LV8121VSLDGEVK

3-Phase Brushless DC Motor Module Solution Kit

ON Semiconductor®

www.onsemi.com

APPLICATION NOTE

This Application Note provides supporting information for the LV8121VSLDGEVK Motor Driver Solution Kit.

KIT OVERVIEW

When developing a motor control application system using any motor driver product provided by ON Semiconductor, it is first necessary to design the hardware after understanding the specifications of the motor driver product. And then proceed to generating operation control signals (for the rotating direction, speed, angle, etc.) that will be inputted into the driver IC. Given that operation control signals like these are generally generated with the use of a microcomputer, it is necessary to develop software for the microcomputer in addition to the hardware design mentioned above.

This kit provides an API function library for motor driver control designed to control the ON Semiconductor motor driver product (LV8121V) via the Arduino Micro microcomputer. It also provides a dedicated GUI for controlling motors connected to the Arduino Micro microcomputer that embedded with the API library from a computer via USB communication.

This means that it is possible to easily tune and otherwise debug motor control sequences and operation parameters without developing motor control software for the Arduino Micro.

In addition, this GUI also has an automatic code generation feature. It outputs control software (source code) to achieve debug control sequences and operation parameters on the Arduino Micro microcomputer in a format (sketch) in which it can be compiled with the Arduino IDE.

As a result, the use of this kit allows users to easily develop a prototype of motor control application system without special knowledge of the specifications of the motor driver on the software development using the API functions. It also reduces the development period and significantly lowers costs.

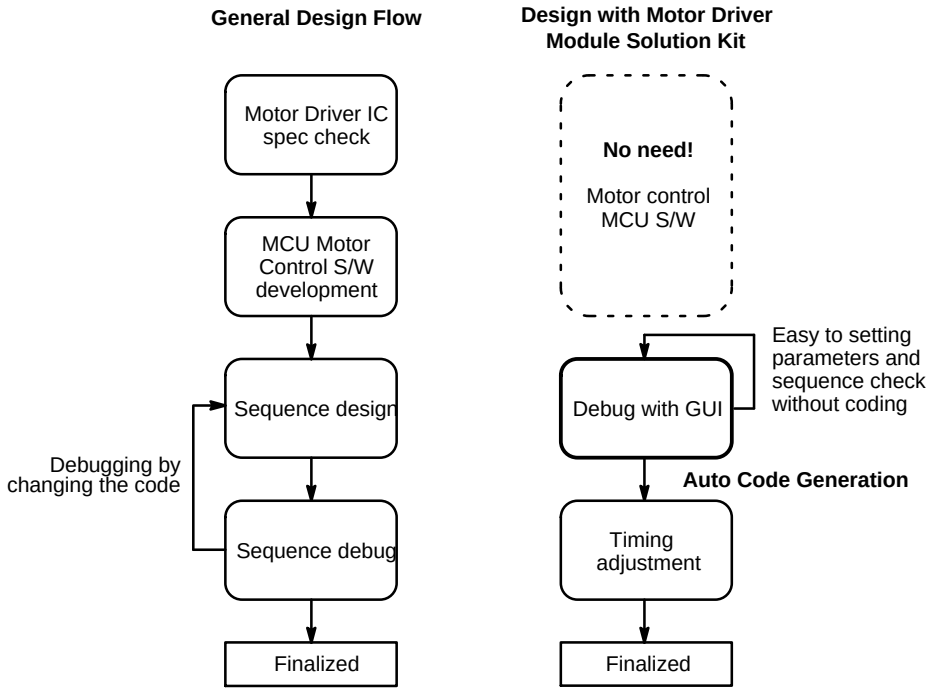


Figure 1. General Development Process Flow and Development Process Flow Using This Kit

GUI DEBUG MODE AND STANDALONE DEVELOPMENT MODES

This kit is presumed to be used in a total of three different debug and development modes: the GUI debug mode, the standalone development mode using the automatic code generation feature, and the standalone development mode using an original sketch.

GUI Debug Mode

In the GUI debug mode, the user will operate the dedicated GUI installed on the computer to change motor control sequences and operation parameters for the LV8121V. The user can tune the parameters while actually operating the motor.

It also has an automatic code generation feature, which outputs a sketch (an .ino file) that can be compiled and written into the Arduino Micro that reflects the motor control sequences and operation parameters set by the user by operating the GUI.

To use this mode, it is necessary to write firmware generated by using Arduino IDE into the Arduino MICRO.

The user must compile the sketch for the GUI (LV8121_Program. ino) with motor driver API Functions library (LV8121_Lib.cpp/h) and TimerOne library which is separately downloaded off the Internet into the Arduino MICRO to generate the firmware by Arduino IDE.

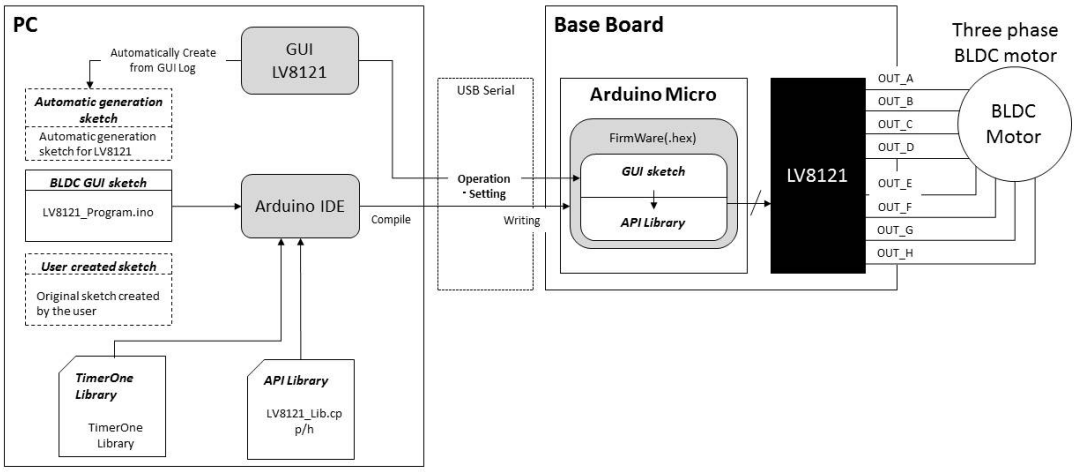


Figure 2. Outline of the GUI Debug Mode

Standalone Development Modes

In the standalone development modes, the user adjusts the timing of the motor drive and adds the user's original source code based on the automatically generated sketch in the GUI debug mode and writes them into the Arduino Micro instead of the sketch that is exclusive for the GUI to facilitate standalone BLDC motor driving using this kit.

Figure 3 provides a graphical representation of this mode.

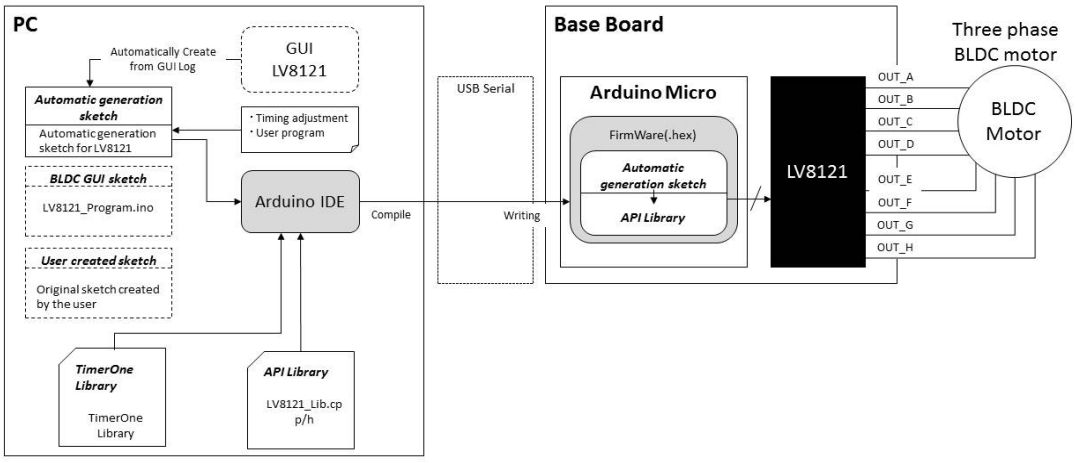


Figure 3. Standalone Development Mode Using Automatic Generation Feature

It is also possible to develop sketches from scratch with the API function library instead of using the automatic code generation feature. With advanced knowledge of programming, it is possible to develop a more complicated and sophisticated motor control application.

Refer to Figure 4.

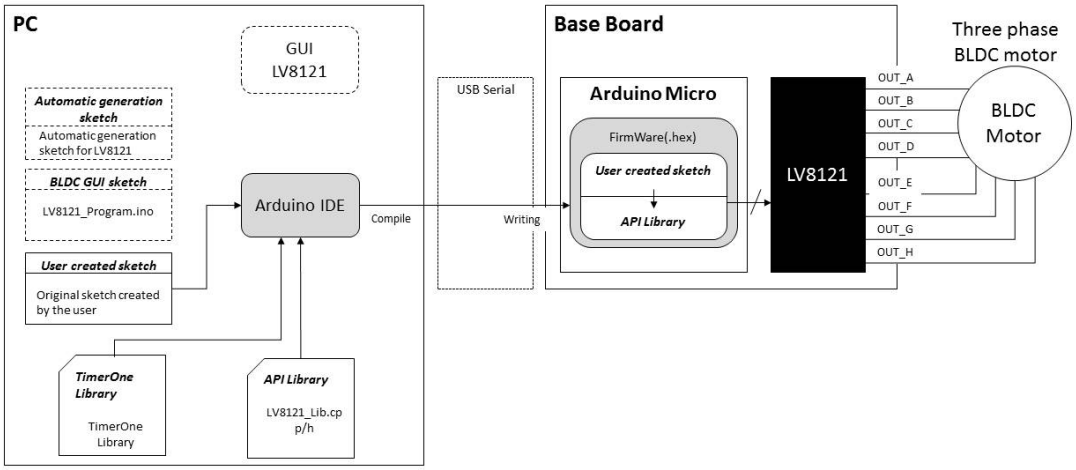


Figure 4. Standalone Development Mode Using Original Sketch

PROGRAMMING GUIDE

ARDUINO SKETCH

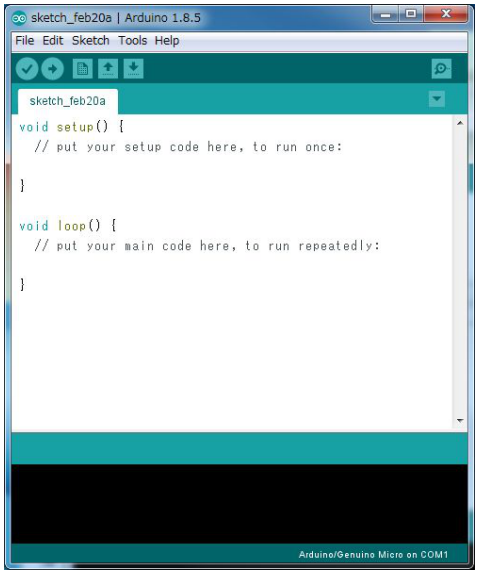
Sketch Overview

A sketch is a program written in Arduino language used for the Arduino. A sketch is a program, or the unit of code that is compiled, uploaded to and run on an Arduino board. The Arduino language is based on the C/C++ languages, and it supports the full structure of the C language and some features of the C++ language.

For details and help regarding Arduino commands, please refer to: https://www.arduino.cc/en/Tutorial/Sketch

setup() and loop()

If you create a new sketch on the Arduino IDE, setup() and loop() will be automatically inserted as below.



setup() is a function that is called only once after the Arduino Board is powered on or reset. This function mainly conducts initial settings including preparation of the libraries to be used, initialization of variables, and initialization of pin modes.

loop() is, as its name suggests, a function that is executed repeatedly after setup() function executed. It contains programs which should be run many times.

https://www.arduino.cc/reference/en/language/structure/sketch/setup/

Overview of Motor Driver Library

The motor driver library (LV8121_APILibrary) provides a library for controlling a BLDC motor with the use of ON Semiconductor LV8121V motor driver from the Arduino Micro. BLDC motor control using the LV8121V is easily achieved by including this API library via the Arduino IDE and calling the API functions suited to the purposes in the sketch.

Table 1. LIST OF MOTOR DRIVER LIBRARY FILES

#	File Name	Description
1	keywords.txt	Keyword file (sets highlighted words in sketches)
2	LV8121_Lib.cpp	Source file
3	LV8121_Lib.h	Header file

Using the BLDC Motor Driver API Library

For instructions for the inclusion of the library, refer to the Quick Start Guide. To use the BLDC motor control API library in the Arduino, include the header file of the BLDC motor control API library at the beginning of the sketch, as explained below, and instantiate the class to be used.

To use the GUI tool, it is necessary to separately call the serial communication API. For details, refer to API Function Specifications. The following shows a sketch that includes the BLDC motor control API library.

Inclusion of a Three−Phase Brushless DC Motor Control API Library

```
#include <LV8121_Lib.h>
// Importing a header file for using the API function for LV8121V
#include <TimerOne.h>
// Importing a header file for using the TimerThree library
Lib_LV8121V Lib                 // Instantiation of LV8121V classes (Note 1)
void setup(){                   // Functions called at start (Refer to setup() and loop())
}
void loop(){                    // Functions executed repeatedly (Refer to setup() and loop())
}
```

1. In this example, the instatntiation was conducted with the name of Lib. The addition of ‘Lib.’ as prefix makes it possible to call the API functions in the Lib_LV8121V class.
   e.g. Lib.initLib();

Compiling and Writing a Sketch into Arduino

For the steps for compiling and writing a sketch (Arduino program), refer to *Compiling an Arduino Program and Writing into Arduino* in Quick Start Guide.

CODING A PROGRAM (SKETCH)

Automatic Code Generation

The following explains the functions which are written by the automatic code generation at the timing of sketch output.

A generated sample sketch is used as an example for explanation.

For details of the API functions of the motor driver library called by the GUI debug operation, refer to section API Function Specifications.

Example of automatically generated code

```
See #include <LV8121_Lib.h>          // Refer to Using the BLDC Motor Driver API Library
See #include <TimerOne.h>            // Refer to Using the BLDC Motor Driver API Library
# define TIMER 10000                 // Interrupt Intervals
#define DELAY 3000                   //delay [ms](*2)
See Lib_LV8121VLib;                  // Refer to Using the BLDC Motor Driver API Library
See void setup()                     // Refer to  Sketch Overview
{
Serial.begin (19200);                // Set the baud rate at 19200 and open the port (Note 2)
Initializing Lib. initLib();         // Intialize the Arduino parameters and registers
Specifying Functions to Call with attachInterrupt(digitalPinToInterrupt(2), pulse_interrupt, CHANGE); // Pin Interrupt
Set Timer1.initialize(TIMER);        // Timer1 is initilalized with 100us interrupt period
Specifying Functions to Call with Timer1.attachInterrupt(time_interrupt);// Timer Interrupt
Lib. setDelay (2000);                // Interval time [ms] after start−up of the Arduino (Note 3)
Lib.setParameters(4,12,0.0044);
Lib.setPIParameter(0.01,0.01);
Lib.setControl(1);
Lib.setSpeed(2000);
Lib.setStartFlag(1);
Lib. setDelay(DELAY;            // Interval time [ms] after start−up of the Arduino (Note 3)
Lib.setStartFlag(0);
Lib. setDelay(DELAY;            // Interval time [ms] after start−up of the Arduino (Note 3)
}
Functions called by the void pulse_interrupt()  //pin interrupt
{
Lib.pulseFire();
}
Functions called by the void time_interrupt()  // timer interrupt
{
Lib.timerFire();
}
See void loop()   // Refer to Sketch Overview
{
}
```

2. This function is required in the case of using serial communication.
   If not using serial communication, you can remove this without affection for the motor operation.
3. The default value of 2000 ms (2 sec) is set.

Using an Automatically Generated Sketch

An automatically generated sketch is so simply structured that it is easy for programming beginners to use. Customization will turn it into a more practical program. This sections sketch is representative of the functionality of the Arduino setup part and motor driver part which can be called using setup() and loop(). An example of this customization is shown in the sketch generated in Automatic Code Generation

```
#include <LV8121_Lib.h>
#include <TimerOne.h>
#define TIMER 10000
#define DELAY 3000
Lib_LV8121V Lib;
Void setup()
{
MotorSetup();// The functionalized initial settings of the Arduino are called with setup().

Lib.setParameters(4,12,0.0044);
Lib.setPIParameter(0.01,0.01);
Lib.setControl(1);
Lib.setSpeed(2000);
}
Void pulse_interrupt()
{
Lib.pulseFire();
}
Void time_interrupt()
{
Lib.timerFire();
}
Void loop()
{
MotorControl();// The functionalized motor drive part is called with loop().
}
// The initial settings of the Arduino are functionalized. //
Void motorSetup(){
Serial.begin(19200);
Lib.initLib();
AttachInterrupt(digitalPinToInterrupt(2),pulse_interrupt,CHANGE);
Timer1.initialize(TIMER);
Timer1.attachInterrupt(time_interrupt);
Lib.setDelay(2000);
}
// The motor drive part is functionalized //
Void motorControl(){
Lib.setStartFlag(1);
Lib.setDelay(DELAY);
Lib.setStartFlag(0);
Lib.setDelay(DELAY);
}
```

API SPECIFICATIONS

Outline of 3−phase Brushless DC−Motor Control API

This section outlines the API for control of the Arduino Micro BLDC motor for LV8121V motor driver.

Overview

This API provides a library for controlling the BLDC motor from the Arduino Micro with the use of ON Semiconductor LV8121V motor driver. It allows the user to easily control the BLDC motor using LV8121V by including the API library in the Arduino IDE and by writing the API functions that match with the user's desired purpose(s) of use in the sketch.

When using this API library, please note that it is also necessary to include the separate TimerOne library. For details of how to include the TimerOne library, refer to Quick Start Guide "Including the TimerOne library."

Library File Configuration

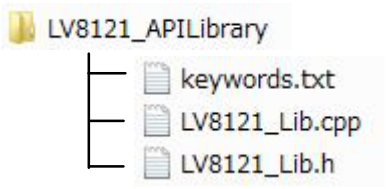


Table 2. LIST OF LIBRARY FILES

#	File Name	Description
1	keywords.txt	Keyword file (sets highlightened words in sketches)
2	LV8121_Lib.cpp	Source file
3	LV8121_Lib.h	Header file

Pin Assignment of the Arduino Micro/LV8121V Base Board

The following portrays the pin assignment of the Arduino Micro/LV8121V Base Board.

The white background color for the Arduino Micro input and the output pins represents the resources available to users.

Remark	Base Board compatible terminal	LV8121compatible terminal	Arduino Micro		LV8121compatible terminal	Base Board compatible terminal	Remark
	NC		MO	SCK		NC	
	NC		SS	MI		NC	
	J2 TX		Tx	5V			
	J2 RX		Rx			SGND SGND	GND
		No physical wiring	RST	RST	No physical wiring		
GND	SGND	SGND		5V		+5V J2 +5V	
1−hole FG output terminal (Open−drain output)	OUT_B	FG	2	NC	No physical wiring		
			3	NC	No physical wiring		
			4	A5		CN8 A5	
Normal/Reverse Control Terminal		FR	5	A4		CN8 A4	
Terminals for varying the DUTY of outputs		CTL	6	A3		CN8 A3	
			7	A2	LIM	CN8 A2	LIM analog voltage input terminal
			8	A1	CTL		CTL analog voltage input terminal
			9	A0		P/N	Input terminal for motor driver identification
			10	RF	No physical wiring		
Terminal for setting the minimum DUTY of outputs	OUT_D	LIM	11	3V			
			12	13	S/S		START/STOP CONTROL TERMINAL
			USB				

Figure 5. Pin Assignment of Arduino Micro/LV8714TA Base Board

INITIAL SETTINGS OF THREE−PHASE BRUSHLESS DC MOTOR CONTROL API

This section describes the initial settings for the BLDC motor control API.

Resources Used by the API

This API uses the Arduino pins shown in Table 3 and the corresponding ATmega32U4 timer register, which are not available to users. See ATmega16U4−32U4_Datasheet.pdf for more information about the register.

Table 3. ARDUINO MICRO PIN AND CORRESPONDING ATmega32U4 TIMER REGISTER

#	Arduino Pins	Timer Register	Description	Explained in
1	D3, IO11	TCCR0B	Timer/Counter 0 Control Register B	TCCR0B
1	D6, D13	TCCR4B	Timer/Counter 4 Control Register B	TCCR4B

Default

The initialization of the parameters, timer registers, and input/output pins is performed by the initLib functions (see Table 6). Be sure to call the initLib functions in setup() to initialize the parameters, timer registers, and I/O pins.

See Table 6 for instructions on using initLib functions.

Table 4. INITIALIZED PIN SETTINGS

#	Arduino I/O Pins	Initial Setting Value	Relationship
1	D2 (FG)	LOW / INPUT	Pin Assignment of the Arduino Micro/LV8121V Base Board
4	D5 (FR)	LOW / OUTPUT	Pin Assignment of the Arduino Micro/LV8121V Base Board
5	D6 (CTL)	LOW / OUTPUT	Pin Assignment of the Arduino Micro/LV8121V Base Board
8	IO11 (LIM)	LOW / OUTPUT	Pin Assignment of the Arduino Micro/LV8121V Base Board
9	IO13 (S/S)	LOW / OUTPUT	Pin Assignment of the Arduino Micro/LV8121V Base Board

API FUNCTION SPECIFICATIONS

Overview of API Functions

The following API functions are used to control the LV8121V motor driver.

Table 5. API FUNCTIONS

#	Function	Description	Chapter
1	initLib	• Register setting • Each parameter default setting	initLib
2	setPwmDuty	• Set the voltage to the LV8121CTL terminal	setPwmDuty
3	setLimPwmDuty	• Set the voltage to the LV8121LIM terminal	setLimPwmDuty
4	setStartFlag	• Set the start/stop control terminal	setStartFlag
5	setDirection	• Setting the rotation direction	setDirection
6	setParameters	• Setting of power supply voltage, electrode logarithm, and back electromotive force constant	setParameters
7	setSpeed	• Set the target Closed Loop velocity	setSpeed
8	setPIParameter	• Set the PI gain of the Closed Loop	setPIParameter
9	setControl	• Open Loop/Closed Loop switching	setControl
10	readAdc	• Measurement of Analog Voltage Values	readAdc
11	readModule	• Acquisition of A0 pin analog voltage value (for model discrimination)	readModule
12	setDelay	• Special−purpose delay functions for Timer0 division ratios	setDelay
13	getLockStatus	• Obtain the ON/OFF status of the restraint	getLockStatus
14	getSpeed	• Gets the motor rotation speed	getSpeed
15	guiSerialRead	• Analyze Bytestream notified by serial communication and Call APIs	guiSerialRead
16	guiSerialParse	• Analysis of Serial communication with GUIs	guiSerialParse

API Function Details

initLib

Table 6. initLib

API function	initLib()		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	void	None	None
Return values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Timer / register initialization 2. Input / Output Initial Settings for Each Pin D2, D5, D6, IO11, IO13, A0 3. Each parameter initialization 4. Each function initialization SetStartFlag(STOP); SetDirection(DIRECTION_FW); SetControl(OPEN_LOOP); SetParameters(REF_POLEPAIRS, REF_VOLTAGE, REF_BEMF); SetSpeed(RPM_MIN); SetPwmDuty(DUTYRATE_MIN); SetLimPwmDuty(DUTYRATE_MIN);		

Example usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	}
	void loop() {
	}		

setPwmDuty

Table 7. setPwmDuty

API function	setPwmDuty(byte dutyRate)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	dutyRate	CTL Duty [Value Range] 0 to 100 [%] (1.2 to 3.4 [V])
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set output voltage to CTL		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib.setPwmDuty(30); // PWM Duty 30[%]
	}
	void loop() {
	}		

setLimPwmDuty

Table 8. setLimPwmDuty

API function	setLimPwmDuty(byte dutyRate)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	dutyRate	LIM Duty [Value Range] 0 to 100 [%] (1.2 to 3.4 [V])
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set output voltage to LIM		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib.setLimPwmDuty(50); // PWM Duty 50[%]
	}
	void loop() {
	}		

setStartFlag

Table 9. setStartFlag

API function	setStartFlag(byte select)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	select	0: STOP 1: START
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set START/STOP to S/S		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib.setControl(0); // Open Loop
	Lib.setPwmDuty(30); // PWM Duty 50[%]
	Lib. setStartFlag (1);// Motor revolution
	Lib. setDelay (1000);//1 second wait
	}
	void loop() {
	}		

setDirection

Table 10. setDirection

API function	setDirection(byte select)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	select	0: Forward Rotation (Forward) 1: Reverse Rotation (Reverse)
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set the rotation direction to FR		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib.setDirection(0); //
	}
	void loop() {
	}		

setParameters

Table 11. setParameters

API function	setParameters(byte polePairs, float Vin, float ke)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	polePairs	Pairs of poles [Value Range] 1 to 128
	float	vin	Power supply voltage [Value Range] 8 to 35 [V]
	float	ke	Back electromotive force constant [Value Range] 0.0001 to 0.0400
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set the electrode logarithm, power supply voltage, and back electromotive force constant. 3. Set the maximum speed derived from the parameters set in (2)		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib. setParameters (4,12,0.0044);//Log of Poles 4 [pole/2], Power Supply Voltage 12 [V], Back Electromotive Force Constant 0.0044 [V/rpm]
	}
	void loop() {
	}		

setSpeed

Table 12. setSpeed

API function	setSpeed(uint16_t spd)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	uint16_t	spd	Target velocity in the Closed Loop [Value Range] 0 to 48000 [rpm]
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set target speed		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib. setParameters (4,12,0.0044);//Log of Poles 4 [pole/2], Power Supply Voltage 12 [V], Back Electromotive Force Constant 0.0044 [V/rpm]
	Lib.setControl(1); // Closed Loop
	}
	void loop() {
	Lib. setSpeed (3000);// Target speed set to 3000 rpm
	Lib. setStartFlag (1);// Motor revolution
	Lib. setDelay (1000);//1 second wait
	}		

setPIParameters

Table 13. setPIParameters

API function	setPIParameter(float p, float i)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	float	p	P gain in the Closed Loop [Value Range] 0.01 to 1.00
	float	i	I gain in the Closed Loop [Value Range] 0.01 to 1.00
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set P Gain and I Gain		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib. setPIParameter (0.01, 0.01); //P gain 0.01 I gain 0.01
	}
	void loop() {
	}		

setControl

Table 14. setControl

API function	setControl(byte ctr)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	ctr	0: Open Loop 1: Closed Loop
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument threshold check 2. Set the control mode		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib. setParameters (4,12,0.0044);//Log of Poles 4 [pole/2], Power Supply Voltage 12 [V], Back Electromotive Force Constant 0.0044 [V/rpm]
	Lib.setControl(0); // Control Mode Open Loop
	}
	void loop() {
	}		

readAdc

Table 15. readAdc

API function	readAdc(byte pin)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	byte	pin	Pin numbering 1: A1 2: A2 3: A3 4: A4 5: A5
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Argument range check 2. Perform analogRead() on the pins of the arguments and return the retrieved values.		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	}
	void loop(){
	int value;
	value = Lib. readAdc (1); //Read 1 pins
	}		

readModule

Table 16. readModule

API function	readModule()		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	void	None	None
Return Values	Type		Description
	int		AnalogRead()
Processing outline	1. Perform analogRead() on the A0 pins and return the retrieved values.		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	}
	void loop(){
	int value;
	value = Lib. readModule(); //Read the analogue voltages of the A0 pins
	}		

setDelay

Table 17. setDelay

API function	setDelay(uint32_t msec)		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	uint32_t	msec	Waiting time
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. Wait for msec		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib. setDelay (1000);//1 second wait
	}
	void loop() {
	}		

getLockStatus

Table 18. getLockStatus

API function	getLockStatus()		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	void	None	None
Return Values	Type		Description
	int		0: Restraint Protection OFF 1: Restraint Protection ON
Processing outline	1. Return the restrained protected ON/OFF status		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib.setControl(0); // Open Loop
	Lib. setStartFlag (1);// Motor revolution
	}
	void loop() {
	Lib.setPwmDuty(30); // PWM Duty 30[%]
	Lib. getLockStatus();// Restriction protected status checked
	Lib. setDelay (1000);//1 second wait
	Lib.setPwmDuty(0); // PWM Duty 0[%]
	Lib. getLockStatus();// Restriction protected status checked
	Lib. setDelay (1000);//1 second wait
	}		

getSpeed

Table 19. getSpeed

API function	getSpeed()		
Class	Lib_LV8121V		
Attribute	Public		
Parameters	Type	Variable	Description
	void	None	None
Return Values	Type		Description
	int		Motor speed [Value Range] 0 to 48000 [rpm]
Processing outline	1. Return the actual motor speed		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Lib. initLib();// Parameter initialization
	Lib. setParameters (4,12,0.0044);//Log of Poles 4 [pole/2], Power Supply Voltage 12 [V], Back Electromotive Force Constant 0.0044 [V/rpm]
	Lib.setControl(1); // Control Mode Closed Loop
	}
	void loop() {
	Lib. setSpeed (3000);//3000 rpm
	Lib. setStartFlag (1);// Motor revolution
	Lib. setDelay (1000);//1 second wait
	Lib. getSpeed();// Gets the motor speed.
	Lib. setSpeed (0);//3000 rpm
	Lib. setDelay (1000);//1 second wait
	Lib. getSpeed();// Gets the motor speed.
	}		

guiSerialRead

Table 20. guiSerialRead

API function	guiSerialRead()		
Class	GuiSerialInterface		
Attribute	Public		
Parameters	Type	Variable	Description
	void	None	None
Return Values	Type		Description
	void		None
Processing outline	1. Bytestream is formed by collecting data in units of Byte transmitted by serial communication. 2. Analyze Bytestream content to Call APIs. See chapter 4.2 for API message format for serial communication. 3. Call from the sketch program loop() function is assumed, and Call by serial communication is not supported.		

Examples usage (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	void setup() {
	Serial.begin(lib.DEFAULT_BAUDRATE);
	}
	void loop() {
	Lib.guiSerialRead();
	}		

guiSerialParse

Table 21. guiSerialParse

API function	guiSerialParse(char *serialRecvStr)		
Class	GuiSerialInterface		
Attribute	Public		
Parameters	Type	Variable	Description
	*char**	serialRecvStr	Analysis of Serial communication with GUIs
Return Values	Type		Description
	int		0: Success 1: Fail
Processing outline	1. If the argument is outside the range, it returns failure (1). 2. Data analysis of Serial communication with GUIs is performed.		

Examples usage 1 (sketch)	#include <LV8121_Lib.h>// Read LV8121VAPI Library
	Lib_LV8121VLib;//Lib_LV8121V instance
	Void setup() {
	Serial.begin(lib.DEFAULT_BAUDRATE);
	}
	Void loop() {
	Lib.guiSerialRead();
	}		

Examples usage 2 (sketch)	// You can customize and use the serial interface by overriding guiSerialParse(). //
	
	//Create a derived class by inheriting the Lib_LV8121V class//
	Class Lib_LV8121_custom : public Lib_LV8121V{
	public:
	virtual ~Lib_LV8121_custom() {}
	int guiSerialParse(char *type) override;
	};
	Lib_LV8121_custom Ex;// instantiation of inheritance class
	
	// Overrides guiSerialParse Functions//
	int Lib_LV8121_custom::guiSerialParse(char *serialRecvStr){
	// Implement for serial code execution.
	Switch (serialRecvStr[0]){
	Case ‘a’:
	{
	Serial.println(”Command”);
	Return SUCCESS;// returns success (0).
	}
	Default:
	{
	Serial.println(”unknown command”);
	}
	}
	return FAILURE;// returns failed (1).
	}
	void setup() {
	Serial.begin(19200); // Baud rate 19200 open ports
	Ex. initLib();// Initialization
	}
	void loop() {
	Call the guiSerialParse process in the guiSerialRead();//guiSerialRead process.
	}		

SERIAL INTERFACE SPECIFICATIONS

This section describes the serial interface for the USB connection between the computer and the Arduino Micro. The BLDC motor may be controlled using LV8121V by implementing the serial communication settings and the guiSerialRead function in a program on the Arduino side (sketch) and by sending messages matched with individual APIs from the computer through serial communication.

For the method of implementing the guiSerialRead function, refer to guiSerialRead.

Table 22. MESSAGE LIST

#	Command Value	Command Name	Command Description	Length	Chapter
1	0x03	None	For acquiring API library identification ID.	1 byte	getId
2	0x04	timeoutPol	Sent at uniform intervals for monitoring the serial connection.	1 byte	timeoutPol
3	0x05	initLib	For calling same name API as command to set the initialize the API settings.	1 byte	initLib
4	0x41	setStartFlag	For calling same name API as command to set the start/stop of ICs.	2 byte	setStartFlag
5	0x42	setControl	For calling same name API as command to set the control modes.	2 byte	setControl
6	0x43	setDirection	For calling same name API as command to set the rotational directions.	2 byte	setDirection
7	0x45	setPwmDuty	For calling same name API as command to set the output voltages at the time of Open Loop.	2 byte	setPwmDuty
8	0x46	setLimPwmDuty	For calling same name API as command to set the minimum output voltage at the time of Open Loop.	2 byte	setLimPwmDuty
9	0x47	setSpeed	For calling same name API as command to set the target speed for Closed Loop.	3 byte	setSpeed
10	0x48	readModule	For calling same name API as command to identify the Module.	1 byte	readModule
11	0x4A	setPIParameter	For calling same name API as command to set the Closed-Loop PI gain.	9 byte	setPIParameter
12	0x4B	getTimerStatus	For calling same name API as command to read the motor rotational speed and the specified analogue input−pin voltage.	2 byte	getTimerStatus
13	0x4D	setParameters	For calling same name API as command to set the power supply voltages, the number of electrode logs of the motor, and the back electromotive force constants to be used.	10 byte	setParameters
15	0x64	readAdc	For calling same name API as command to read the ADC converted value of the any analog input−pin(A1−A5) voltage.	2 byte	readAdc

Message Configuration Details

getId

- Command description

This is a command for acquiring library identification data.

Upon the receipt of this command, the API returns identification data that specifies the corresponding motor driver name, the API version and other details.

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x03

timeoutPol

- Command description

This command is for monitoring the state of the serial connection with the Arduino Micro.

The GUI sends this command every second to the Arduino Micro. If three seconds elapse without receiving data from serial communication (including this command), the API will call the timeoutPole function to automatically stop the motor for failsafe purposes.

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x04

initLib

- Command description

This command is used to initialize API settings.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the initLib function.

For details of the initLib function, refer to initLib

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x05

setStartFlag

- Command description

This command is for calling the setStartFlag function to configure the Power ON/OFF of ICs.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setStartFlag function.

For details of the setStartFlag function, refer to setStartFlag

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	SELECT
Value	0x41	0x00 − 0x01

Field SELECT: Start/Stop switching

setControl

- Command description

This command is for calling the setControl function to set the control mode.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setControl function.

For details of the setControl function, refer to setControl

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	CONTROL
Value	0x42	0x00 − 0x01

Field CONTROL: Rotational Forward/Reverse switching

setDirection

- Command description

This command is for calling the setDirection function to set the direction of the motor.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setDirection function.

For details of the setDirection function, refer to setDirection

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	SELECT
Value	0x43	0x00 − 0x01

Field SELECT: Rotational Forward/Reverse switching

setPwmDuty

- Command description

This command is for calling the setPwmDuty function to set the output voltage at the time of Open Loop.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setPwmDuty function.

For details of the setPwmDuty function, refer to setPwmDuty

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	DUTY
Value	0x45	0x00 − 0x64

Field DUTY: Output Voltage Duty [%]

setLimPwmDuty

- Command description

This command is for calling the setLimPwmDuty function to set the minimum output voltage at the time of Open Loop.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setLimPwmDuty function.

For details of the setLimPwmDuty function, refer to setLimPwmDuty

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	DUTY
Value	0x46	0x00 − 0x64

Field DUTY: Minimum Output Voltage Duty [%]

setSpeed

- Command description

This command is for calling the setSpeed function to set the rotational speed at the time of Closed Loop.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setSpeed function.

For details of the setSpeed function, refer to setSpeed

Command from GUI to Motor Driver Kit

Byte	0	1 Low	2 High
Field	CMD	SPEED	
Value	0x47	0x00 − 0xBB80	

Field SPEED: Rotational speed [rpm]

readModule

- Command description

This command is for calling the readModule function to identify connected modules.

The guiSerialParse function calls the readModule function and returns the read analog−voltage values to the sender along with the CMD 0x49.

For details of the readModule function, refer to readModule

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x48

Field PIN: Read pin number

Command from Motor Driver Kit to GUI

Byte	0	1 Low	2 High
Field	CMD	RECVMODULE	
Value	0x49	0x0000 − 0x03FF	

Field RECVMODULE: Analog Voltage by Model (0 to 1023)

setPIParameter

- Command description

This command is for calling the setPIParameter function to set the PI Gain at the time of Closed Loop.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setPIParameter function.

For details of the setPIParameters function, refer to setPIParameters

Command from GUI to Motor Driver Kit

Byte	0	1 Low	2	3	4 High
Field	CMD	PVALUE			
Value	0x4A	0x00000000 − 0x03F80000			

Byte	5 Low	6	7	8 High
Field	IVALUE			
Value	0x00000000 − 0x03F80000			

Field PVALUE: P gain
Field IVALUE: I gain

getTimerStatus

- Command description

This command is for calling the getLockStatus function to reads the current motor speed, the specified terminal analog voltage, and the constraint protection status.

The guiSerialParse function calls the getLockStatus function and returns the reads the current motor speed, the specified terminal analog voltage, and the constraint protection status to the sender along with the CMD 0x4C.

For details of the getLock function, refer to getLockStatus

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	PIN
Value	0x4B	0x01 − 0x05

Field PIN: Terminal No. (A1−A5)

Command from GUI to Motor Driver Kit

Byte	0	1 Low	2 High	3 Low	4 High	5
Field	CMD	ADC		ACT_SPEED		LOCK_STATUS
Value	0x4C	0x0000 − 0x03FF		0x00 − 0xBB80		0x00 − 0x01

Field ADC: Analogue Voltage Values (0 to 1023)
Field ACT_SPEED: Current Rotational Speed [rpm]
Field LOCK_STATUS: Restriction protected status

setParameters

- Command description

This command is for calling the getLockStatus function to set the power supply voltage, electrode logarithm, and back electromotive force constant.

The guiSerialParse function converts receieved parameters into function parameter (argument) format and calls the setParameters function.

For details of the setParameters function, refer to setParameters

Command from GUI to Motor Driver Kit

Byte	0	1	2 Low	3	4	5 High
Field	CMD	POLEPAIRS	VOLT			
Value	0x4D	0x01 − 0x80	0x41000000 − 0x420C0000			

Byte	6 Low	7	8	9 High
Field	BEMFCON			
Value	0x38D1B717 − 0x3D23D70A			

Field POLEPAIRS: polar logarithm
Field VOLT: Power supply
Field BEMFCON: Back EMF Constants

readAdc

- Command description

This command is used to call readAdc function and readthe analog − voltage.

guiSerialParse function call readAdc function and returns the read analog − voltage values to the transmitter along with the CMD 0x62.

For details of the readAdc function, refer to readAdc

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	CH
Value	0x64	0x01 − 0x05

Field CH: Analog pins (A1−A5)

Command from Motor Driver Kit to GUI

Byte	0	1 Low	2 High
Field	CMD	RECVADC	
Value	0x62	0x0000 − 0x03FF	

Field RECVADC: Analog Voltage Values (0 to 1023)

OVERVIEW OF INTERNAL PARAMETERS, TABLES AND FUNCTIONS

Internal Parameters List

The table below provides a list of internal parameters.

Table 23. LIST OF INTERNAL PARAMETERS

#	Parameter	Initial Value	Description	Timing of Update
1	isRotation	STOP	Motor rotation state 0: STOP 1: START	Change by setStartFlag
2	controlMode	OPEN_LOOP	Control mode 0: Open Loop 1: Closed Loop	Change by setControl
3	pole	POLEPAIRS_MIN * 2	Number of the poles	Change by setParameters
4	v_in	VOLTAGE_MIN	Power supply voltage	Change by setParameters
5	bemf_const	BEMF_MIN	Back electromotive force constant	Change by setParameters
6	diff[2]	0	Closed Loop deviations Diff[0]: Previous deviations Diff[1]: this time deviations	Change by pidControl
7	max_rpm	RPM_MIN	Maximum rotation speed [rpm]	Change by setParameters
8	target_spd	RPM_MIN	Target rotation speed [rpm]	Change by setSpeed
9	act_spd	RPM_MIN	Current rotation speed [rpm]	calcSpeed timerFire Change by judgeLock
10	pps	0	Number of FG pulses per second	Change by calcSpeed
11	time[3]	{0, 0, 0}	Array for measuring one cycle time of FG pulse	Change by pulseFire
12	pulse_switch	False	FG pulse edge detection flag	timerFire Change by pulseFire
13	pulse_count	0	FG pulse period counter	timerFire Change by judgeLock
14	Kp	0.01	P gain	Change by setPIParameter
15	Ki	0.01	I gain	Change by setPIParameter
16	Mv	300	Manipulated variable	Change by pidControl
17	integral	0	I manipulated variable	Change by pidControl
18	lock_status	LOCK_OFF	Restraint protection state	Change by judgeLock
19	STOP	0	Motor stop	Fixed value
20	START	1	Motor start	Fixed value
21	OPEN_LOOP	0	Open Loop	Fixed value
22	CLOSED_LOOP	1	Closed Loop	Fixed value
23	CTLVOLT_MIN	1.2	CTL minimum voltage	Fixed value
24	CTLVOLT_MAX	3.4	Maximum CTL voltage	Fixed value
25	LIMVOLT_MIN	1.2	LIM minimum voltage	Fixed value
26	LIMVOLT_MAX	3.4	LIM maximum voltage	Fixed value
27	DUTYRATE_MIN	0	Duty[%] Minimum value (0)	Fixed value
28	DUTYRATE_MAX	100	Duty[%] Maximum value (100)	Fixed value

Table 23. LIST OF INTERNAL PARAMETERS (continued)

#	Parameter	Initial Value	Description	Timing of Update
29	RPM_MIN	0	Motor minimum speed [rpm]	Fixed value
30	RPM_MAX	48000	Maximum motor speed [rpm]	Fixed value
31	DIRECTION_FW	0	Normal rotation	Fixed value
32	DIRECTION_RV	1	Reversal	Fixed value
33	POLEPAIRS_MIN	1	Extreme log minimum	Fixed value
34	POLEPAIRS_MAX	128	Extreme log maximum	Fixed value
35	VOLTAGE_MIN	8	Minimum power supply voltage	Fixed value
36	VOLTAGE_MAX	35	Maximum power supply voltage	Fixed value
37	BEMF_MIN	0.0001	Minimum counter electromotive force constant	Fixed value
38	BEMF_MAX	0.0400	Maximum counter electromotive force constant	Fixed value
39	REF_POLEPAIRS	4	Reference motor rating Pairs of poles	Fixed value
40	REF_VOLTAGE	12	Reference rated power supply voltage	Fixed value
41	REF_BEMF	0.0044	Reference motor rating Back electromotive force constant	Fixed value
42	PIN_NUMBER_MIN	1	Minimum pin number (A1)	Fixed value
43	PIN_NUMBER_MAX	5	Maximum pin number (A5)	Fixed value
44	LOCK_OFF	0	Restraint Protection OFF	Fixed value
45	LOCK_ON	1	Restraint protection ON	Fixed value
46	KP_MIN	0.01	Minimum P gain	Fixed value
47	KP_MAX	1.00	Maximum P gain	Fixed value
48	KI_MIN	0.01	I gain minimum	Fixed value
49	KI_MAX	1.00	Maximum I gain	Fixed value
50	DELTA_T	0.01	PI control sampling period	Fixed value
51	ADC_READ_PINSETS[]	[PIN_NUMBER_MAX + 1]		Fixed value
52	FAILURE_ADC	0xFFFF	Return values for failed readAdc	Fixed value
53	SRMES_GET_ID	0x03	getId compliant serial messages Identifiers (0x03)	Fixed value
54	SRMES_POLLING_ID	0x04	timeOutPole compliant serial messages Identifiers (0x04)	Fixed value
55	SRMES_SET_INITIAL	0x05	initLib API compliant serial messages Identifiers (0x05)	Fixed value
56	SRMES_START_FLAG	0x41	setStartFlag API response Serial message Identifiers (0x41)	Fixed value
57	SRMES_SET_CONTROL	0x42	setControl API compliant serial messages Identifiers (0x42)	Fixed value
58	SRMES_DIRECTION	0x43	setDirection API compliant serial messages Identifiers (0x43)	Fixed value
59	SRMES_PWM_DUTY	0x45	setPwmDuty API compliant serial messages Identifiers (0x45)	Fixed value
60	SRMES_LIM_PWM_DUTY	0x46	setLimPwmDuty API response Serial message Identifiers (0x46)	Fixed value

Table 23. LIST OF INTERNAL PARAMETERS (continued)

#	Parameter	Initial Value	Description	Timing of Update
61	SRMES_SET_SPEED	0x47	setSpeed API compliant serial messages Identifiers (0x47)	Fixed value
62	SRMES_READ_MODULE	0x48	readModule API compliant serial messages Identifiers (0x48)	Fixed value
63	SRMES_RES_READ_MOD-ULE	0x49	Serial messages for readModule API re-sponses Identifiers (0x49)	Fixed value
64	SRMES_SET_PIPARAME-TER	0x4A	setPIParameter API response Serial message Identifier (0x4A)	Fixed value
65	SRMES_GET_TIMER_STA-TUS	0x4B	getTimerStatus compliant serial messages Identifier (0x4B)	Fixed value
66	SRMES_RES_TIMER_STA-TUS	0x4C	Serial messages for getTimerStatus re-sponses Identifier (0x4C)	Fixed value
67	SRMES_SET_PARAME-TERS	0x4D	setParameter API response Serial message Identifier (0x4D)	Fixed value
68	SRMES_RES_READ_ADC	0x62	Serial messages for readAdc responses Identifiers (0x62)	Fixed value
69	SRMES_READ_ADC	0x64	readAdc API response Serial message Identifiers (0x64)	Fixed value

List of Internal Structures

A list of internal structures is shown below.

Table 24. LIST OF INTERNAL STRUCTURES

#	Structure	Description
1	SrMesDivStartFlag	Structure containing serial communication parameters for setStartFlag
2	SrMesDivSetControl	Structure containing serial communication parameters for setControl
3	SrMesDivDirection	Structure containing serial communication parameters for setDirection
4	SrMesDivPwmDuty	Structure containing serial communication parameters for setPwmDuty
5	SrMesDivLimPwmDuty	Structure containing serial communication parameters for setLimPwmDuty
6	SrMesDivSetSpeed	Structure containing serial communication parameters for setSpeed
7	SrMesDivPIParameter	Structure containing serial communication parameters for setPIParameter
8	SrMesDivGetTimerStatus	Structure containing serial communication parameters for getTimerStatus
9	SrMesDivRecvGetTimerStatus	Structure containing serial communication parameters for getTimerStatus (transmission)
10	SrMesDivReadAdc	Structure containing serial communication parameters for readAdc
11	SrMesDivRecvReadAdc	Structure containing serial communication parameters for readAdc (transmission)
12	SrMesDivParameters	Structure containing serial communication parameters for setParameter
13	SrMesDivRecvReadModule	Structure containing serial communication parameters for readModule (transmission)

DETAILS OF TIMER RESISTOR SETTINGS

The register setting of the ATmega32U4 used in this API will be described. See ATmega16U4−32U4_Datasheet.pdf for more information about the register.

Table 25. RELATED ATmega32U4 REGISTERS

#	Register name	Description	Chapter
1	TCCR0B	Timer/counter 0 control register B	TCCR0B
2	TCCR4B	Timer/counter 4 control register B	TCCR4B

TCCR0B Setting

TCCR0B is a register for the 8bit counter/timer 0. The contents of the table below for each bit are shown.

Table 26. TCCR0B REGISTERS

Bit	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	FOC0A	FOC0B	−	−	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W
Initial value Arduino Micro	0	0	0	0	0	0	1	1

For LV8121V drivers, the PWM frequency used in the IO11 Pin is fixed to 62.500 [kHz]. Therefore, the register is set so that the division ratio becomes ”1” according to the following calculation formula.

[Fast PWM Frequency Calculation Formula]

(clock frequency/division ratio) × (1/counter number) [Hz]

= {(16.0 × 10^6 / 1) × (1/256)} / 10^3 = 62.500 [kHz]

Since the division ratio is determined by the TCCR0B CS(Clock Selector) 3−bit combinations as shown in the following table, (CS02,CS01,CS00)=(0, 0, 1) is set.

Table 27. TIMER COUNTER 1 INPUT CLOCK SELECTION

CS02	CS01	CS00	Division Ratio
0	0	0	Timer counter Stop operation
0	0	1	1/1 (without pre−division)
0	1	0	1/8
0	1	1	1/64
1	0	0	1/256
1	0	1	1/1024
1	1	0	External clocking
1	1	1	External clocking

TCCR4B Setting

TCCR4B is a 10−bit fast timer/counter 4 register. The contents of the table below for each bit are shown.

Table 28. TCCR4B REGISTERS

Bit	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	PWM4X	PSR4	DTPS41	DTPS40	CS43	CS42	CS41	CS40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value Arduino Micro	0	0	0	0	0	1	1	1

In the LV8121V motor driver control, the PWM frequency used in the D6 Pin is set to a fixed frequency of 31.373 [kHz]. Therefore, according to the following calculation formula, it is necessary to set the register so that the frequency division ratio becomes “1”.

[Phase Correct PWM Frequency Calculation Formula]

(clock frequency/division ratio) × (1/(TOP × 2))[Hz]

= {(16.0 × 10^6 /1) × (1/510)} / 10^3 = 31.373 [kHz]

Since the division ratio is determined by the TCCR4B CS(Clock Selector) 4−bit combinations as shown in the following table, (CS43,CS42,CS41,CS40)=(0, 0, 0, 1) is set.

Table 29. TIMER/COUNTER CLOCK 4 INPUT CLOCK SELECTION

CS43	CS42	CS41	CS40	Division ratio
0	0	0	0	Timer counter Stop operation
0	0	0	1	1/1
0	0	1	0	1/2
0	0	1	1	1/4
0	1	0	0	1/8
0	1	0	1	1/16
0	1	1	0	1/32
0	1	1	1	1/64
1	0	0	0	1/128
1	0	0	1	1/256
1	0	1	0	1/512
1	0	1	1	1/1024
1	1	0	0	1/2048
1	1	0	1	1/4096
1	1	1	0	1/8192
1	1	1	1	1/16384

Setting of the register is performed as part of initialization in the LV8121V motor−compatible API Lib_LV8121V::initLib functions. (See API Function Specifications for details.)

Special Notes

Impact of TCCR0 Change

The LV8121V Motor Driver Control APIs have changed TCCR0, which affects the default library functions of the Arduino Micro. Since the following functions do not operate normally, it is necessary to make them as necessary.

tone()、analogRead()、micros()、millis()、delayMicroseconds()、delay()(*)

* In this API, setDelay functions corresponding to delay() are prepared, and the API can obtain a result equivalent to normal delay() by using the API.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

ON Semiconductor Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative