Phi Robotics

Product Manual

SmartLynx

Version 1.1

Phi Robotics Research Pvt. Ltd.

www.phi-robotics.com

Table of Contents

1 Introduction

SmartLynx is a micro stepping stepper motor driver board powered by L6470. A 1/128 steps resolution can be achieved due to its high quality control system. The digital control core can generate user defined motion profiles with acceleration, deceleration, speed or target position through a set of dedicated registers. All commands and data register values are sent through SPI interface. A rich set of protection features makes SmartLynx an ideal choice for demanding motor control applications.

SmartLynx is compatible with Phi Robotics xLynx-CB board.

2 Board Features

- Supports up to 128 micro steps per full step
- SPI interface for communication with microcontroller
- On-board 32 MHz crystal oscillator
- Programmable speed profile and positioning
- Two level over temperature protection
- 5 bit ADC for battery voltage monitoring
- Stall detection

3 Specifications

- L6470 package: POWERSO36
- Motor supply voltage: 8V to 45V
- Maximum output current: 7 A

3.1 PCB Details

- PCB size: 57.15 mm x 38.1 mm
- PCB type: FR4
- Solder mask: Black
- Board thickness: 1.6 mm
- Surface finish: Immersion gold

4 Hardware Connections

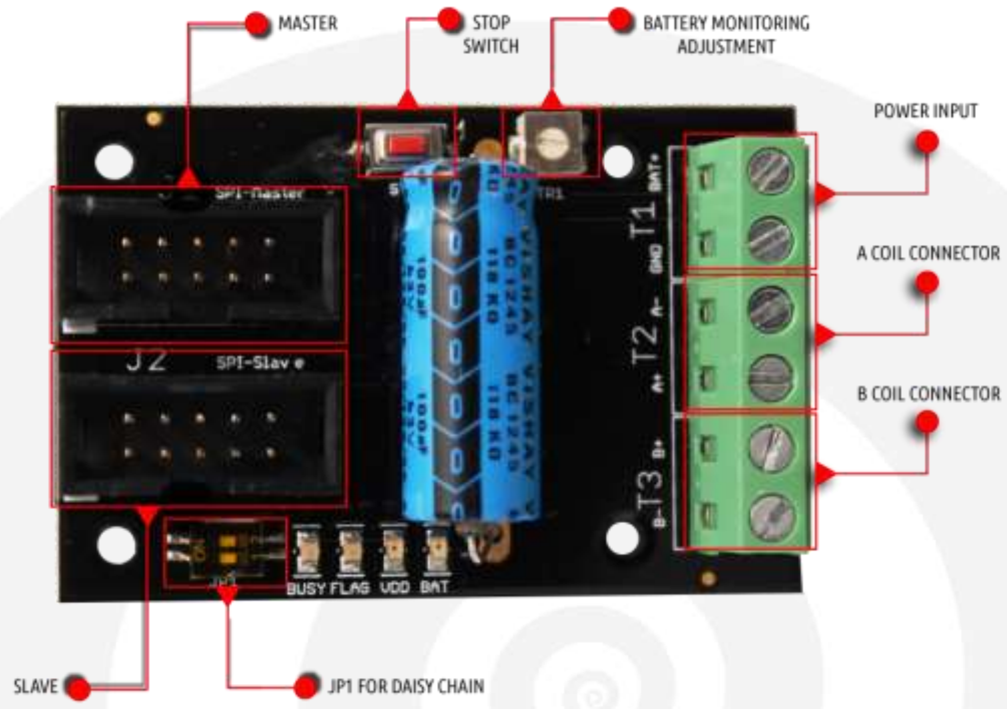

Figure 1 - SmartLynx top view

4.1 Battery Connection

Terminal T1 is used to connect an 8V to 45V external power supply to the motor.

4.2 Motor Connections

Terminals T2 and T3 are used to connect SmartLynx to bipolar stepper motor. T2 is for connecting coil A of the motor and T3 for coil B of the motor.

4.3 Microcontroller Connections

Connector J1, a 5x2 pin FRC header, is provided for connecting the SmartLynx with microcontroller. J1 has SPI communication lines as well as other control and status pins for the motor driver. The connector is compatible with Phi Robotics xLynx-CB board. Figure 2 below shows the J1 header pin layout.

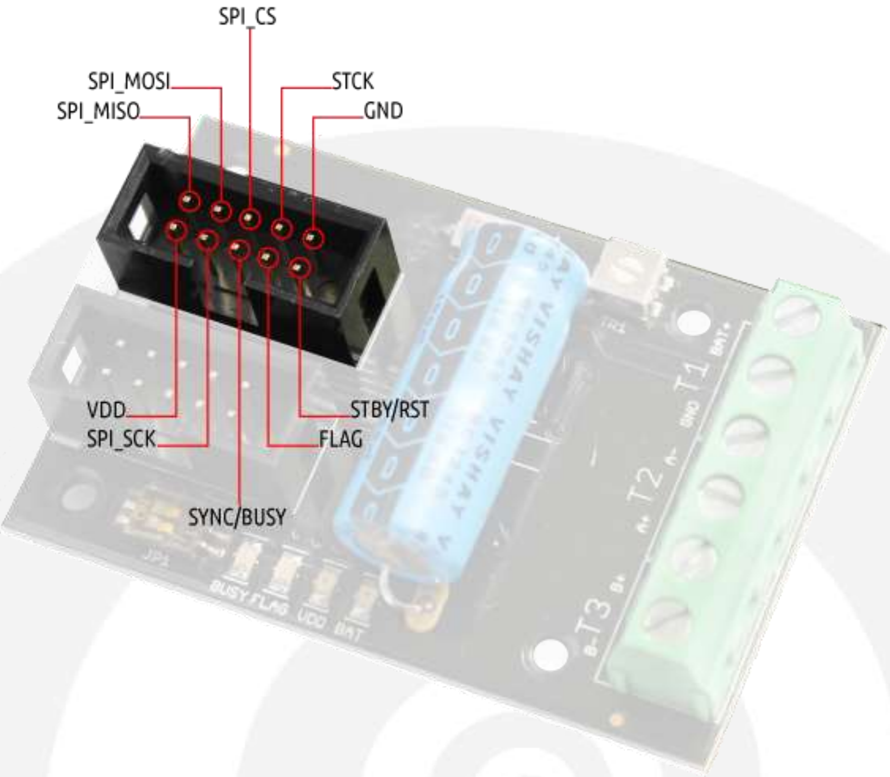

Figure 2 - J1 header pin layout

4.4 Daisy Chain

L6470 stepper motor driver used in SmartLynx allows daisy chain connections. Multiple SmartLynx boards can be connected in daisy chain thus allowing user to control multiple motors using same SPI bus and chip select pin. The incoming connection of the chain should be connected to J1 and outgoing connection for the next board of the chain should be connected to J2. Figure 3 below shows daisy chain connection details.

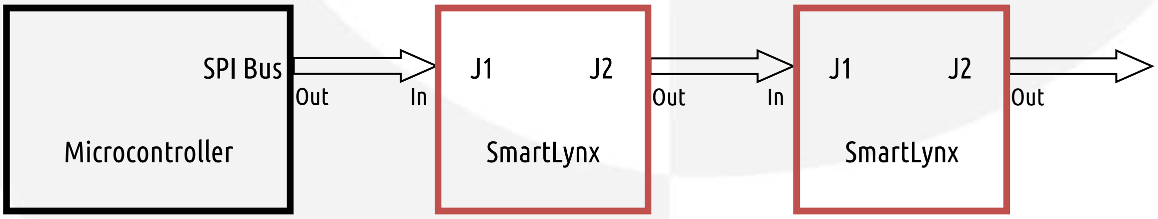

Figure 3 – SmartLynx daisy chain connections

4.5 Jumper Switch Settings

Following table explains the settings of jumper JP1.

Table 1 - JP1 settings

Configuration	JP1 switch 1 position
Single board (without daisy chain)	On
Daisy chain (except for last board in the chain)	Off
Last board in daisy chain	On

4.6 Stop Switch

Stop switch provides the user with an option to stop the motor. When the switch is pressed, the hard stop command is executed.

4.7 Battery Voltage Monitoring

L6470 motor driver has on-chip battery voltage monitor. ADC for this battery voltage monitor cannot accept more than 3.3V. However, the motor supply range is 8 to 45V. Hence, we need to use a voltage divider to make the battery voltage compatible with the battery voltage monitor. Potentiometer TR1 is provided for the same. User should adjust TR1 such that voltage fed to ADC should not be more than 3.3V. Value to set for TR1 can be calculated using the following equation.

$$R_{TR1} = (3.03K \ \times V_{batt}) - 43K$$

As a 5 bit ADC is used for battery voltage monitoring, following equation can be used to calculate voltage fed to the ADC.

$$V_{ADC} = \ 0.103125 \ \times ADC\ value$$

5 Pseudo Code

- Speed values mentioned in steps/seconds.
- Acceleration values mentioned in steps/seconds^2.

5.1 Register Addresses and Commands

```
// smartlynx register addresses
SMARTLYNX_ABS_POS = 0x01
SMARTLYNX_SPEED = 0x04
SMARTLYNX_ACC = 0x05
SMARTLYNX_DEC = 0x06
SMARTLYNX_MAX_SPEED = 0x07
SMARTLYNX_MIN_SPEED = 0x08
SMARTLYNX_FS_SPD = 0x15
SMARTLYNX_STEP_MODE = 0x16
SMARTLYNX_STATUS = 0x19

//smartlynx commands
SMARTLYNX_SET_PARAM = 0x00
SMARTLYNX_GET_PARAM = 0x20
SMARTLYNX_RUN = 0x50
SMARTLYNX_MOVE = 0x40
SMARTLYNX_SOFT_STOP = 0xB0
SMARTLYNX_HARD_STOP = 0xB8
SMARTLYNX_RESET_DEVICE = 0xC0
SMARTLYNX_GET_STATUS = 0xD0

// step size selection
SMARTLYNX_STEP_FULL = 0
SMARTLYNX_STEP_HALF = 1
SMARTLYNX_STEP_QUARTER = 2
SMARTLYNX_STEP_ONE_EIGHT = 3
SMARTLYNX_STEP_ONE_16TH = 4
SMARTLYNX_STEP_ONE_32ND = 5
SMARTLYNX_STEP_ONE_64TH = 6
SMARTLYNX_STEP_ONE_128TH = 7
```

5.2 Basic Data Transfer to SmartLynx

```
uint32_t smartlynxDataTransfer (uint32_t data, uint8_t size)
{
      uint8_t wrData=0;
      uint32_t rdData=0;

      switch(len)
      {
            case 3:
                  //transfer 3rd data byte
                  wrData = data >>16;
                  gpioClear(SPI_CS);        // assert chip select
                  spiWriteByte(wrData);     // write byte over SPI bus
                  rdData = spiReadByte();   // read received byte
                  gpioSet(SPI_CS);          // de-assert chip select
                  rdData = rdData << 8;

            case 2:
                  //transfer 2nd  data byte
                  wrData = data >> 8;
                  gpioClear(SPI_CS);
                  spiWriteByte(wrData);
                  rdData |= spiReadByte();
                  gpioSet(SPI_CS);
                  rdData = rdData << 8;

            case 1:
                  //transfer 1st data byte
                  wrData = data;
                  gpioClear(SPI_CS);
                  spiWriteByte(wrData);
                  rdData |= spiReadByte();
                  gpioSet(SPI_CS);
            default:
                  break;
      }
      return rdData;
}
```

5.3 Configuring Various Parameters

```
void smartLynxInit(void)
{
      uint32_t dummy=0;
      uint32_t status;

      // hardware reset SmartLynx device
      // using port pin
      gpioClear(SMARTLYNX_RESET_PIN);
      delay_us(250);    // delay in microseconds
      gpioSet(SMARTLYNX_RESET_PIN);
      delay_us(250);

      // read status register is must to set ULV0(under-voltage lockout)
      // flag
      smartlynxDataTransfer(SMARTLYNX_GET_STATUS, 1);
      // read smartlynx status
      status = smartlynxDataTransfer(dummy, 2);
}

void smartlynxSetMaxSpeed(float maxSpeed)
{
      uint32_t data;
      // data conversion for smart lynx
      data = 0.065536 * maxSpeed;
      // set maximum speed command
      smartlynxDataTransfer( (SMARTLYNX_SET_PARAM |
                              SMARTLYNX_MAX_SPEED), 1);
      // set maximum speed of rotation
      smartlynxDataTransfer(data, 2);
}

void smartlynxSetMinSpeed(float minSpeed)
{
      uint32_t data;
      // data conversion for smart lynx
      data = 4.1943 * minSpeed;           // minimum speed can be 0
      // set minimum speed command
      smartlynxDataTransfer( (SMARTLYNX_SET_PARAM |
                              SMARTLYNX_MIN_SPEED), 1);
      //set minimum speed of rotation
      smartlynxDataTransfer(data, 2);
}

void smartLynxResetDevice(void)
{
      // reset smartLynx driver
      smartlynxDataTransfer(SMARTLYNX_RESET_DEVICE, 1);
}
```

```
void smartLynxSetFullStepSpeed(float speed)
{
      uint32_t data;
      // data conversion for smart lynx
      data = (0.0655*speed) - 0.5;
      // set full step speed command
      smartlynxDataTransfer( (SMARTLYNX_SET_PARAM |
                              SMARTLYNX_FS_SPD), 1);
      // set full step speed value
      smartlynxDataTransfer(speed, 2);
}

void smartLynxSetAccelerationProfile (float acc, float dec)
{
      uint32_t data;
      // data conversion for smart lynx
      data = 0.0687 * acc;
      // set acceleration command
      smartlynxDataTransfer( (SMARTLYNX_SET_PARAM | SMARTLYNX_ACC), 1);
      // set acceleration value
      smartlynxDataTransfer(data, 2);

      // data conversion for smart lynx
      data = 0.0687 * dec;
      // set deceleration command
      smartlynxDataTransfer( (SMARTLYNX_SET_PARAM | SMARTLYNX_DEC), 1);
      // set deceleration value
      smartlynxDataTransfer(data, 2);
}
```

5.4 Running and Stopping the Motor

```
// this function will put motor in continuous running mode
void smartlynxRun(uint8_t dir, float speed)
{
      uint32_t data;
      // data conversion for smart lynx
      data = 67.1*speed;
      // run command
      smartlynxDataTransfer( (SMARTLYNX_RUN | dir), 1);
      // set speed
      smartlynxDataTransfer(data, 3);
}

// this function will move motor by specified number of steps (stepCount)
void smartlynxMove(uint8_t dir, uint32_t stepCount)
{
      // move command
      smartlynxDataTransfer( (SMARTLYNX_MOVE | dir), 1);
      // set number of steps to move
      smartlynxDataTransfer(stepCount, 3);
}

// single step driving for manual control
void smartlynxSingleStep(void)
{
      // pulse to STEP pin
      gpioClear(STEP);
      delay_us(1);
      gpioSet(STEP);
}

// this function will stop motor by decelerating it
// to minimum speed set
void smartlynxSoftStop(void)
{
      // soft stop command
      smartlynxDataTransfer(SMARTLYNX_SOFT_STOP, 1);
}

// this function will stop the motor immediately
void smartlynxHardStop(void)
{
      // hard stop command
      smartlynxDataTransfer(SMARTLYNX_HARD_STOP, 1);
}
```

6 References

L6470 Datasheet: http://www.st.com/web/catalog/sense_power/FM142/CL851/SC1794/SS1498/LN1723/PF248592

Phi Robotics Research Pvt. Ltd.

www.phi-robotics.com