EVALUATION BOARD MANUAL

Motor Driver

Stepper Motor Driver IC EVK STEPMO_EVK_20x

ROHM Stepper Motor Driver IC Evaluation Kit based on Arduino/Genuino Platform

● Abstract

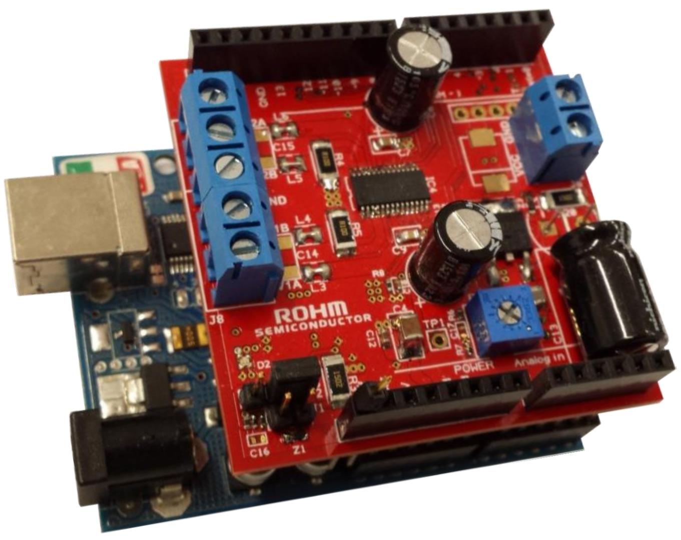
This evaluation board manual describes the usage of ROHM's stepper motor driver IC evaluation kit (EVK) called STEPMO_EVK_20x. It is designed as a plug-in board (Shield) for popular Arduino microcontroller platform. This document provides guidelines to quickly setup the hardware and software for fast and easy stepper motor driver IC performance evaluation.

● Description of Supported ICs

This EVK supports a variety of pin-compatible ROHM stepper motor driver ICs from standard, microstep, low voltage and high voltage families with CLK-IN (clock input) or PARA-IN (parallel input) control. They integrate PWM constant-current drive with adjustable decay ratio and the ability for full, half and microstepping. The ICs feature single supply operation by integrating the voltage regulator for the low power logic together with highly efficient DMOS output power stages. These do not require an internal charge pump so the motor drivers achieve low EMI performance. The motor drivers integrate various protection functions such as Ghost Supply Prevention (GSP), Thermal Shutdown (TSD), Over-Current Protection (OCP), Under / Over Voltage Lockout (UVLO / OVLO) high ESD resistance and Pin Short Protection for robust and reliable operation. The ICs are housed in compact HTSSOP-B28 power packages. For further details, please also consult the relevant product datasheets at http://www.rohm.com.

● EVK Key Specifications (Across Model Versions)

- Input voltage range………………………..……8~42V
- Maximum Output Current Range..........1A~2.5A
- Stepping Modes……………………….⅟₁,½,¼,⅛,⅟₁₆



● **EVK Features**

- Designed as plug-in Shield for Arduino platform
- Recommended: Arduino Uno
- Several model variants covering wide range of pin- compatible stepper driver ICs
- Support of bipolar or unipolar stepper motors
- Adjustable current limit
- Adjustable current decay mode
- Single supply operation
- Stackable design to allow controlling two motors at the same time
- Reverse power supply protection
- Software library for Arduino IDE
- Arduino example programs (Sketches)
- Visit http://www.rohm.com/web/eu/arduino-stepper-motor-shield for details

© 2016 ROHM Co., Ltd. All rights reserved.

Table of Contents

DRAFT

1. Introduction

This evaluation board manual describes the usage of ROHM's stepper motor driver IC evaluation kit (EVK) called STEPMO_EVK_20x. The purpose of the EVK is to allow the test and evaluation of the stepper driver IC in professional research and development environments. It is designed as a plug-in board (Shield) for popular Arduino microcontroller platform. This document provides guidelines to quickly setup the hardware and software for fast and easy stepper motor driver IC performance evaluation. For further details about the motor driver ICs and other ROHM products, please also consult the relevant product datasheets and general information at http://www.rohm.com.

The EVK provided by ROHM consists of hardware and software. This document refers to hardware revision 1605.
Please check http://www.rohm.com/web/eu/arduino-stepper-motor-shield for downloading additional useful information such as application notes, gerber files, schematics, bill of materials, software libraries, example programs etc.

For operation of the EVK an Arduino motherboard is required. For all tests and performance evaluation by ROHM the EVK has been used together with an Arduino Uno R3 motherboard.
Arduino (also Genuino) is an open-source electronics microcontroller platform based on easy-to-use hardware and software.
The concept of Arduino includes a defined IO and Power pinning which allows the easy connection of daughter application boards into the microcontroller mother board. These plug-in boards are also called Shields. The microcontroller programs created using the Arduino IDE are called Sketches. For more information about Arduino please visit https://www.arduino.cc/.

Of course, the EVK hardware can be operated with any other microcontroller platform if desired if this is preferred for evaluation of the IC performance. For this case the required information about the necessary IO and power signals is also provided within this document.

1.1 Block Diagram

A block diagram of the EVK hardware is depicted in Figure 1. The motor voltage applied to screw terminal J6 is also the supply for the motor driver IC. For single supply operation a Buck regulator creates a 7V input voltage for the Arduino motherboard (Vin) which again uses this voltage to create the logic supplies of 5V/3.3V. Therefore it is not necessary to power the microcontroller board separately. The 5V created by the Arduino motherboard is used by the EVK to supply logic blocks and voltage levels to configure the stepper motor driver IC.

The EVK is designed to be stackable to allow evaluating two stepper driver ICs at the same time. For this purpose the Jumper J1 is used to configure the board from Master (default, used for the first Shield) and Slave (only used for a second stacked Shield). As Slave the Buck Regulator is turned off with high impedance output because the Vin Voltage is already generated by the Master Shield.

Based on the setting of J1 the control signals for the motor driver IC are routed via multiplexers to different Arduino IOs to allow independent motor drive. The Master or Slave configuration is indicated by an LED (Master: green, Slave red).
The screw terminal J7/J8 allows the connection of a 4-wire bipolar or a 5-to-6-wire unipolar stepper motor.

DRAFT

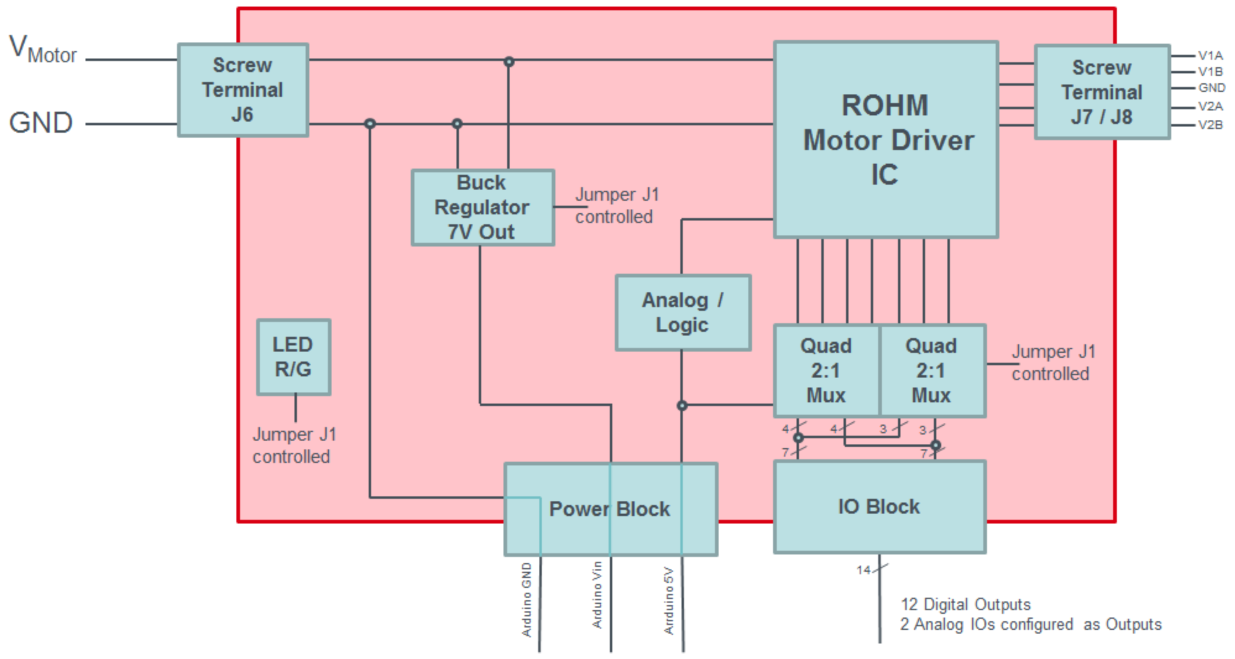


Figure 1: EVK Shield Block Diagram

DRAFT

1.2 Model Overview

This stepper motor EVK has several different model variants. The main difference between the variants is the equipped ROHM stepper motor driver IC. Table 1 gives an overview of the different IC features and the according naming of the EVK. The ID suffix “x” of the EVK name STEPMO_EVK_20x denotes the model version. For further information about the ICs please follow the hyperlinks to the datasheets on ROHM’s global web site.

ID	**ROHM Motor Driver IC**	**Supply Voltage / V**		**Max. Current per Phase / A**	**Supported Step Modes**	**Control Type**
x		Min	Max	25°C		
1	BD63510AEFV [1]	8	28	1.0	1/1,1/2,1/4,1/16	CLK-IN
2	BD63520AEFV [1]	8	28	2.0 [2]	1/1,1/2,1/4,1/16	CLK-IN
3	BD63524AEFV [1]	8	28	2.5 [2]	1/1,1/2,1/4,1/8	CLK-IN
4	BD63710AEFV	19	28	1.0	1/1,1/2,1/4	CLK-IN
5	BD63715AEFV	19	28	1.5	1/1,1/2,1/4	CLK-IN
6	BD63720AEFV	19	28	2.0 [2]	1/1,1/2,1/4	CLK-IN
7	BD63843EFV	19	28	1.0	1/1,1/2,1/8,1/16	CLK-IN
8	BD63847EFV	19	28	2.0 [2]	1/1,1/2,1/8,1/16	CLK-IN
9	BD63873EFV	19	28	1.0	1/1,1/2,1/4	CLK-IN
A	BD63875EFV	19	28	1.5	1/1,1/2,1/4	CLK-IN
B	Reserved					
C	BD63872EFV	19	28	1.0	1/1,1/2,1/4	PARALLEL-IN
D	Reserved					
E	BD63876EFV	19	28	2.0 [2]	1/1,1/2,1/4	PARALLEL-IN
F	BD6425EFV	19	42	1.5	1/1,1/2,1/4	CLK-IN

Note 1: In development.
Note 2: Will require additional heat sinking or cooling techniques to achieve the maximum current continuously.

Table 1: EVK Model Variants

2. Hardware Description

The schematic of the EVK is shown in Figure 2 and Figure 3 and the Bill-of-Materials (BOM) is summarized in Table 2. Please note that the depicted schematics and BOM refer to model version 2 of the EVK. The other model versions differ in some details while the major differences are the in assembled motor driver IC part number (see Table 1) and the different current sense resistor values. For schematics and BOMs of all available model versions please visit http://www.rohm.com/web/eu/arduino-stepper-motor-shield.

Main part of the EVK is the ROHM stepper motor driver labelled IC4. Q2, R20 and Z3 form a reverse power supply protection. Z2 is a TVS diode to protect against transient surge voltages. Supply decoupling and filtering is accomplished by C13, L2, and C6-C9. Voltage dividers R9-R11 and R6, R19, R7 set the necessary voltage levels for VREF and MTH. Test pins are connected to GND via 0Ω-resistors.

The PWM frequency is set by C10 and R8 to the standard value as suggested in the IC datasheet. The resulting chopping frequency is ~ 25 kHz. For fine tuning the stepper motor performance it is usually recommended to find the optimum frequency as a trade-off between motor noise and current waveform distortion. Thus, if desired, the values of C10 and R8 could be changed by the user by soldering different components. Please refer to the IC datasheet for recommended values.

An optional low pass filter in the motor phase outputs is formed by L3-L6 and C14-C15. D2 is an instance of ROHM’s ultra-compact PICOLED^{TM}-Duo two-color type LED indicating Master/Slave mode. It is switched by Q1 and controlled by Jumper J1.

J5 is the defined Arduino compatible set of header rows for IOs and power.

IC1 and IC2 are ROHM BU4551 parts which contain 4 times a 2:1 multiplexer each. They are required to route the correct control signals to the motor driver IC depending on the setting of Jumper J1.

IC3 is ROHM’s BD9G101G wide input range DC/DC step-down regulator which integrates a high-side power MOSFET into its small SOT-23 package while able to provide up to 0.5A DC output. It is used together with adjacent circuitry to create the 7V input voltage to the Arduino with a high efficiency.

2.1 Schematic

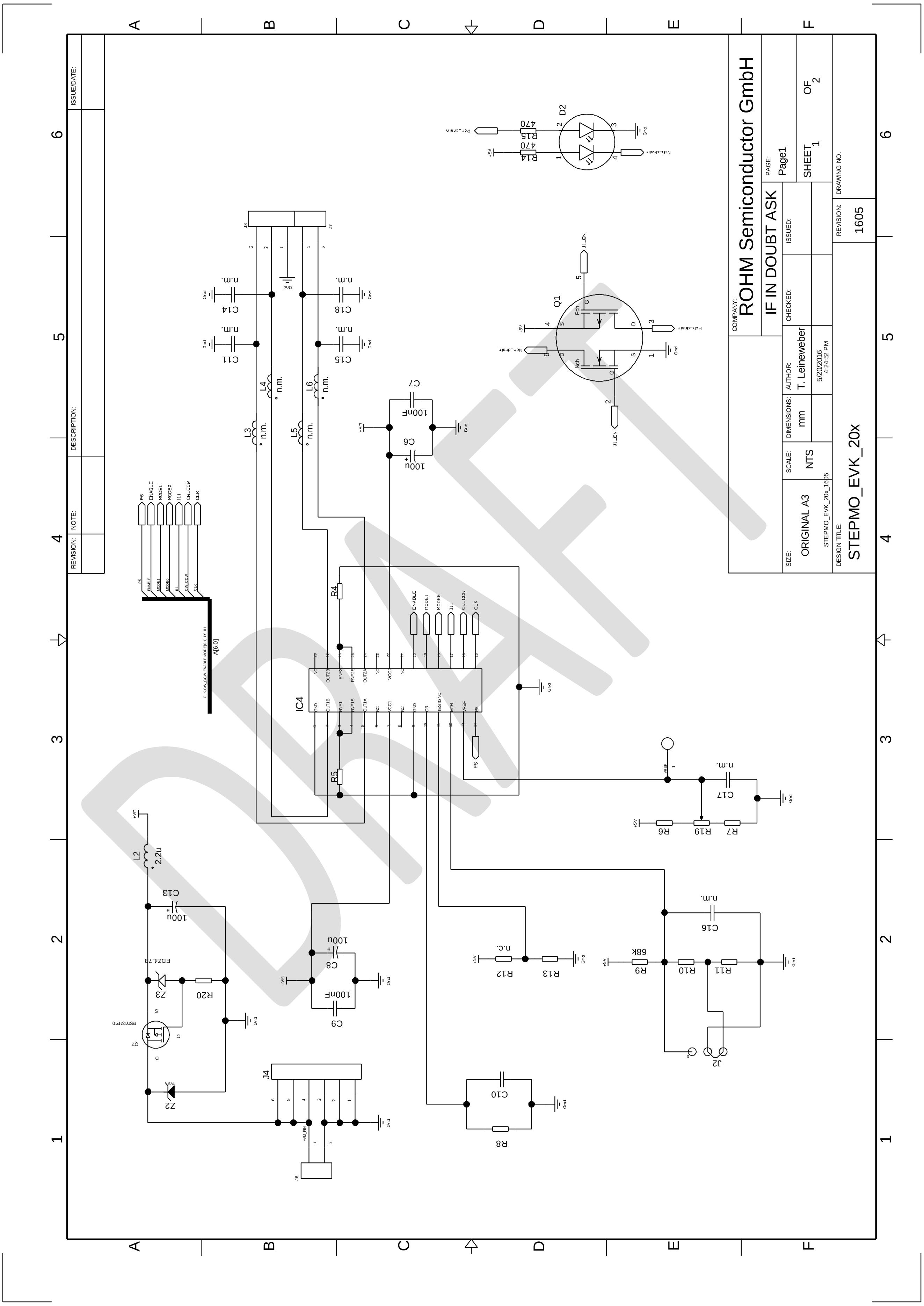


Figure 2: EVK Schematic – STEPMO_EVK_20x – Page 1

© 2016 ROHM Co., Ltd. All rights reserved.

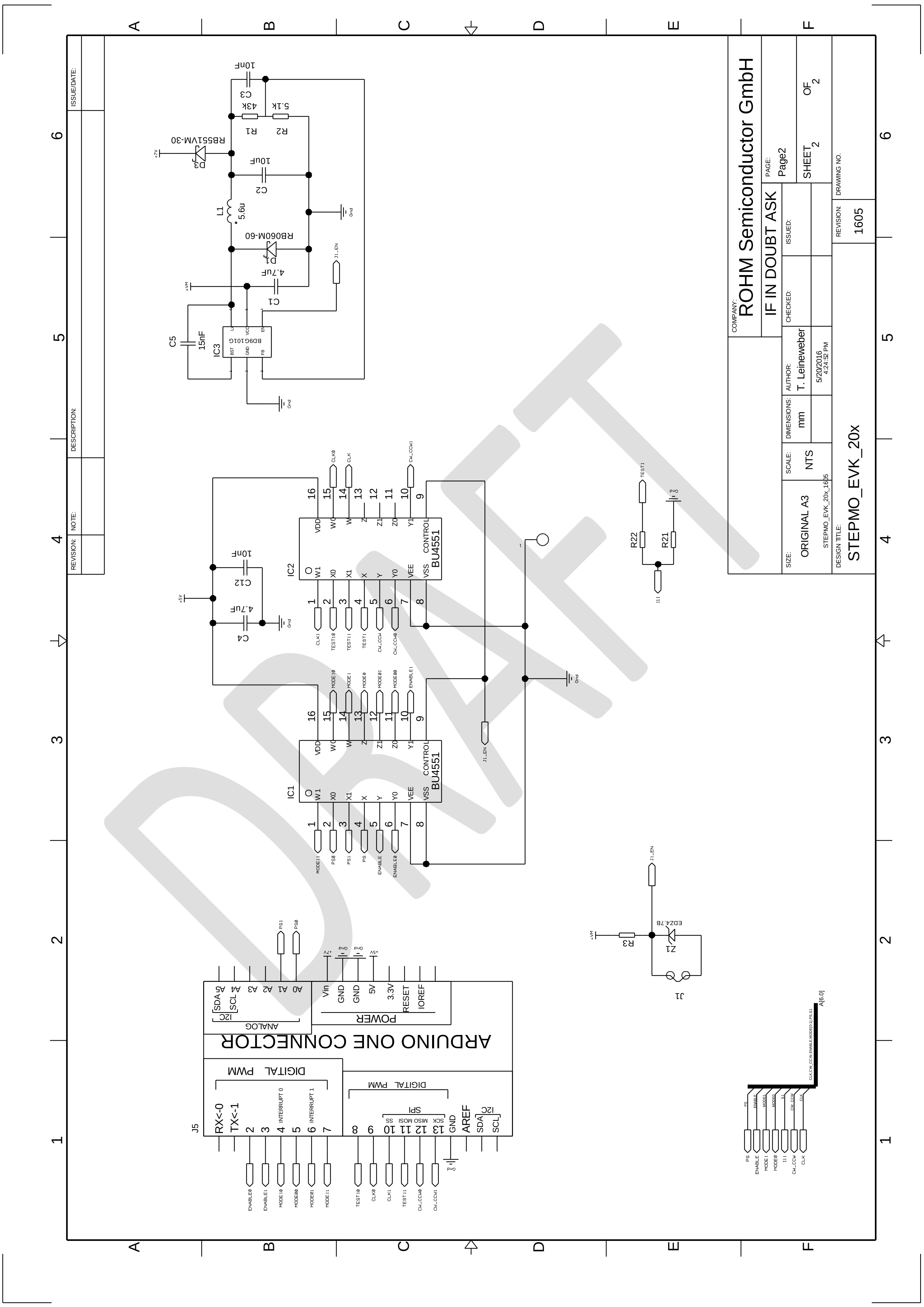


Figure 3: EVK Schematic – STEPMO_EVK_20x – Page 2

The schematic of the EVK is plotted in Figure 2 and Figure 3. Component values valid for all model versions are annotated, for all other values please refer to the BOM of each model version available at http://www.rohm.com/web/eu/arduino-stepper-motor-shield.

2.2 Bill of Materials

The bill of materials is given Table 2 as example for model version 2.
For all other BOMs please refer to http://www.rohm.com/web/eu/arduino-stepper-motor-shield.

Component Name	Value	Description	Supplier	Part number	Qty
C1, C4	4.7u	Cer Cap 22uF 20% 16V X7R 1210	TDK	C3225X7R1H475K250AB	2
C2	10u	Cer Cap 22uF 20% 16V X7R 1210	TDK	C3225X7R1H106M250AC	1
C3, C12	10n	Cer Cap 10nF 10% 25V X7R 0603	TDK	C1608X7R1E103K080AA	2
C5	15n	Cer Cap 15nF 20% 50V X7R 0603	TDK	C1608X7R1H153M080AA	1
C6, C8	100u	100uF 20% 63V Electrolyte Capacitor 8mm x 11.5mm	Nichicon	UVY1J101MPD	2
C7, C9	100n	Ceramic capacitor 100nF 10% 50V X7R 0805 Wurth	Wurth	885012207098	2
C10	1n	Ceramic capacitor 1nF 20% 50V X7R 0402	TDK	C1005X7R1H102M050BA	1
C13	100u	100uF 20% 100V Electrolyte Cap. 10mm x 16mm	Nichicon	UVK2A101MPD	1
C11, C14, C15, C16, C17, C18	n.m.	Ceramic capacitor, not mounted			6
D1		Schottky Diode 2A 60V	ROHM	RB060M-60	1
D2		PICOLED^{TM}-Duo two-color type LED (green + red) in 1 package	ROHM	SML-P24MUWT86	1
D3		Schottky Diode 0.5A 20V	ROHM	RB551VM-30	1
IC1, IC2		Quad 2ch Analog Multiplexer/Demultiplexer	ROHM	BU4551BFV-E2	2
IC3		Wide Input Range DCDC Regulator	ROHM	BD9G101G-TR	1
IC4		Stepper Motor Driver	ROHM	BD63520EFV-E2	1
J1		Jumper 2 Way	FCI	77311-118-02LF	1
J2		Jumper 3 Way	FCI	77311-118-03LF	1
J4		ARDUINO Stackable Header 6pin 14mm5 height	Arduino	A000084	1
J5 C		ARDUINO Stackable Header 6pin 14mm5 height	Arduino	A000084	1
J5 A, J5 D		ARDUINO Stackable Header 8pin 14mm5 height	Arduino	A000085	2
J5 B		ARDUINO Stackable Header 10pin 14mm5 height	Arduino	A000086	1
J6, J7		Connector Screw 5mm pitch	Wurth	691102710002	2
J8		Connector Screw 5mm pitch	Wurth	691102710003	1
L1	5.6u	5.6µH 2.8A 81mOhm Shielded 20% Tolerance	Wurth	74438356056	1
L2	2.2u	2.2µH 3A 84mOhm Shielded 20% Tolerance	Toko	DFE252012P-2R2M=P2	1
L3, L4, L5, L6	n.m.	Shorted on board, no need to assemble			4
Q1		Nch+Pch MOSFET	ROHM	US6M2TR	1
Q2		Pch MOSFET, 100V, 13A	ROHM	RSD131P10TL	1
R1	36k	Resistor 36kOhm 1% 1/16W	ROHM	MCR01MZPF3602	1
R2	4.3k	Resistor 4.3kOhm 1% 1/16W	ROHM	MCR01MZPF4301	1
R3, R20	3.9k	Resistor 3.9kOhm 1% 0.5W	ROHM	MCR50JZHF3901	2
R4, R5	0.1	Resistor 0.1Ohm 1% 0.5W	ROHM	MCR50JZHFLR100	2
R6	62k	Resistor 62kOhm 1% 1/16W	ROHM	MCR01MZPF6202	1
R7	1.8k	Resistor 4.3kOhm 1% 1/16W	ROHM	MCR01MZPF1801	1
R8	39k	Resistor 39kOhm 1% 1/16W	ROHM	MCR01MZPF3902	1
R9	68k	Resistor 68kOhm 1% 1/16W	ROHM	MCR01MZPF6802	1
R10	12k	Resistor 12kOhm 1% 1/16W	ROHM	MCR01MZPF1202	1
R11	56k	Resistor 56kOhm 1% 1/16W	ROHM	MCR01MZPF5602	1
R12, R22	n.m.	open			2
R13, R21	0	Resistor short, 50m Ohm max, 0.5A max.	ROHM	MCR01MZPJ000	2
R14, R15	470	Resistor 470 Ohm 1% 1/16W	ROHM	MCR01MZPF4700	2
R19	20k		Bourns	3362P-1-203LF	1
TP0, TP1	n.m.				2
Z1, Z3		ZENER DIODE 4.7V 150mW	ROHM	EDZTE614.7B	2
Z2		TVS DIODE	Fairchild Semiconductor	SMCJ26A	1

Table 2: EVK Bill of Materials – STEPMO_EVK_202

3. Setup Instruction

3.1 Hardware

● Master/Slave Mode

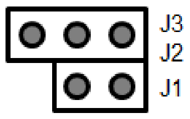
This section describes how to set up the Shield in Master or Slave mode. This selection is made by Jumper J1. In most of the configurations (default) the Jumper J1 is simply left open and the Shield is configured as Master. Jumper J1 needs only to be in closed position when the Shield is stacked as a Slave Board on a Master Board for operation of two Shields with a single Arduino microcontroller.

Note:

- **For the electrical operation the physical order of Master and Slave Shield actually does not matter.**
- **Do not change the setting of J1 while the power supply is applied.**

Master Board (Default)

Jumper J1 is **open**:



- Buck regulator supply (7V) is enabled to generate VIN for Arduino
- Green LED of D2 is lit
- Arduino IO mapping according to the following table

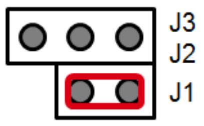
ALL Model EVK Versions except C, E						
ARDUINO IO	**Schematic**	**MUX Routing Between**		**Signal**	**Motor Driver IC Pin**	
No.	**Net Name**	**Pin**	**Pin**	**Name**	**No.**	**Name**
A1	PS1	IC1_X1	IC1_X	PS	14	PS
3	ENABLE1	IC1_Y1	IC1_Y	ENABLE	20	ENABLE
7	MODE11	IC1_W1	IC1_W	MODE1	19	MODE1
6	MODE01	IC1_Z1	IC1_Z	MODE0	18	MODE0
13	CW_CCW1	IC2_Y1	IC2_Y	CW_CCW	16	CW_CCW
10	CLK1	IC2_W1	IC2_W	CLK	15	CLK
Model EVK Versions C, E						
ARDUINO IO	**Schematic**	**MUX Routing Between**		**Signal**	**Motor Driver IC Pin**	
No.	**Net Name**	**Pin**	**Pin**	**Name**	**No.**	**Name**
A1	PS1	IC1_X1	IC1_X	PS	14	PS
3	ENABLE1	IC1_Y1	IC1_Y	ENABLE	20	I12
7	MODE11	IC1_W1	IC1_W	MODE1	19	I02
6	MODE01	IC1_Z1	IC1_Z	MODE0	18	PHASE2
11	TEST11	IC2_X1	IC2_X	TEST1	17	I11
13	CW_CCW1	IC2_Y1	IC2_Y	CW_CCW	16	I01
10	CLK1	IC2_W1	IC2_W	CLK	15	PHASE1

Table 3: Mapping of Arduino IOs to the motor driver IC pins **(Master Shield)**

DRAFT

Slave Board (for a stacked Shield)

Jumper J1 is **closed**:



- Buck regulator supply (7V) is disabled with high impedance output. VIN is delivered by Master Shield.
- Red LED of D2 is lit
- Arduino IO mapping according to the following table

ALL Model EVK Versions except C, E						
ARDUINO IO	**Schematic**	**MUX Routing Between**		**Signal**	**Motor Driver IC Pin**	
No.	**Net Name**	**Pin**	**Pin**	**Name**	**No.**	**Name**
A0	PS0	IC1_X0	IC1_X	PS	14	PS
2	ENABLE0	IC1_Y0	IC1_Y	ENABLE	20	ENABLE
4	MODE10	IC1_W0	IC1_W	MODE1	19	MODE1
5	MODE00	IC1_Z0	IC1_Z	MODE0	18	MODE0
12	CW_CCW0	IC2_Y0	IC2_Y	CW_CCW	16	CW_CCW
9	CLK0	IC2_W0	IC2_W	CLK	15	CLK
Model EVK Versions C, E						
ARDUINO IO	**Schematic**	**MUX Routing Between**		**Signal**	**Motor Driver IC Pin**	
No.	**Net Name**	**Pin**	**Pin**	**Name**	**No.**	**Name**
A0	PS0	IC1_X0	IC1_X	PS	14	PS
2	ENABLE0	IC1_Y0	IC1_Y	ENABLE	20	I12
4	MODE10	IC1_W0	IC1_W	MODE1	19	I02
5	MODE00	IC1_Z0	IC1_Z	MODE0	18	PHASE2
8	TEST10	IC2_X0	IC2_X	TEST1	17	I11
12	CW_CCW0	IC2_Y0	IC2_Y	CW_CCW	16	I01
9	CLK0	IC2_W0	IC2_W	CLK	15	PHASE1

Table 4: Mapping of Arduino IOs to the motor driver IC pins **(Slave Shield)**

DRAFT

Note:

- **In case a stacked Slave Shield is not used the IOs listed in Table 4 are free to use and can be accessed on the corresponding EVK pin header. However, the additional capacitive load of ~10pF by the turned-off multiplexer path should be considered in this case.**

● Current Limitation Value

The ROHM stepper motor driver ICs supported by this EVK have a current limitation function. This must not to be confused with over current protection (OCP) which is another feature of the IC (check datasheet for details). Instead, the purposes of the current limitation are:

- Constant motor current operation independent from the supply voltage
- Operation of motors with low impedance phase without exceeding the rated motor current
- Operation with high supply voltages for faster current rise in the phase windings to achieve higher motor torque
- Operation in microstepping mode

The current limitation is achieved by chopping the output current with pulse width modulation (PWM) as soon as the set limit is reached. As described in the datasheet the set current limit depends on the current sense resistor RNF (resistors R4 and R5 in the schematic) and the voltage applied to the VREF pin. In this EVK the current sense resistor is fixed but the voltage on the VREF pin can be adjusted by the potentiometer R19. The current limit value will depend linearly on the potentiometer setting. For reference, the approximate values for minimum and maximum current limit values are given in Table 5. With the corresponding values of VREF and RNF the set current limit value can be calculated with the formula:

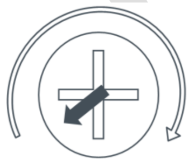
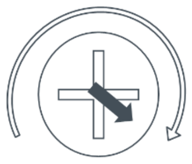
$$IOUT\ (per\ phase) = \frac{VREF}{5 \cdot RNF}$$

VREF can be accessed at test point TP1 for indirect measurement of the current limit.

Note:

- **The highest value which can be set by the potentiometer is limited to the maximum continuous current per phase as allowed by the motor driver IC specifications.**
  Nevertheless, for high output currents (~1.5A and above) additional cooling or heat sinking will need to be applied to the IC and PCB. Please always check the IC temperature!
  Also please take care to set the current limit to a value not exceeding the rated maximum current per phase of your connected motor.

DRAFT

	Minimum Setting		**Maximum Setting**		
Potentiometer Setting					
EVK Model Versions	**VREF / V**	**Current Limit / A**	**VREF / V**	**Current Limit / A**	**RNF Value / Ω**
1, 4, 7, 9, C	0.11	~ 0.067	1.64	~ 1.0	0.33 [1]
F	0.16	~ 0.13	1.88	~ 1.5	0.25 [1]
5, A	0.1	~ 0.15	0.98	~ 1.5	0.13 [1]
2, 6, 8, E	0.11	~ 0.17	1.3	~ 2.0	0.13 [1]
3	0.25	~ 0.35	1.75	~ 2.5	0.14 [1]

Note 1: Including parasitic board resistance of ~0.03Ω

Table 5: Current limit settings by potentiometer R19 with references to VREF and RNF values

● Current Decay Mode

As explained in the datasheet the ROHM stepper motor driver IC used by this EVK allows external configuration of the current decay mode. It is a way to fine tune the motor performance between vibration and current waveform distortion/harmonics. The optimum decay mode setting depends very much on the application so the influence of each setting should be investigated by lab experiments.
For reference some general recommendations are given:

• Slow decay:	Full step mode, low pulse rate half- and micro-stepping modes
• Fast decay:	High pulse rate half- and micro-stepping modes
• Mixed decay:	Trade-off

The EVK allows selecting the different decay modes with the Jumper J2/J3 as described in Table 6. In mixed decay mode it is also possible to apply an external voltage (not supported by model version F) to find the optimum setting.

Note:

- **The current decay setting is only effective when the IC is operating in current limitation mode. For details please refer to the according section in this document.**

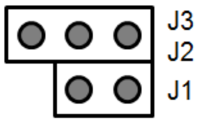
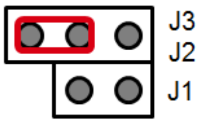
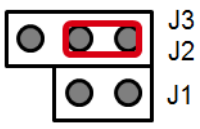
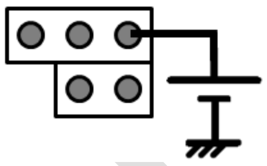
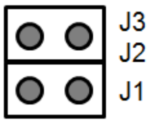
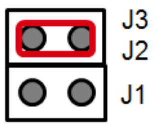
ALL Model EVK Versions except F	
	**Jumper J2/J3 open:** ▪ Fast Decay Mode (MTH Voltage ~ 2.5V)
	**Jumper J2/J3 Pos. 1:** ▪ Mixed Decay Mode (MTH Voltage ~ 0.75V)
	**Jumper J2/J3 Pos. 2:** ▪ Slow Decay Mode (MTH Voltage ~ 0V)
	**Analog Input Voltage:** ▪ Allows optimum tuning of current decay setting with analog voltage source ▪ Effective voltage range is ~0.4V to ~1.0V ▪ Voltage source should be able to drive load of ~30kΩ
Model Version F	
	**Jumper J2/J3 open:** ▪ Mixed Decay Mode (Fast Decay at 40%)
	**Jumper J2/J3 closed:** ▪ Slow Decay Mode

Table 6: Setting of Current Decay Mode

Some further explanations about the supported decay modes are summarized in Table 7.

DRAFT

MTH Voltage / V	**Current Decay Mode**	**Explanation**
0.0 – 0.3	Slow	Any voltage in the given range will result in the same “slow” decay mode. Typically just pull to GND to activate.
0.3 – 0.4	Transition region	Do not set the MTH voltage in this region because the resulting decay mode is not guaranteed.
0.4 – 1.0	Mixed	The mixture of “slow” and “fast” decay is not fixed in this region. The optimum setting can be tuned by applying an analog voltage to the MTH terminal. By jumper configuration (see EVK manual) just one predefined setting (~ 0.75V) is provided. If a different Mixed Mode MTH voltage is desired the voltage divider built by R9, R10, R11 must be adjusted.
1.0 – 1.5	Transition region	Do not set the MTH voltage in this region because the resulting decay mode is not guaranteed.
1.5 – 3.5	Fast	Any voltage in the given range will result in the same “fast” decay mode. Simply pull to 3.3V logic pin or use a 1:1 voltage divider for 5V logic.

Table 7: Current decay mode explanations

DRAFT

● Supply and Motor Connection

The STEPMO_EVK_20x allows the connection of a 4-wire bipolar or 5-wire unipolar stepper motor to the screw terminal J7/J8. The outputs V1A and V1B belong to one phase of the motor while V2A and V2B to the other. If the motor spins clockwise when it should go counter clockwise the polarity of one phase should be reversed, i.e. the wiring position of one phase should be swapped (either V1A with V1B OR V2A with V2B not both).

The connection of a bipolar motor is depicted in Figure 4. In case of a unipolar motor the additional common mode wire(s) are connected to the GND middle pin.

The power supply is connected to screw terminal J6. Although the EVK is equipped with a reverse power supply protection care should be taken about the polarity of the supply.

To meet the requirements from electromagnetic compatibility (EMC) all connected cables must not exceed a length of 3m.

Note:

- **In case of stacking two EVK Shields in Master/Slave configuration only the Master Shield must be supplied by the external power supply. The Slave Shield is powered via terminal J4 with the same applied voltage.**
- **When applying power to the Shield please double check the EVK model version you are using and take care to stay within the rated power supply limits as listed in Table 1. Operating the EVK over the given supply voltage ratings may permanently damage the EVK.**

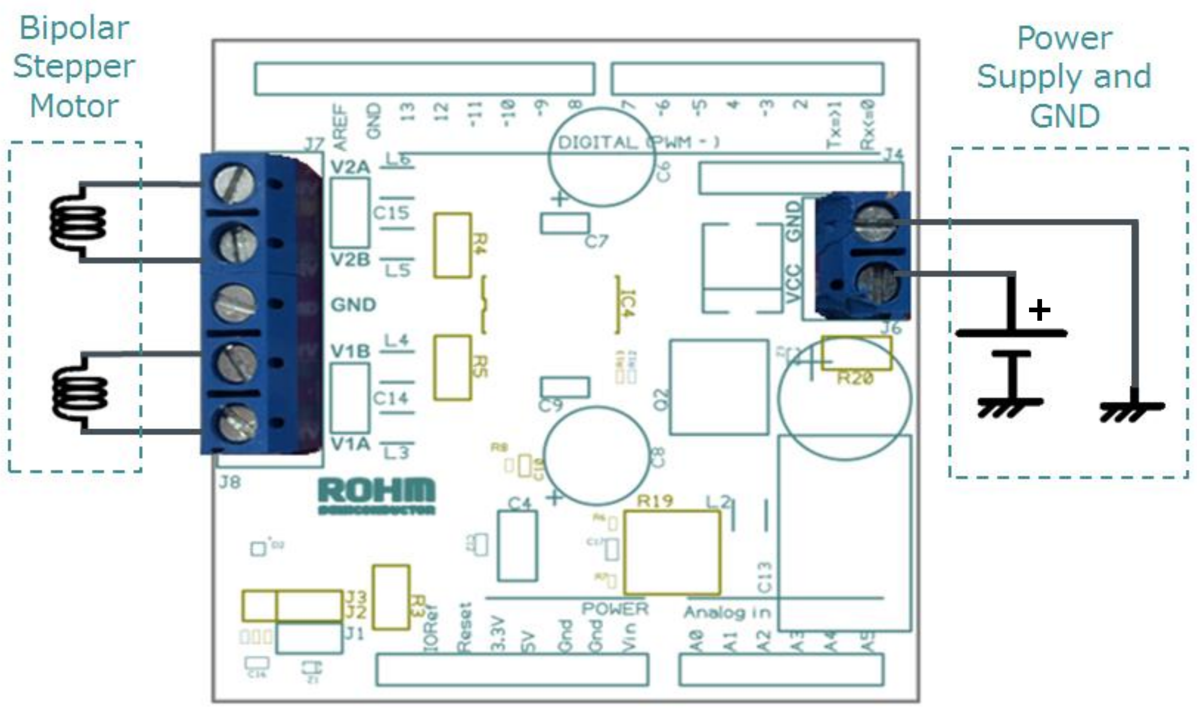


Figure 4: Connection of Stepper Motor and Power Supply

● Connection to Microcontroller Motherboard

Since the EVK is designed as Arduino Shield it can be directly plugged into an Arduino microcontroller motherboard such as the Arduino Uno. If the EVK is used with another microcontroller, all control signals and power supplies can be accessed / supplied via the pin header row interfaces as documented in the tables below.

-Pin-	Alias	Function in EVK			
		All Model Versions except C, E		**Models C, E**	
		1 Shield	**2 Shields**	**1 Shield**	**2 Shields**
0	RX	Reserved for Serial Communication via FTDI/USB			
1	TX	Reserved for Serial Communication via FTDI/USB			
2	IRQ0	Free to use	SH2_ENABLE	Free to use	SH2_I12
3	IRQ1	SH1_ENABLE		SH1_I12	
4	-	Free to use	SH2_MODE1	Free to use	SH2_I02
5	-	Free to use	SH2_MODE0	Free to use	SH2_PHASE2
6	-	SH1_MODE0		SH1_PHASE2	
7	-	SH1_MODE1		SH1_I02	
8	-	Free to use	Reserved	Free to use	SH2_I11
9	-	Free to use	SH2_CLK	Free to use	SH2_PHASE1
10	SS	SH1_CLK		SH1_PHASE1	
11	MOSI	Reserved		SH1_I11	
12	MISO	Free to use	SH2_CW_CCW	Free to use	SH2_I01
13	SCK	SH1_CW_CCW		SH1_I01	
14	A0	SH1_PS			
15	A1	Free to use	SH2_PS	Free to use	SH2_PS
16	A2	Free to use			
17	A3	Free to use			
18	A4	Free to use			
19	A5	Free to use			

Table 8: Arduino IO Mapping to EVK Function

-Pin-	Value	Function in EVK
Vin	7V	This pin provides a supply voltage generated by EVK from the applied motor voltage at screw terminal J6. This voltage is used as input voltage for the Arduino motherboard or any other microcontroller board which may be used. The maximum current drawn from this pin must not exceed 500mA (tbc).
5V	5V	This voltage is supplied by the Arduino motherboard to the EVK. In case of using another microcontroller board, please supply a regulated 5V voltage to this pin with a minimum current delivery capability of 10mA.
Gnd	0V	All pins labelled “Gnd” are connected to the ground (0V) of the EVK and are connected to the ground of the Arduino motherboard. In case of using another microcontroller board, please connect these pins to its ground terminal.

Table 9: EVK power supply interface to microcontroller

In operation, the Arduino motherboard is supplied by the motor voltage connected to the EVK so no additional power supply to the microcontroller board is necessary. However, in a typical lab set up the Arduino motherboard may be connected to the USB port of a host while evaluating, testing and programming. The following table gives an overview about the allowed power supply connections in this case if one or two EVKs are plugged into the microcontroller board.

EVK motor voltage	Additional Arduino supply via		
	USB	**Vin**	**DC Jack**
Turned off / not connected	Not recommended	Not allowed	Not allowed
Turned on	Allowed		

Table 10: Matrix of allowed additional Arduino supplies while EVK(s) plugged in

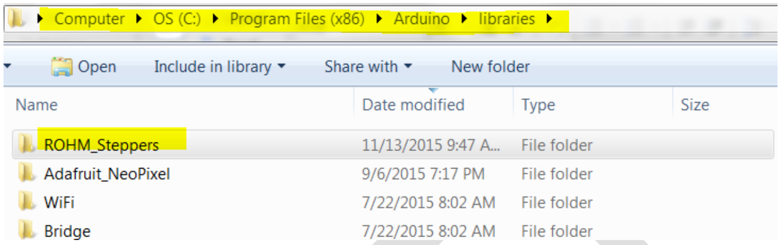
DRAFT

3.2 Software

● Installation Procedure

1. The latest Arduino Software (IDE) can be downloaded here:
   https://www.arduino.cc/en/Main/Software
   Please download and install it.

2. The latest ROHM STEPMO_EVK_20x software delivery package can be downloaded here:
   http://www.rohm.com/web/eu/arduino-stepper-motor-shield
   Please download it and unzip the package in the subfolder `Arduino\libraries\`.
   This folder is typically located at `C:\Program Files (x86)\`.
   After unzipping a folder called "ROHM_Steppers" is created. For contents of this folder please see the bullet point "Content of the Software Delivery Package".



Computer ▸ OS (C:) ▸ Program Files (x86) ▸ Arduino ▸ libraries ▸

Open | Include in library ▾ | Share with ▾ | New folder

Name	Date modified	Type	Size
ROHM_Steppers	11/13/2015 9:47 A...	File folder	
Adafruit_NeoPixel	9/6/2015 7:17 PM	File folder	
WiFi	7/22/2015 8:02 AM	File folder	
Bridge	7/22/2015 8:02 AM	File folder	

Figure 5: System path showing the location of the unzipped folder "ROHM_Steppers"

3. Run the Arduino IDE and open the provided ROHM example sketches.
   For more information about how to use the Arduino IDE please refer to
   https://www.arduino.cc/en/Guide/Environment

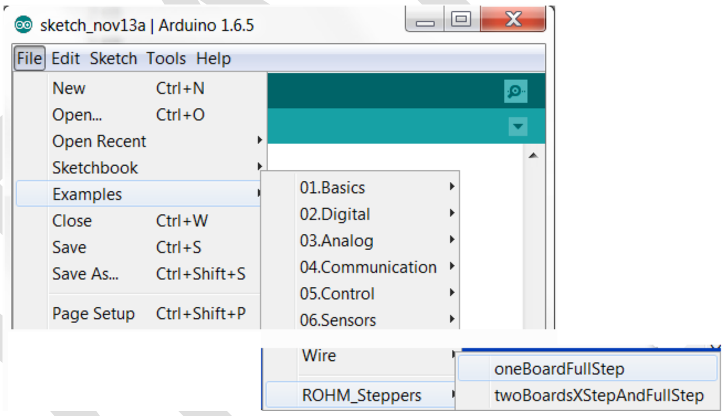


Figure 6: Arduino IDE with the "ROHM_Steppers" Menu Item and the available example programs

DRAFT

● Content of the Software Delivery Package

The ROHM STEPMO_EVK_20x software delivery package is part of this EVK. It contains the library with all required functions and also some example programs (Arduino Sketches) to demonstrate the usage of this library.

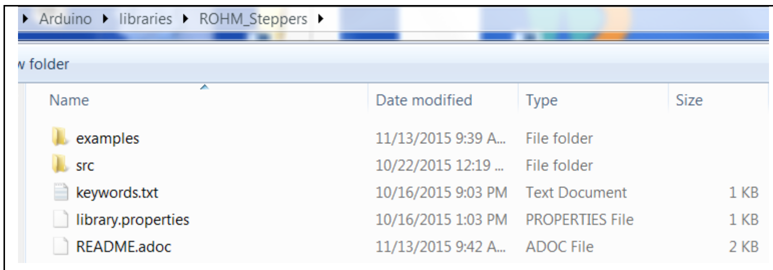
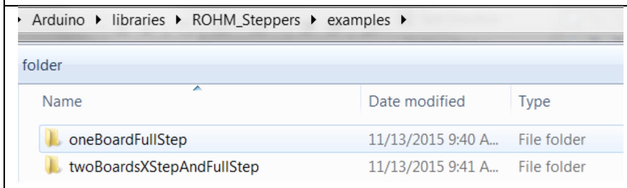
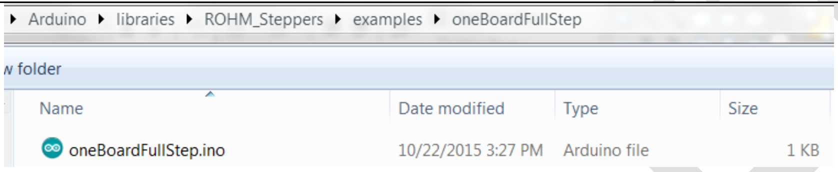
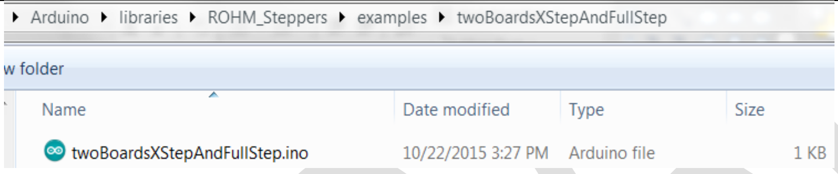
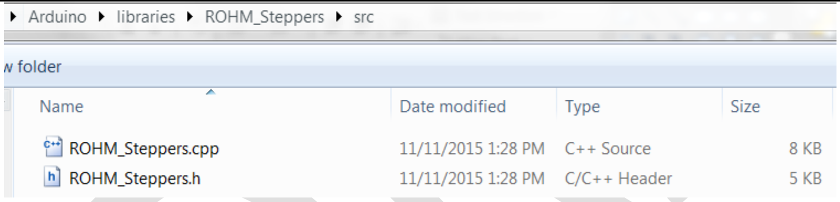
 ▸ Arduino ▸ libraries ▸ ROHM_Steppers ▸ w folder Name | Date modified | Type | Size examples | 11/13/2015 9:39 A... | File folder | src | 10/22/2015 12:19 ... | File folder | keywords.txt | 10/16/2015 9:03 PM | Text Document | 1 KB library.properties | 10/16/2015 1:03 PM | PROPERTIES File | 1 KB README.adoc | 11/13/2015 9:42 A... | ADOC File | 2 KB	**Standard content for Arduino library**
 ▸ Arduino ▸ libraries ▸ ROHM_Steppers ▸ examples ▸ folder Name | Date modified | Type oneBoardFullStep | 11/13/2015 9:40 A... | File folder twoBoardsXStepAndFullStep | 11/13/2015 9:41 A... | File folder	**Arduino GUI accessible examples**
 ▸ Arduino ▸ libraries ▸ ROHM_Steppers ▸ examples ▸ oneBoardFullStep w folder Name | Date modified | Type | Size oneBoardFullStep.ino | 10/22/2015 3:27 PM | Arduino file | 1 KB	**Example Sketch:** ▪ **One board and full step mode**
 ▸ Arduino ▸ libraries ▸ ROHM_Steppers ▸ examples ▸ twoBoardsXStepAndFullStep w folder Name | Date modified | Type | Size twoBoardsXStepAndFullStep.ino | 10/22/2015 3:27 PM | Arduino file | 1 KB	**Example Sketch:** ▪ **Two boards with micro stepping and full step modes**
 ▸ Arduino ▸ libraries ▸ ROHM_Steppers ▸ src w folder Name | Date modified | Type | Size ROHM_Steppers.cpp | 11/11/2015 1:28 PM | C++ Source | 8 KB ROHM_Steppers.h | 11/11/2015 1:28 PM | C/C++ Header | 5 KB	**Implementation of the library**

Table 11: Contents of the ROHM STEPMO_EVK_20x software delivery package

● Library API Description

Class Instantiations:

```
ROHM_Stepper (int BoardNumber);   board number
ROHM_Stepper (int BoardNumber, int StepsType); board number and steps type
```

- Where **BoardNumber** is **BRDN1** or **BRDN2**
- **StepsType** is **FS** full step, **HS** half step (1/2), **HSA** half step mode A, **HSB** half step mode B, **QS** quarter step (1/4), **ES** eighth part of step (1/8), **XS** sixteenth part of step (1/16)
  Note: Available stepping modes depend on model version (see Table 1)

Examples for using in Arduino sketch:

```
    // initialize the ROHM Stepper class for BRD1 and BRD2 without stepping type
    ROHM_Stepper S1(BRDN1);
    ROHM_Stepper S2(BRDN2);
    // initialize for BRD1 and BRD2 classes with stepping type
    ROHM_Stepper S1(BRDN1,XS);
    ROHM_Stepper S2(BRDN2,FS);
```

Main API methods:

```
void enable();    chip enable  IC pin 20
void disable();   chip disable

void active();    chip active  mode  IC pin 14
void standby();   chip standby mode

void setStepsType(int StepsType); set stepping mode IC pins 18,19
```

Values from header file for StepsType:

```
ND Not defined, FS full step, HS half step (1/2),
QS quarter step (1/4), ES eighth part of step (1/8), XS sixteenth part of step (1/16)

int  getStepsType(); return current stepping mode
ND, FS, HS, QS, ES or XS

void go(int STEPS); run N steps CW or CCW depends of (+/-)STEPS IC pins 15,16
```

API methods for step timing setting up:

By default for half a clocking period are predefined values for
XS 1*16 (16) us,
QS 2*16 (32) us,
HS 4*16 (64) us,
FS 16 *16 (256) us

For reading back values of a half of period and setting new values use pairs of:

```
int getXSstep(); void setXSstep(int XShalfPeriod);
int getQSstep(); void setQSstep(int QShalfPeriod);
int getHSstep(); void setHSstep(int HShalfPeriod);
int getFSstep(); void setFSstep(int FShalfPeriod);
```

By default there is a no delay from clock to clock.
For reading back clock to clock delay or setting up new value use pair of:

```
int  getStSdelay( ); void setStSdelay(int usec );
```

All values in positive microseconds

DRAFT

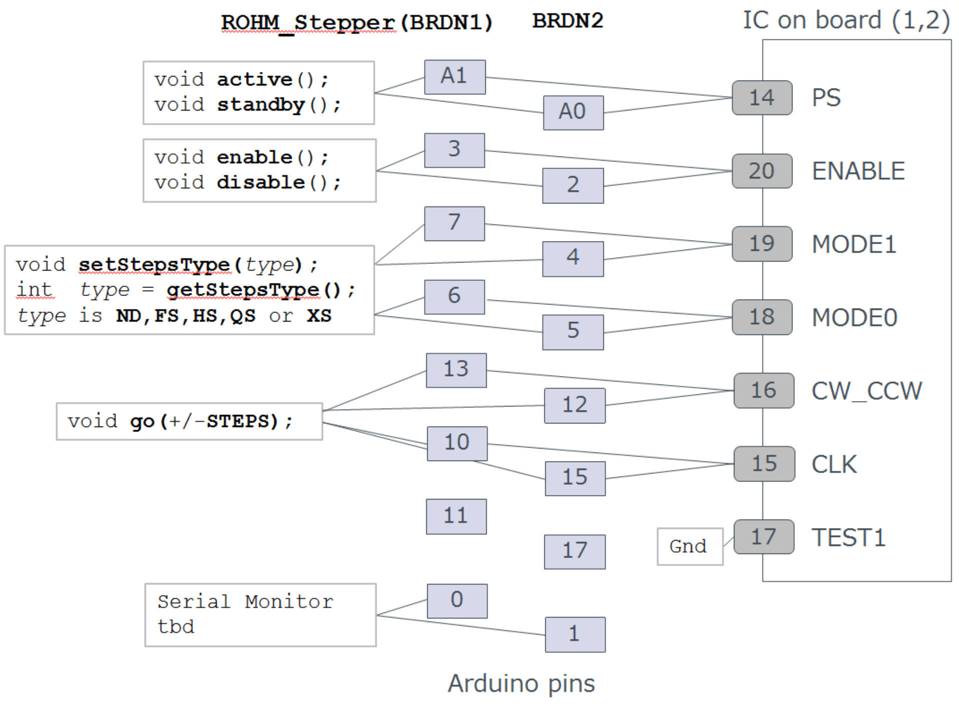


Figure 7: Functional connections

© 2016 ROHM Co., Ltd. All rights reserved.

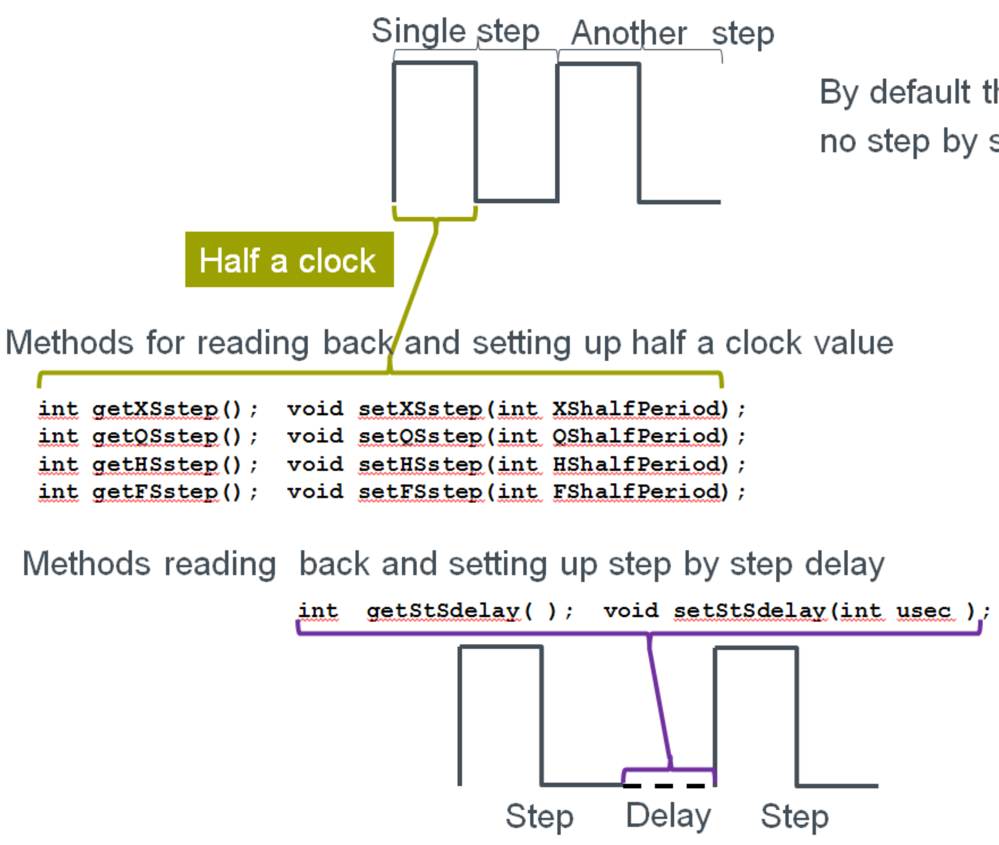
Declared but not yet implemented methods:

```
// return type of connected boards and initState
unsigned int present();

// return version of the library
int version(void);

// Operation with brushed motors
// start rotation of brushed motor to CW or CCW or STOPvoid
void run(int Direction);
void run(int Direction, int PWM);
void stop(void);
```

● Reading/Setting Clock Parameters


Single step
Another step
By default there is no step by step delay
Half a clock
Methods for reading back and setting up half a clock value
Methods reading back and setting up step by step delay
Step
Delay
Step

```
int getXSstep();  void setXSstep(int XShalfPeriod);
int getQSstep();  void setQSstep(int QShalfPeriod);
int getHSstep();  void setHSstep(int HShalfPeriod);
int getFSstep();  void setFSstep(int FShalfPeriod);

int  getStSdelay( );  void setStSdelay(int usec );
```

Figure 8: Reading and Setting Clock Parameters

● DEMO Sketch

```
/*Demo for one rotation cycle at full step mode*/
#include <ROHM_Steppers.h> // include header file
/* StepsType      FS HS QS XSBoardNumber    BRDN1 BRDN2*/
// initialize the ROHM Steppers libs for BRD1 and full step
ROHM_Stepper S1(BRDN1,FS); // one board full step
int  i = 200; // one rotation cycle at full step (200*1.8 degree = 360 degree)
void setup() // Arduino init. phase.
{
  S1.enable(); // enable board
  S1.active();
}
void loop()
{
  while(i)
  {
    S1.go(1);  // execute one step
    i--;
  }
}
```

● FAQ

How to initialize motor shield?

At sketch before init() function by instantiating ROHM_Stepper class

```
    ROHM_Stepper S1(BRDN1);  // board number without stepping mode
    ROHM_Stepper S1(BRDN1,FS); // board number with stepping  mode
```

Predefined values for boards: BRDN1, BRDN2 for stepping modes: FS, HS, HSA, HSB, QS, ES, XS
See examples for references.

How to set up stepping mode (full step, half step, etc)?

Here is two ways: at class instantiating and by class method

```
    ROHM_Stepper S1(BRDN1,FS); // at class instantiating
    S1.setStepsType(QS);      // by class method
```

return the current stepping mode by class method

```
    int type = getStepsType();
```

Predefined values for stepping modes: FS, HS, QS, ES, XS
FS - stepping mode by default.

How to set clock speed and direction

Predefined values for stepping are **XS** 16 us, **QS** 32 us, **HS** 64 us, **FS** 256 us and no time delay from step to step.

How to run motor for N steps

With class method go(+/-STEPS)
Example1 (5 steps CW):

```
    S1.go(5);
```

Example2 (10 steps CCW)

```
    S1.go(-10);
```

DRAFT

Notes

This EVK is

- only to be used as device under test for research and development purposes.
- only to be operated by trained professionals.

1. The information contained herein is subject to change without notice.
2. Before you use our Products, please contact our sales representative and verify the latest specifications :
3. Although ROHM is continuously working to improve product reliability and quality, semiconductors can break down and malfunction due to various factors. Therefore, in order to prevent personal injury or fire arising from failure, please take safety measures such as complying with the derating characteristics, implementing redundant and fire prevention designs, and utilizing backups and fail-safe procedures. ROHM shall have no responsibility for any damages arising out of the use of our Products beyond the rating specified by ROHM.
4. Examples of application circuits, circuit constants and any other information contained herein are provided only to illustrate the standard usage and operations of the Products. The peripheral conditions must be taken into account when designing circuits for mass production.
5. The technical information specified herein is intended only to show the typical functions of and examples of application circuits for the Products. ROHM does not grant you, explicitly or implicitly, any license to use or exercise intellectual property or other rights held by ROHM or any other parties. ROHM shall have no responsibility whatsoever for any dispute arising out of the use of such technical information.
6. The Products are intended for use in general electronic equipment (i.e. AV/OA devices, communication, consumer systems, gaming/entertainment sets) as well as the applications indicated in this document.
7. The Products specified in this document are not designed to be radiation tolerant.
8. For use of our Products in applications requiring a high degree of reliability (as exemplified below), please contact and consult with a ROHM representative: transportation equipment (i.e. cars, ships, trains), primary communication equipment, traffic lights, fire/crime prevention, safety equipment, medical systems, servers, solar cells, and power transmission systems.
9. Do not use our Products in applications requiring extremely high reliability, such as aerospace equipment, nuclear power control systems, and submarine repeaters.
10. ROHM shall have no responsibility for any damages or injury arising from non-compliance with the recommended usage conditions and specifications contained herein.
11. ROHM has used reasonable care to ensure the accuracy of the information contained in this document. However, ROHM does not warrants that such information is error-free, and ROHM shall have no responsibility for any damages arising from any inaccuracy or misprint of such information.
12. Please use the Products in accordance with any applicable environmental laws and regulations, such as the RoHS Directive. For more details, including RoHS compatibility, please contact a ROHM sales office. ROHM shall have no responsibility for any damages or losses resulting non-compliance with any applicable laws or regulations.
13. When providing our Products and technologies contained in this document to other countries, you must abide by the procedures and provisions stipulated in all applicable export laws and regulations, including without limitation the US Export Administration Regulations and the Foreign Exchange and Foreign Trade Act.
14. This document, in part or in whole, may not be reprinted or reproduced without prior consent of ROHM.

DRAFT

ROHM SEMICONDUCTOR

Thank you for your accessing to ROHM product information.
More detailed product information and catalogs are available, please contact us.

ROHM Customer Support System

http://www.rohm.com/contact/