GENERAL DESCRIPTION

The MC3479 is a small form factor, integrated digital output 3-axis accelerometer with a feature set optimized for cell phones and IoT product motion sensing. Applications include user interface control, gaming motion input, electronic compass tilt compensation for cell phones, game controllers, remote controls and portable media products.

The MC3479 features a dedicated motion block which implements embedded algorithms to support “any motion” and shake detection, tilt/flip and tilt 35 position detection.

The EV3479A is a prebuilt circuit board with MC3479 LGA-12 3-axes sensor. The MC3479 has internal sample rate from 0.5 to 1000 samples / second and measures acceleration with a wide usage range, from +/-2g up to +/-16g, and 16-bit high precision ADC output, which is easy to fit on top of the microcontroller, such as an Arduino. The accelerometer communicates via I2C/SPI and gives out motion detection or sample acquisition conditions to trigger an interrupt toward an MCU.

The sensor data is easily readable by connecting DVDD to 3.3V, GND to ground, and SCL/SDA pins to your Arduino I2C clock and data pin respectively. Download the MC3479 library from MEMSIC onto the board, run the example sketch, and then sensor data shortly comes out in raw data count and SI unit accelerometer measurements. An easy-to-use demonstration on EV3479A using the Arduino platform is depicted in this document.

MC3479 FEATURES

Range, Sampling & Power

- ±2,4,8,12 or 16g range
- 16-bit single sample resolution
- 16-bit resolution with FIFO
- 0.5 to 1000 Hz Output Data Rate
- 4 µA typical Standby current
- Low typical active current

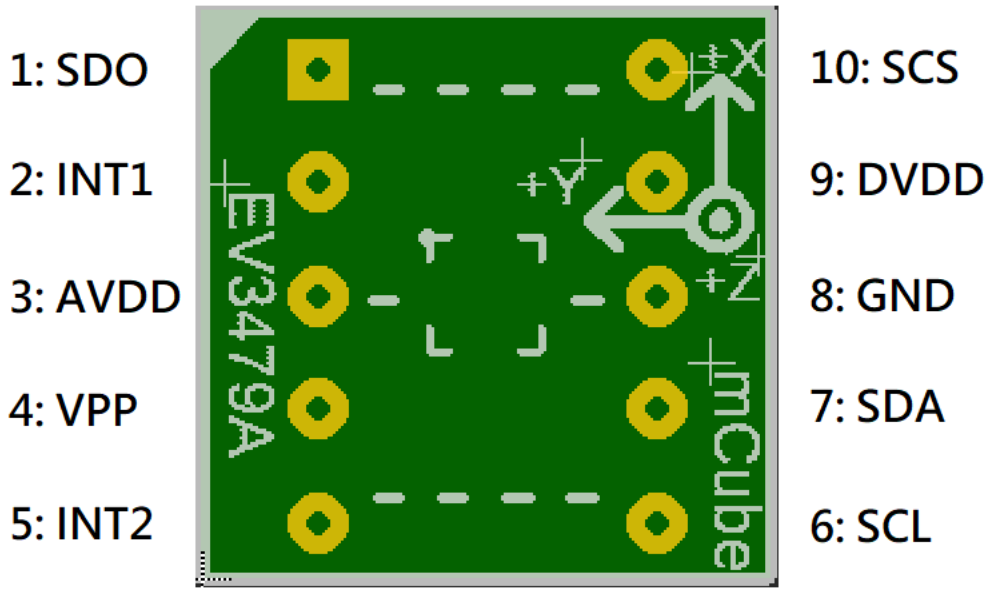
Simple System Integration

- I2C interface, up to 1 MHz
- SPI Interface, up to 10 MHz
- 2×2×0.92 mm 12-pin LGA package
- Single-chip 3D silicon MEMS
- RoHS compliant

TABLE OF CONTENTS

1 GENERAL OPERATION

1.1 PINOUTS

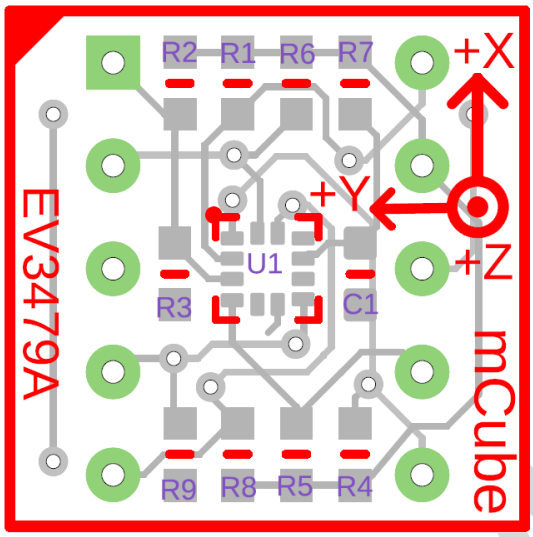


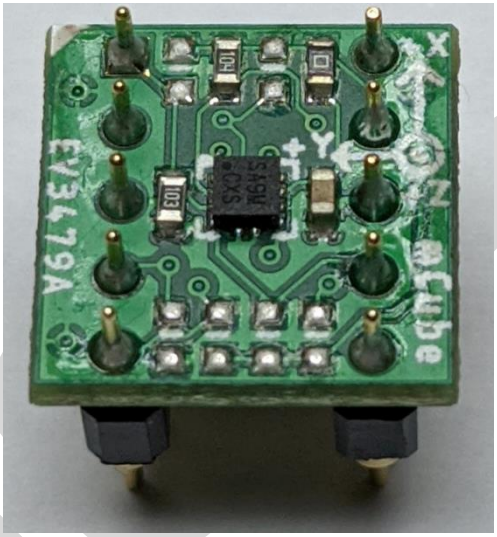
1.2 POWER PINS

- **DVDD** – 3.3V Power Supply Input
- **GND** – Ground Pin for Power and Logic
- **R7**: The current draw from the sensor can be measured by putting an ammeter in place of R7.
- In the following demonstration, an Arduino DUE is used to illustrate on how to test the evaluation board with a microcontroller.
- Please be advised that if an Arduino UNO is used instead, hardware modification on Arduino UNO MUST be made for it to output at 3.3V. (WARNING: attempting to power the part at 5V is likely to damage it.)
  By default, Arduino UNO operates at 5V, which is higher than the maximum voltage rating for the evaluation board. Please refer to an excellent tutorial on modifying Arduino UNO to output at 3.3V:
  https://learn.adafruit.com/arduino-tips-tricks-and-techniques/3-3v-conversion

1.3 I2C PINS

- Connect the **SCL** (I2C clock pin) to your microcontroller's I2C clock line.
- Connect the **SDA** (I2C data pin) to your microcontroller's I2C data line.





R4, R5: If using I2C and I2C pull-up resistors are needed for your application then install~4.7KΩ resistors into R4 (SCL clock pin) and R5 (SDA data pin) which are not installed by factory default. In addition, besides soldering resisters on R4/R5, you can add axial lead 4.7K ohm resistors to the SDA and SCL pin respectively. It will work the same either way.

NOTE: DO NOT install more than one setup pull-up resistors per I2C bus.

1.4 SPI PINS

With an SPI connection, there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically, there are four wires commonly connected to all the devices:

Connect the **SCS** (Slave Chip Select) to the pin on the device that the master can use to enable and disable SPI cycles.

Connect the **SCL** (Serial clock) to the pin where the clock pulses synchronize data transmission generated by the master

Connect **SDO** to the pin where the Slave sends data to the master (Master Input, Slave Output).

Connect **SDA** to the pin where the Master sends data to the peripherals (Master Output, Slave Input).

1.5 INTERRUPT PINS

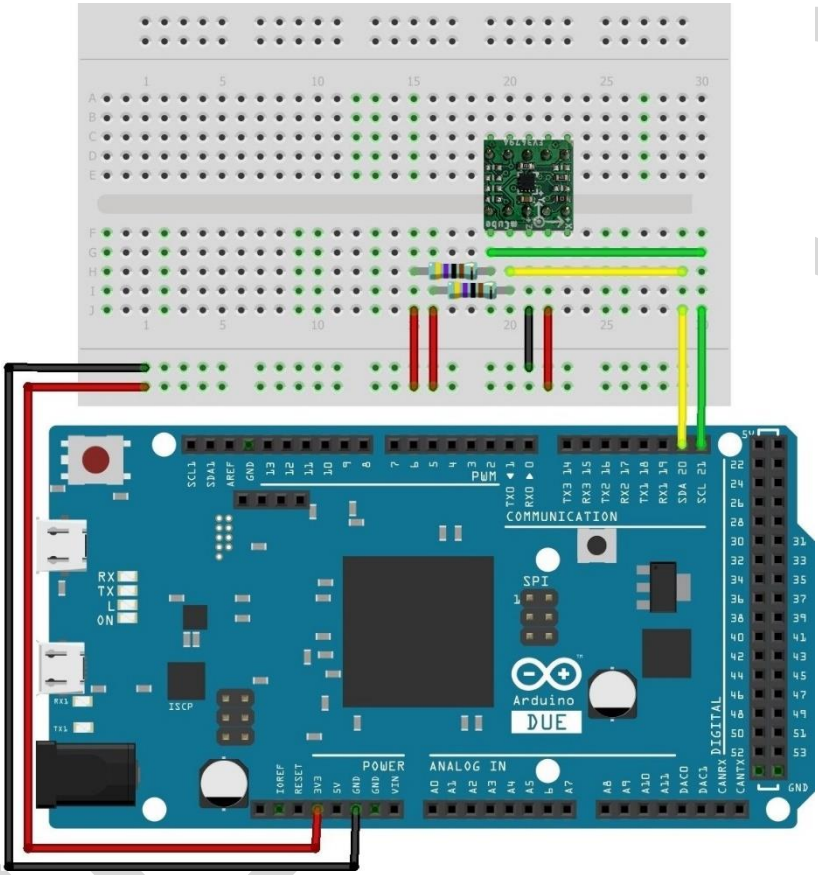
INT - HW interrupt signal pin. This pin will be triggered by the device when data is ready to read, or a motion event is detected by the accelerometer. (Not currently supported in the library for the interrupt pin, so please check the datasheet for the I2C commands and related registers).

R6: If using the sensor interrupt signal as open-drain, then install pull-up resistor ~4.7KΩ into R6 (not installed by default).

2 ASSEMBLY AND TEST

Please note that the SPI and I2C interfaces cannot both be active at the same time as the clock (SCK) and data (SDA) are shared between the two protocols.

2.1 I2C INTERFACE

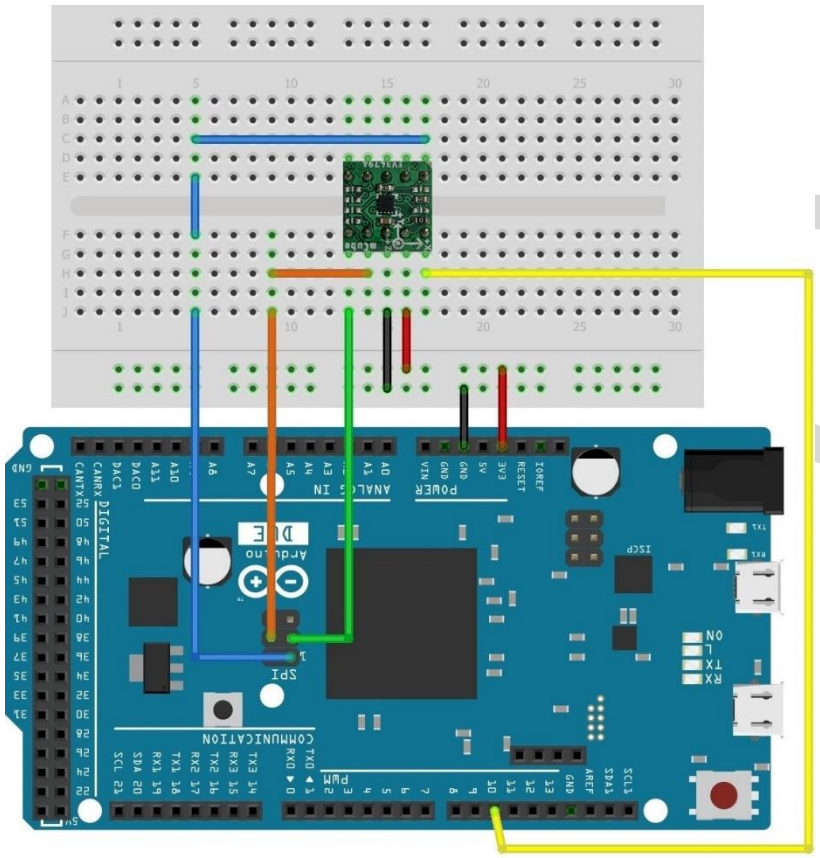


The EV3479A evaluation board can be easily wired to any microcontroller. This example shows a typical Arduino DUE platform. For other microcontrollers, be sure it has I2C with repeated-start support, then port the code. Please refer to the illustration below to connect the related pins.

- Connect **DVDD** to the power supply, **3.3V**. (WARNING: attempting to power the part at a voltage exceeds the maximum rating of 3.6V is likely to damage it.)
- Connect **GND** to common power/data ground.
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino.
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino.

The MC3479 has a default I2C address of 0x4C and it can be changed to 0x6C by tying the SDO pin to VDD.

2.2 SPI INTERFACE



The EV3479A evaluation board can be easily wired to any microcontroller. This example shows a typical Arduino DUE platform. Please refer to the illustration below for connecting the related pins and then port the code to get the raw X, Y, Z sensor data.

- Connect **DVDD** to the power supply, **3.3V**. (WARNING: attempting to power the part at a voltage exceeds the maximum rating of 3.6V is likely to damage it.)
- Connect **GND** to common power/data ground.
- Connect **SCL** to ICSP-3 as Serial Clock.
- Connect **SDO** to ICSP-1 as Master Input, Slave Output.
- Connect **SDA** to ICSP-4 as Master Output, Slave Input.
- Connect **SCS** to digital I/O pin **10** as Slave Chip Select.

3 DEMO

3.1 GET THE DRIVER FROM MEMSIC

To begin reading sensor data, you will need to get the MC3479 Library from MEMSIC.

Get the ZIP file from MEMSIC and check the zip file consisting of **MC34X9.cpp**, **MC34X9.h** and examples directory.

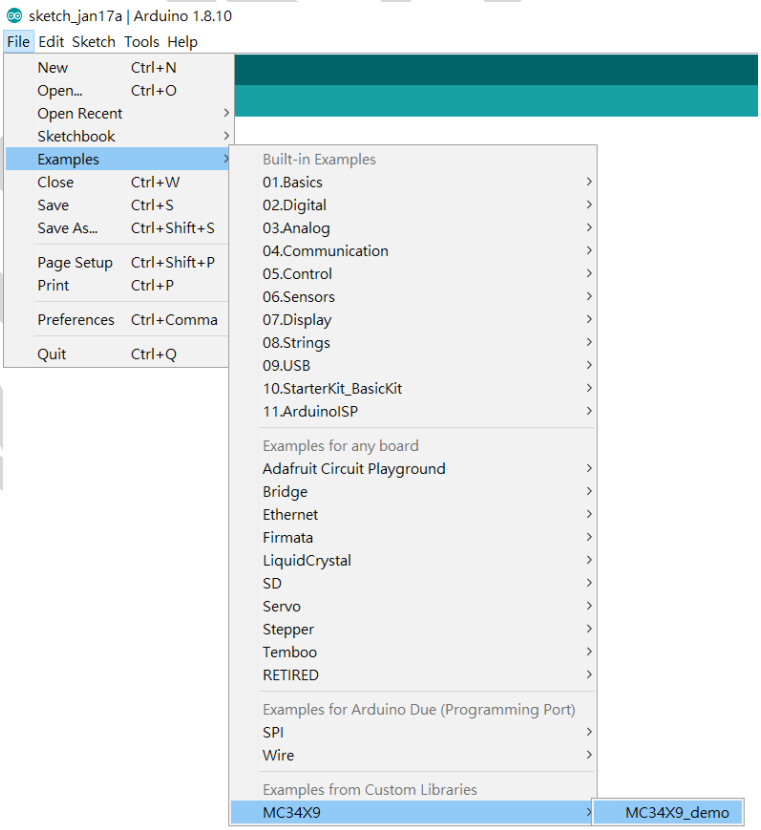
An excellent tutorial on Arduino library installation is located at:

https://www.arduino.cc/en/guide/libraries#toc4

You may need to create the library subfolder if it is your first library files. Then just restart the IDE.

3.2 LOAD THE DEMO

Open File->Examples->MC34X9-> MC34X9_demo and upload to your Arduino while it is wired to the sensor.



If you need the sensor running on SPI, please configure the bus as SPI in the MC34X9_demo.ino shown as below. Otherwise, the default is I2C bus.

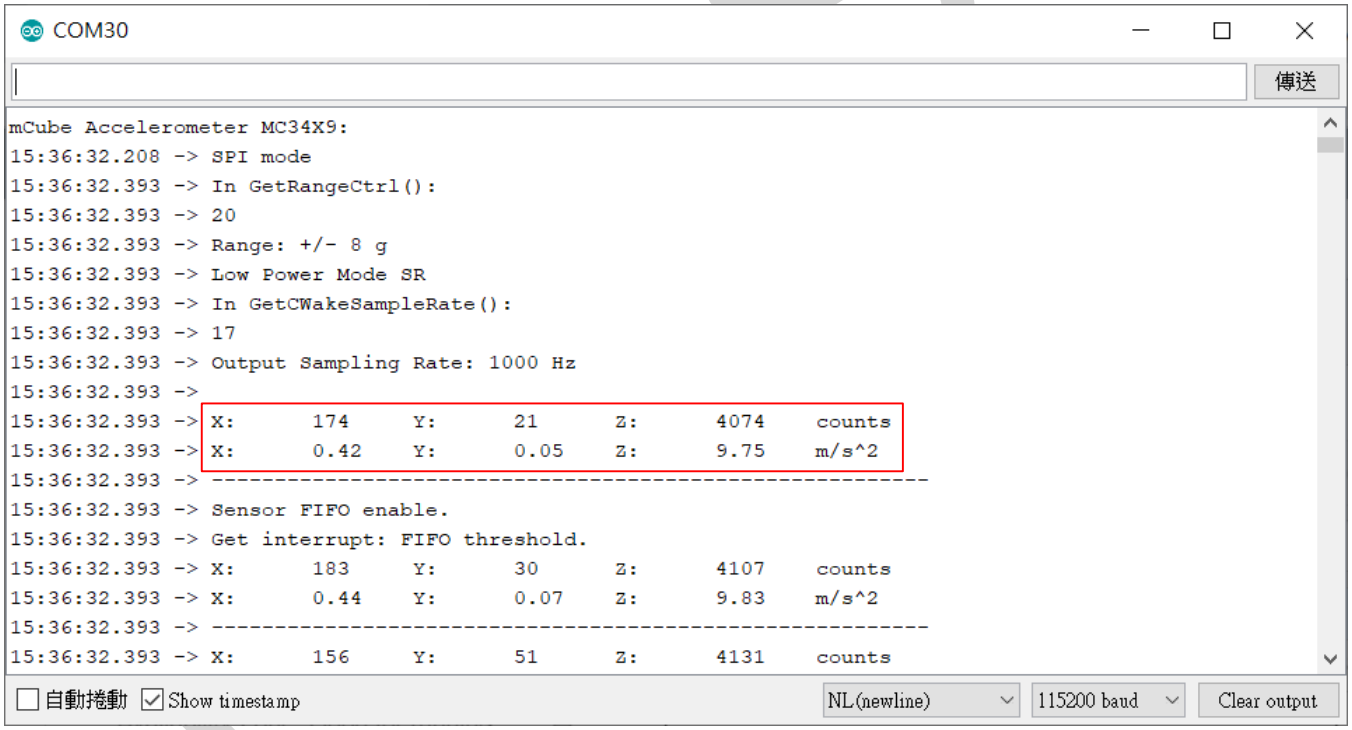
```
/*** bSpi: I2C/SPI bus selection. SPI: 0, I2C: 1 ***/
const uint8_t bSpi = 1;
```

Setup I2C interface as 2.1 will get I2C address 0x4C.

Setup SPI interface as 2.2 will get SPI select pin 10.

```
/* Chip Select & Address */
uint8_t chipSelect = 0;
const uint8_t SPIChipSelectPin = 10; // SPI chipSelectPin
const uint8_t I2CAddress = 0x4c; // I2C address
```

Now open the serial terminal window at 115,200 baud rate speed to begin the test.



You will see the output from the serial terminal showing the current range scale and resolution of the sensor in the first three lines followed by two lines of output sensor data at some output data rate which depict “raw count" data for line 1: X: 174 Y: 21 Z: 4074 with 8G range, 16bit ADC resolution. Line 2 indicates the SI units for measuring acceleration as X: 0.42 m/s^2 Y: 0.05 m/s^2 Z: 9.75 m/s^2.

To enable FIFO mode, which is controlled by the configurations below. FIFO size could be set up to maximum 31 samples. Enable FIFO threshold interrupt will automatically enable FIFO feature.

```
/*** FIFO control ***/
int FIFO_THRE_SIZE = 30;
/* FIFO Interrupt */
const bool enableFifoThrINT = false;
/* For FIFO feature, enable FIFO interrupt will automatically enable FIFO feature */
bool enableFIFO = false;
```

To control embedded motion blocks, which is enabled by the configurations below. Enable individual motion mode will also enable corresponding interrupt.

These modes can only be enabled separately.

```
/*** Motion control ***/
/* Enabling motion feature below also enables corresponded motion interrupt */
const bool enableTILT = false;
const bool enableFLIP = false;
const bool enableANYM = true;
const bool enableSHAKE = false;
const bool enableTILT_35 = false;
```

4 LIBRARY REFERENCE

4.1 CREATE MEMSIC_MC34X9 OBJECT

You can create the MEMSIC_MC34X9 object with:

```
MC34X9 MC34X9_acc = MC34X9();
```

4.2 INITIALIZE AND CONFIGURE SENSOR

Initialize and configure the sensor with:

```
MC34X9_acc.start(bSpi, chipSelect);
```

Wake up sensor with your own configuration, it will follow the factory default setting:

```
MC34X9_acc.wake();
```

Stop sensor to change setting:

```
MC34X9_acc.stop();
```

4.3 SET RANGE

Set the accelerometer max range to ±2g, ±4g, ±8g, ±16g or ±12g with:

```
MC34X9_acc.SetRangeCtrl(MC34X9_RANGE_8G);
```

4.4 READ RANGE

Read the current range with:

```
MC34X9_acc.GetRangeCtrl();
```

It returns: 0 for ±2g, | 1 for ±4g, | 2 for ±8g | 3 for ±16g | 4 for ±12g.

4.5 SET SAMPLING RATE

Set the accelerometer sampling rate with:

```
MC34X9_acc.SetSampleRate(MC34X9_ SR_DEFAULT_1000Hz);
```

4.6 READ SAMPLING RATE

Read the current sampling rate with:

```
MC34X9_acc.GetSampleRate ();
```

It returns sampling rate from 25 ~ 1000 Hz.

4.7 CONFIG MOTION FEATURE

To configure motion feature, open MC34X9.h

Set the threshold and debounce value used by the tilt and flip logic for activity detection:

```
#define s_bCfgFTThr                     100
#define s_bCfgFTDebounce                50
```

Set the threshold and debounce value used by the any motion logic for activity detection:

```
#define s_bCfgANYMThr                   200
#define s_bCfgANYMDebounce              100
```

Set the threshold, duration and count value used by the shake logic for activity detection:

```
#define s_bCfgShakeThr                  300
#define s_bCfgShakeP2PDuration          10
#define s_bCfgShakeCount                1
```

Set the threshold and duration value used by the tilt35 logic for activity detection:

```
#define s_bCfgTILT35Thr                 20
#define s_bCfgTILT35Timer               MC34X9_TILT35_2p0
```

4.8 CONFIG INTERRUPT MODE

Configure the FIFO interrupt mode with:

```
MC34X9_acc.SetFIFOINTCtrl(false, false, enableFifoThrINT);
```

Configure the motion interrupt mode with:

```
MC34X9_acc.SetINTCtrl(enableTILT, enableFLIP, enableANYM, enableSHAKE,
enableTILT_35);
```

MC34X9 has 5 motion interrupt modes – TILT | FILP | ANYM | SHAKE | TILT35.

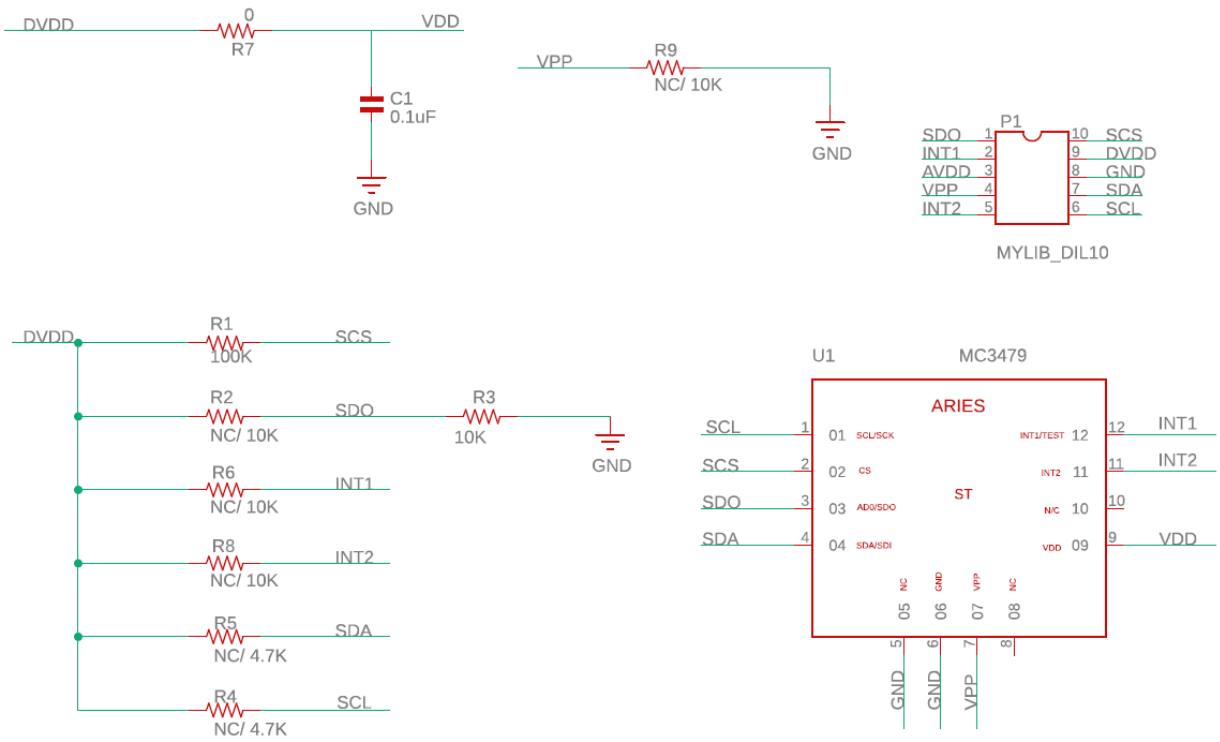
These modes can only be enabled separately.

4.9 READ RAW COUNT DATA

Read the raw count data and SI unit measurement with:

```
MC34X9_acc.readRawAccel();
```

5 SCHEMATICS



Above is a schematic on EV3479A. This is the factory preset when receiving the part.

For other options, please refer to the following table:

Interface	R1	R2	R3
SPI or I2C 0x4C (**Factory default**)	100KΩ	DNI[1]	10KΩ
SPI or I2C 0x6C	100KΩ	10KΩ	DNI

R4, R5: Install ~4.7KΩ as the I2C pull-ups if using I2C mode and there is no I2C pull-ups installed. (DNI by default)

Note: It is recommended not to install more than one pull-ups per I2C bus.

R6: Install ~4.7KΩ pull-up resistor if setting the sensor interrupt pin to open-drain. (DNI by default)

R7: Sensor's driving current can be measured by putting an ammeter in place of R7.

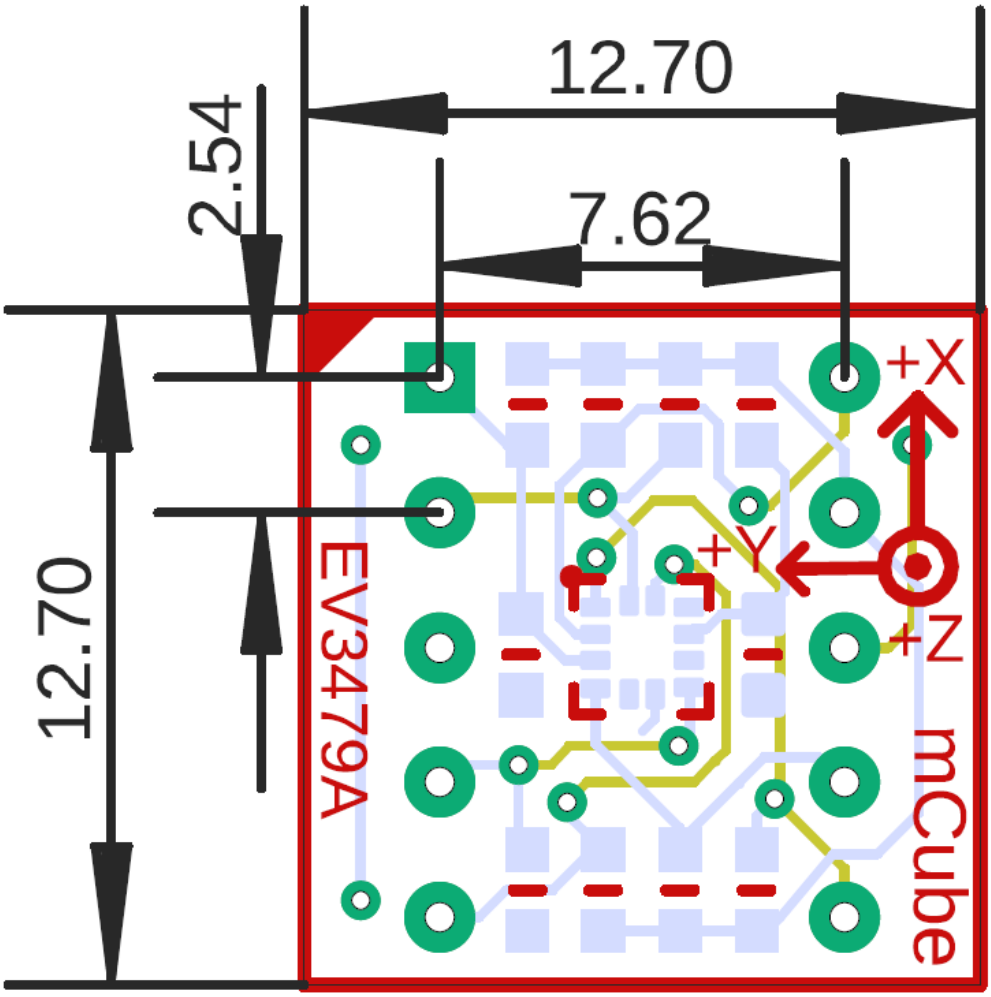
The physical location of the resistor is in the diagram in Section 8.

[1] DNI: Do Not Install

6 BILL OF MATERIALS

Item	Part	Value	Package	Vendor	P/N	Install
1	C1	0.1uF	CAP-0603	Walsin	0603B104K500	Yes
2	J1	DIL10-A	DIL10-A	-	-	Yes
3	R1	100K	RES-0603	Walsin	WR06X103JTL	Yes
4	R3	10K	RES-0603	Walsin	WR06X102JTL	Yes
5	R7	0R	RES-0603	Walsin	WR06X000PTL	Yes
6	U1	MC3479	LGA-12_2.0 x 2.0	MEMSIC	MC3479	Yes
7	R2	10K	RES-0603	Walsin	WR06X472JTL	No
8	R4	4.7K	RES-0603	Walsin	WR06X472JTL	No
9	R5	4.7K	RES-0603	Walsin	WR06X472JTL	No
10	R6	4.7K	RES-0603	Walsin	WR06X472JTL	No
11	R9	10K	RES-0603	Walsin	WR06X472JTL	No

7 FABRICATION PRINT



NOTE: All dimensions are in millimeters.

8 NOTE

All the mCube logo in the document will be replaced later by MEMSIC logo.

9 REVISION HISTORY

Date	Revision	Description
2020-02-05	APS-045-0034v1.0	First release.
2020-08-13	APS-045-0034v1.1	Change to MEMSIC format based on the License Agreement with mCube.