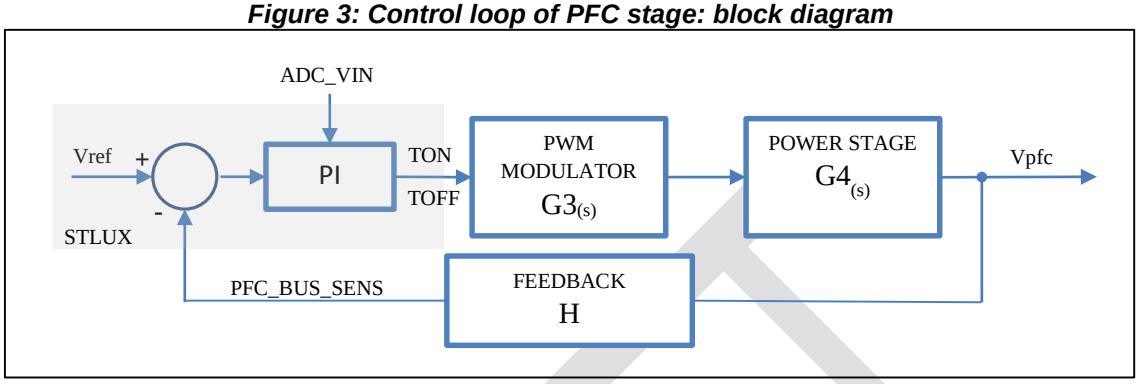
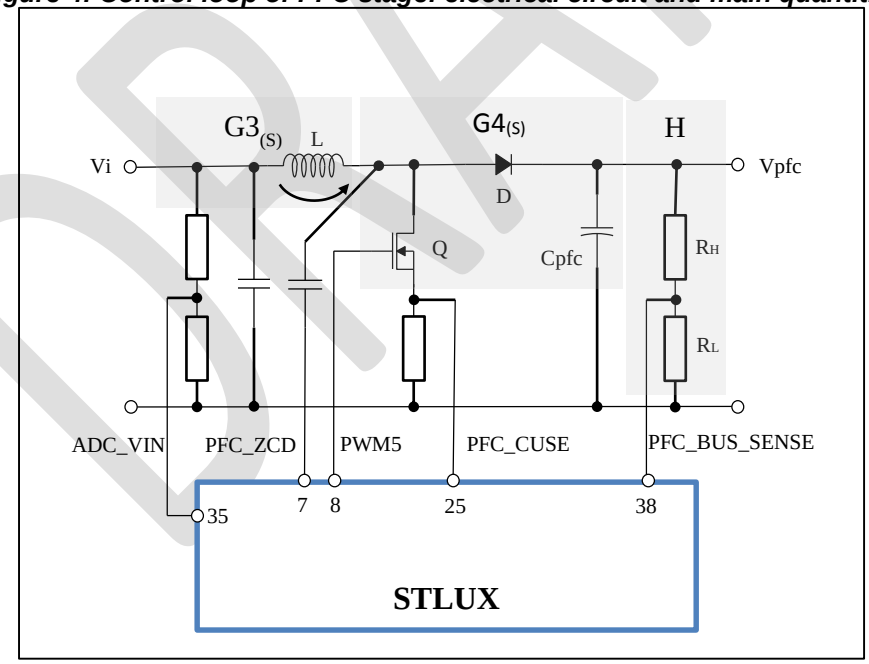


Figure 4 illustrates how the various blocks of *Figure 3* relate with the electrical circuit, both external and inside the STLUX. The loop gain of PFC must have a very low crossover frequency (fc) so as to maintain PWM5 fairly constant over a given line cycle and ensure a high PF. Inside the SMED the PFC current peak is detected only to protect the PFC stage from a short circuit. The SMED manage directly also the ZCD event to start the ON time when the inductor current are into valley.

Figure 4: Control loop of PFC stage: electrical circuit and main quantities



The User have the possibility to select itself a good hardware component based on the own experience exactly as an analog chip. The value of the "P" and of "I" are modify/adapt by the User using the four parameters defined into the parameters file.

The STLUX implement a standard "PI" (Proportional-Integral) algorithm but have also a gain scaler used to make a fast response during PFC load change. The parameters #46 and #47 define the PFC outside limit when

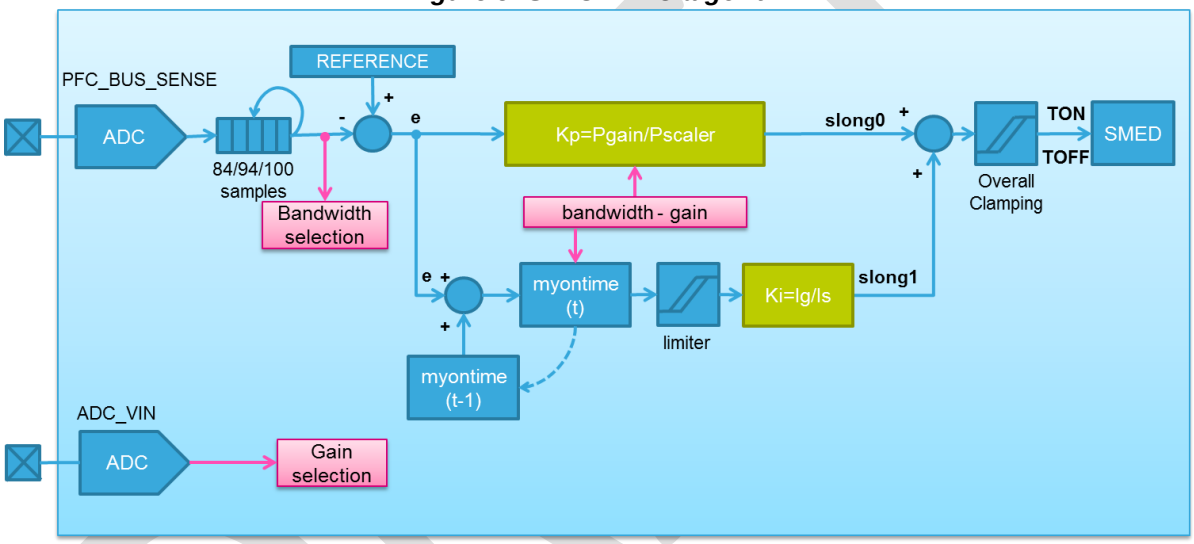
the fast response is active. The #48 define the correction slope when a PFC level is below of #47 and fast response is active.

The *Figure 5* show the PFC algorithm implement on the ILL066V1 demo board. The user adapt the parameters #42, #43, #44 and #45 to the own hardware. See the *chap. 5* to understand the parameters behavior.

The "*PI*" loop sensing the PFC output voltage using the "*PFC_BUS_SENSING*" ADC input voltage. The ADC samplings are accumulate to an internal 32bit variable and every half AC cycles is apply using the Kp and Ki parameters. The result of this "PI" algorithm define the PFC TON-TOFF time used during the successive half AC cycles. The accumulation phases is performed using a sampling rate equal to 100uS. Using this timing rate the samples is 84 at 60Hz and 100 at 50Hz. When the board start, because the input frequency is unknown, a fixed time is used, this is imposed to 55Hz (93 samples). The PFC output voltage target point used into this "*PI*" algorithm is fixed and is 1.1143V at the ADC input pin (PFC_BUS_SENSE - STLUX pin 38).

The output of the "PI" algorithm are clamping to a maximum PFC_Ton (1800 SMED.S3 time) and to a maximum PFC_Toff (32000 SMED.S2 time).

Figure 5: STLUX PFC algorithm



The bandwidth selection is a function to speed-up the PFC reaction during transaction (load change, increases or decreases). When the value of the accumulation is over or below to a predefined value (approx. +3.35% or -6% respect the target PFC - see parameters #46 and #47), the value of Kp gain is increased or decreased by 25%.

On the "PI" algorithm there is a Gain selector applied on the Kp gain. This value depends on the input AC value acquired on the "ADC_VIN" (STLUX pin 35). The gain apply on Kp is showed on the *Table 1*.

Input voltage (STLUX pin 35) (ADC_VIN)	***Approximated Vac input on ILL066V1 demo board***	***Gain applied on Kp***
More than 0.899V	More than 233Vac	1/4
From 0.899V to 0.727V	From 233Vac to 189Vac	2/4
From 0.727V to 0.555V	From 189Vac to 144Vac	3/4
Below 0.555V	Below 144Vac	4/4

Table 1: Kp Gain function of Vac input voltage

The “standard” PI equation on the literature is:

$$u(t) = Kp\, e(t) + Ki \int_{t_0}^{t} e(\tau) d\tau$$

Moreover, the simplified version is:

$$R_{PI}(s) = K_P \frac{1+T_I S}{T_I S} \qquad \text{where} \qquad T_I = \frac{K_P}{K_I}$$

By default the ILL066V1 demo board with the V3R35 FW revision implement the parameters defined below when the PFC output voltage are within the #46,#47 limit.

$$Kp = \frac{\#44}{2^{\#45}} * Kp\_g = \frac{100}{2^{12}} * Kp\_g = 0.024414 * Kp\_g$$

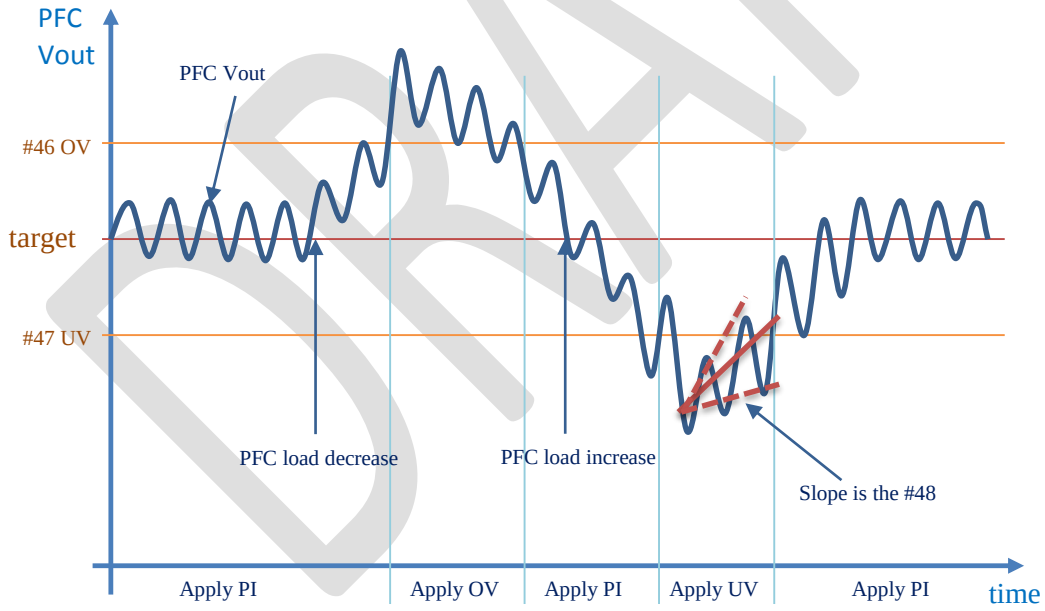
$$Ki = \frac{\#42}{2^{\#43}} = \frac{160}{2^{15}} = 0.0048828$$

Where:

- “#42, #43, #44, #45” is the value present on the parameters file described on the *chap. 5*
- “*Kp_g”* is a function of input Vac, see *Table 1*.

The figure below shows the relation for the PFC PI implementation.

Figure 6: STLUX PFC PI parameters relationship



3.2 HB algorithm detail

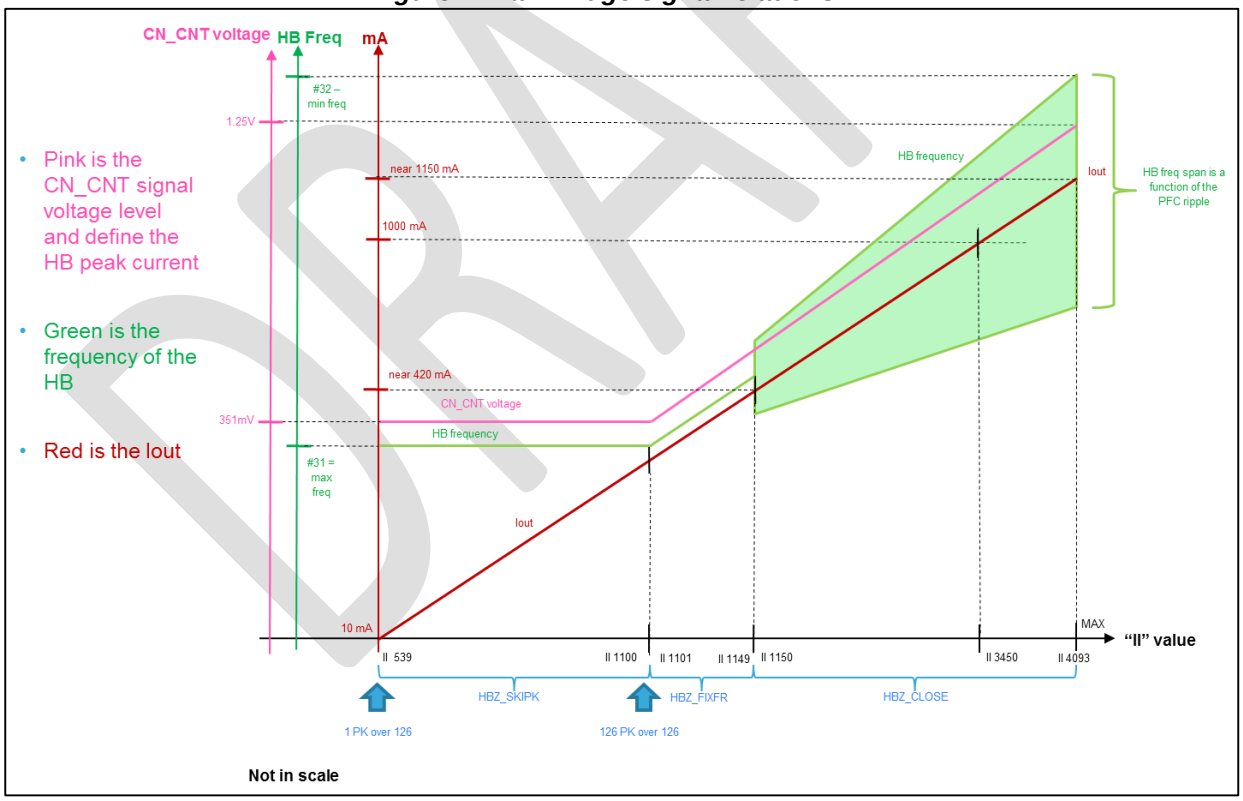
The Half Bridge topology is a voltage fed LC converter. A frequency define the current on the LC tank and, because the used frequency is five time higher than the resonant frequency, the tank current is approximated to a triangular shape. The primary pick current is near the half of the output current if the output transformer ratio is 1:1. The real output transformer ratio used on ILL066V1 demo board is 1:1.85.

On the ILL066V1 demo board there is the “*ll*” serial command used to modify the output current. This number define some internal behaviors. The relation of “*ll*” numbers and the output current is defined on the ILL066V1 User Manual.

On *Figure 7* showed the relation of output current, HB frequency and CN_CNT voltage. If you see on the figure there is three functional "*zone*". The first, used when the output current is "lower" has name "*HBZ_SKIPK*", the second has name "*HBZ_FIXFR*" and the last has name "*HBZ_CLOSE*". The detail of this "zone" is explained below.

- "*HBZ_SKIPK*": This "*zone*" define the "lower" output current. The algorithm implemented here use the SMED3 to force the Half Bridge to start and stop the proper activity. The SMED 3 on/off time is a function of the output current requested by the User (form DALI, 0-10V or "*ll*" command). To set the lowers output current ("ll 539") the SMED 3 active the half bridge frequency only one cycles and set off for 125 cycles. In the same way, this zone set the maximum output current when all the 126 cycles set active the HB switching. The Half Bridge frequency, in this zone, is fixed to a maximum frequency defined by the #31 parameter file. This "*zone*" is selected when the "*ll*" value is between 539 and 1100.
- "*HBZ_FIXFR*": This "*zone*" is a small and unstable area to connect the other two "*zone*". This "*zone*" is selected when the "*ll*" value is between 1100 and 1150.
- "*HBZ_CLOSE*": This "*zone*" control the output current using the voltage on "*CN_CNT*" signal. The CN_CNT" signal is a function of the output current defined by the User. The Half Bridge frequency is moved into a predefined frequency during the output current transaction. After the output current transaction, the Half Bridge frequency is regulated by the "*CN_CNT*" signal. This "*zone*" is selected when the "*ll*" value is between 1150 and 4093.

Figure 7: Half Bridge signal relations



4 Library interfaces

This chapter defines the available function called by the main (or other source module), that control the PFC and the Half Bridge hardware. To identify signal name or component described into this manual, please refer to the ILL066V1 demo board schematics present into the AN4461.

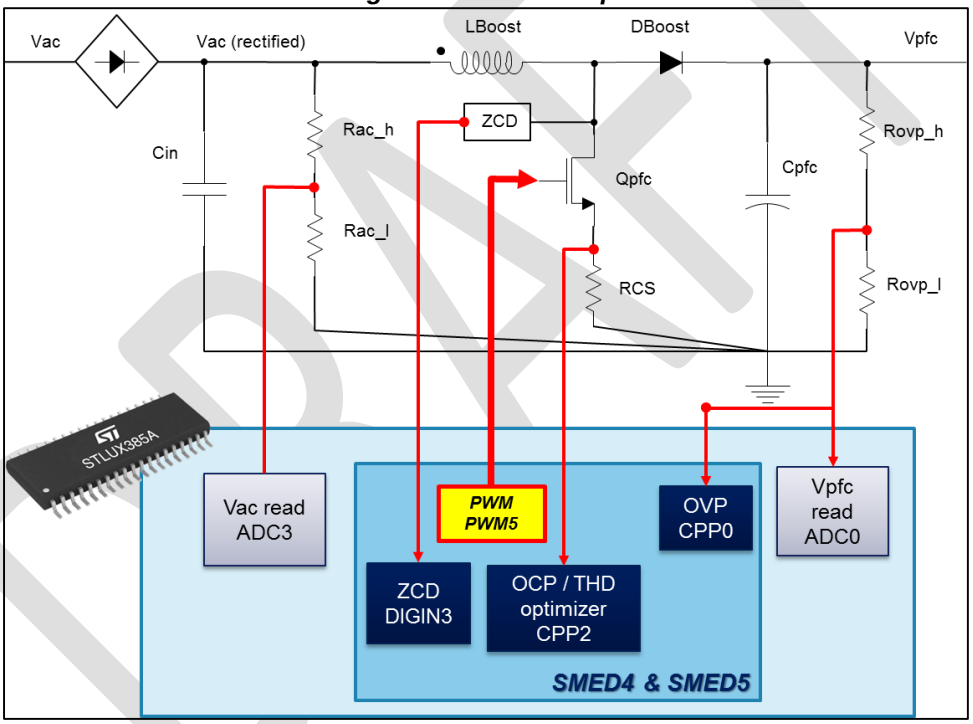
Each library module has routines to initialize your own variables, to start or stop the output PWM PIN, some other routine is called during the STMR timer interrupt. There is a global variable to control the module behavior. Inside every library module there is a routine called during the associated SMED interrupt.

4.1 PFC library Interface

This paragraph describes the available function implemented into the PFC library and makes the necessary action to manage the PFC signal. The PFC module is implemented using the SMED4 and the SMED5 resources. The PFC module control the SMED5 PWM output signal connected to the PFC gate driver.

A PFC circuit reminder is shown on the next figure:

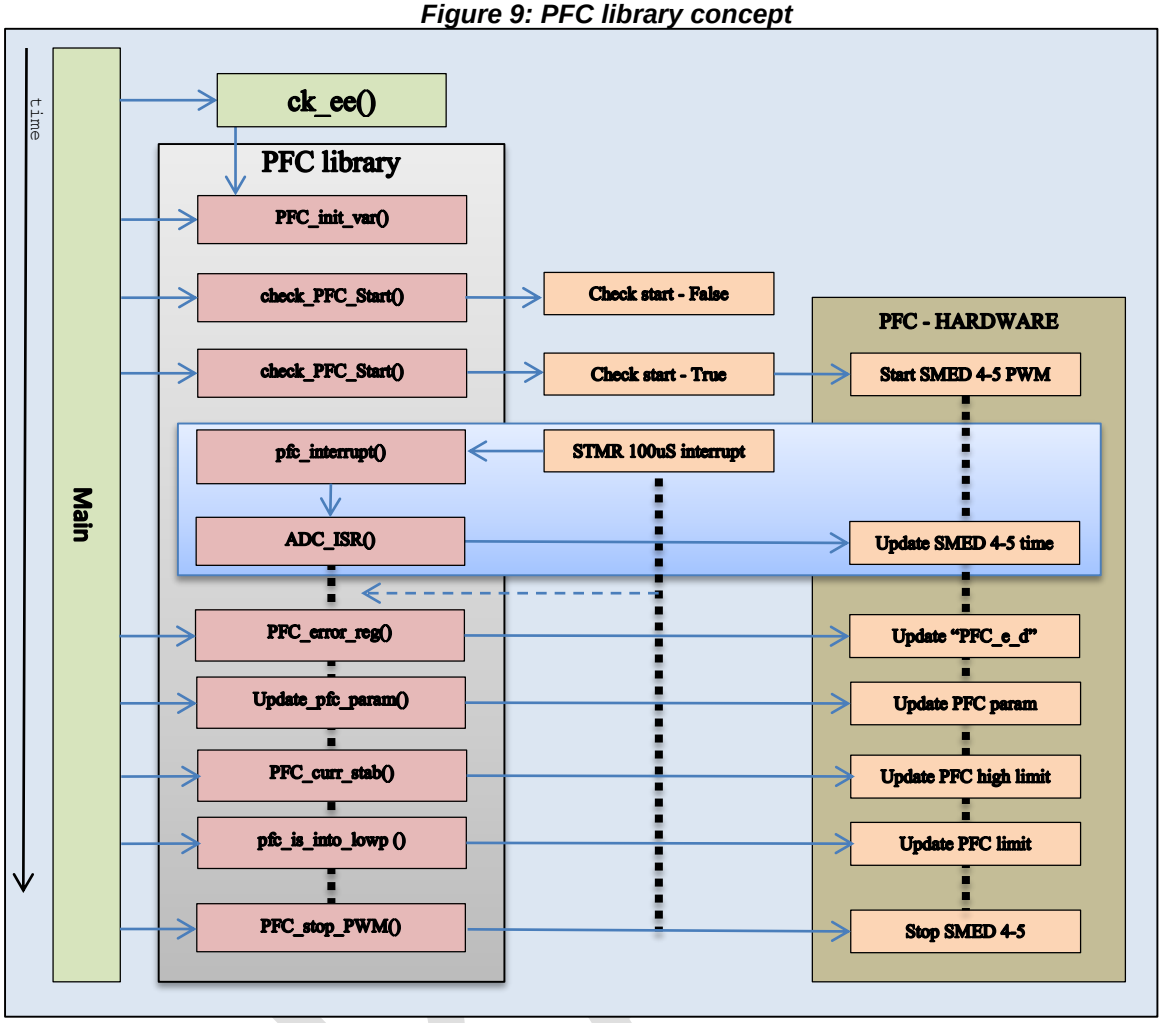
Figure 8: PFC concept



Into the PFC library module is referenced the SMED4 and the SMED5 interrupt routines and the users do not reference on the owner code.

Figure 9 offers a system overview on STLUX385A PFC library implementation.

Figure 9: PFC library concept



The previous figure shown the call sequence of the most important PFC library routine. There are more than these routines on the PFC library.

4.1.1 u8 PFC_init_var(void)

Call this routine from the main to initialize the PFC internal variable. Some variables used here are stored into the EEPROM data area and, before call this routine, call the "*ck_ee*" subroutine. Call this function only one time and during the startup. The number returned by the subroutine identifies the PFC library revision.

4.1.2 void check_PFC_Start(u8)

Call this function from the main to start the PFC only when the PFC is not running. Before PFC start, the subroutine checks some external signal to verify the possibility to start the PFC. For example, the PFC start only if the input rectified Vac voltage is greater than a predefined level (*PFC_in_min*). When not all condition is true, the PFC do not start. When the parameter passed to these subroutines is equal to 0, the routines detect the startup time overlap, set an error (ERR_PFC_OT) and activate the restart time procedure (using SERV_START service).

This routine called also some other internal functions. The function called by this routine read the rectified Vac voltage and identify the input frequency. The routine is disabled when the PFC start because the PFC behavior is automatically updated during the interrupt routines.

4.1.3 void pfc_interrupt(void)

This is PFC interrupts routine and start the PFC control loop procedure. The "*main.c*" call this function during the STMR timer interrupts. This routine synchronized by the STMR interrupt routine enable the ADC interrupt routine to start the PFC control loop procedure. Inside this routine there is a call to others routines. The called routine verify the input Vac top voltage and detect the input frequency.

Into this routine is also managed the PFC startup procedure enabled during the "*check_PFC_Start()*" routine.

4.1.4 INTERRUPT_HANDLER (ADC_ISR, 22)

Called by the ADC ISR interrupt when a new ADC conversion is available and only when the STMR interrupt request it. This routine is used to accumulate the PFC output voltage into one AC cycles. When one ac cycles ends, calculate the new ON/OFF time used during the next AC cycles. The algorithm used into this routine is a Proportional Integration system. The proportional and the Integration value is defined by some parameters present on the parameters file. The users adapt these parameters value to the proper hardware.

Into this interrupt routine is implement the entire PFC main algorithm. To regulate the PFC output voltage the PFC on time or the PFC off time are modified. The function is not called when the PFC is stopped.

4.1.5 void pfc_error_reg(u8)

This routine, called by the main, computes the internal PFC error value and defines when PFC is into good regulation or are out of a predefined level defined by the "*PFC dE to LPOK*" parameters file (parameters #38, #39 and #40). These parameters identify when the PFC output voltage is into "good zone".

4.1.6 u8 Update_PFC_param(void)

This function updates the PFC parameters according to the input AC voltage. The subroutine check the input voltage using the "#22, #23, #24" parameters specify on the "*parameter file*". There is three range defined low range (typically a USA range), medium range (between low and high) and high range (typically the European range). The PFC parameters changed by this routine define the PFC behavior according to the input voltage. For example, define the OCP protection levels according the input voltage.

This subroutine is called every 800uS.

The return parameters identify when a change is performed during the call. If the return is 1, the high level is selected (default is into EU range). If the return value is 2, the middle range is selected (lower then EU but higher then USA range). If the return value is 3, the lower level is selected (USA range). If the return is 0, no change is performed during the call.

4.1.7 void update_PFC_MM(void)

Update the internal variable used during the PFC debug phases and showed using the "*gp*" serial command.

This subroutine is called every 800uS.

4.1.8 void PFC_set_vout_story(void)

This subroutine is called every 100uS to update the PFC Vout story. The output of this routine is some global variable used to identify the input frequency.

4.1.9 u8 update_PFC_V(void)

This routine read the PFC Vout voltage and calculate the medium value every AC cycle. Check also the PFC output voltage to verify if is lower than predefined value (*PFC_v_low_lim*). When is lower, the variable returned is not zero to inform the main of this problem. The main make the appropriate action (output UVLO protection).

This routine is called every 100uS.

4.1.10 void PFC_stop_PWM(void)

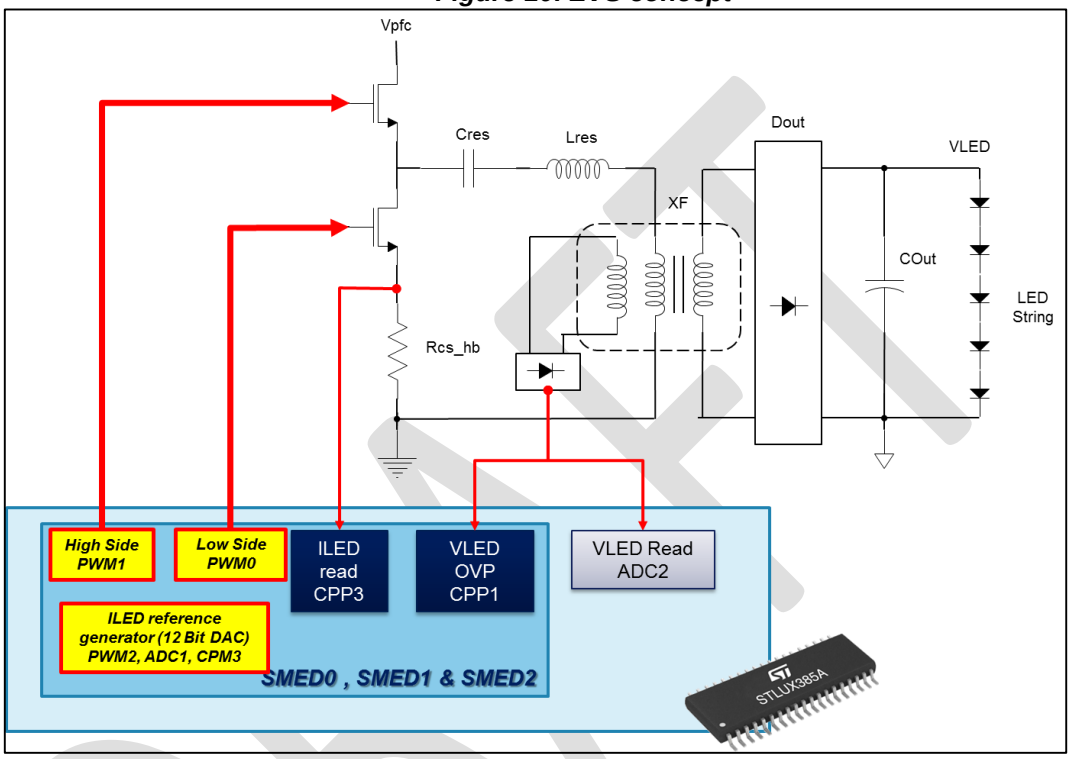
This subroutine stops the PFC module (and also the PWM output signal) and clear internal variable to inform other PFC routine to this situation. It is important the main call this routine only when the HB is also switched off otherwise the output current is out of control.

4.2 Half Bridge library Interface

This paragraph describes the available function implemented into the Half Bridge library and makes all the necessary action to manage the Half Bridge MOSFET gate. The Half Bridge module is implemented using the SMED0 for low side and the SMED1 for high side. The SMED2 as used to generate the analog reference voltage. This reference voltage defines the top current used by the HB module. The SMED3 is used to manage the low current on the HB module.

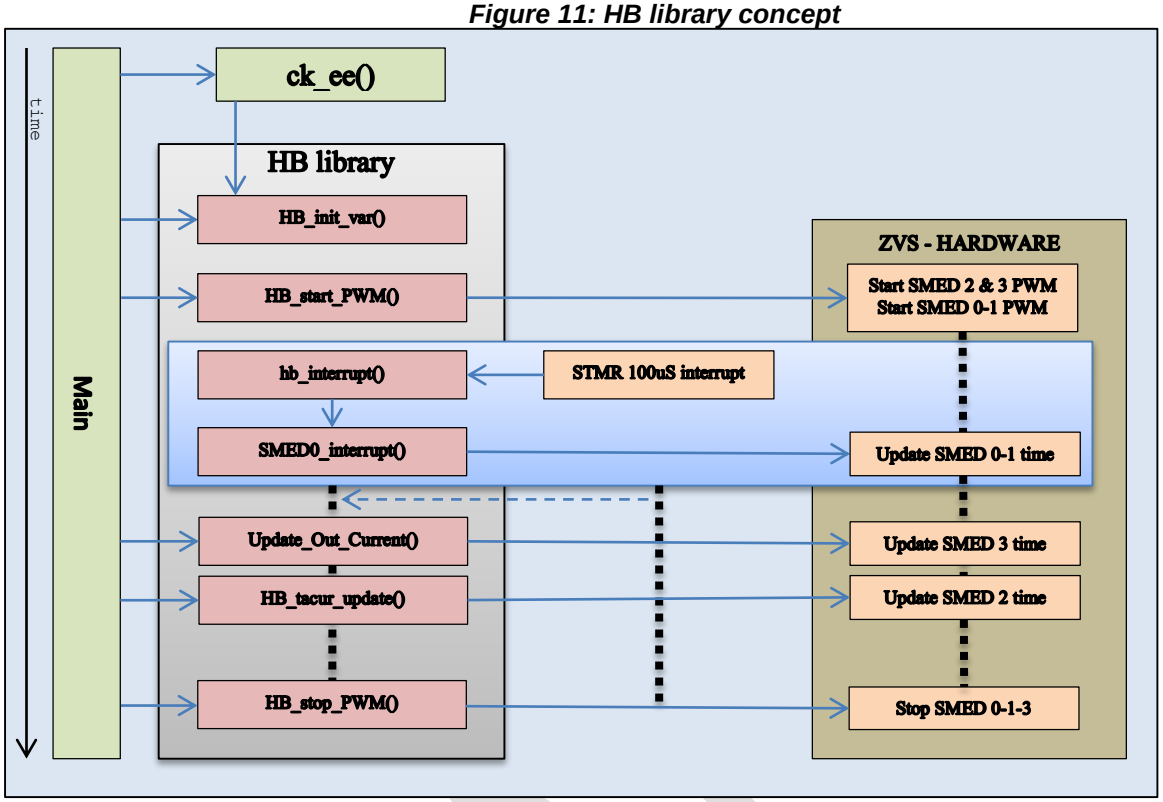
A circuit reminder is shown on the next figure:

Figure 10: ZVS concept



Into this library module is referenced the SMED0, the SMED1, the SMED2 and the SMED3 interrupt routines. One routine is called during the STMR timer interrupt to startup the HB current regulation function.

Figure 11: HB library concept



On the previous figure is showed the call sequence of the most important HB library routine. There are more than these routines on the HB library.

4.2.1 u8 HB_init_var(void)

Call this routine by the main to initialize the Half Bridge internal variable. Some ram variables used here is stored into the EEPROM data area and, before call this routine, call the "*ck_ee*" subroutine to initialize this variable. Call this function only one time and during the startup phases. The number returned by the subroutine identifies the HB library revision.

4.2.2 void hb_interrupt(void)

The STMR timer interrupt call this routine to check/adjust of the output current. Call this routine every 100uS synchronous from the STMR timer. The execution time depend from the HB frequency. When called, and when is necessary, this routine enable the SMED0 interrupt time.

4.2.3 INTERRUPT_HANDLER(SMED0_ISR, 6)

The SMED0 interrupt update the ON time of SMED0 and SMED1. The ON time modification change the output current on the LEDs string. The correct time is acquired using the SMED "dump" facility.

4.2.4 void HB_start_PWM(void)

This function activates the half bridge PWM signal. The startup frequency of the half bridge module is defined as the maximum frequency (defined by parameter #31). The #31 parameter represents the SMED time (expressed using a 96MHz SMED tick) during the state 2 of SMED 0 and SMED1. The startup output current is defined by the parameter #4 and is apply after the HB startup phases.

4.2.5 u8 check_S_CH0(void)

This subroutine acquire the input voltage present on the ADC CH2 (S_CH0 signal) and check if this voltage is lower than a parameter #33 present on configuration file. If the compare is higher return a TRUE value

indicating the output voltage, present on the J4 connector is higher than expected. In this case the main codes apply the error #7 (ERR_OVP_CH0) and stop the PFC and HB activity.

The subroutine is called every 100uS but require a maximum of 400uS to identify a new value of S_CH0 signal.

4.2.6 void update_CPM_val(void)

This function acquires the value of ADC CH1 (CN_CNT) and store this value into an "accumulation variable" used during the CN_CNT voltage loop regulation. This function is called every 100uS and use a few uS of the processor time.

4.2.7 void HB_tacur_update(void)

This function regulates the CN_CNT voltage and consequently regulates the output current. To regulate the CN_CNT voltage change the SMED2 state 0 time. The function compensate the external tolerance (VCC, resistor and capacitor variation) using the voltage acquired on ADC CH1 by the update_CPM_val function. The function is called every 800uS. The minimum output voltage is defined by HB_OL_LIM limit (default 1150) consequently the minimum output voltage at ADC-CH1 pin (CN_CNT) is equal to 0,3512V (typ). The maximum output voltage (theoretical) at ADC CH1 pin is 1.25V but depends also from R3, R4, C8 and VCC value.

This function setup also the ON time of the HB stage to support the required Iout value (specified on "*PWM2_on_time*" variable) during the output current modification. A table define every intermediate point and the maximum Iout slope ("*hb_ont[]*" table).

4.2.8 void HB_burst_mode(u16)

This function is used to define the SMED3 ON/OFF time. The SMED3 define the "packet skipping" mode on the HB PWM line. The parameters passed to the subroutines represent the level of the output current expressed into equivalent "ll" value. The correct parameter range is from 1 and "*HB_SM3_SM01*" (default 1100). If the parameter passed to the subroutine is lower than "*HB_MIN_PWM2*" (default 539), the routine apply the lowest value. If the parameters passed to the subroutines is higher the "*HB_MIN_PWM2*" the routine do noting.

4.2.9 void New_PWM2(B2W)

This function is used to initialize the value of PWM2 state 0 time. This state timer defines the output off time of PWM2 and, consequently, define the voltage on the CNT_CN signal. The minimum allowed value is 2 and represents the maximum output voltage on CN_CNT signal. The maximum allowed value is 4094 and represents the minimum voltage on CN_CNT signal.

4.2.10 void Update_Out_Current(void)

This routine regulates the output current when the "open loop" is forced. The open loop is defined as when the "*ll*" (output current) value is lower than "*HB_SM3_SM01*" value. This routine is called every 1.6mS and defines the rising and the falling "rate" of output current during the "open loop" phases. This routine control the SMED3 activity, start or stop the SMED3 behaviour, regulate also the HB frequency when the "*HBZ_FIXFR"* mode is necessary and other internal Half Bridge jobs. To define the proper action this routine look on some internal HB variable.

This routine update also the "*HB_zone*" to identify in which zone are now the HB. The user modifying the "PWM2_on_time" variable defines the target current. When the output current is regulated using the "*HBZ_CLOSE*" zone, this routines do noting because the output current is regulated modify the "*CN_CNT*" voltage.

The routine entering into "*HBZ_CLOSE*" zone when the output current is bigger than "*HB_OL_LIM*" value (default is 1150). The routine entering into "*HBZ_SKIPK*" zone when the output current is lower than "*HB_SM3_SM01*" and bigger than "*HB_SM3_LIM*". The middle values use the "*HBZ_FIXFR*" zone is not stable and is used only during the transaction from "*HBZ_SKIPK*" to "*HB_OL_LIM*" and vice versa.

4.2.11 void HB_stop_PWM(void)

This routine makes the necessary action to STOP the Half Bridge PWM frequency and setup some internal variable for the next Half Bridge start.

4.2.12 void search_mm_HB_on(void)

This routine is used for debug purpose and stores the minimum and the maximum of the SMED0-S2 timer value. This routine is called every 800uS only. The output variable is showed using the "*gh*" serial command.

5 Variable parameter description

Into this chapter all the available variable parameters are described. There are three types of variable parameters present on the ILL066V1 demo board library.

- The parameters define the configuration of the board and are usable to disable or enable some interfaces or function of the demo board. For example, into this parameters type is possible to enable the DALI interface or enable the Half Bridge current loop control. These parameters are defined as: "*application*" level. These parameters are changed only by the serial line and are apply only after the power off or reset.
- The parameters define the PFC functionality. Not all parameter defined here control the total PFC activity. Some PFC parameters depend also from hardware value. For example, the PFC output voltage is defined by external resistor value because the value of the PFC target voltage is fixed into STLUX FW. These parameters are defined as: "*PFC*" level.
- The parameters define the Half Bridge functionality. Also into this area some parameters are defined by the external hardware. These parameters are defined as: "*HB*" level.

On this chapter there is a list of all FW modifiable parameters used into ILL066V1 demo board. There is a dedicated paragraph to clarify each parameter itself.

To modify the parameters value the "*ip*" and "*in*" serial command are available.

Some PFC parameters are split into three areas, called high, middle and low range. This split area refers to the input Vac level. The value of these three PFC areas is specifying by #22, #23 and #24 parameters.

The list of all available parameters is:

FW IDX	Level	Description	Default value	Unit	Range		See Paragraph
					Min	Max	
1	APP	PFC activation enable when 1	1	Bit	0	1	*5.1*
2	APP	HB activation enable when 1	1	Bit	0	1	*5.2*
3	APP	Dali interface enable when 1	1	Bit	0	1	*5.3*
4	HB	Start-up output current level	539	Pure number	539	4093	*5.4*
5	n.u.	Reserved					
6	APP	Half Bridge current loop compensation enable when 1	1	Bit	0	1	*5.5*
7	APP	Debug mode enable when 1	1	Bit	0	1	*5.6*
8	APP	0-10 V interface enable when 1	0	Bit	0	1	*5.7*
9	PFC	PFC THD maximum time	2000	1/96MHz	1	9600	*5.8*
10	APP	Enable the ROP code protection when 1	0	Bit	0	1	*5.9*
11	n.u.	Reserved					
12	PFC	DAC level (OCP) during emergency mode	7	DAC-bit	4	12	*5.10*
13	PFC	DAC level (OCP) during high level mode	8	DAC-bit	5	13	
14	PFC	DAC level (OCP) during mid level mode	11	DAC-bit	6	14	
15	PFC	DAC level (OCP) during low level mode	13	DAC-bit	7	15	
16	PFC	Define the PFC activation level - high	503	ADC-bit	181	1023	*5.11*
17	PFC	Define the PFC activation level - mid	407	ADC-bit	181	1023	
18	PFC	Define the PFC activation level - low	296	ADC-bit	181	1023	
19	PFC	Define the PFC input voltage stop level - high	445	ADC-bit	180	1023	*5.12*
20	PFC	Define the PFC input voltage stop level - mid	376	ADC-bit	180	1023	
21	PFC	Define the PFC input voltage stop level - low	267	ADC-bit	180	1023	
22	PFC	Define the PFC level entering into high zone	518	ADC-bit	180	1023	*5.13*
23	PFC	Define the PFC level entering into mid zone	422	ADC-bit	180	1023	
24	PFC	Define the PFC level entering into low zone	251	ADC-bit	180	1023	
25	PFC	PFC UVLO on output level - high	329	ADC-bit	180	1023	*5.14*
26	PFC	PFC UVLO on output level - mid	289	ADC-bit	180	1023	
27	PFC	PFC UVLO on output level - low	189	ADC-bit	180	1023	
28	PFC	PFC Start On time	70	1/96MHz	10	2047	*5.15*
29	PFC	PFC Start Off time	1	1/96MHz	1	336	*5.16*

30	PFC	Delay to detect input Vac missing	100	mS	2	255	*5.17*
31	HB	Maximum available HB frequency - SMED0&1-S2	61	1/96MHz	61	260	*5.18*
32	HB	Minimum available HB frequency - SMED0&1-S2	950	1/96MHz	260	1120	*5.19*
33	HB	No load level protection - ADC CH2	818	ADC-Bit	736	1023	*5.20*
34	HB	Propagation delay mismatch compensation	30	1/96MHz	0	100	*5.21*
35	HB	Trimming HIGH side delay	4	1/96MHz	0	20	*5.22*
36	HB	HB dead time	33	1/96MHz	24	192	*5.23*
37	n.u.	Reserved					
38	PFC	Δ error to entering into LOOP OK - high zone	7	Pure number	0	100	*5.24*
39	PFC	Δ error to entering into LOOP OK - mid zone	10		0	100	
40	PFC	Δ error to entering into LOOP OK - low zone	16		0	100	
41	HB	When true activate the fast output current change	1	Bit	0	1	*5.25*
42	PFC	PFC Integral Gain	160	Number	1	254	*5.26*
43	PFC	PFC Integral Gain Scaler	15	Number	1	24	*5.27*
44	PFC	PFC Proportional Gain	150	Number	1	254	*5.28*
45	PFC	PFC Proportional Gain Scaler	12	Number	1	24	*5.29*
46	PFC	PFC Over Voltage high speed level enable	940	ADC-Bit	920	1023	*5.30*
47	PFC	PFC Under Voltage high speed level enable	855	ADC-Bit	600	900	*5.31*
48	PFC	PFC under voltage high speed gain	6000	Number	1	10000	*5.32*

Table 2: Variable parameters table

Unit legend:

- Bit — As defined two state value. If is 1, activate the function, if is 0, disable the function.
- 1/96Mhz — This unit represents the clock cycle of the SMED. For the SMED0, SMED1 SMED4 and SMED5 this time is 10.416666 nS (or 1/96MHz).
- DAC-Bit — This unit is used with the STLUX DAC peripherals. The value defined with this unit is copied, when necessary, into the related DAC register.
- ADC-Bit — This unit is the value read from the ADC register. To collect the voltage at the STLUX pin multiply the value to 1.221896mV (1.25V/1023).
- mS — Is a 1mS unit time. This specified a time before start action.

The “reserved” parameters are used only for back compatibility whit the old source code and are not modifiable by the users.

5.1 PFC activation enable

This parameter identify if the PFC firmware module is enable (when 1) or is disable (when 0). It’s used during the hardware debug phases to verify the external circuit and the external resistor partition. The default value is 1 means PFC module active.

5.2 HB activation enable

This parameter identify if the HB firmware module is enable (when 1) or is disable (when 0). It’s used during the hardware debug phases to verify the external circuit and the external resistor partition. The default value is 1 means HB module active.

5.3 Dali interface enable

This parameter identify if the DALI interfaces module is enable (when 1) or is disable (when 0). It’s used during the hardware debug phases to verify the external circuit and the external resistor partition. The default value is 1 means DALI interfaces active. NOTE: to enable the 0-10 interfaces (#8) it’s necessary to disable the DALI interface because this two interfaces share some STLUX internal resources.

5.4 Start-up output current level

This parameter identifies the output current when the demo board is powered. This value is used only if the DALI and 0-10V interfaces are disabling. The valued used here is the same of the value used during the “*ll*” command.

The minimum available value is 539 (minimum output current) and the maximum available level is 4093 (maximum output current).

5.5 Half Bridge current loop compensation

This parameter identify if the automatic current control loop is active on the HB module. If the current loop compensation is disabling, the HB frequency is fixed. The default value is "1" and means HB current compensation active, the value "0" means disable. To change the output current (changing the HB frequency), when the current loop compensation is disable, use the "*hb*" serial command.

5.6 Debug mode enable

This parameter identifies if the board is active or totally disable. When this parameter is "1" all the activity of the board is froze. This mode is the default mode when a new firmware is loaded into the demo board.

5.7 0-10 V interface enable

This parameter identifies if the 0-10V interfaces are active and control the output current. This interface is automatically disabled when the DALI interface is enabling. To enable the 0-10 V interface disable the DALI interface. The default value of this parameter is 0 that means disable.

5.8 PFC THD maximum time

This parameter identify the maximum numbers of clock cycles (tick) used to wait the PFC peak current arrive to a predefined level. This numbers is expressed in clock cycles (1/96MHz) and is apply on the SMED5 state 2. This wait time is applied outside the PFC startup period and outside the PFC light output power. The available range for this parameters start from 1 (means: no THD optimization) to 9600 tick (means: wait for a maximum of 100uS the minimum current). During this time the PFC gate driver is enabling and the MOSFET is closed. See the STLUX Reference Manual and the SMED configuration program to understand the SMED activity. The PFC minimum peak current on the ILL066V1 demo board is 206mA (typ.) and is defined by the U18 comparator and R100, R101 partition.

5.9 Enable the ROP code protection

This parameter identify, when 1, if the program code is protected from download and to any modification of configurable parameters. When this parameters is 1, disable all command used to modify the parameters and disable the code download using the SWIM. To remove this setting the only way is to erase the entire chip (also loose the actual configuration file).

5.10 PFC DAC level (OCP)

This parameter identifies the Over Current Protection level. This parameter is used to setup the DAC level of the STLUX used on the PFC stage. There is the possibility to make four current level setting. The first level is defined using an internal behavior define "emergency" and is triggered when the PFC module detect an immediate protection of the PFC hardware stage. The other three levels define three different current protection level when the input AC voltages is into a predefined level. All the values of these parameters are referred to a DAC value. The protection voltages have a step of 0.082V and the PFC current protection depends from the PFC shunt resistor value. To identify the real input voltage please multiply the single step to the value of the parameters. For example the default "emergency" level is equal to 0.082*7=0.574V. The resistor defines the maximum current available every protection level. The SMED5 triggered an interrupt when the OCP event arrives.

5.11 PFC activation level

This parameter is the PFC input voltage, at the STLUX pin, which identifies the activation level of the PFC module. When the PFC output voltage is below the level defined into these parameters, the PFC don't start but if it's running stay active. These three levels depends from the input Vac level defined on parameters #25, #26 and #27. The value defined into these parameters is the ADC conversion value.

To obtain the “equivalent” input voltage at STLUX pin please multiply the parameter value to 1.221896mV. To obtain the “equivalent” input voltage at the V_IN signal sees the external resistor partition. On the ILL066V1 demo board to obtain the correct V_IN voltage divide the parameters on this field to 2.225929283772 constant.

5.12 PFC input voltage stop level

These parameters identify the PFC input level to stop the PFC modules. The PFC module controls the V_IN signal and, when stay lower below this parameters for a minimum of 80mS, the PFC module stop the PFC activity and entering into “emergency mode”. These three levels depends also from the input Vac level defined on parameters #25, #26 and #27.

To obtain the “equivalent” input voltage at STLUX pin please multiply the parameter value to 1.221896mV. To obtain the “equivalent” input voltage at the V_IN signal sees the external resistor partition. On the ILL066V1 demo board to obtain the correct V_IN voltage divide the parameters on this field to 2.225929283772 constant.

5.13 PFC level entering into “zone”

This parameter identifies the three zones which is divided the total input voltage (V_IN signal). The PFC module checks if the top of the AC cycle on input voltage is higher than the “high” zone to entering into high zone. If the input voltage is lower than “high” zone check if is higher than the “mid” zone to entering into mid zone. If the input voltage is lower than “mid” zone check if is higher than the “low” zone to entering into low zone.

To obtain the “equivalent” input voltage at STLUX pin please multiply the parameter value to 1.221896mV. To obtain the “equivalent” input voltage at the V_IN signal sees the external resistor partition. On the ILL066V1 demo board to obtain the correct V_IN voltage divide the parameters on this field to 2.225929283772 constant.

5.14 PFC UVLO on output level

This parameter identifies the level of UVLO protection on the PFC output voltage. The PFC output voltage is continuously acquired and compared with the value defined into these parameters. When the PFC output voltage is less than the value defined in this parameters then the PFC module stop its activity.

To obtain the “equivalent” input voltage at STLUX pin please multiply the parameter value to 1.221896mV. To obtain the “equivalent” input voltage at the PFC_OUT signal sees the external resistor partition. On the ILL066V1 demo board to obtain the correct PFC_OUT voltage divide the parameters on the field to 2.225929283772 constant.

5.15 PFC On time during startup

This parameter identifies the SMED5-S3 time used during the start-up of the PFC module. This parameter depends primarily from the PFC inductor value, from the Vac input voltage and from the load when the PFC starts. Those parameters are expressed into SMED clock cycle (1/96MHz) and define the value of the SMED5-S3 (state 3) time during start-up of the PFC. Basically these parameters define the PFC output voltage ramp-up. The SMED5-S3 values are automatically changed when the PFC output voltage arrive near the target value.

To calculate the real PFC on time during the PFC ramp-up add the SMED5-S1 (fixed to 40), the SMED5-S2 (during ramp-up fixed to 1) and the value of this parameter. For example, the default values of this parameter on the ILL066V1 demo board is: 40+1+70 SMED clock cycles which are 1.15uS.

Before the PFC start there is 30mS to acquire the Vac input voltage to identify the applied input level before start the PFC activity. To understand the SMED behavior on PFC stage, see SMED4 and SMED5 evolution using the SMED configurator.

5.16 Off time during PFC start

This parameter identifies the minimum PFC OFF time and is used to identify the minimum OFF time before search the ZCD event. To understand see the PFC (SMED4 and SMED5) evolution using the SMED

configurator. Those parameters are expressed into SMED clock cycle (1/96MHz) and define the minimum value of the SMED4 State 0 time.

5.17 Delay to detect PFC input Vac missing

This parameter identifies the minimum time to detect the missing of the Vac input voltage. The minimum time to detect this event is a one Vac rectified cycle and, for 50Hz is equal to 10mS. Below this time is impossible to detect because the FW inside the STLUX verify the Vac peak to peak voltage and this is detect every Vac cycle. The unit of this parameter is 1 mS.

5.18 Maximum available HB frequency

This parameter identifies the maximum available frequency using during the HB current regulation. This numbers identify the maximum frequency of the Half Bridge stage. Please see the AN4461 to understand the ZVS topology. This parameter is expressed into SMED clock cycle (1/96MHz). This time define the minimum number load into SMED0&SMED1 state 2. To calculate the HB frequency this parameter is added to the Dead Time plus the LEB time plus the HOLD transaction (2.5 plus 2.5 cycle) multiply by 2 SMED used for the HB stage. On the PSR ZVS demo board this time is: (((33+24+61)*2)+5)*(1/96) = 241*10.41nS near 398KHz.

5.19 Minimum available HB frequency

This parameter identifies the minimum available frequency using during the HB current regulation. This numbers identify the minimum frequency of the Half Bridge stage. This parameter is expressed into SMED clock cycle (1/96MHz). This time define the maximum number load into SMED0&SMED1 state 2. On the PSR ZVS Demo board this time is: (((33+24+750)*2)+5)*(1/96) = 1619*10.41nS near 60KHz. This minimum frequency depends from load, output transformer characteristic and HB supply voltage.

5.20 HB No load level protection

This parameter identifies the ADC value witch define the “no load” detecting during the HB operation. This value, on the ILL066V1 demo board, is near 1V. To found the real voltage present on the output connector, the T2 transformer ratio and the R92-R28-R29 resistor ratio is necessary to know. In case of the ILL066V1 demo board the T2 transformer ratio is 5.2 and the shunt resistor ratio is near 18.43. The maximum default output voltage on the ILL066V1 demo board at J4 connector pins is near to 95V.

5.21 Propagation delay mismatch compensation

This parameter identifies the propagation delay mismatch compensation due to the delay of external circuit (the HB driver), the external RC filter and STLUX internal propagation delay. This parameter is expressed into SMED clock cycle (1/96MHz) and the default value for the ILL066V1 demo board is 30 SMED cycle or 312.5nS.

5.22 Trimming HIGH side delay

This parameter identifies the time difference from low side to high side on the HB stage. On the ILL066V1 demo board the circuit delay is near 41nS. The difference is due to the external driver delay difference from low side to the high side. This parameter is expressed into SMED clock cycle (1/96MHz). This time is added to the HB High Side signal to compensate the external different time.

5.23 Dead time delay

This parameter defines the dead time on the Half Bridge stage. The time is applied on the low side and on the high side stage together. On the ILL066V1 demo board the default value is 343.75nS (or 33 SMED cycle) and the valid range is from 250nS to 2uS (or from 24 to 192 SMED cycle).

NOTE: The external drive (L6388E) forces the minimum time due to an internal minimum dead time (typ. 320nS).

5.24 PFC error to LOOP OK

These parameters are the numbers of error to enable the PFC_B_LPOK flag. There is three zone defined by the Vac input value.

5.25 Fast HB current regulation

When this flag is true (default) enable the fast Iout regulation. When true the typical switch on time (from 10mA to 1A output) is 150mS typical. If case the DALI interfaces is used, the DALI stack regulated the fade time.

When the flag is false (equal to 0) the minimum fade time manage by the demo board is 333mA/S.

The use of the bit equal to 0 is deprecated because some DALI request are not satisfied.

5.26 PFC Integral Gain

This parameter identify the Integral Gain used into the PFC PI algorithm. The default value applied on the ILL066V1 demo board is 160 and is the multiplication factory applied on the error accumulated during the last AC cycle. The sampling rate of the ADC is 100uS. Refer to *chap. 3.1* to understand the PI algorithm.

5.27 PFC Integral Gain Scaler

This parameter identify the Integral Gain scaler used into the PFC PI algorithm. The default value applied on the ILL066V1 demo board is 15 and is the exponent applied on the error accumulated during the last AC cycle. The sampling rate of the ADC is 100uS. Refer to *chap. 3.1* to understand the PI algorithm.

5.28 PFC Proportional Gain

This parameter identify the Proportional Gain used into the PFC PI algorithm. The default value applied on the ILL066V1 demo board is 150 and is the multiplication factory applied on the error accumulated during the last AC cycle. The sampling rate of the ADC is 100uS. Refer to *chap. 3.1* to understand the PI algorithm.

5.29 PFC Proportional Gain Scaler

This parameter identify the Proportional Gain scaler used into the PFC PI algorithm. The default value applied on the ILL066V1 demo board is 12 and is the exponent applied on the error accumulated during the last AC cycle. The sampling rate of the ADC is 100uS. Refer to *chap. 3.1* to understand the PI algorithm.

5.30 PFC Over Voltage high speed level enable

This parameter define the PFC high level where the high-speed gain is enable. When the PFC output voltage is more than this value, the PI bandwidth is increased to speed-up the time to arrive to the target. When the PFC output voltage is less than this value, the PI bandwidth is the default. The default PFC level to enable the high-speed PFC gain is approx. around the +3.35%.

To obtain the “equivalent” input voltage at STLUX pin please multiply the parameter value to 1.221896mV (obtained from: 1.25V/1023).

5.31 PFC Under Voltage high speed level enable

This parameter define the PFC low level where the high-speed gain is enable. When the PFC output voltage is less than this value, the PI bandwidth is increased to speed-up the time to arrive to the target. When the PFC output voltage is more than this value, the PI bandwidth is the default. The default PFC level to enable the high-speed PFC gain is approx. around the -6%.

To obtain the “equivalent” input voltage at STLUX pin please multiply the parameter value to 1.221896mV (obtained from: 1.25V/1023).

5.32 PFC under voltage high speed gain

This parameter define the slope of the correction convergence when the voltage is lower than the value specified on the parameter #47. A big number define less time, a small number define long time.

6 Fixed parameter description

On this chapter there is the description of fixed value defined into the PFC or HB library. This value is "fixed" inside the STLUX because some external circuit (typically a resistor partition) defines the level of the controlled variable.

Into next table the PFC-ZVS demo board fixed value is showed and is related to the PFC and to the Half Bridge library.

HW IDX	Types	Description	Default value	Unit	Circuit	See Parag.
1	PFC	PFC output target voltage	1.1151	V	R46-R52-R55-R58	*6.1*
2	PFC	PFC input voltage rectified @220Vac or 311Vdc	0.8462	V	Fixed on STLUX	*6.2*
3	PFC	PFC output - second OVP level	1.250	V	Fixed on STLUX	*6.3*
4	HB	HB time increment during time stamp	30	1/96MHz	Fixed on STLUX	*6.4*
5	HB	HB control voltage @ 1A	1.0539	V	R36-R37	*6.5*
5	APP	ADC input voltage on ADC[5] (pin 33) @ 14V	1.0370	V	R96-R97	*6.6*
6	APP	ADC input voltage on ADC[6] (pin 32) @ 3V3	1.1647	V	R98-R99	*6.7*

Table 3: Fixed parameters table

The value of the voltage expressed on ***Table 3*** is related to the STLUX input pin.

6.1 PFC output target voltage

This fixed parameter identifies the target PFC voltage apply to the STLUX pin. It's fixed because some external resistor define the real PFC voltage. This level, on the ILL066V1 demo board, is fixed at 1.1151 Volt and is used as a reference for the PFC output voltage. In case of ILL066V1demo board, the PFC output voltage is around 410V. The R46, R52, R55 value is equal to total 6M6Ω and the R58 is fixed to 18KΩ. The internal ADC conversion value is 912 and corresponding to 1.1151 Volt at the STLUX pin. If the user changes the R46, R52, R55 resistor value limit the total resistor partition to a maximum of 10MΩ. The total capacitor present on this pin also limits the bandwidths of the ADC acquisition. The actual internal ADC bandwidths is fixed, for the ILL066V1 demo board, at 10KHz (one acquisition every 100uS).

6.2 PFC input voltage rectified

This fixed parameter identifies the PFC input voltage. The partition resistors on this STLUX input have the same value of the PFC output resistor partition. The ILL066V1 demo board read the 0.8462 voltage at STLUX pin when 220Vac is applied on the Vac input pin.

6.3 PFC output - second OVP level

This parameter identifies the voltage witch define the triggering level of the second PFC output voltage protection. The second PFC output voltage protection is implemented directly into the SMED function without any firmware activity. This is the last resources to limit the PFC output voltage and, for the ILL066V1 demo board, is fixed at 452V (or 1.23V at the STLUX pin).

6.4 HB time increment during time stamp

This parameter identifies the SMED cycle increment value when a new time stamp is start. This time define also the maximum $\Delta i/\Delta t$ variation during the change of the output current.

6.5 HB control voltage

This parameter identifies the voltage present on the LLC_CN signal (and also on the CN_CNT signal) when the output current is 1A. This fixed voltage is the level of peak current on the R36//R37 when the output current on LED string is 1A. This parameter depends from T2 inductor specification, T1 inductance value, output current, output voltage and frequency of the HB stage. The LLC_CN signal is the feedback from the sensing resistor and is compared to the target value define by the CN_CNT level. The default value for 1A Iout is 1.0539V. The

minimum value (due to noise rejection) is 0.351V (Iout near 400mA). Below this point, the output current is regulated using the "*HBZ_SKIPK*" mode.

The customer change the R36//R37 shunt resistor when the output current is changed respect the ILL066V1 demo board. The value for LLC_CN pin is 1.0539V when the output current is at target level. Also, if is possible, the lower range is necessary to respected (0.351V).

6.6 ADC input voltage on ADC[5]

This parameter identifies the level of the ADC[5] (pin 33 on STLUX385A) when the VD_14 signal is equal to 14V. This value is acquired to monitor the VD_14 signal.

6.7 ADC input voltage on ADC[6]

This parameter identifies the level of the ADC[6] (pin 32 on STLUX385A) when the VD_3V3 signal is equal to 3.3V. This value is acquired to monitor the VD_3V3 signal.

7 PFC check procedure

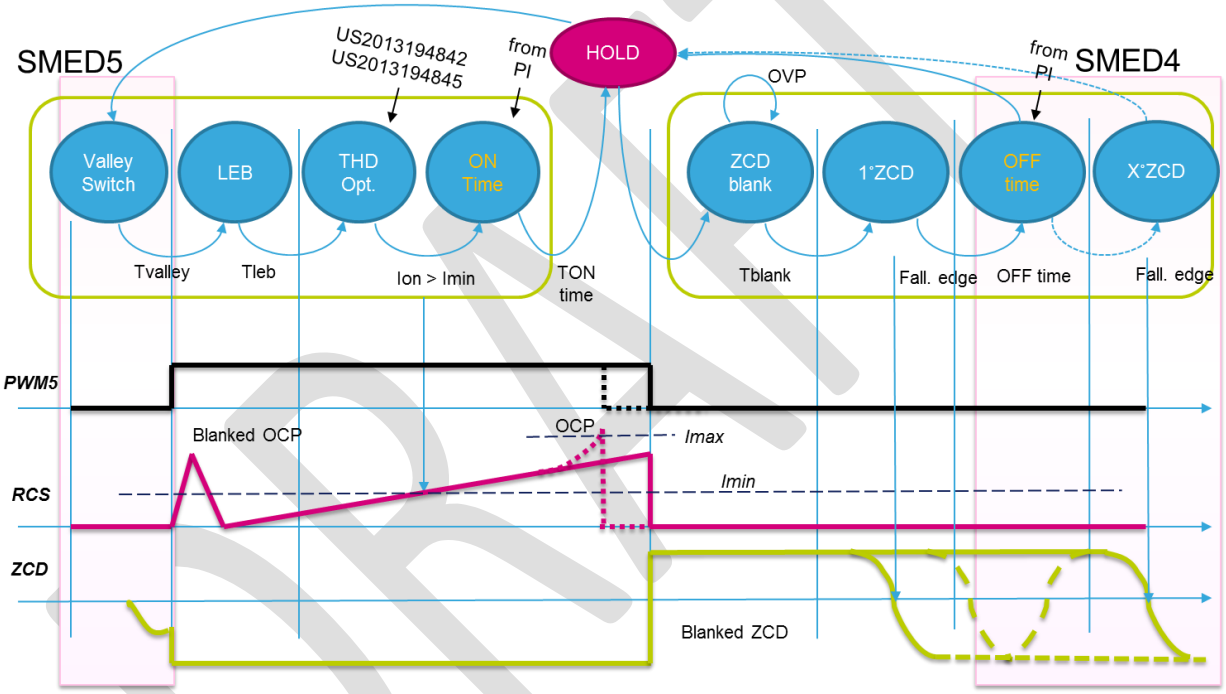
This chapter describes a suggested procedure to check the PFC functionality using the STLUX and the PSR-ZVS topology. Some serial command is used to perform / check the PFC behaviors. This procedure are generic for the PFC functionality.

NOTE: This procedure apply a dangerous high voltage on the DUT and, DUT create internal high voltage (in some circumstance). Please use caution and do not touch the board (also, when no power is applied). Only people prepared for this should perform the test procedure.

7.1 Verify the PFC functionality

To check the PFC functionality the SMED peripherals need to understand. The next figure show the SMED states machine implementation on the current PFC library.

Figure 12: STLUX PFC - SMED state machine



If you look the previous figure, the PFC PI regulation is based only to the ON time and the OFF time.

- The PFC maximum OFF time allowed used in the PI regulation is 32000 SMED tick @96MHz
- The PFC maximum ON time allowed used in the PI regulation is 1800 SMED tick @96MHz

The suggested procedure to check the PFC functionality are defined as sequential steps described here.

1. Connect the electronic load on the PFC output. Use an electronic load in CCH mode and setup the load to 10mA – 500Vdc.
2. Connect one high voltage probe to the PFC output voltage.
3. Setup the power generator to 220V-50Hz and connect to the AC input of the DUT. After power on look on the PFC output voltage if is the expected value defined by design.

4. After power-on look on the Hyper Terminal the output strings of the DUT. Verify the highlighted strings below.

```
STEVAL385LEDPSR V3R35 03:02
Build Apr  1 2015 08:07:16 core is 385A ok
Option R enabled
Ready
frq=99
```

5. Using the "*st*" command verify the highlighted strings below.

```
st
PSR-ZVS status is 0:0
Vin Vi=689 Vp=693 Vmin=689 Vmax=692 freq=100
PFC st=0x0 Vo=689 S0=1 S3=70 I=8
HB st=0x0 Z=2 S0=33 S3=61 dt=30 O0 L=9 H=9 CH0=0
PW2 ta=0:0 re=11 now=539:16507 O1:0 dump=0 ndump=0
```

NOTE: The highlighted value do not define the exactly value showed on the DUT. The allowable error is around 5 unit and depends on the AC output voltage and the component mounted on the DUT.

6. Using the "*gp*" command verify the highlighted strings below.

```
gp
PFC
f=0x0 S0=1 S3=70 e=132 Vo=689 ta=912 pw=0 Vol=691 is=1
on=70:70 off=1:1 s_t=3 ZCD=104 L=5988 fl=17
L0=:224:224:224:223:224:224
```

7. Setup the PFC to working using the "in 1 1" command.
8. Power off the DUT, wait the complete discard of the PFC output capacitor.
9. Power on the DUT and look the PFC output voltage rump up using a high voltage probe connected to the PFC output voltage. This step is the first step that involve the STLUX PFC functionality, use caution.
10. Look the highlighted output strings on the Hyper Terminal below.

```
STEVAL385LEDPSR V3R35b9 03:02
Build Apr  1 2015 08:07:16 core is 385A ok
Option PFC R enabled
Ready
frq=99
```

Note: the "*frq=99*" strings define the numbers of 100uS step into half AC cycles.

11. Using the "*st*" command verify the highlighted strings below.

```
st
PSR-ZVS status is 0:0
Vin Vi=150 Vp=693 Vmin=127 Vmax=690 freq=100
PFC st=0x41 Vo=909 S0=1688 S3=1 I=8
HB st=0x0 Z=2 S0=33 S3=61 dt=30 O0 L=9 H=9 CH0=0
PW2 ta=0:0 re=11 now=539:16507 O1:0 dump=0 ndump=0
```

NOTE: The highlighted value do not define the exactly value acquired on the DUT. The allowable error is around ±5 unit and depends on the AC output voltage and the component mounted on the DUT. S0 parameter is an indication of the OFF time used on the last AC cycles. The "S0" parameter is a function of PFC input voltage, PFC inductor and PFC load.

12. Using the "*gp*" command verify the highlighted strings below.

```
gp
PFC
f=0x0 S0=2280 S3=1 e=164 Vo=909 ta=912 pw=0 Vol=911 is=1
on=1:1 off=2280:2280 s_t=0 ZCD=104 L=5988 fl=0
L0=:4:4:4:4:4:4
```

NOTE:

- The "Vo" define the PFC output voltage during the last AC cycles. The showed value do not define the exactly value acquired on the DUT. The allowable error is around ±5 unit.
- The "*on*" parameters report the ON time used during the PFC regulation algorithm. When the load is the minimum (10mA on electronics load), this value must be to 1 (as the minimum on time).

- The "*off*" parameter define the OFF time used to regulate the PFC output voltage. The value depends on the AC input voltage, the PFC inductor value used on the DUT and the load on the PFC output. Using this value is also possible to verify the inductor tolerance during the board production if the others parameters are constants. The two numbers define the minimum-maximum OFF time during the last half AC cycle.
- The "*L0*" parameter report the PFC error during the last 153.6mS. If the six numbers are the same, the PFC output voltage is stable. A tolerance of one digit is possible.

13. Increase/decrease the load of electronic load to the target defined by design. Monitor the PFC output voltage to verify if the PFC output voltage remaining on the target voltage.
14. Power off the DUT

8 HB – fine-tuning procedure

This paragraph suggest a HB trimming procedure when the Customer debug the own hardware board. The procedure suggest a sequential step to trim the HB stage.

The fine tuning procedure is performed by changing the HB R_sensing resistor or changing the parameters file describes above. Another suggestion is to perform this check only when all hardware problem was solved. Also the PFC stage need to work correctly before perform this tuning. To understand this procedure the *Figure 7* is used as the reference.

8.1 Check the HB output power

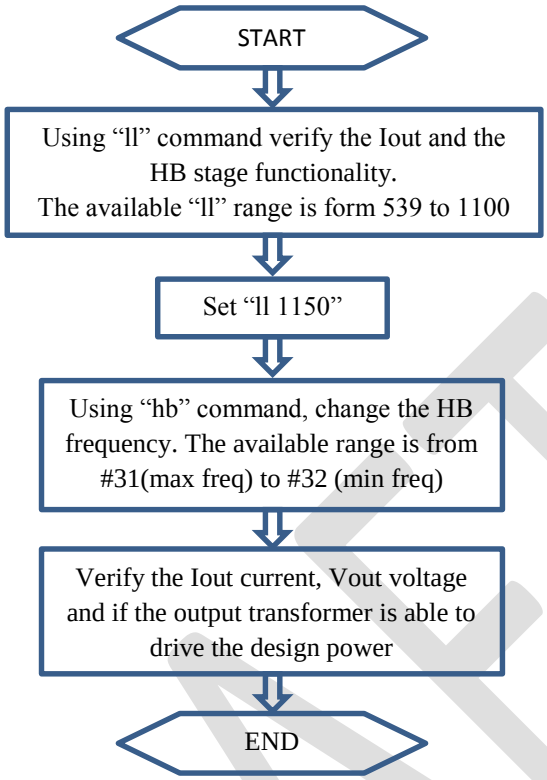
Before do anything the HB output stage need to be verified. To check the HB output performance, the HB stage behavior is modified to working into open loop mode. This is set by the #6 parameter file. When the #6 parameter file is set to "0", the output current is not defined by the "ll" value but from the HB frequency (only when the "ll" value is more than "1150").

To start the procedure some prerequisite are necessary.

1. The board used do not have any hardware issue pending (especially on PFC and HB stage)
2. The serial line is available and working
3. The PFC is working and stable into all output power range
4. The PFC Vout is equal to the PFC Vout target
5. Vin is set to the fixed voltage (for example, 220Vac)
6. Vout is at nominal LED string voltage (maximum)
7. The load is a real LED
8. The HB #34, #35, #36 parameters is controlled and adapted to their HW
9. The HB control loop was disable (#6 = 0), is working and the Iout is stable (means, no hardware problem)

To check the HB output stage the suggested step are described below.

Figure 13: HB output power check procedure



DETAIL:

- The “ll 539” set the HB stage to force one HB switching cycles every 112 HB switching cycles, this drive the minimum Iout current
- The “ll 1100” set the HB stage to force all the 112 HB switching cycles, this force the maximum current for the “HBZ_SKIPK” mode
- The real minimum current is defined during the next fine-tuning procedure.

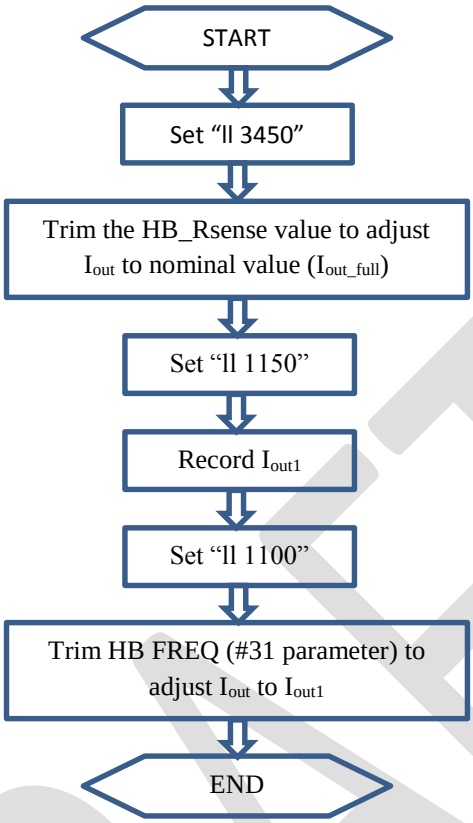
8.2 Fine-tuning the HB stage

To start the procedure a prerequisite is necessary.

1. The board used do not have any hardware issue pending (especially on PFC and HB stage)
2. The serial line is available and working
3. The PFC is working and stable into all output power range
4. The PFC Vout is equal to the PFC Vout target
5. Vin is set to the fixed voltage (for example, 220Vac)
6. Vout is at nominal LED string voltage (maximum)
7. The load is a real LED
8. The HB #34, #35, #36 parameters is controlled and adapted to their HW
9. The HB control loop was enable (#6 = 1), is working and the Iout is stable (means, no hardware problem)

After this prerequisite, the necessary step to fine tuning the HB stage as described by the next figure

Figure 14: HB fine tuning procedure



DETAIL:

- The “ll 3450” setup the CN_CNT control voltage to 1.054V, the Iout(nominal) is fixed to this voltage level.
- The “ll 1150” setup the minimum output control using the CN_CNT voltage on HBZ_CLOSE mode
- The “ll 1100” setup the maximum Iout using the HBZ_SKIPK mode

Please refer to the Chap *3.2* for the HB detail.

9 What to do when

9.1 Double the secondary output voltage

If the double output voltage is desired, a very simple change can be performed. Remove the SMD resistor R102 and put D42 and D43 on the PCB. D42 and D43 is a STTH3R06S (ultrafast rectifier - SMC case) diode. When the circuit is modify, the maximum output voltage is around 200V but the maximum output current is 500mA.

9.2 Input voltage range and PFC Vout

In the eval board, the input voltage range is 90Vac-265Vac. The PFC output voltage is 410Vdc. For 277Vac input, the max input voltage is 305Vac. Therefore, it is necessary to modify the PFC output voltage to 460Vdc.

From the schematic, the resistor network R46, R52, R55, and R58 provide voltage feedback to STLUX385A. The internal reference voltage Vref is 1.1151v.

We have: Vref=Vout*R58/(R58+R55+R52+R46). So we can change R58 from 18K to 16K to get 460V.

The OVP level will be changed as well. The internal OVP setting point is 1.23v. Therefore, with the change of R58, the OVP level is moved to 507V.

Also it is necessary to change MOSFET to 600V rating and increase bulk capacitor voltage up to 510V accordingly.

9.3 Output LED current

The output LED current can be calculated by the equation: ILED=Ipk*n/2

Where Ipk is the primary side peak current, n is the turn's ratio of the transformer. On the ILL066V1 demo board the max output current 1A is corresponding to "ll" command value "3450". The reference voltage Vref for the max current is 1.053v. The turn's ratio of the transformer is 1.85. Therefore, max LED current can be obtained: ILED=(Vref/Rcs)*1.85/2. Current sensing resistor Rcs is R36 and R37 in parallel. Because of propagation delay, the calculated value is lower than actual value. The resistor value needs to be fine-tuned in the circuit to get exact right LED current.

The maximum output current on the LEDs can be changed by modifying the value of the R36 // R37 For example, to change the output current to 1.28A at the same output voltage, R36//R37 can be changed to 0.90Ω value (or use two 1.8 Ω resistor). Similarly, if the max output LED current needs to be reduced, the value of R36//R37 can be increased. For detail, see *paragraph 6.5*.

9.4 Change the Vout on LEDs string

If the max output voltage needs to be different from 100v, try to select transformer turns ratio such that the reflected is same as the eval board. For example, if the maximum output voltage needs to be 42v, the new turn's ratio can be N=n*(100+Vd)/(42+Vd),

Where N is the new turns ratio, n is the old turns ratio 1.85; Vd is output diode voltage drop, 1v. We can calculate N=1.85*101/43=4.3.

9.5 Design of the circuit

A design spreadsheet is available upon request. Users can design the circuit according to their own requirements. It is a very helpful tool to design PFC choke, resonant inductor, and output transformer

10 Firmware download procedure

	High voltage is present on the ILL066V1 demo board.	

Users must take care about relevant safety procedure before handling the board, even before initiating a firmware upgrade.

The firmware upgrade procedure requires the "*STM8_pgm.exe*" program provided by Raisonance. The user shall install the Raisonance RIDE 7 tool. Please refer to www.raisonance.com. The "*STM8_pgm.exe*" program do not require licensing, only the use of R-link hardware, one PC with windows and the Raisonance RIDE7 tools plus the extension of STM8 microprocessor plus the STLUX patch files.

10.1 Install a new Firmware

The new firmware is normally provided into a zipped file containing:

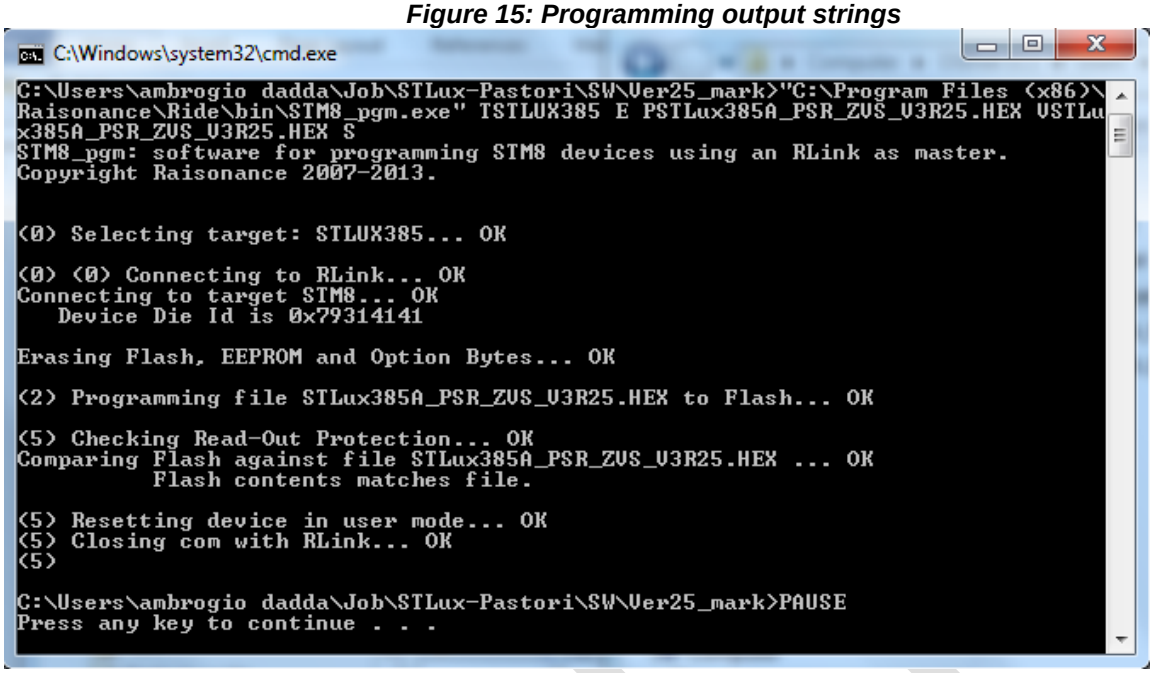
- firmware (STLUX385_PSR_ZVS_V3Rxx.hex file, where the final "xx" refers to the release number)
- Batch file (program.bat)
- SW release note document

The batch file instruments the "*STM8_pgm.exe*" program to erase and program the new firmware to the STLUX385A. The batch file also erases all the variable parameters to the default level. The user must take account of this action. The batch file also verifies that the download was successful.

The following steps shall be performed:

1- Modify the "*program.bat*" batch file to map the correct "*STM8_pgm.exe*" directory.
2- Use an isolated power supply and apply 14V (+/- 0,5V) via TP31 (+14V) and TP25 (GND) without any other power voltage (especially on AC power line). This condition guarantees the safest firmware upload condition for both the user and the board.
3- Power on the ILL066V1 demo board using the isolated power supply.
4- Connect the R-LINK hardware tool to the PC and after, connect the SWIM cable to the board (connect to J1 connector - SWIM IF). Make sure the pin 1 is correctly connected.
5- Use the "program.bat" procedure. The result of this action is show in the next picture:

Figure 15: Programming output strings



The user should verify that the output values for each step are marked with "OK", especially phase (2) "*programming*" and (5) "*checking*".

6- Connect the serial interface to the computer.

7- After a correct download and after a reset or power-on, the STLUX385A starts and shows the version/release of new firmware, as illustrated into next figure:

```
EVAL STLUX385-PSR-ZVS V3R25
Build Jul  1 2013 16:01:26 core is 385A
HALT - Debug Mode Activated
```

8- After a firmware upgrade, the application starts in safety mode. All the output PWMs are kept to low levels and both PFC and Half Bridge stages are disabled. This mode is identified by the string "HALT - Debug Mode Activated" shown on screen. Also the green status LED toggles (1 second on and 1 second off) on the green LED (RUN).

9- To enable the board operation, the users has to use the "*in*" command on the serial line, using the proper parameters as described into the command section chapter. The command used to enable the full functionality is "*in 7 0*". Only after a new power-on or a reset, the new configurations are applied and STEVAL-ILL066V1 demo board starts with full functionality and the startup message is changed into:

```
EVAL STLUX385-PSR-ZVS V3R25
Build Jul  1 2013 16:01:26 core is 385A
Option PFC HB Dali enabled
Ready
```

As an example, in the screen capture, the PFC is active, the Half Bridge is active and the DALI interfaces is enable. Note the "*Ready*" message; it states that the STEVAL-ILL066V1 board is ready to receive a new serial line command.

10.2 Duplicate Firmware and Configuration file

When the user wants to duplicate the firmware and the configuration file loaded into one board to make one exactly copy into one other board, there is a simple procedure described here. This procedure uses only the "*STM8_pgm.exe*" program described before and the R-LINK hardware tools. To use the "*STM8_pgm.exe*" program, the entire Raisonance environment must be installing on the PC.

NOTE: if the ROP code protection is enable ("1") this procedure is not allowed.

1- Use an isolated power supply and apply 14V (+/- 0,5V) via TP31 (+14V) and TP25 (GND) without any other power voltage (especially on AC power line). This condition guarantees the safest firmware upload condition for both the user and the board.

2- Power on the ILL066V1 demo board with14V.

3- Connect the R-LINK hardware tool to the PC and after, connect the SWIM cable to the board (connect to J1 connector - SWIM IF). Make sure the pin 1 is correctly connected.

4- Move into a dedicated directory, open the "*command line*" into this directory (using "cmd.exe") and write this command:

```
"C:\Program Files (x86)\Raisonance\Ride\bin\STM8_pgm.exe" TSTLUX385 DORIGINAL.HEX S
```

Adapt the path to your installation. This command create the "ORIGINAL.HEX" file where it is stored the EEPROM value (all the EEPROM value, also the DALI EEPROM area), the option byte of the STLUX and the CODE (from 0x8000 to the end of program area). This "ORIGINAL.HEX" file is the physical dump of all STLUX memory area.

5- The second step is to write the "ORIGINAL.HEX" file into a new board. This action erases all data (EEPROM, Option Byte and FLASH) into the STLUX mounted into this board. The command is showed into next figure write the code saved during first command into a new board. This command restores all data area; FLASH, Option Byte and code.

```
"C:\Program Files (x86)\Raisonance\Ride\bin\STM8_pgm.exe" TSTLUX385 E
PORIGINAL.HEX VORIGINAL.HEX S
```

6- It there is more than one board to duplicate the firmware loaded, simple change the board and repeat step "5" of this procedure.

NOTE: to understand the "*STM8_pgm.exe*" programs execute it without any option.

11 Revision history

DATE	REVISION	Change
February 11, 2015	0-0	Start-up - Initial release.
September 8, 2015	0-1	Update to V3R35

Table 4: Document revision history