

BLDC Shield with TLE9879QXA40 for Arduino

About this document



Scope and purpose

This application note describes the BLDC shield for Arduino equipped with the TLE9879QXA40, a single chip 3-phase motor driver that integrates the industry standard Arm® Cortex®-M3 core.

This document provides detailed information on the board's content, layout and use. It should be used along with the [TLE9879QXA40 datasheet](#), which contains full technical details on the device specification and operation.

An Arduino baseboard is mandatory for operation.

Intended audience

This document is intended for anyone working with the BLDC shield for Arduino equipped with the TLE9879QXA40.

* Arm and Cortex are registered trademarks of Arm Limited, UK

Table of contents

	About this document	1
	Table of contents	2
1	Acronyms	3
2	Introduction	4
2.1	TLE9879 BLDC shield for Arduino overview	4
2.2	Key features	5
2.3	Application and block diagrams	5
3	TLE9879 BLDC shield for Arduino description	7
3.1	Overview	7
3.2	Schematics	8
3.3	Layout	11
3.4	Bill of material	12
3.5	Pin assignment and functions	14
4	Arduino software documentation	16
4.1	Constructor	16
4.2	Commands	17
4.3	Code example	22
4.4	Parameter sets for BEMF, FOC, and HALL	23
5	TLE9879QXA40 overview	26
5.1	Key features	26
5.2	Block diagram	27
5.3	Pin assignment	28
5.4	Pin definitions and functions	29
6	BLDC shield programming	31
7	Revision history	32
	Disclaimer	33

1 Acronyms

1 Acronyms

The following acronyms and terms are used within this document.

Table 1 Acronyms

Acronyms	Names
ADC	Analog Digital Converter
BLDC	Brushless Direct Current
DC	Direct Current
EMC	Electro Magnetic Compatibility
GND	Ground
IDE	Integrated Development Environment
I/O	Input/Output
LIN	Local Interconnect Network
n.u.	not used
NFET	Negative Channel Field Effect Transistor
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
VDDEXT	External Voltage Supply Output

2 Introduction

2 Introduction

2.1 TLE9879 BLDC shield for Arduino overview

The BLDC Motor Control shield with TLE9879QXA40 for Arduino Uno makes it easy to control and drive a BLDC motor. It features an Infineon TLE9879QXA40, a single chip 3-phase motor driver with integrated MOSFETs.

The shield can be controlled by a microcontroller via an SPI interface - here the Arduino Uno R3 is used as master. Both the shield and the Arduino board are connected via the pin headers.

Up to four shields can be stacked via the pin headers, to control up to four BLDC motors. Dedicated pins are used to select the relevant shield (see [Chapter 3.5](#) on page 14).

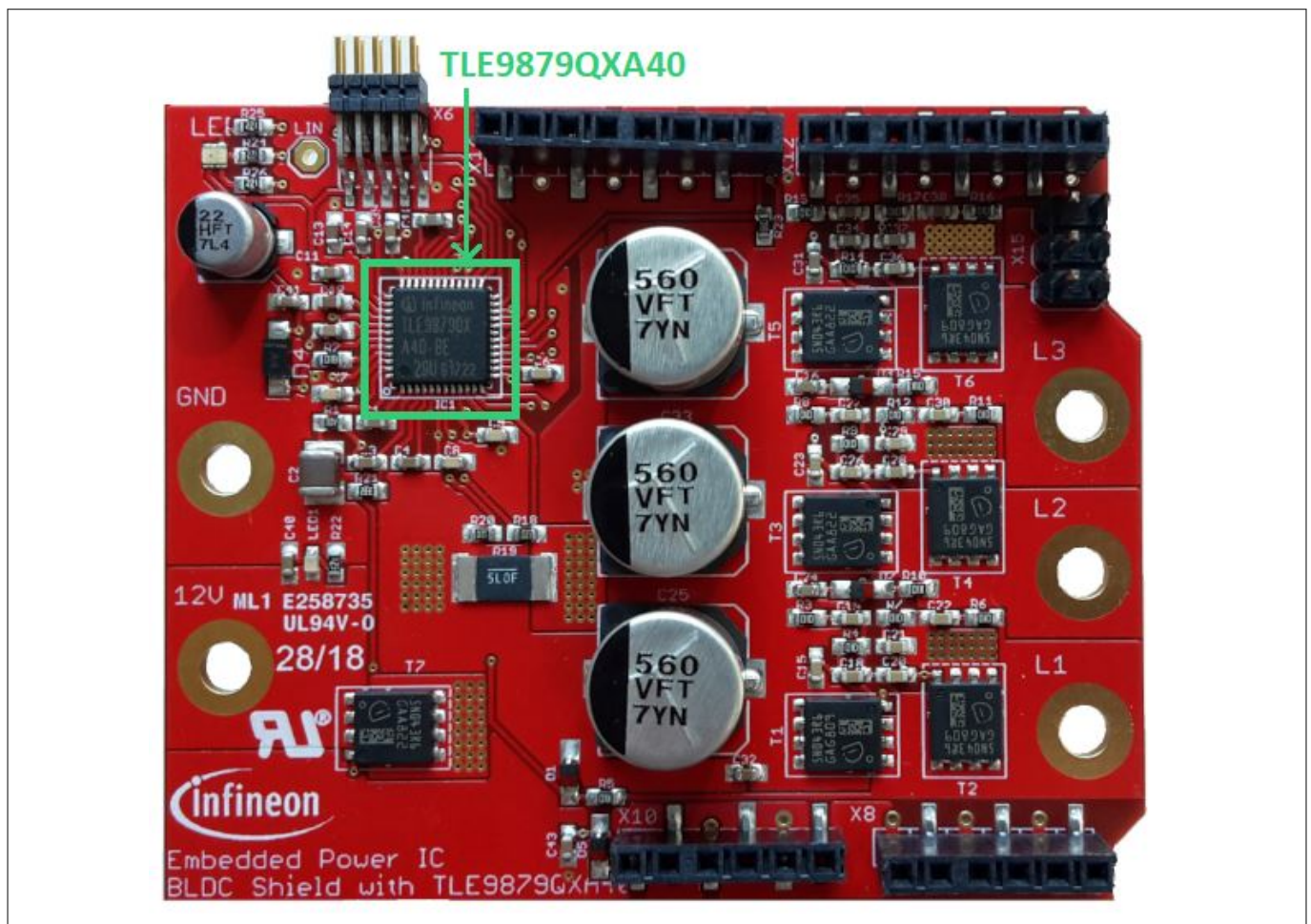


Figure 1 TLE9879 BLDC shield for Arduino overview

2 Introduction

2.2 Key features

The TLE9879 BLDC shield for Arduino has the following features:

- An Arduino Uno R3 connected to the shield can control a single chip 3-phase motor driver via the SPI interface
- BLDC motor control
 - 5.5 V - 28 V normal operating input voltage
 - 28 V - 40 V extended operating input voltage
- Possibility to stack up to four shields
- SPI interface for high configurability and detailed diagnosis
- Protections:
 - Overtemperature
 - Overcurrent
 - Undervoltage
 - Overvoltage
- Motor speed control

2.3 Application and block diagrams

The BLDC shield is meant to be controlled with an Arduino Uno or Arduino compatible baseboard. The application diagram (Figure 2 on page 5) shows the simplified connection scheme.

The auto-addressing functionality will configure a dedicated chip select, regarding the board position in the stack. An external 12 V supply voltage has to be attached to the BLDC shields to provide the right amount of current for the motor control.

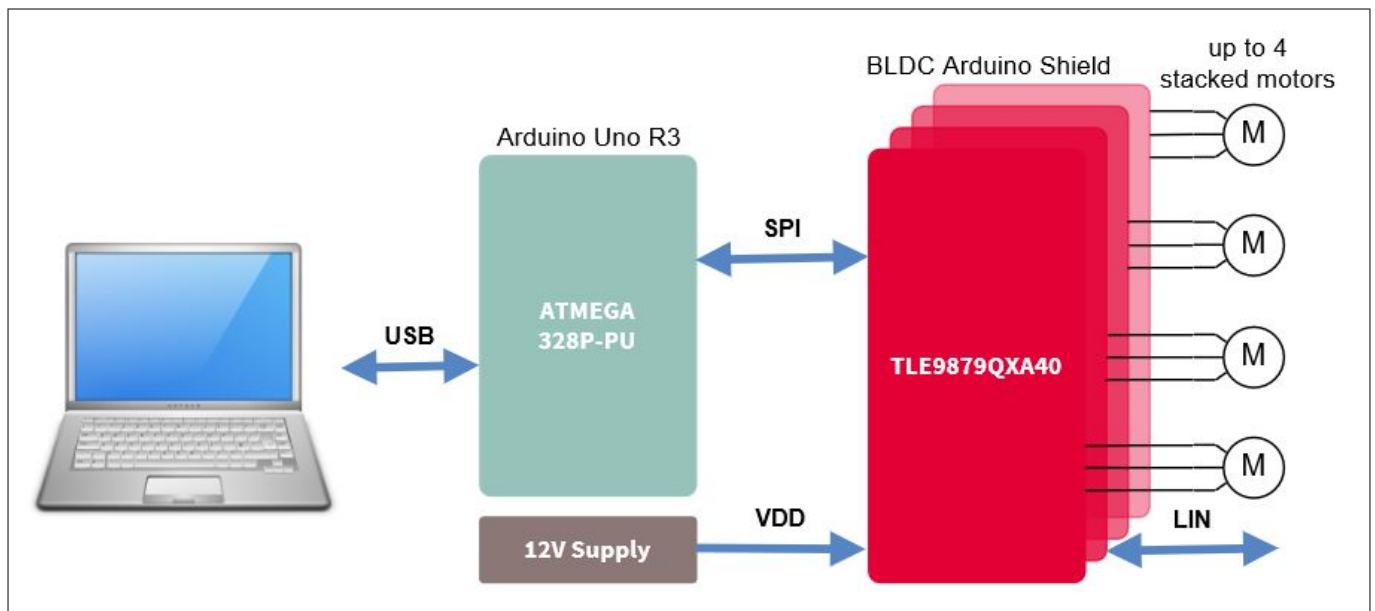


Figure 2 Simplified application diagram for TLE9879 BLDC shield for Arduino

2 Introduction

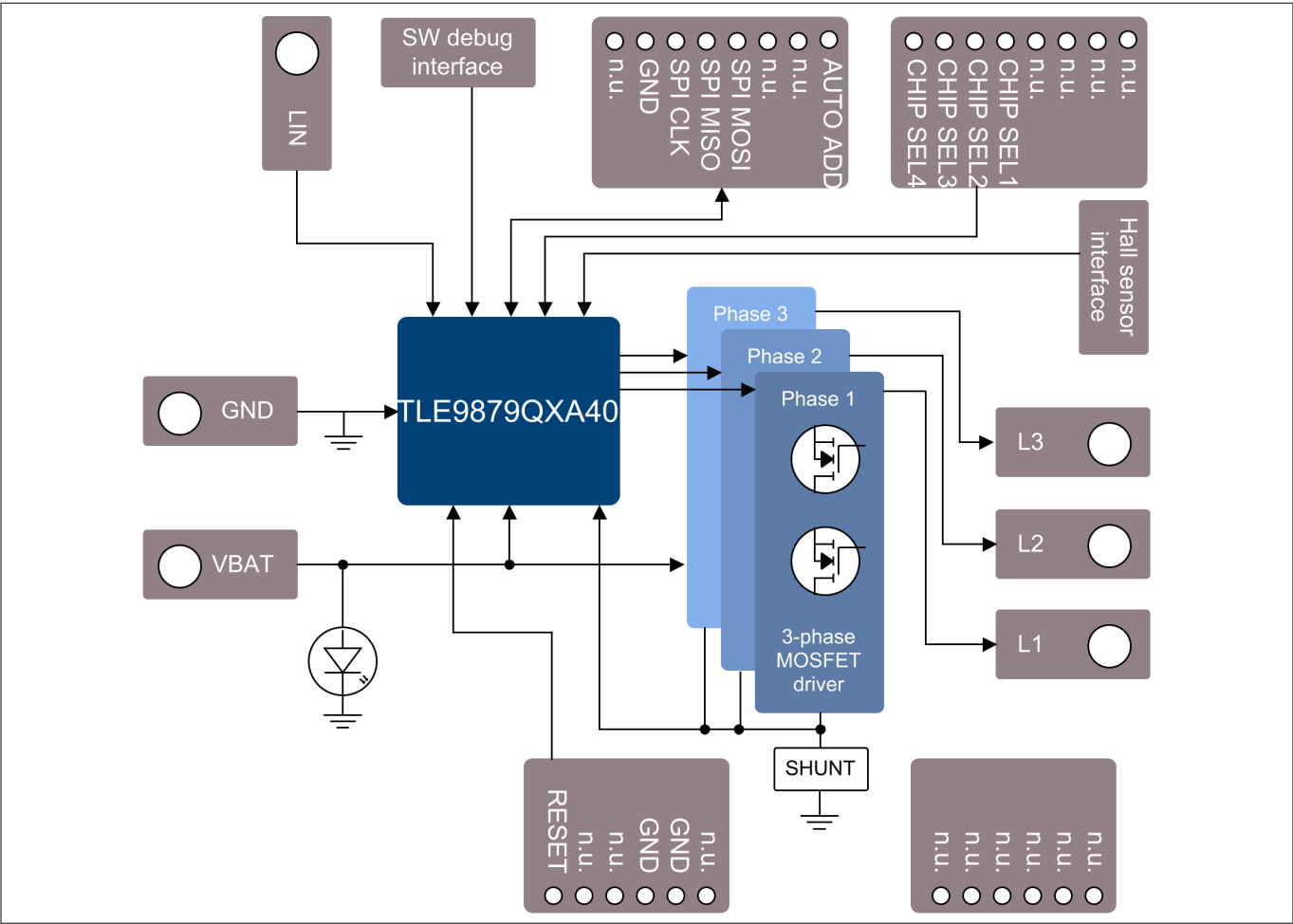


Figure 3 Block diagram for TLE9879 BLDC shield for Arduino

Refer to [Chapter 5](#) on page 26 or the corresponding datasheet for more information.

3 TLE9879 BLDC shield for Arduino description

3 TLE9879 BLDC shield for Arduino description

3.1 Overview

The TLE9879 BLDC shield for Arduino provide the following features (see [Figure 4](#) on page 7):

- A single chip 3-phase motor driver, TLE9879QXA40
- Solder holes to connect a BLDC motor - marked red
- Several communication interfaces:
 - Pin headers to connect the shield to the Arduino board - X8, X10, X12, X14, marked yellow. See the pin assignment [Chapter 3.5](#) on page 14.
 - LIN via a test point - marked green. It is not populated yet and has to be soldered in the designated solder hole.
 - Debug pin headers - X6, marked blue. It can be used by connecting a Segger J-Link debugger.
 - Hall connector - X15, marked purple. It can be used by connecting a Hall sensor for motor control purpose.
- A RGB LED - marked orange - in the upper left corner, which indicates that the shield is working properly.

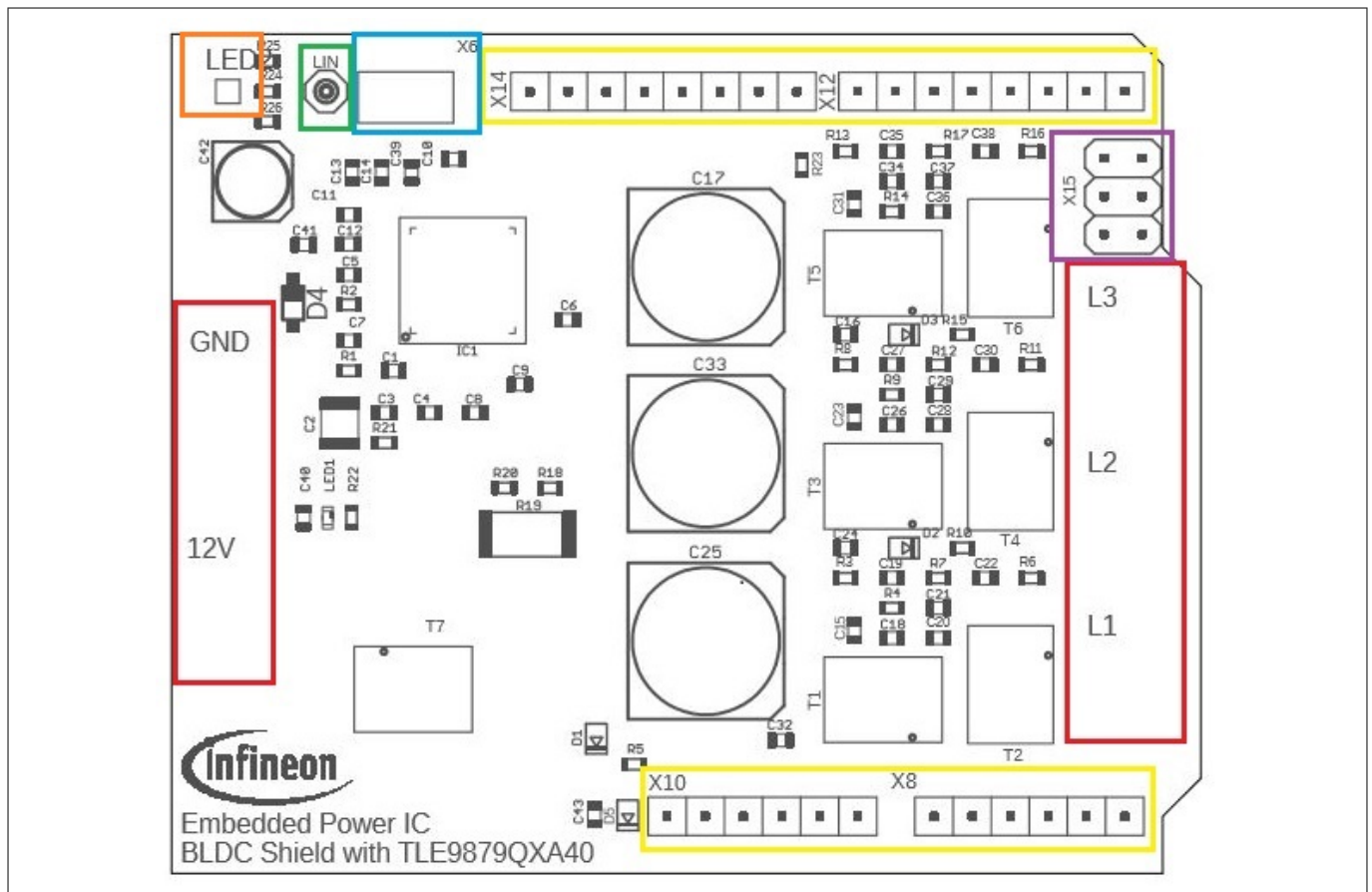


Figure 4 TLE9879 BLDC shield for Arduino placements

3 TLE9879 BLDC shield for Arduino description

3.2 Schematics

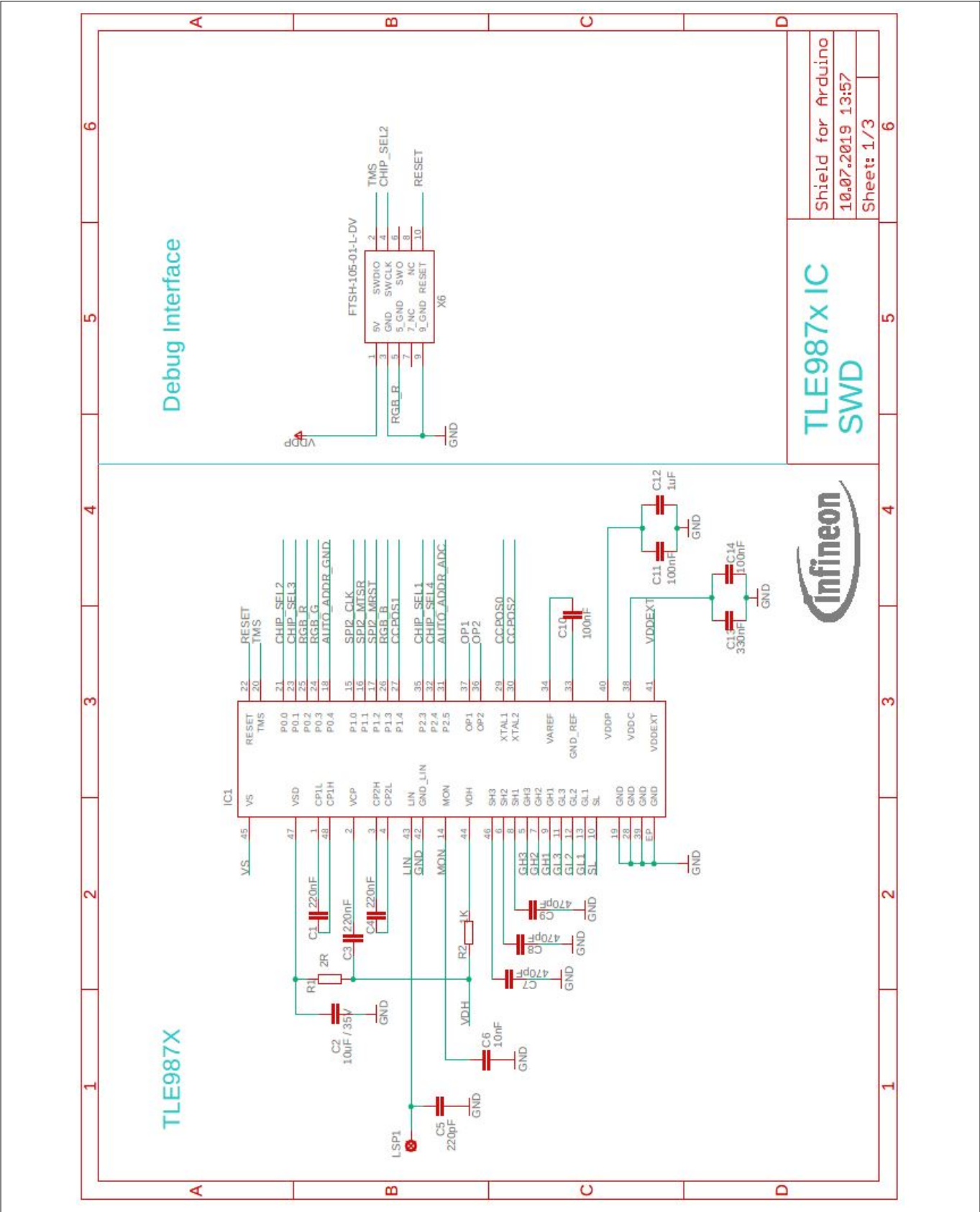


Figure 5 Schematics TLE9879 BLDC shield for Arduino (1/3)

3 TLE9879 BLDC shield for Arduino description

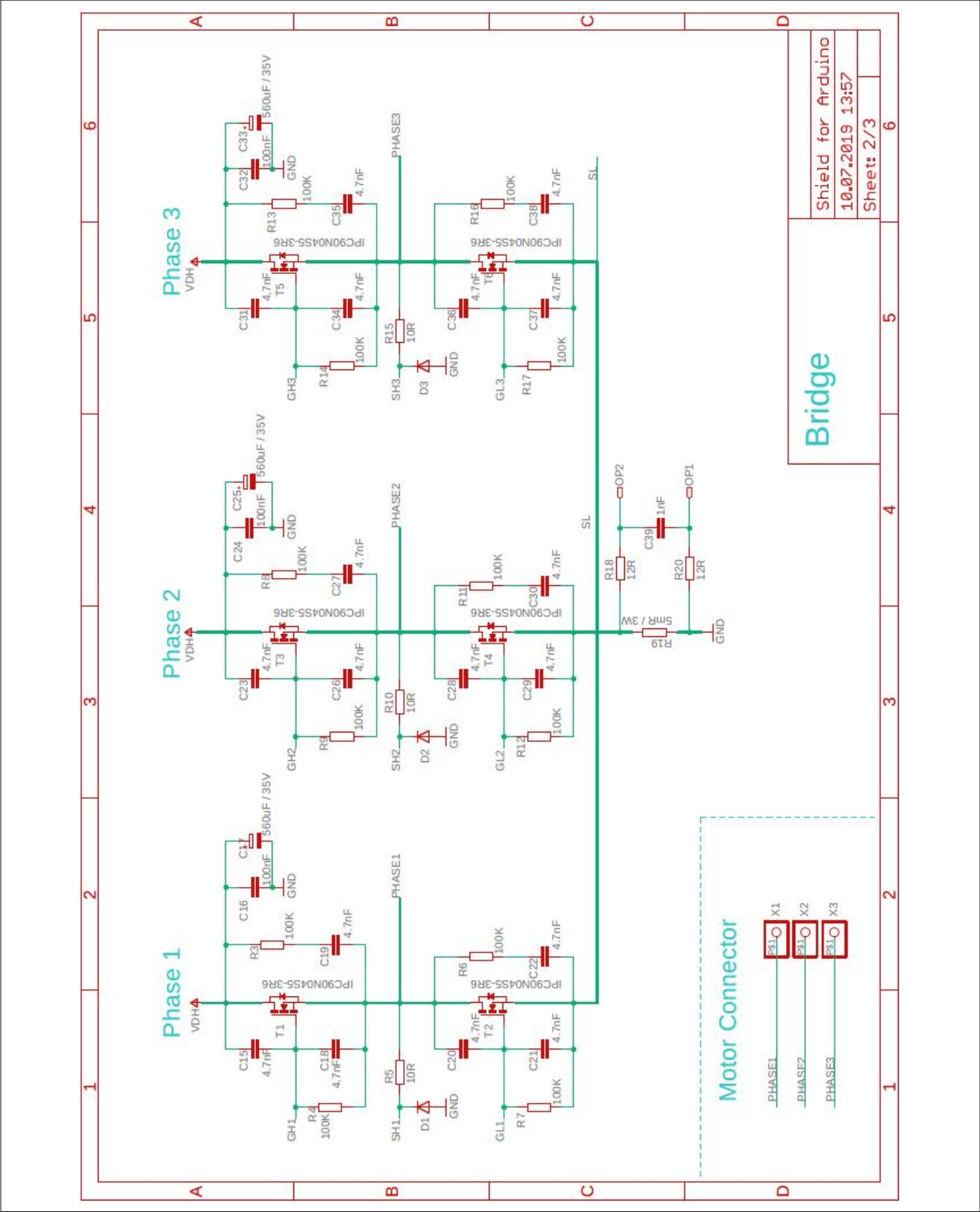


Figure 6 Schematics TLE9879 BLDC shield for Arduino (2/3)

3 TLE9879 BLDC shield for Arduino description

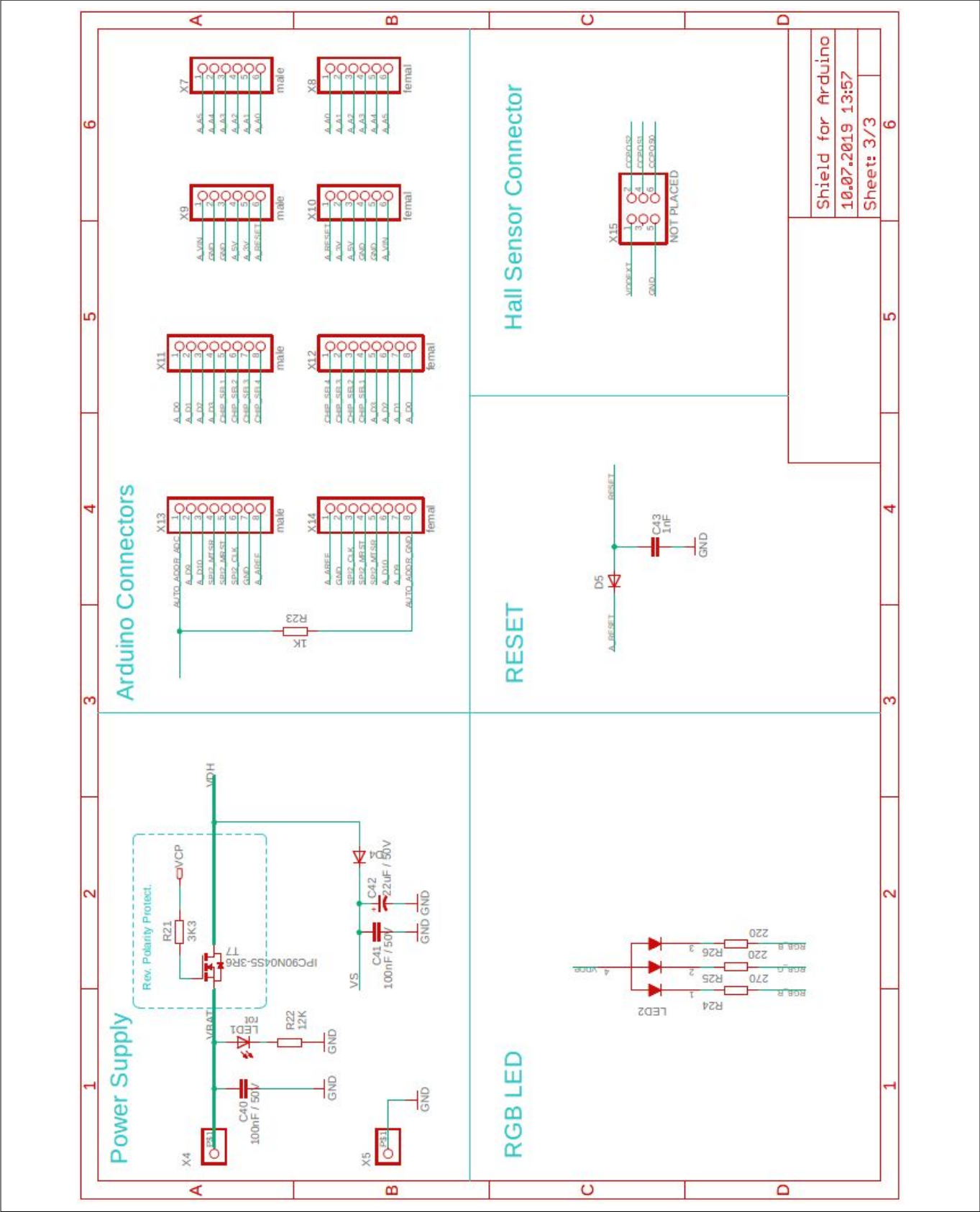


Figure 7 Schematics TLE9879 BLDC shield for Arduino (3/3)

3 TLE9879 BLDC shield for Arduino description

3.3 Layout

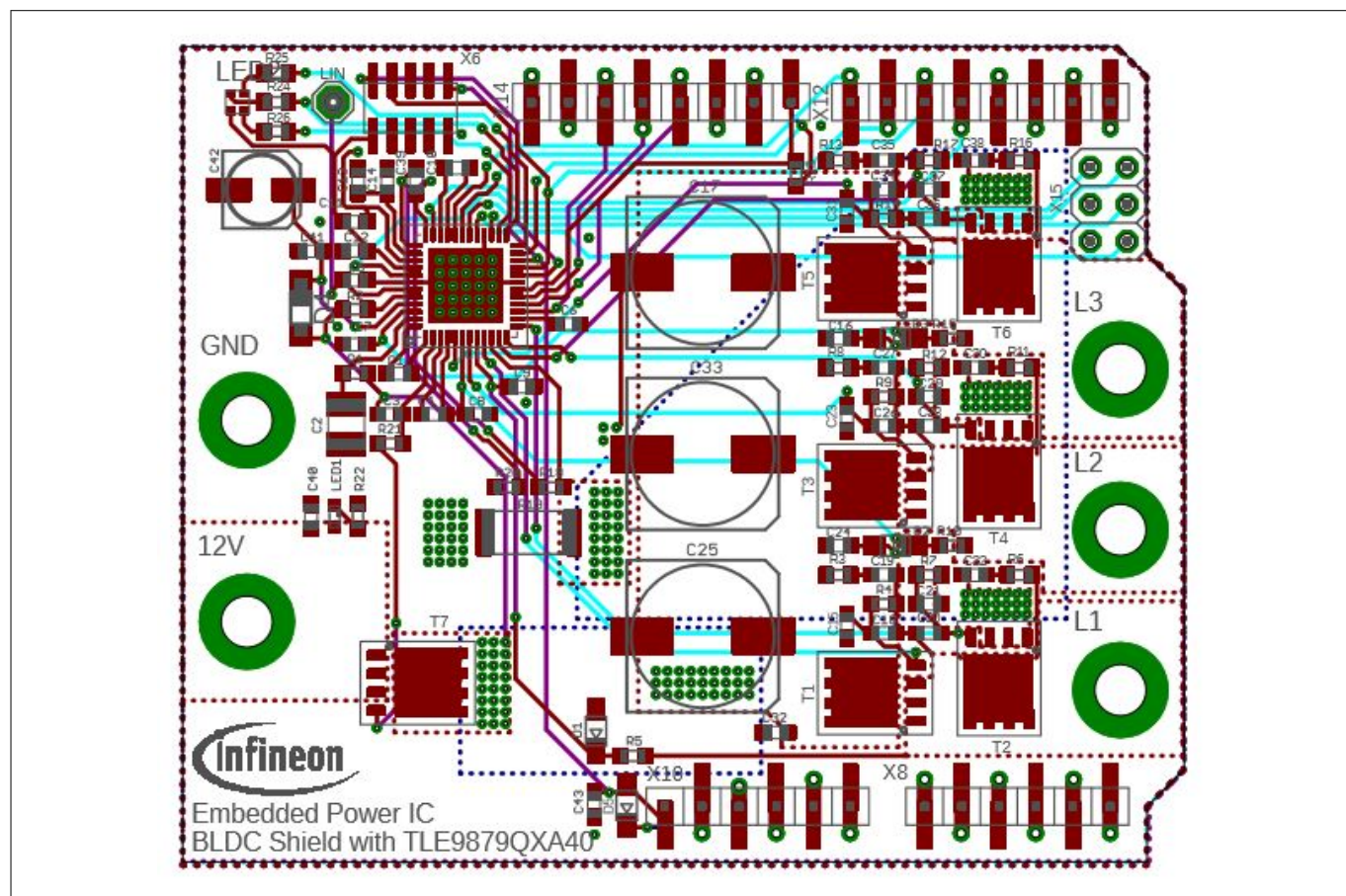


Figure 8 Top layout

3 TLE9879 BLDC shield for Arduino description

3.4 Bill of material

Table 2 Bill of material TLE9879 BLDC shield for Arduino

Part	Value	Device	Package	Description
C1, C3, C4	220 nF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C2	10 uF / 35 V	C-EUC1210	C1210	CAPACITOR, European symbol
C5	220 pF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C6	10 nF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C7, C8, C9	470 pF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C10, C11, C14, C16, C24, C32, C40, C41	100 nF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C12	1 uF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C13	300 nF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C15, C18, C19, C20, C21, C22, C23, C26, C27, C28, C29, C30, C31, C34, C35, C36, C37, C38	4.7 nF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C17, C25, C33	560 uF / 35 V	CPOL- EU153CLV-1014	153CLV-1014	POLARIZED CAPACITOR, European symbol
C39, C43	1 nF / 50 V	C-EUC0603	C0603	CAPACITOR, European symbol
C42	22 uF / 50 V	CP-153CLV-0505	153CLV-0505	POLARIZED CAPACITOR, European symbol
D1, D2, D3, D5	BAS3010B03WE6 327HTSA1	DIODE-SOD323- W	SOD323-W	DIODE
D4	SM4004	DIODE- DO-214AC	DO-214AC	DIODE
IC1	TLE9879QXA40	TLE9879QXA40	VQFN48-15_7X7	Infineon SoC
LED1	red	LEDSML0603	SML0603	LED
LED2	GM5WA94310A	GM5WA94310A	GM5WA94310A	Chip LED RGB
LSP1	MCS10B	MCS10B	MCS10B	SOLDER PAD
R1	2 / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R2, R23	1K / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R3, R4, R6, R7, R8, R9, R11, R12, R13, R14, R16, R17	100K / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R5, R10, R15	10R / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R18, R20	12R / 1%	R-EU_R0603	R0603	RESISTOR, European symbol

(table continues...)

3 TLE9879 BLDC shield for Arduino description

Table 2 (continued) Bill of material TLE9879 BLDC shield for Arduino

Part	Value	Device	Package	Description
R19	5mR / 3 W / 0.5%	R-EU_R6332	R6332	RESISTOR, European symbol
R21	3K3 / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R22	12K / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R24, R26	220 / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
R25	270 / 1%	R-EU_R0603	R0603	RESISTOR, European symbol
T1, T2, T3, T4, T5, T6, T7	IPC90N04S5-3R6	IPC90N04S5-3R6	PG-TDSON-8	
X1, X2, X3, X4, X5	SCREW HOLE	SCREW HOLE	SCREW HOLE	SCREW HOLE
X6	FTSH-105-01-L-DH	SWD	SWD	SWD
X7, X9	TSM-106-03-L-SV	PINHD-1X6_2.54-SMD	SAMTEC	PINHD
X8, X10	SSM-106-L-SV	PINHD-1X6_2.54-SMD	SAMTEC	PINHD
X11, X13	TSM-108-03-L-SV	PINHD-1X8_2.54-SMD	SAMTEC	PINHD
X12, X14	SSM-108-L-SV	PINHD-1X8_2.54-SMD	SAMTEC	PINHD
X15	PINHD-2X3_2.54	PINHD-2X3_2.54	2X03	PINHD

3 TLE9879 BLDC shield for Arduino description

3.5 Pin assignment and functions

Figure 9 on page 14 shows the pin-out/connectors of the TLE9879 BLDC shield for Arduino.

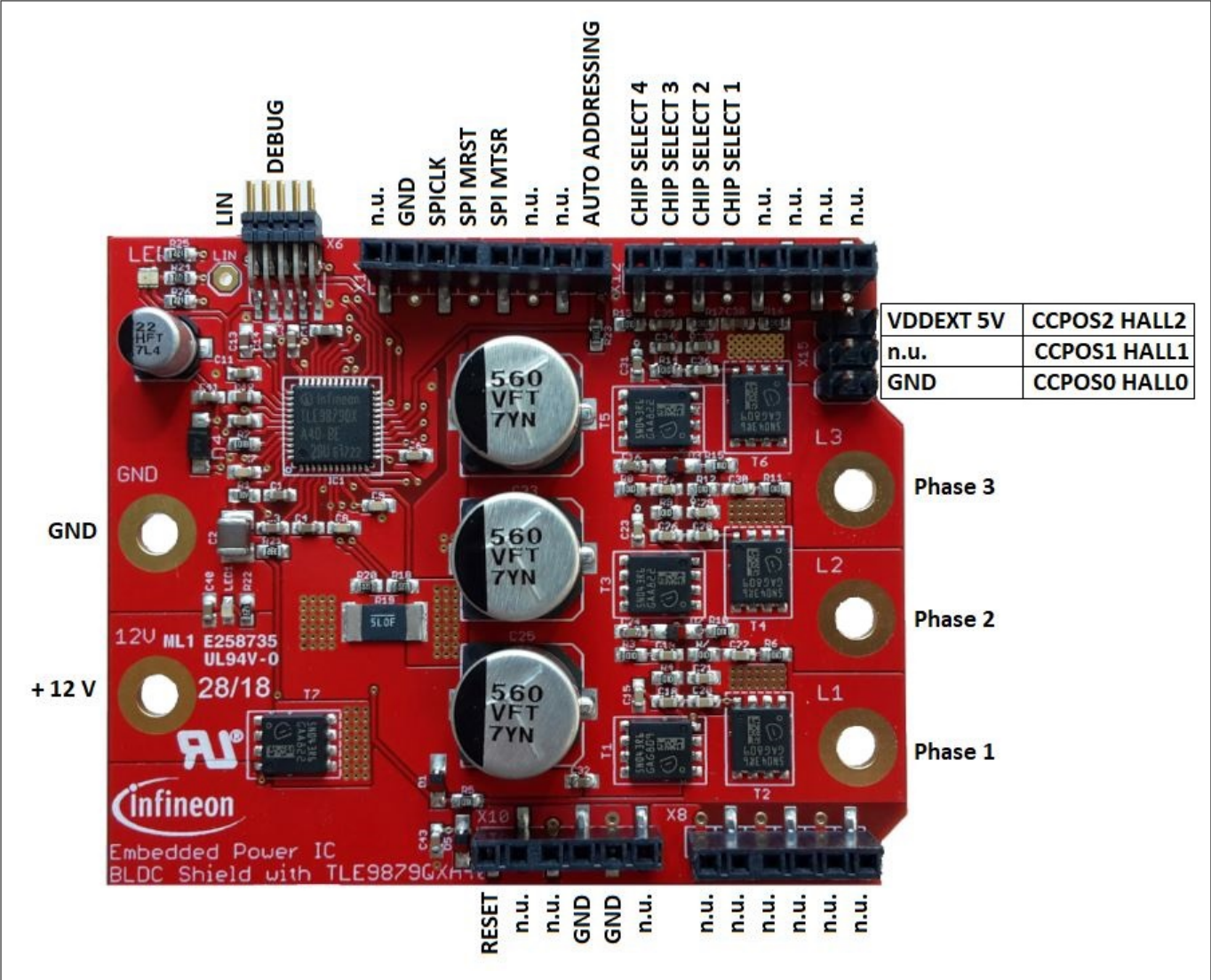


Figure 9 Pin assignment TLE9879 BLDC shield for Arduino

3 TLE9879 BLDC shield for Arduino description

Table 3 **Pin assignment**

Pin	Function
GND	Ground
SPICLK	Serial clock
SPI MRST	SPI Master Receive Slave Transmit (similar to MISO)
SPI MTSR	SPI Master Transmit Slave Receive (similar to MOSI)
Auto Addressing	When starting after a reset, gets which shield (out of the four possible) is active
CHIP SELECT 1,2,3,4	Enables to select the shield to control
RESET	Resets the shield
VDDEXT 5V	Supply voltage for the Hall sensor
CCPOS 0,1,2 HALL 0,1,2	Inputs for the Hall sensors
Phase 1,2,3	Inputs for the motor phases

4 Arduino software documentation

The TLE9879QX BLDC shield comes with several software examples for 3-phase motor control. The shield can be controlled using the SPI. Infineon provides the software library for Arduino Uno.

This chapter describes the application programming interface to control either one or multiple shields.

4.1 Constructor

At the beginning a pointer to a class TLE9879_Group needs to be created. A single shield or all of them can be addressed.

There can be up to four boards used with the Arduino library and the constructor sends a reset-request to all connected boards.

Example:

```
TLE9879_Group *shields;  
shields = new TLE9879_Group(2);
```

4.2 Commands

setMode

With setMode the mode can be changed or the current mode can be found out. A single shield or all of them can be addressed.

The following modes exist:

- BOOTLOADER: is started at the beginning as standard. Not all commands are available here only modeControl, boardControl, LEDOn and LEDOff).
- BEMF: control motor using block commutation with back-EMF.
- HALL: control motor using block commutation with hall sensors.
- FOC: control motor using FOC.
- GETCURRENTMODE: returns the current mode.

Example:

```
shields->setMode(FOC, BOARD1);
```

setMotorMode

With setMotorMode the motor can be started and stopped. A single shield or all of them can be addressed.

- START_MOTOR
- STOP_MOTOR

With invalid parameters the motor may not run.

Example:

```
shields->setMotorMode(START_MOTOR, BOARD1);
```

setMotorSpeed

SetMotorSpeed changes the motor speed in rpm. It can be changed while the motor is running. A single shield or all of them can be addressed.

Example:

```
shields->setMotorspeed(2000, BOARD1);
```

setParameter

SetParameter sets a parameter of the current dataset to a specific value. A single shield or all of them can be addressed. A list of all addressable parameter can be found in chapter [Parameter sets for BEMF, FOC, and HALL](#).

Example:

```
shields->setParameter(FOC_MAX_SPEED, 5000, BOARD1);
```

4 Arduino software documentation

saveDataset

With saveDataset the current dataset will be saved in ROM of the shield, the position needs to be named. A single shield or all of them can be addressed. See [Table 4](#) for ROM allocation.

- POSITION0
- POSITION1
- POSITION2
- POSITION3

These are the positions for custom datasets. CUSTOMDATASET0 is saved at POSITION0 and so on.

Example:

```
shields->saveDataset(POSITION0, BOARD1);
```

Table 4 ROM allocation

4k-block	NVM sector number	Start address	End address	Comment
---	32	0x1101F000		0x1101EFFF
FOC Custom Data Set	31	0x1101E000		
HALL Custom Data Set	30	0x1101D000		
BEMF Custom Data Set	29	0x1101C000		
	28	0x1101B000		
	27	0x1101A000		
	26	0x11019000		
	25	0x11018000		
FOC Infineon Data Set	24	0x11017000	0x11017FFF	End FOC
FOC	23	0x11016000		
FOC	22	0x11015000		
FOC	21	0x11014000		
FOC	20	0x11013000		
FOC	19	0x11012000		
FOC	18	0x11011000		
FOC	17	0x11010000		Start FOC
	16	0x1100F000		
Hall Infineon Data Set	15	0x1100E000	0x1100EFFF	End BC Hall
BC Hall	14	0x1100D000		
BC Hall	13	0x1100C000		
BC Hall	12	0x1100B000		
BC Hall	11	0x1100A000		Start BC Hall
	10	0x11009000		
BEMF Infineon Data Set	9	0x11008000	0x11008FFF	End BC BEMF

(table continues...)

4 Arduino software documentation

Table 4 (continued) ROM allocation

4k-block	NVM sector number	Start address	End address	Comment
BC BEMF	8	0x11007000		
BC BEMF	7	0x11006000		
BC BEMF	6	0x11005000		
BC BEMF	5	0x11004000		
BC BEMF	4	0x11003000		Start BC BEMF
	3	0x11002000		
Bootloader	2	0x11001000	0x001FFF	End BL
Bootloader	1	0x11000000		Start BL

loadDataset

The requested dataset is loaded and used. A single shield or all of them can be addressed.

There are prepared Infineon datasets:

- DATASET0
- DATASET1
- DATASET2
- DATASET3

And custom datasets:

- CUSTOMDATASET0
- CUSTOMDATASET1
- CUSTOMDATASET2
- CUSTOMDATASET3

Custom datasets are empty by default and need to be adapted before use.

Example:

```
shields->loadDataset(DATASET0, BOARD1);
```

readDataset

With readDataset the currently used dataset is sent to the master (Arduino).

Then it is stored in the appropriate class variable:

- data_BMEF
- data_FOC
- data_HALL

Example:

```
shields->readDataset(BOARD1);
```

4 Arduino software documentation

writeDataset

When requesting writeDataset, the values currently in data_BMEF, data_FOC or data_HALL will be sent to the TLE9879_Board. A single shield or all of them can be addressed.

After correct transfer the data will be used.

Note: *Always call readDataset() before calling writeDataset(). By default, no dataset is loaded into the Arduino. Calling writeDataset() in that situation causes an issue, so a dataset must be loaded first.*

Example:

```
shields->writeDataset(BOARD1);
```

copyDataset

With copyDataset dataset can be transferred from one shield to another.

Note: *The correct sequence of calls is:
readDataset() -> copyDataset() (from one board to another) -> writeDataset()*

Example:

```
shields->copyDataset(BEMF,BOARD1,BOARD2);
```

sendMessageToAll

This command sends data to all shields at once, for example, to start or stop all motors connected to the shields with a single command.

Examples:

```
shields->sendMessageToAll(MOTORCONTROL+STOP_MOTOR);  
shields->sendMessageToAll(MOTORCONTROL+START_MOTOR);
```

4 Arduino software documentation

setLed

With setLed the mode of the onboard LED can be set. A single shield or all of them can be addressed.

The modes are:

- LED_ON: switches the LED on
- LED_OFF: switches the LED off

Select the LED to be turned on or off also.

- LED_RED: Red LED
- LED_GREEN: Green LED
- LED_BLUE: Blue LED

Example:

```
shields->setLed(LED_RED, LED_ON, BOARD1);
```

setLedColor

With setLedColor different LED colors can be flashed using the RGB combination.

The colors are: COLOR_BLACK, COLOR_BLUE, COLOR_GREEN, COLOR_TURQUOISE, COLOR_RED, COLOR_PURPLE, COLOR_YELLOW, COLOR_WHITE

Example:

```
shields->setLedColor(COLOR_RED, BOARD1);
```

getMotorSpeed

getMotorSpeed outputs the current speed of the motor to the Arduino IDE terminal. (Used for FOC and Hall example).

Example:

```
shields->getMotorSpeed(BOARD1);
```

4.3 Code example

```
#include " TLE9879_Group.h"
TLE9879_Group *shields;

void setup()
{
    shields = new TLE9879_Group(1);
    // the command "start" sets motor speed to 1000 and starts the motor.
    // The configuration from the ConfigWizard is used.
    shields->setMode(FOC, BOARD1);
    // FOC is used for motor control
    shields->setMotorSpeed(1000, BOARD1);
    // motor speed is set to 1000 rpm
    shields->setMotorMode(START_MOTOR, BOARD1);
}

void loop()
{
    // do nothing here
}
```

4.4 Parameter sets for BEMF, FOC, and HALL

BEMF

Table 5 BEMF parameter set

Parameter	Range	Description
BEMF_PWM_FREQ	1-25000	PWM frequency
BEMF_SPIKE_FILT	1/2/4/8	Provides low pass filtered signal of BEMF
BEMF_BLANK_FILT	3/6/8/12/16	The blank filter ensures that the latching only takes place if the corresponding PWM phase has a width greater than the blank time
BEMF_POLE_PAIRS	1-32	Number of motor pole pairs
BEMF_SPEED_KP	0-32767	Kp: The proportional gain of the PI regulator
BEMF_SPEED_KI	0-32767	Ki: The integral gain of the PI regulator
BEMF_SPEED_FILT_TIME	0.01-1	Time constant for actual speed
BEMF_SPEED_TEST_ENABLE	0/1	0: Startup without speedtest 1: Startup with speedtest
BEMF_RUNBEGIN_SPEED_PWM_MIN	0-0.95	Speedcontroller.Outmin = BEMF_RUNBEGIN_SPEED_PWM_MIN × Period T12 if (Motorstatus = START6)
BEMF_START_FREQ_ZERO	0/1	0: Start without frequency = 0 1: Start with frequency = 0
BEMF_ZERO_VEC_TIME	0-10	Time for frequency = 0 if(BEMF_START_FREQ_ZERO = 1)
BEMF_START_SPEED_PWM_MIN_OFFSET	0-0.95	$V_t = ((\text{BEMF_START_SPEED_PWM_MIN} / \text{BEMF_END_START_SPEED}) \times \text{SpeedSlope}) + \text{BEMF_START_SPEED_PWM_MIN_OFFSET}) \times \text{Period T12}$
BEMF_END_SPEED_PWM_MIN	0-0.95	BEMF_RUN_SPEED_PWM_MIN + BEMF_START_SPEED_PWM_MIN_OFFSET
BEMF_SWITCH_ON_SPEED	all	Turn on with speed reference > BEMF_SWITCH_ON_SPEED
BEMF_END_START_SPEED	all	Speed value for end of startup (Motorstatus ≤ Start5)
BEMF_START_ACCEL	all	Acceleration for start (Motorstatus ≤ Start5)
BEMF_RUN_SPEED_PWM_MIN	0-0.95	Speedcontroller.Outmin = BEMF_RUN_SPEED_PWM_MIN × PeriodeT12 if (Motorstatus = RUN)
BEMF_RUN_ACCEL	all	Acceleration for run (Motorstatus > Start5)

4 Arduino software documentation

FOC

Table 6 FOC parameter set

Parameter	Range	Description
FOC_R_SHUNT	0-10	Value of the on-board shunt resistor (H/W)
FOC_NOM_CUR	(-10) - (10)	Nominal current: Maximum expected phase current in A
FOC_PWM_FREQ	1-25000	PWM frequency in Hz: 10 kHz ~ 25 kHz
FOC_R_PHASE	0-100	Phase resistance of the motor in Ω
FOC_L_PHASE	0-100	Phase inductance of the motor in H
FOC_POLE_PAIRS[7:0]	0-255	Number of pole pairs of the motor
FOC_MAX_SPEED	0-65535	Maximum speed of the motor in rpm mech., as determined by the motor specification
FOC_CUR_ADJUST	0.01-1	Current controller amplification as a multiplier for the current-control loop speed from 0.01 (slow) to 1 (fast)
FOC_FLUX_ADJUST	0.01-1	Timer constant for flux estimator: KB for flux estimator
FOC_SPEED_KP	0-32767	Kp: The proportional gain of the PI regulator
FOC_SPEED_KI	0-32767	Ki: The integral gain of the PI regulator
FOC_SPEED_FILT_TIME	0.01-1	The time for the low-pass speed filter in s, from 0.01 (slow) to 1 (fast)
FOC_SWITCH_ON_SPEED	0-65535	Switch-on Speed [rpm.mech.]: Beginning open loop speed
FOC_END_START_SPEED	0-65535	End Start-up Speed [rpm,mech.]: Speed at end of open loop
FOC_START_ACCEL	0-65535	Start-up acceleration [rpm/s,mech.]: Acceleration during open loop
FOC_START_CUR_IF	(-10) - (10)	I/F initial start current [A]: Current limit during open loop
FOC_START_FREQ_ZERO	0/1	Rotor alignment time [s]: Time to place the rotor in a known position before running
FOC_ZERO_VEC_TIME	0 - (10)	Timer for zero speed [s]
FOC_MIN_POS_REF_CUR	0 - (10)	FOC current in A for lower positive speeds
FOC_MAX_CUR_SPEED	0-65535	Positive switch over speed [rpm,mech.]
FOC_MAX_POS_REF_CUR	0 - (10)	FOC current in A for higher positive speeds
FOC_MIN_NEG_REF_CUR	<0	FOC current in A for lower negative speeds
FOC_MIN_CUR_SPEED	<0	Negative switch over speed [rpm,mech.]
FOC_MAX_NEG_REF_CUR	<0	FOC current in A for higher negative speeds

4 Arduino software documentation

HALL

Table 7 HALL parameter set

Parameter	Range	Description
HALL_PWM_FREQ	0-25000	PWM frequency
HALL_POLE_PAIRS	1-32	Number of motor pole pairs
HALL_INIT_DUTY	Integer numbers 0-100	Initial Hall duty cycle
HALL_INPUT_A	0/1/2	Selection of Hall pin (check ConfigWizard)
HALL_INPUT_B	0/1/2	Selection of Hall pin
HALL_INPUT_C	0/1/2	Selection of Hall pin
HALL_OFFSET_60DEGREE_EN	0/1	0: Disable HALL offset 1: Enable HALL offset (use when high current consumption while running is observed)
HALL_ANGLE_DELAY_EN	0/1	0: Disable Hall angle 1: Enable Hall angle
HALL_DELAY_ANGLE	Integer numbers 0-59	Hall delay angle value
HALL_DELAY_MINSPEED	Integer numbers 0-2000	Minimum speed for applying the angle delay
HALL_SPEED_KP	0-65535	Kp: The proportional gain of the PI regulator
HALL_SPEED_KI	0-65535	Ki: The integral gain of the PI regulator
HALL_SPEED_IMIN	0-100	Minimum I output duty cycle
HALL_SPEED_IMAX	0-100	Maximum I output duty cycle
HALL_SPEED_PIMIN	0-100	Minimum PI output duty cycle
HALL_SPEED_PIMAX	0-100	Maximum PI output duty cycle

5 TLE9879QXA40 overview

5 TLE9879QXA40 overview

The TLE9879QXA40 is a single chip 3-phase motor driver that integrates the industry standard Arm® Cortex®-M3 core, enabling the implementation of advanced motor control algorithms such as field-oriented control.

It includes six fully integrated NFET drivers optimized to drive a 3-phase motor via six external power NFETs, a charge pump enabling low voltage operation and programmable current along with current slope control for optimized EMC behavior. Its peripheral set includes a current sensor, a successive approximation ADC synchronized with the capture and compare unit for PWM control and 16-bit timers. A LIN transceiver is also integrated to enable communication to the device along with a number of general purpose I/Os. It includes an on-chip linear voltage regulator to supply external loads.

It is a highly integrated automotive qualified device enabling cost and space efficient solutions for mechatronic BLDC motor drive applications such as pumps and fans.

5.1 Key features

- Six current programmable drivers with charge pump for N-Channel MOSFET
- Integrated LIN transceiver compatible with LIN 2.2 and SAEJ2602
- Two full-duplex serial interface (UART) with LIN support
- Two synchronous serial channel (SSC)
- On-chip OSC and PLL for clock generation
- One high-voltage monitoring input with wake up functionality
- High-speed operational amplifier for motor current sensing via shunt
- Measurement unit:
 - 8-bit ADC module with 10 multiplexed inputs
 - 10-bit ADC module with 8 multiplexed inputs, 5 external analog inputs
 - On-chip temperature and battery voltage measurement unit
- Independent programmable window watchdog
- 5 V/1.5 V internal supplies
- External supply (VDDEXT): 5 V $\pm$ 2% @ 20 mA
- Power saving modes:
 - MCU slow-down mode
 - Sleep mode
 - Stop mode
 - Cyclic wake-up Sleep mode

5 TLE9879QXA40 overview

5.2 Block diagram

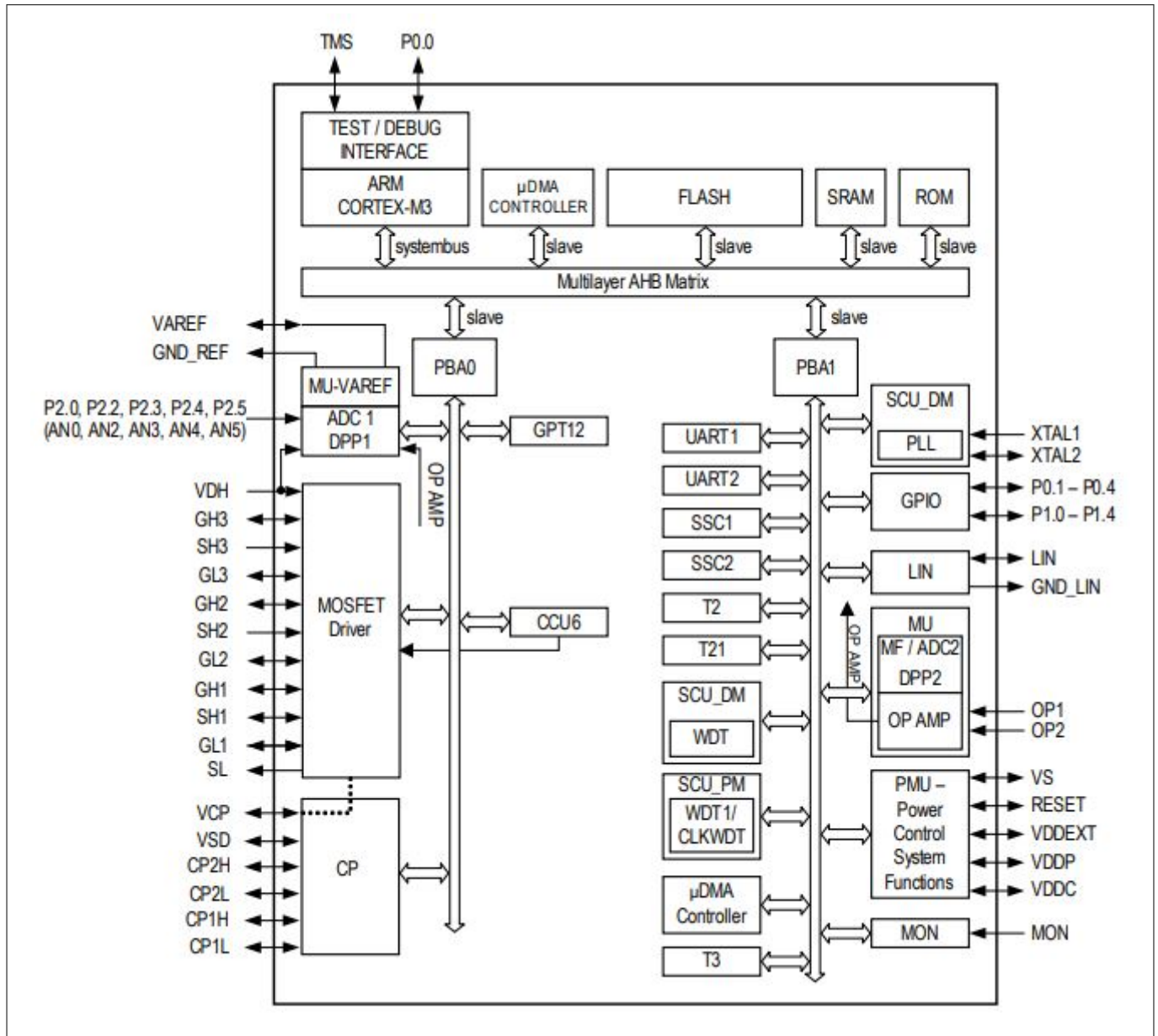


Figure 10 Block diagram TLE9879QXA40

5 TLE9879QXA40 overview

5.3 Pin assignment

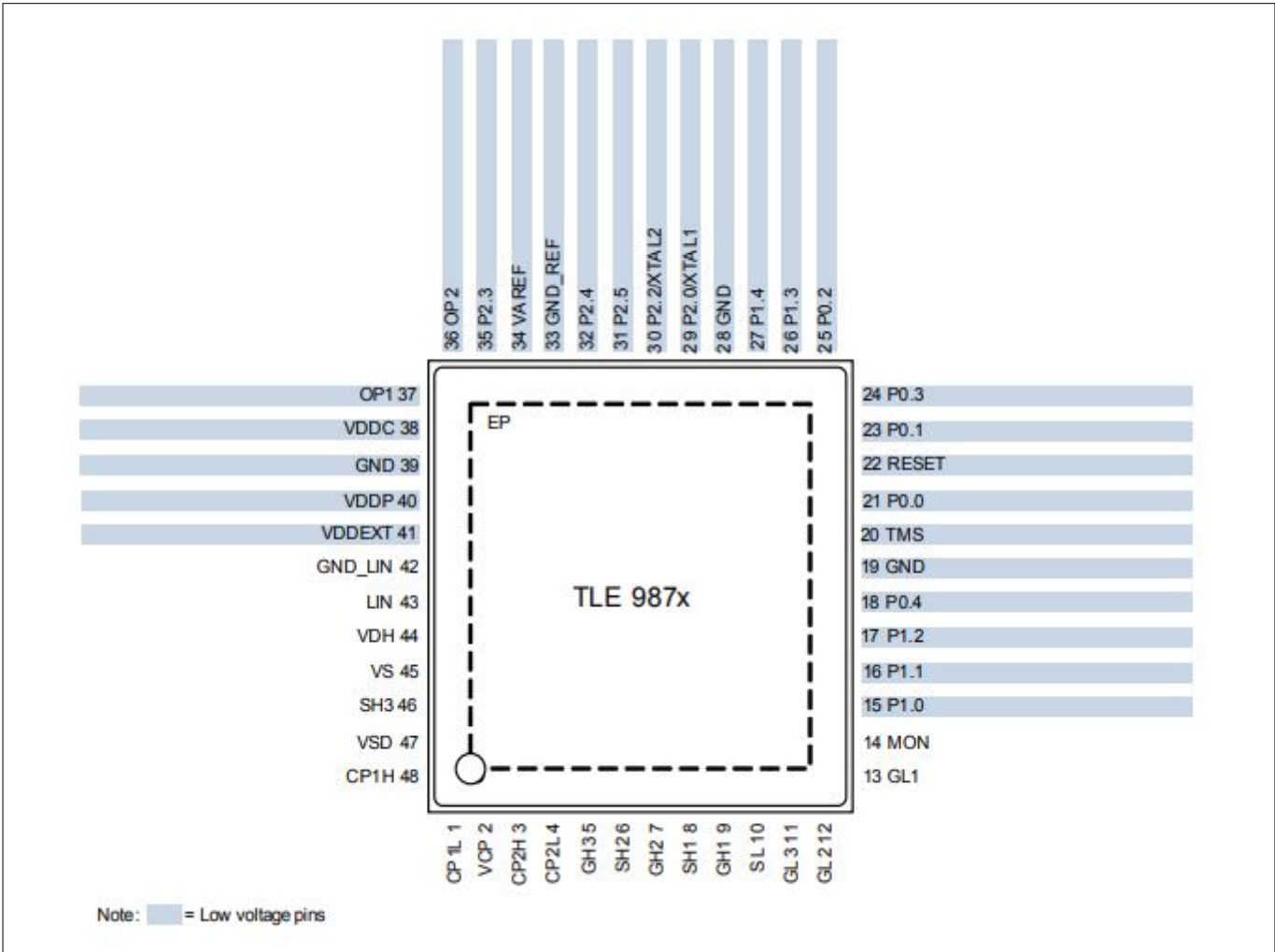


Figure 11 Device pinout TLE9879QXA40

5 TLE9879QXA40 overview

5.4 Pin definitions and functions

Table 8 Pin definitions and functions

Pins	Symbols	Functions
1	CP1L	Charge Pump Capacity 1 Low, connect external C
2	VCP	Charge Pump Capacity
3	CP2H	Charge Pump Capacity 2 High, connect external C
4	CP2L	Charge Pump Capacity 2 Low, connect external C
5	GH3	Gate High Side FET 3
6	SH2	Source High Side FET 2
7	GH2	Gate High Side FET 2
8	SH1	Source High Side FET 1
9	GH1	Gate High Side FET 1
10	SL	Source Low Side FET
11	GL3	Gate Low Side FET 3
12	GL2	Gate Low Side FET 2
13	GL1	Gate Low Side FET 1
14	MON	High Voltage Monitor Input
15	P1.0	GPIO
16	P1.1	GPIO
17	P1.2	GPIO
18	P0.4	GPIO
19	GND	GND Digital
20	TMS	Test Mode Select Input
21	P0.0	Serial Wire Debug Clock
22	RESET	Reset Input, not available during Sleep Mode
23	P0.1	GPIO
24	P0.3	GPIO
25	P0.2	GPIO
26	P1.3	GPIO
27	P1.4	GPIO
28	GND	GND Digital
29	P2.0/XTAL1	ADC Analog Input 0
30	P2.2/XTAL2	ADC Analog Input 2
31	P2.5	ADC Analog Input 5
32	P2.4	ADC Analog Input 4

(table continues...)

5 TLE9879QXA40 overview

Table 8 (continued) Pin definitions and functions

Pins	Symbols	Functions
33	GND_REF	GND for VAREF
34	VAREF	5 V ADC1 Reference Voltage
35	P2.3	ADC Analog Input 3
36	OP2	Positive Operational Amplifier Input
37	OP1	Negative Operational Amplifier Input
38	VDDC	Core Supply
39	GND	GND Analog
40	VDDP	I/O Port Supply
41	VDDEXT	External Voltage Supply Output
42	GND_LIN	GND for LIN
43	LIN	LIN Bus Interface I/O
44	VDH	Voltage Drain High Side MOSFET Driver
45	VS	Battery Supply Input
46	SH3	Source High Side FET 3
47	VSD	Battery Supply Input for Charge Pump
48	CP1H	Charge Pump Capacity 1 High, connect external C
-	EP	Exposed Pad, connect to GND

6 BLDC shield programming

6 BLDC shield programming

The .hex files to be programmed for BLDC shield are stored in the 03_shield_software\batch_flashing_shield_software directory.

If there are any changes in the firmware – for example, FOC, Hall, or BEMF – make those changes in the Keil project and generate its .hex files. Copy these files to the 03_shield_software\batch_flashing_shield_software directory as shown below. (Make sure that the file names are the same as shown below.)

	all_applications.jlinkscript		JLINKSCRIPT File	1 KB
	bemf.hex		HEX File	59 KB
	bootloader.hex		HEX File	15 KB
	foc.hex		HEX File	90 KB
	hall.hex		HEX File	48 KB
	JLink		Application	225 KB
	JLinkARM.dll		Application extension	14,501 KB
	TLE9879_Shield_Flash		Windows Batch File	1 KB

Figure 12 Directory "03_shield_software\batch_flashing_shield_software"

- Connect the 12 V to the BLDC shield and do not connect the Arduino UNO to the BLDC shield
- Connect the 10-pin connector to the BLDC shield
- Run the TLE9879_Shield_Flash.bat script to transfer all examples to the BLDC shield

Note: Programming individual example code may not work sometimes.

7 Revision history

7 Revision history

Revision	Date	Changes
v1.3	2024-05-24	Added section BLDC shield programming . Editorial changes.
v1.2	2020-05-25	Added chapter Arduino software documentation . Editorial changes.
v1.1	2019-09-26	Added 2nd and 3rd pages of Schematics .
v1.0	2018-08-23	Initial creation.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-05-24

Published by

Infineon Technologies AG
81726 Munich, Germany

© 2024 Infineon Technologies AG
All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-cru1533549003952

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenhheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.