ST Microelectronics

L647X Linux Library

User Manual

1 Introduction

This document presents the L647X (L6470/L6472) library for Linux operating system. This library is based on standard linux file system paradigms to access to the hardware. Thanks to this solution, the library can be used on other Linux based system.

This library has been developed on Raspberry Pi version B platform and has been tested on Raspberry B, B+, 2 and 3. The development has been realized using Eclipse environment, for Raspberry Pi, but it is applicable to other platforms.

This document does not cover the l647x chips operation. This is done for the L6470 in the datasheet: "DS6582: l6470™ fully integrated microstepping motor driver with motion engine and SPI" and for the L6472 in "DS8858: l6472™ fully integrated microstepping motor driver".

2 Library package

The library package contains two folders:

- A **"doc**" folder which contains this user manual.
- A "**motion_library**" folder which contains the files which are specific to the l647x library and its example projects. It is divided in two folders:
  - a "**binaries**" folder which contains the l647x library file and the test application, generated for Raspberry Pi platform.
  - a "**src**" folder which contains the Linux library source code.

2.1 Source code folder

The source code folder ("src") contains 8 files:

- **l647x_linux.c**: core functions of the l647x library
- **l647x_linux.h**: declaration of the l647x functions and the associated definitions
- **l647x_target_config.h**: predefines values for the l647x registers
- **l647x_RPi_host_config.h**: constants definitions for the Raspberry Pi platform: GPIOs, SPIs, …
- **gpio.c**: functions dedicated to GPIO management using the Linux file system paradigms.
- **gpio.h**: declaration of the GPIO management functions and of the associated definitions. (constants, variables, structures)
- **main.c**: Test source code to demonstrate library features, using SPI, available in l657x_linux.c
- **main_stepclock.c**: Test source code to demonstrate library features, using step clock, available in l657x_linux.c
- **Makefile**: Makefile used to generate the library and the test application.

3 Library features

The Linux l647x library has the following features:

- l647x registers read, write

- Linux file system configuration to facilitate access to GPIOs and SPI
- GPIO management (IRQs, step clock, ..)
- Motion commands
- BUSY and FLAG interrupts handling (alarms reporting)
- Microstepping handling
- Daisy chaining handling
- Step clock mode

By starting the library, the user specifies the number of l647x chips which are connected to the Linux platform. Once set, the number of l647x devices must not be changed.

Depending on the device number, the library will:

- Setup the required GPIOs to handle the FLAG interrupt,
- Start the SPI device to communicate with the l647x chips,
- Release the reset of each of the l647x chips,
- Disable the power bridge and clear the status flags of the l647x chips,
- Load the registers of each of the l647x with the predefined values from “l647x_target_config.h” or from a configuration file, depending of the method called.

Once the initialization is done, the user can modify the l647x registers as desired. Most of the functions of the library take a device Id (from 0 to 254) as input parameter. It gives the user the possibility to specify which of the devices configuration he wants to modify.

The user can also write callback functions and attach them to:

- the flag interrupt handler depending on the actions he wants to perform when an alarm is reported (read the flags, clear and read the flags…),
- the busy interrupt handler which is called each time the busy pin position is changed.

Then, he can request the move of one or several motors (till using the same principle of device Id). This request can be:

- to move for a given number of steps in a specified direction,
- to go to a specific position,
- to run till reception of a new instruction.

The speed profile is completely handled by the l647x. The motor starts moving by using the programmed minimum speed (set in MIN_SPEED register). At each step, the speed is increased using the acceleration value (ACC register).

If the target position is far enough, the motor will perform a trapezoidal move:

- accelerating phase using the acceleration value (ACC register),
- steady phase where the motor turns at constant speed (MAX_SPEED register),
- decelerating phase using the deceleration value (DEC register),
- stop at the targeted position.

Else, if the target position does not allow reaching the max speed, the motor will perform a triangular move:

- accelerating phase using the acceleration value,
- decelerating phase using the deceleration value,

- stop at the targeted position.

A moving command can be stopped at any moment:

- Either by a soft stop which progressively decreases the speed using the deceleration parameter. Once the minimum speed is reached, the motor is stopped.
- Or by a hard stop command which immediately stops the motor.

To avoid sending a new command to a device before the completion of the previous one, the library offers a L647X_WaitWhileActive() command which locks the program execution till the motor ends moving.

The library also offers the possibility to change the step mode (from full step till 1/128 microstep mode for L6470 or till 1/16 microstep mode for L6472) for a given device. When the step mode is change, the current position (ABS_POSITION register) is automatically reset but it is up to the user to update the speed profile (max and min speed, acceleration deceleration registers).

To limit the memory usage of the library, the maximum number of devices in the daisy chains is limited by default to 8 in file l647x_linux.h via the following line:

```
#define MAX_NUMBER_OF_DEVICES                    (8)
```

This number correspond to up to four stacked EVAL6470H-RPi board. It can be increased up to 255 without problem.

If the library supports both L6470 and L6472 chips, only one configuration is supported at a time. The choice of the enabled configuration is done by preprocessor option:

- Option `L6470` must be enabled for L6470 chips
- Option `L6472` must be enabled for L6472 chips

Selection of the chip is realized in Makefile. Once configured, the compilation is realized by the command (under Linux) :

➢ `make all`

4 Library required resources

The communication between the l647x and the Linux platform is mainly done through the SPI interface.

For the handling of the flag interrupt, the library uses only one external interrupt for all devices as all flag pins are connected together.

This is the same for the busy interrupt handling: only one external interrupt is used for all devices.

For the step clock mode, the generation of the step clock via a PWM requires a supplementary GPIO.

The SPI device name and GPIO configurations are done in a specific include file per platform. For the Raspberry Pi platform, this include file is l647x_RPi_host_config.h

5 Library Interface functions

The functions of the library are presented below.

5.1 L647X_Begin ()

void ***L647X_Begin*** *(* *uint8_t* ***nbDevices*** *)*

Description
Starts the l647x library.

The l647x registers will be set to the predefined values from l647x_target_config.h

Parameters
[in] **nbDevices** Number of l647x devices to use (from 1 to MAX_NUMBER_OF_DEVICES)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 3 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(3);


  ...
}
```

5.2 L647X_Close ()

*void **L647X_Close** (void)*

Description
Stop the l647x library.

The platform devices (SPI, GPIO) will be closed and memory will be released to leave the system in a clean state.

Parameters
None

Return values
None

Example
```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 3 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(3);

  ...

  L647X_Close();
}
```

5.3 L647X_LoadMotorConfigValuesFromFile ()

void ***L647X_LoadMotorConfigValuesFromFile*** *(FILE * **ptr**)*

Description
Load l647x parameters from a text file.

This method allow configuration of l647x dynamically.

Parameters
[in] **ptr** Pointer on the configuration file. This file has a text format.

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  FILE *pMotorConfigFile = NULL;

  /* Start the l647x library to use 3 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(3);

   /* Open motor configuration file is present */
   pMotorConfigFile = fopen("/home/pi/motor_config.txt", "r");
   if( NULL != pMotorConfigFile)
       L647X_LoadMotorConfigValuesFromFile( pMotorConfigFile);
...

  L647X_Close();
}
```

5.4 L647X_SetMotorConfigToPredefinedValues ()

*void **L647X_SetMotorConfigToPredefinedValues**(void)*

Description
Set l647x parameters to predefined values.

The predifined values are configured in file l647x_target_config.h

Parameters
[in] **ptr** Pointer on the configuration file. This file has a text format.

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  FILE *pMotorConfigFile = NULL;

  /* Start the l647x library to use 3 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(3);

   /* Open motor configuration file is present */
   pMotorConfigFile = fopen("/home/pi/motor_config.txt", "r");
   if( NULL != pMotorConfigFile)
       L647X_LoadMotorConfigValuesFromFile( pMotorConfigFile);
...

  L647X_Close();
}
```

5.5 L647X_AttachBusyInterrupt ()

void ***L647X_AttachBusyInterrupt****(void(*)(void)* ***callback*** *)*

Description
Attaches a user callback to the busy Interrupt. The call back will be then called each time the busy pin is set or reset.

Parameters
[in] **callback** Name of the callback to attach to the busy interrupt

Return values
None

Example

```
#include "l647x_linux.h"

static void MyBusyInterruptHandler(void);

int main(void)
{

  /* Start the l647x library to use 2 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(2);


  /* Attach the function MyBusyHandler (defined below) to the busy
interrupt */
  L647X_AttachBusyInterrupt(&MyBusyInterruptHandler);
...
}

/**
  * @brief  This function is the user handler for the busy interrupt
  * @param  None
  * @retval None
  */
static void MyBusyInterruptHandler(void)
{
   bool L647X_IsDeviceBusy(uint8_t deviceId)
}
```

5.6 L647X_AttachFlagInterrupt ()

*void **L647X_AttachFlagInterrupt** (void(*)(void) **callback**)*

Description

Attaches a user callback to the flag Interrupt. The call back will be then called each time the status flag pin will be pulled down due to the occurrence of a programmed alarms (OCD, thermal pre-warning or shutdown, UVLO, wrong command, non-performable command).

Parameters

[in] **callback** Name of the callback to attach to the Flag Interrupt

Return values

None

Example

```
#include "l647x_linux.h"

static void MyFlagInterruptHandler (void);

int main(void)
{
  /* Start the l647x library to use 2 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(2);

  /* Attach the function MyFlagInterruptHandler (defined below) to the
flag interrupt */
  L647X_AttachFlagInterrupt(&MyFlagInterruptHandler);
…
}

/**
  * @brief  This function is the User handler for the flag interrupt
  * @param  None
  * @retval None
  */
void MyFlagInterruptHandler(void)
{
  /* Get the value of the status register via the l647x command
GET_STATUS */
  uint16_t statusRegister0 = L647X_CmdGetStatus(0);
  uint16_t statusRegister0 = L647X_CmdGetStatus(1);
  …
}
```

5.7 L647X_CheckBusyHw ()

*uint8_t **L647X_CheckBusyHw** (void)*

Description
Checks if at least one l647x is busy by checking busy pin position. The busy pin is shared between all devices.

Parameters
None

Return values
One if at least one l647x is busy, otherwise **zero**

Example

```
#include "l647x_linux.h"


static void MyBusyInterruptHandler(void);

int main(void)
{
  /* Start the l647x library to use 2 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(2);

  /* Attach the function MyBusyHandler (defined below) to the busy
interrupt */
  L647X_AttachBusyInterrupt(MyBusyInterruptHandler);
...
}

/**
  * @brief  This function is the user handler for the busy interrupt
  * @param  None
  * @retval None
  */
static void MyBusyInterruptHandler(void)
{
   if (L647X_CheckBusyHw())
   {
     /* Busy pin is low, so at list one L647x chip is busy */
     /* To be customized (for example Switch on a LED) */
   }
   else
   {
     /* To be customized (for example Switch off a LED) */
   }
```

```
}
```

5.8 L647X_CheckStatusHw ()

*uint8_t **L647X_CheckStatusHw** (void)*

Description
Checks if at least one l647x has an alarm flag set by reading flag pin position.
The flag pin is shared between all devices.

Parameters
None

Return values
One if at least one l647x has an alarm flag set , otherwise **zero**

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(1);

  /* Move device 0 of 16000 steps in the FORWARD direction*/
  L647X_CmdMove(0, FORWARD, 16000);

  /* Wait for the motor of device 0 ends moving */
  L647X_WaitWhileActive(0);

  /* Wait for 2 seconds */
  HAL_Delay(2000);

  /* Request device 0 to come back to home if no alarm was set */
  if (L647X_CheckStatusHw() == 0)
  {
      L647X_CmdGoHome(0);
   }

   ...
}
```

5.9 L647X_CmdGetParam ()

uint32_t ***L647X_CmdGetParam*** *(* *uint8_t* ***deviceId****,*
l647x_Registers_t ***param***
)

Description
Issues a l647x Get Param command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
[in] **param** Register address of the l647x from *l647x_Registers_t* enum:

For L6470:

```
typedef enum {
        L647X_ABS_POS           =((uint8_t)0x01),
        L647X_EL_POS            =((uint8_t)0x02),
        L647X_MARK              =((uint8_t)0x03),
        L647X_SPEED             =((uint8_t)0x04),
        L647X_ACC               =((uint8_t)0x05),
        L647X_DEC               =((uint8_t)0x06),
        L647X_MAX_SPEED         =((uint8_t)0x07),
        L647X_MIN_SPEED         =((uint8_t)0x08),
        L647X_FS_SPD            =((uint8_t)0x15),
        L647X_KVAL_HOLD         =((uint8_t)0x09),
        L647X_KVAL_RUN          =((uint8_t)0x0A),
        L647X_KVAL_ACC          =((uint8_t)0x0B),
        L647X_KVAL_DEC          =((uint8_t)0x0C),
        L647X_INT_SPD           =((uint8_t)0x0D),
        L647X_ST_SLP            =((uint8_t)0x0E),
        L647X_FN_SLP_ACC        =((uint8_t)0x0F),
        L647X_FN_SLP_DEC        =((uint8_t)0x10),
        L647X_K_THERM           =((uint8_t)0x11),
        L647X_ADC_OUT           =((uint8_t)0x12),
        L647X_OCD_TH            =((uint8_t)0x13),
        L647X_STALL_TH          =((uint8_t)0x14),
        L647X_STEP_MODE         =((uint8_t)0x16),
        L647X_ALARM_EN          =((uint8_t)0x17),
        L647X_CONFIG            =((uint8_t)0x18),
        L647X_STATUS            =((uint8_t)0x19),
        L647X_RESERVED_REG2     =((uint8_t)0x1A),
```

```
                    L647X_RESERVED_REG1         =((uint8_t)0x1B)
                } l647x_Registers_t;
```

For L6472:

```
typedef enum {
                    L647X_ABS_POS               =((uint8_t)0x01),
                    L647X_EL_POS                =((uint8_t)0x02),
                    L647X_MARK                  =((uint8_t)0x03),
                    L647X_SPEED                 =((uint8_t)0x04),
                    L647X_ACC                   =((uint8_t)0x05),
                    L647X_DEC                   =((uint8_t)0x06),
                    L647X_MAX_SPEED             =((uint8_t)0x07),
                    L647X_MIN_SPEED             =((uint8_t)0x08),
                    L647X_FS_SPD                =((uint8_t)0x15),
                    L647X_TVAL_HOLD             =((uint8_t)0x09),
                    L647X_TVAL_RUN              =((uint8_t)0x0A),
                    L647X_TVAL_ACC              =((uint8_t)0x0B),
                    L647X_TVAL_DEC              =((uint8_t)0x0C),
                    L647X_RESERVED_REG5         =((uint8_t)0x0D),
                    L647X_T_FAST                =((uint8_t)0x0E),
                    L647X_TON_MIN               =((uint8_t)0x0F),
                    L647X_TOFF_MIN              =((uint8_t)0x10),
                    L647X_RESERVED_REG4         =((uint8_t)0x11)
                    L647X_ADC_OUT               =((uint8_t)0x12),
                    L647X_OCD_TH                =((uint8_t)0x13),
                    L647X_RESERVED_REG3         =((uint8_t)0x14),
                    L647X_STEP_MODE             =((uint8_t)0x16),
                    L647X_ALARM_EN              =((uint8_t)0x17),
                    L647X_CONFIG                =((uint8_t)0x18),
                    L647X_STATUS                =((uint8_t)0x19),
                    L647X_RESERVED_REG2         =((uint8_t)0x1A),
                    L647X_RESERVED_REG1         =((uint8_t)0x1B)
                } l647x_Registers_t;
```

Return values

Register value

Example

```c
#include "l647x_linux.h"

int main(void)
{
  uint32_t registerValue;

  /* Start the l647x library to use 1 device */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(1);

  /* Read EL_POS register of device 0*/
  registerValue = L647X_CmdGetParam (0, L647X_EL_POS);
…
}
```

5.10 L647X_CmdGetStatus ()

uint16_t ***L647X_CmdGetStatus*** *(* *uint8_t* ***deviceId*** *)*

Description
Issues a l647x Get Status command to the specified device

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
Status value of the STATUS register

Note
Once the GetStatus command is performed, the flags of the status register are reset. This is not the case when the status register is read with the GetParam command (via the function **L647X_CmdGetParam**).

Example

```
#include "l647x_linux.h"

static void MyFlagInterruptHandler (void);

int main(void)
{
  /* Start the l647x library to use 2 devices */
  /* The l647x registers are set with the predefined values */
  /* from file l647x_target_config.h */
  L647X_Begin(2);


  /* Attach the function MyFlagInterruptHandler (defined below) to the
flag interrupt */
  L647X_AttachFlagInterrupt(MyFlagInterruptHandler);
…
}

/**
  * @brief  This function is the User handler for the flag interrupt
  * @param  None
  * @retval None
  */
void MyFlagInterruptHandler(void)
{
  /* Get the value of the status register via the l647x command
GET_STATUS */
  uint16_t statusRegister = L647X_CmdGetStatus(0);
```

```
/* Then test all flags of the status register */
  /* Please note that no action is performed -> to be customized */

  /* Check HIZ flag: if set, power brigdes are disabled */
  if ((statusRegister & L647X_STATUS_HIZ) == L647X_STATUS_HIZ)
  {
    // HIZ state
  }

  /* Check BUSY flag: if not set, a command is under execution */
  if ((statusRegister & L647X_STATUS_BUSY) == 0)
  {
    // BUSY
  }

  /* Check SW_F flag: if not set, the SW input is opened */
  if ((statusRegister & L647X_STATUS_SW_F ) == 0)
  {
     // SW OPEN
  }
  else
  {
    // SW CLOSED
  }
  /* Check SW_EN bit */
  if ((statusRegister & L647X_STATUS_SW_EVN) == L647X_STATUS_SW_EVN)
  {
    // switch turn_on event
  }
  /* Check direction bit */
  if ((statusRegister & L647X_STATUS_DIR) == 0)
  {
    // BACKWARD
  }
  else
  {
    // FORWARD
  }
  if ((statusRegister & L647X_STATUS_MOT_STATUS) ==
L647X_STATUS_MOT_STATUS_STOPPED )
  {
       // MOTOR STOPPED
  }
  else  if ((statusRegister & L647X_STATUS_MOT_STATUS) ==
L647X_STATUS_MOT_STATUS_ACCELERATION )
  {
           // MOTOR ACCELERATION
  }
```

```
  else  if ((statusRegister & L647X_STATUS_MOT_STATUS) ==
L647X_STATUS_MOT_STATUS_DECELERATION )
  {
           // MOTOR DECELERATION
  }
  else  if ((statusRegister & L647X_STATUS_MOT_STATUS) ==
L647X_STATUS_MOT_STATUS_CONST_SPD )
  {
       // MOTOR RUNNING AT CONSTANT SPEED
  }

  /* Check Not Performed Command flag: if set, the command received by
SPI can't be performed */
  /* This often occures when a command is sent to the l647x */
  /* while it is in HIZ state */
  if ((statusRegister & L647X_STATUS_NOTPERF_CMD) ==
L647X_STATUS_NOTPERF_CMD)
  {
       // Command can't be performed
  }

  /* Check Wrong Command Error flag: if set, the command does not exist
*/
    if ((statusRegister & L647X_STATUS_WRONG_CMD) ==
L647X_STATUS_WRONG_CMD)
  {
       // Command does not exist
  }

  /* Check UVLO flag: if not set, there is an undervoltage lock-out */
  if ((statusRegister & L647X_STATUS_UVLO) == 0)
  {
     //undervoltage lock-out
  }


  /* Check thermal warning flags: if  not set, there is a thermal
warning */
  if ((statusRegister & L647X_STATUS_TH_WRN) != 0)
  {
    //thermal warning
  }

  /* Check thermal shutdown flags: if  not set, there is a thermal
shutdown */
  if ((statusRegister & L647X_STATUS_TH_SD) != 0)
  {
    //thermal shutdown
```

```
  }

  /* Check OCD  flag: if not set, there is an overcurrent detection */
  if ((statusRegister & L647X_STATUS_OCD) == 0)
  {
    //overcurrent detection
  }
#ifdef L6470
  /* Check Step Loss A flag: if not set, there is a Stall condition on
bridge A */
  if ((statusRegister & L647X_STATUS_STEP_LOSS_A) == 0)
  {
    //stall detected on bridge A
  }

  /* Check Step Loss B flag: if not set, there is a Stall condition on
bridge B */
  if ((statusRegister & L647X_STATUS_STEP_LOSS_B) == 0)
  {
    //stall detected on bridge B
  }
#endif
  /* Check Step Clock Mode flag: if  set, the device is working in step
clock mode */
  if ((statusRegister & L647X_STATUS_SCK_MOD) == L647X_STATUS_SCK_MOD)
  {
    //step clock mode is enabled
  }
}
```

5.11 L647X_CmdGoHome ()

void ***L647X_CmdGoHome*** *(uint8_t **deviceId**)*

Description
Issues a L647x Go Home command (Shorted path to zero position)

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 devices */
  L647X_Begin(2);

  /* Set home position for device 1*/
  L647X_CmdResetPos(1);

  /* Request device 1 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(1, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request device 1 to make a soft stop */
  L647X_CmdSoftStop(1);

  /* Wait for device 1 end moving */
  L647X_WaitWhileActive(1);

  /* Request device 1 to go to home */
  L647X_CmdGoHome(1);

  /* Wait for device 1 end moving */
  L647X_WaitWhileActive(1);


  …
}
```

5.12 L647X_CmdGoMark ()

void ***L647X_CmdGoMark*** *(* *uint8_t* ***deviceId*** *)*

Description
Issues a l647x Go Mark command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to go to position 200 */
  L647X_CmdGoTo(0, 200);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

  /* Set current position to become the Mark position of device 0*/
  L647X_SetMark(0);

  /* Request device 0 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(0, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request device 0 to make a soft stop */
  L647X_CmdSoftStop(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

  /* Request device 0 to come back to its Mark position */
  L647X_CmdGoMark(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);
```

```
  …
}
```

5.13 L647X_CmdGoTo ()

void ***L647X_CmdGoTo*** *(* *uint8_t* ***deviceId,***
int32_t ***abs_pos***
)

Description
Issues a L647x Go To command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
[in] **targetPosition** absolute position in steps

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to go to position -200 */
  L647X_CmdGoTo(0, -200);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);
…
}
```

5.14 L647X_CmdGoToDir ()

```
void L647X_CmdGoToDir  (      uint8_t         deviceId,
                              l647x_Direction_t direction,
                              int32_t         abs_pos
                       )
```

Description
Issues a L647x Go To DIR command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
[in] **direction** movement direction (FORWARD or BACKWARD)
[in] ***abs_pos*** absolute position in steps

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

 /* Request device 0 to go to position 200 in the backward direction */
  L647X_CmdGoToDir(0, BACKWARD , 200);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);
…
}
```

5.15 L647X_CmdGoUntil ()

```
void L647X_CmdGoUntil   (       uint8_t         deviceId,
                                l647x_Action_t  action,
                                l647x_Direction_t direction,
                                uint32_t        speed
                        )
```

Description
Issues a L647x Go Until command to the specified device.

Parameters
- [in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
- [in] **action** action to be performed once the targeted position is reached:
  ACTION_RESET to reset the ABS_POS
  ACTION_COPY to copy ABS_POS to MARK register
- [in] **direction** movement direction (FORWARD or BACKWARD)
- [in] ***speed*** speed

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to move at 400 steps/s in the backward direction
  /* till an external turn-on event occurs */
  /* When this happens, the position is saved into the MARK register */
  L647X_CmdGoUntil(0, ACTION_COPY , BACKWARD , Speed_Steps_to_Par(400);

  /* Wait for device 0 end moving (Need turn-on event) */
  L647X_WaitWhileActive(0);

  /* Request device 0 to go home */
  L647X_CmdGoHome(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

  /* Request device 0 to go to Mark position which was saved via the GO
  Until command */
  L647X_CmdGoMark(0);

  /* Wait for device 0 end moving */
```

```
  L647X_WaitWhileActive(0);

…
}
```

5.16 L647X_CmdHardHiZ ()

void ***L647X_CmdHardHiZ*** *(* *uint8_t* ***deviceId*** *)*

Description
Issues a L647x Hard HiZ command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 1 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(1, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Immediately stop the device 0 and put it in HiZ state */
  L647X_CmdHardHiZ(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

…
}
```

5.17 L647X_CmdHardStop ()

void ***L647X_CmdHardStop*** *(uint8_t **deviceId**)*

Description
Issues a L647x Hard Stop command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 1 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(1, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Immediately stop the device 0 */
  L647X_CmdHardStop(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

…
}
```

5.18 L647X_CmdMove ()

void ***L647X_CmdMove*** *(uint8_t* ***deviceId,***
l647x_Direction_t ***direction****,*
uint32_t ***n_step****)*

Description
Issues a l647x Move command

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  /* Start the l647x library to use 3 devices */
  L647X_Begin(3);

  /* Request device 1 to move of 16000 steps in the FORWARD direction*/
  L647X_CmdMove(1, FORWARD, 16000);

  /* Wait for device 1 end moving */
  L647X_WaitWhileActive(1);

  /* Request device 2 to move of 6000 steps in the BACKWARD direction*/
  L647X_CmdMove(2, BACKWARD, 6000);

  /* Wait for device 2 end moving */
  L647X_WaitWhileActive(2);

 /* Request device 0 to move of 8000 steps in the FORWARD direction*/
  L647X_CmdMove(0, FORWARD, 8000);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);


…
}
```

5.19 L647X_CmdNop ()

void ***L647X_CmdNop*** *(* *uint8_t* ***deviceId*** *)*

Description
Issues a L647x Nop command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Issue a Nop comman to device 0 */
  L647X_CmdNop(0);
…
}
```

5.20 L647X_CmdReleaseSw ()

void ***L647X_CmdReleaseSw*** *(uint8_t* ***deviceId,***
l647x_Action_t ***action,***
l647x_Direction_t ***direction)***

Description
Issues a L647x Release SW command.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
[in] **action** action to be performed once the targeted position is reached:
`ACTION_RESET` to reset the ABS_POS
`ACTION_COPY` to copy ABS_POS to MARK register
[in] **direction** movement direction (`FORWARD` or `BACKWARD`)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to move at minimum speed till SW is released */
  /* then ABS_POS is reset */
  L647X_CmdReleaseSw (0, ACTION_RESET, FORWARD);
…
}
```

5.21 L647X_CmdResetDevice ()

void ***L647X_CmdResetDevice*** *(* *uint8_t* ***deviceId****)*

Description
Issues a L647x Reset Device command.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Reset device 0 to power up conditions*/
  L647X_CmdResetDevice (0);
…
}
```

5.22 L647X_CmdResetPos ()

void ***L647X_CmdResetPos*** *(* *uint8_t* ***deviceId****)*

Description
Issues a L647x Reset Pos command.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Reset ABS_POS of device 0 to 0 (set home position) */
  L647X_CmdResetPos (0);
…
}
```

5.23 L647X_CmdRun ()

void ***L647X_CmdRun*** *(* *uint8_t* ***deviceId,***
l647x_Direction_t ***direction,***
uint32_t ***speed****)*

Description
Issues a l647x Move command

Parameters

[in]	**deviceId**	Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
[in]	**direction**	movement direction (FORWARD or BACKWARD)
[in]	**speed**	speed

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 3 devices */
  L647X_Begin(3);

  /* Request device 0 to run BACKWARD at 400 step/s*/
  L647X_CmdRun(0, BACKWARD,Speed_Steps_to_Par(400));

  /* Request device 1 to run FORWARD at 400 step/s*/
  L647X_CmdRun(1,FORWARD,Speed_Steps_to_Par(400));

  /* Request device 2 to run FORWARD at 200 step/s*/
  L647X_CmdRun(2,FORWARD,Speed_Steps_to_Par(200));

  /* Wait for 5 seconds */
  HAL_Delay(5000);
…
}
```

5.24 L647X_CmdSetParam()

void ***L647X_CmdSetParam*** *(uint8_t* ***deviceId****,*
l647x_Registers_t ***param****,*
uint32_t ***value****)*

Description
Issues the SetParam command to the l647x of the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

 /* Set ALARM_EN register of device 0 to predefined value of device 1*/
 L647X_CmdSetParam(0,
                   L647X_ALARM_EN,
                   L647X_CONF_PARAM_ALARM_EN_DEVICE_1);
…
}
```

5.25 L647X_CmdSoftHiZ ()

void ***L647X_CmdSoftHiZ*** *(* *uint8_t* ***deviceId*** *)*

Description
Issues a L647x Soft HiZ command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 1 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(0, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request a soft stop of the device 0 and put it in HiZ state */
  L647X_CmdSoftHiZ (0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

…
}
```

5.26 L647X_CmdSoftStop ()

void ***L647X_CmdSoftStop*** *(uint8_t **deviceId**)*

Description
Issues a L647x Soft Stop command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(0, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request a soft stop of the device 0 */
  L647X_CmdSoftStop(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

…
}
```

5.27 L647X_CmdStepClock ()

void ***L647X_CmdStepClock*** *(* *uint8_t* ***deviceId*** *)*

Description
Issues a L647x Step Clock command to the specified device.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to switch in step clock mode */
  L647X_CmdStepClock(0);

   …
}
```

5.28 L647X_FetchAndClearAllStatus ()

*void **L647X_FetchAndClearAllStatus** (void)*

Description
Fetches and clear status flags of all devices by issuing a GET_STATUS command simultaneously to all devices. It can be used alone to clear the status of several devices which are connected in daisy chain. Or it can be used to simultaneously get the status of several devices in association with function ***L647X_GetFetchedStatus()***

Parameters
None

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{

  uint16_t status;

  /* Start the l647x library to use 3 devices */
  L647X_Begin(3);

  /* Request device 0 to run BACKWARD at 400 step/s*/
  L647X_CmdRun(0, BACKWARD,Speed_Steps_to_Par(400));

  /* Request device 1 to run FORWARD at 400 step/s*/
  L647X_CmdRun(1,FORWARD,Speed_Steps_to_Par(400));

  /* Request device 2 to run FORWARD at 200 step/s*/
  L647X_CmdRun(2,FORWARD,Speed_Steps_to_Par(200));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request a soft stop of the device 0 */
  L647X_CmdSoftStop(0);

  /* Request a hard stop of the device 1 */
  L647X_CmdHardStop(1);

  /* Request a soft HiZ stop of the device 2 */
  L647X_CmdSoftHiZ (2);
  /* Wait for all devices end moving */
```

```
  L647X_WaitForAllDevicesNotBusy();

  /* Fetch and clear status of all devices */
  L647X_FetchAndClearAllStatus();

  /* Get fetched status of device 1 */
  status = L647X_GetFetchedStatus(1);

  …

}
```

5.29 L647X_GetFetchedStatus ()

Uint16_t ***L647X_GetFetchedStatus*** *(* *uint8_t* ***deviceId*** *)*

Description
Get the value of the STATUS register which was fetched by using

L647X_FetchAndClearAllStatus
The fetched values remain available as long as there is no other call to functions which use the SPI.

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
Status Last fetched value of the STATUS register

Example

```
#include "l647x_linux.h"

int main(void)
{

  uint16_t status;

  /* Start the l647x library to use 3 devices */
  L647X_Begin(3);

  /* Request device 0 to run BACKWARD at 400 step/s*/
  L647X_CmdRun(0, BACKWARD,Speed_Steps_to_Par(400));

  /* Request device 1 to run FORWARD at 400 step/s*/
  L647X_CmdRun(1,FORWARD,Speed_Steps_to_Par(400));

  /* Request device 2 to run FORWARD at 200 step/s*/
  L647X_CmdRun(2,FORWARD,Speed_Steps_to_Par(200));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request a soft stop of the device 0 */
  L647X_CmdSoftStop(0);

  /* Request a hard stop of the device 1 */
  L647X_CmdHardStop(1);

  /* Request a soft HiZ stop of the device 2 */
  L647X_CmdSoftHiZ (2);
  /* Wait for all devices end moving */
```

```
  L647X_WaitForAllDevicesNotBusy();

  /* Fetch and clear status of all devices */
  L647X_FetchAndClearAllStatus();

  /* Get fetched status of device 1 */
  status = L647X_GetFetchedStatus(1);

  …

}
```

5.30 L647X_GetFwVersion ()

Uint8_t ***L647X_GetFwVersion*** *(void)*

Description
Returns the FW version of the library

Parameters
None

Return values
L647X_FW_VERSION Fw version of the library

Example

```
#include "l647x_linux.h"


int main(void)
{

  uint16_t version;

 /*  Get the FW version of the library */
  version = L647X_GetFwVersion();

  …

}
```

5.31 L647X_GetMark ()

int32_t ***L647X_GetMark*** *(* *uint8_t* ***deviceId*** *)*

Description
Returns the mark position of the specified device

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
Mark register value converted in a 32b signed integer

Example

```
#include "l647x_linux.h"


int main(void)
{
  int32_t pos;

  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to go to position -200 */
  L647X_CmdGoTo(0, -200);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

  /* Set current position to become the Mark position of device 0*/
  L647X_SetMark(0);

 /* Get mark position of deveice 0*/
  pos  = L647X_GetMark(0);

…
}
```

5.32 L647X_GetNbDevices ()

uint8_t ***L647X_GetNbDevices*** *(void)*

Description
Returns the number of devices in the daisy chain.

Parameters
None

Return values
numberOfDevices number of devices from 1 to MAX_NUMBER_OF_DEVICES

Example

```
#include "l647x_linux.h"


int main(void)
{
  int32_t nbDevices;

  /* Start the l647x library to use 1 device */
  L647X_Begin(3);

  /* Request the number of devices in the daisy chain */
  nbDevices = L647X_GetNbDevices();
…

}
```

5.33 L647X_GetPosition ()

int32_t ***L647X_GetPosition*** *(* *uint8_t* ***deviceId*** *)*

Description
Returns the ABS_POSITION of the specified device.

Parameters
None

Return values
Position ABS_POSITION register value converted in a 32b signed integer

Example

```
#include "l647x_linux.h"


int main(void)
{
  int32_t nbDevices;

  /* Start the l647x library to use 1 device */
  L647X_Begin(3);

  /* Request the number of devices in the daisy chain */
  nbDevices = L647X_GetNbDevices();
…

}
```

5.34 L647X_IsDeviceBusy ()

bool ***L647X_IsDeviceBusy*** *(* *uint8_t* ***deviceId*** *)*

Description
Checks if the specified device is busy by reading the Busy flag bit of its status Register

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values
True if busy, else **False**

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 1 to move of 16000 steps in the FORWARD direction*/
  L647X_CmdMove(1, FORWARD, 16000);

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Check if device 0 is still busy */
  if (L647X_IsDeviceBusy(0))
  {
     /* If still busy, request an hard stop of the device 0 */
     L647X_CmdHardStop(0);
  }

…

}
```

5.35 L647X_QueueCommands ()

void ***L647X_QueueCommands***	*(*	*uint8_t*	***deviceId****,*
		uint8_t	***param****,*
		uint32_t	***value****)*

Description

Puts commands in queue before synchronous sending done by calling **L647X_SendQueuedCommands**.

Any call to functions that use the SPI between the calls of **L647X_QueueCommands** and **L647X_SendQueuedCommands** will corrupt the queue.

A command for each device of the daisy chain must be specified before calling **L647X_SendQueuedCommands**.

With **L647X_SendQueuedCommands**, this function allows a simultaneous commands sending to several devices.

Parameters

[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

[in] **param** Command to queue (all l647x commmands from **l647x_Commands_t** except L647X_SET_PARAM, L647X_GET_PARAM, L647X_GET_STATUS)

[in] **value** argument of the command to queue

Return values

None

Example

```
#include "l647x_linux.h"


int main(void)
{
  uint32_t loop;

  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Queue command "Move by 60000 steps in reverse direction" for
device 0 */
  L647X_QueueCommands (0, (uint8_t) L647X_MOVE |(uint8_t) BACKWARD,
60000);

  /* Queue comamnd "Run at 400 steps/s in forward direction" for device
1 */
  L647X_QueueCommands (1, (uint8_t) L647X_RUN |(uint8_t) FORWARD,
Speed_Steps_to_Par(400));

  /* No operation for other devices */
  for (loop = 2; loop <= L647X_GetNbDevices();  loop++)
  {
    L647X_QueueCommands (loop, L647X_NOP, 0);
```

```
  }

  /* Issue simultaneously the queued commands */
  L647X_SendQueuedCommands();

  /* Wait for all devices end moving */
  L647X_WaitForAllDevicesNotBusy();

…
}
```

5.36 L647X_ReleaseReset ()

*void **L647X_ReleaseReset** (void)*

Description
Releases the l647x reset of all devices (reset pin set to high).

Parameters
None

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 2 devices */
  L647X_Begin(2);

  /* Release l647x reset of all devices */
  L647X_ReleaseReset();
…
}
```

5.37 L647X_Reset ()

*void **L647X_Reset** (void)*

Description
Resets the l647x of all devices (reset pin set to low).

Parameters
None

Return values
None

Example
```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 2 devices */
  L647X_Begin(2);

  /* Reset the l647x of all devices */
  L647X_Reset ();
…
}
```

5.38 L647X_SelectStepMode ()

void ***L647X_SelectStepMode*** *(* *uint8_t* ***deviceId***, *l647x_StepSel_t* ***stepMod*** *)*

Description
Sets the stepping mode

Parameters
[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)
[in] **stepMod** **for L6470**, from full step to 1/128 microstep as specified in enum

```
typedef enum {
      L647X_STEP_SEL_1      = ((uint8_t)0x00), //full step
      L647X_STEP_SEL_1_2    = ((uint8_t)0x01), //half step
      L647X_STEP_SEL_1_4    = ((uint8_t)0x02), //1/4 microstep
      L647X_STEP_SEL_1_8    = ((uint8_t)0x03), //1/8 microstep
      L647X_STEP_SEL_1_16   = ((uint8_t)0x04), //1/16 microstep
      L647X_STEP_SEL_1_32   = ((uint8_t)0x05), //1/32 microstep
      L647X_STEP_SEL_1_64   = ((uint8_t)0x06), //1/64 microstep
      L647X_STEP_SEL_1_128  = ((uint8_t)0x07)  //1/128 microstep
                } l647x_StepSel_t;
```

for L6472, from full step to 1/16 microstep as specified in enum

```
typedef enum {
      L647X_STEP_SEL_1      = ((uint8_t)0x00), //full step
      L647X_STEP_SEL_1_2    = ((uint8_t)0x01), //half step
      L647X_STEP_SEL_1_4    = ((uint8_t)0x02), //1/4 microstep
      L647X_STEP_SEL_1_8    = ((uint8_t)0x03), //1/8 microstep
      L647X_STEP_SEL_1_16   = ((uint8_t)0x04), //1/16 microstep
                } l647x_StepSel_t;
```

Return values
None

Note
The devices parameters (acceleration, deceleration, min and max speed) have to be adapted to the new stepping mode.
The ABS_POS register is automatically reset to 0 when changing the stepping mode.
The MARK register value becomes inconsistent.

Example

```
#include "l647x_linux.h"


int main(void)
```

```
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Select full step mode for device 0 */
  L647X_SelectStepMode(0,L647X_STEP_SEL_1);

  /* Set speed  to be consistent with full step mode */
  L647X_CmdSetParam(0,L647X_MAX_SPEED, MaxSpd_Steps_to_Par(50));

  /* Request device 0 to go position 200 */
  L647X_CmdGoTo(0,200);

  /* Wait for the motor of device 0 ends moving */
  L647X_WaitWhileActive(0);

  /* Get current position */
  pos = L647X_GetPosition(0);

  /* Wait for 2 seconds */
  HAL_Delay(2000);

  /* Reset device 0 to its initial  microstepping mode */
  L647X_SelectStepMode(0,L647X_CONF_PARAM_STEP_MODE_DEVICE_0);

  /* Update speed, acceleration, deceleration for initial microstepping
mode*/
  L647X_CmdSetParam(0,L647X_MAX_SPEED,

  MaxSpd_Steps_to_Par(L647X_CONF_PARAM_MAX_SPEED_DEVICE_0));


…
}
```

5.39 L647X_SendQueuedCommands ()

void L647X_SendQueuedCommands (void)

Description

Sends commands which were queued by a previous call to function **L647X_QueueCommands**.

Any call to functions that use the SPI between the calls of **L647X_QueueCommands** and **L647X_SendQueuedCommands** will corrupt the queue.

With **L647X_ QueueCommands**, this function allows a simultaneous commands sending to

several devices.

Parameters
None

Return values
None

Example

```
#include "l647x_linux.h"

int main(void)
{
  uint32_t loop;

  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Queue command "Move by 60000 steps in reverse direction" for
device 0 */
  L647X_QueueCommands (0, (uint8_t) L647X_MOVE |(uint8_t) BACKWARD,
60000);

  /* Queue comamnd "Run at 400 steps/s in forward direction" for device
1 */
  L647X_QueueCommands (1, (uint8_t) L647X_RUN |(uint8_t) FORWARD,
Speed_Steps_to_Par(400));

  /* No operation for other devices */
  for (loop = 2; loop <= L647X_GetNbDevices();  loop++)
  {
    L647X_QueueCommands (loop, L647X_NOP, 0);
  }

  /* Issue simultaneously the queued commands */
  L647X_SendQueuedCommands();

  /* Wait for all devices end moving */
  L647X_WaitForAllDevicesNotBusy();

…
}
```

5.40 L647X_SetMark ()

void ***L647X_SetMark*** *(uint8_t* ***deviceId*** *)*

Description
Sets current position to be the Mark position.

Parameters

[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values

None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to go to position 200 */
  L647X_CmdGoTo(0, 200);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

  /* Set current position to become the Mark position of device 0*/
  L647X_SetMark(0);

  /* Request device 0 to run in FORWARD direction at 400 step/s */
  L647X_CmdRun(0, FORWARD, Speed_Steps_to_Par(400));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request device 0 to make a soft stop */
  L647X_CmdSoftStop(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);

  /* Request device 0 to come back to its Mark position */
  L647X_CmdGoMark(0);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);
  …
}
```

5.41 L647X_StartStepClock ()

void ***L647X_StartStepClock*** *(* *uint16_t* ***newFreq*** *)*

Description
Starts the step clock by using the given frequency

Parameters
[in] **newFreq** frequency in Hz of the step clock

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Enable Step Clock Mode of the l647x*/
  L647X_CmdStepClock(0, FORWARD);

  /* Wait for 1 second */
  HAL_Delay(1000);

  /* Enable the step clock at 333 Hz*/
  L647X_StartStepClock(333);

  /* Let the motor runs for 5 second at 333 step/s*/
  HAL_Delay(5000);

  /* Stop the step clock */
  L647X_StopStepClock();

 }
```

5.42 L647X_StopStepClock ()

*void **L647X_StartStepClock** (void)*

Description
Stops the step clock.

Parameters
None

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Enable Step Clock Mode of the l647x*/
  L647X_CmdStepClock(0, FORWARD);

  /* Wait for 1 second */
  HAL_Delay(1000);

  /* Enable the step clock at 333 Hz*/
  L647X_StartStepClock(333);

  /* Let the motor runs for 5 second at 333 step/s*/
  HAL_Delay(5000);

  /* Stop the step clock */
  L647X_StopStepClock();
  …
}
```

5.43 L647X_WaitWhileActive ()

void ***L647X_WaitWhileActive*** *(* *uint8_t* ***deviceId*** *)*

Description

Locks until the device state becomes Inactive. The use of this function is particularly useful to wait for the stop of a given device before sending it a new moving command (`GoTo`, `GoMark`, `GoHome`, `Move`, `Run`). Without using a `WaitWhileActive`, the last moving command will be executed before the completion of the previous one.

Parameters

[in] **deviceId** Id of the device (from 0 to MAX_NUMBER_OF_DEVICES-1)

Return values

None

Example

```
#include "l647x_linux.h"


int main(void)
{
  int32_t pos;

  /* Start the l647x library to use 1 device */
  L647X_Begin(1);

  /* Request device 0 to go to position -200 */
  L647X_CmdGoTo(0, -200);

  /* Wait for device 0 end moving */
  /* Without this command, the GoTo -200 will be interrupted */
  /* by the GoTo 200 */
  L647X_WaitWhileActive(0);

  /* Request device 0 to go to position 200 */
  L647X_CmdGoTo(0, 200);

  /* Wait for device 0 end moving */
  L647X_WaitWhileActive(0);
  …
}
```

5.44 L647X_WaitForAllDevicesNotBusy ()

void ***L647X_WaitForAllDevicesNotBusy*** *(void)*

Description
Locks until all devices become not busy

Parameters
None

Return values
None

Example

```
#include "l647x_linux.h"


int main(void)
{
  uint16_t status;

  /* Start the l647x library to use 3 devices */
  L647X_Begin(3);

  /* Request device 0 to run BACKWARD at 400 step/s*/
  L647X_CmdRun(0, BACKWARD,Speed_Steps_to_Par(400));

  /* Request device 1 to run FORWARD at 400 step/s*/
  L647X_CmdRun(1,FORWARD,Speed_Steps_to_Par(400));

  /* Request device 2 to run FORWARD at 200 step/s*/
  L647X_CmdRun(2,FORWARD,Speed_Steps_to_Par(200));

  /* Wait for 5 seconds */
  HAL_Delay(5000);

  /* Request a soft stop of the device 0 */
  L647X_CmdSoftStop(0);

  /* Request a hard stop of the device 1 */
  L647X_CmdHardStop(1);

  /* Request a soft HiZ stop of the device 2 */
  L647X_CmdSoftHiZ (2);
  /* Wait for all devices end moving */
  L647X_WaitForAllDevicesNotBusy();

  …
}
```

6 Revision history

Table 1. Revision history

Date	Revision	Changes
2015-02-18	0.1	Initial release
2017-02-06	0.2	Update for EVAL6470H-RPi evaluation board