GENERAL DESCRIPTION

The MC3635 is an ultra-low power, low-noise, integrated digital output 3-axis accelerometer with a feature set optimized for wearables and consumer product motion sensing. Applications include wearable consumer products, IoT devices, user interface control, gaming motion input, electronic compass tilt compensation for cell phones, game controllers, remote controls and portable media products.

The EV3635A/B is a prebuilt circuit board with MC3635 3-axes sensor. The MC3635 has internal sample rate from 14 to 1300 samples / second and measures acceleration with a wide usage range, from +/-2g up to +/-16g, and 6-bit to 14-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 MC3635 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 EV3635A/B using the Arduino platform is included in this document.

MC3635 FEATURES

Range, Sampling & Power

- ±2, 4, 8, 12 or 16g ranges
- 8, 10 or 12-bit resolution with FIFO
  - 14-bit single samples
- Sample rate 14 - 1300 samples/sec
- Sample trigger via internal oscillator, clock pin or software command
- Sniff and Wake modes
- 0.4 μA Sniff current @ 6Hz
- Separate or combined sniff/wake
- Ultra-Low Power with 32 sample FIFO
  - 0.9 μA typical current @ 25Hz
  - 1.6 μA typical current @ 50Hz
  - 2.8 μA typical current @ 100Hz
  - 36 μA typical current @ 1300Hz

Simple System Integration

- I2C interface, up to 1 MHz
- SPI Interface, up to 8 MHz
- 1.6 × 1.6 × 0.94 mm 10-pin package
- Single-chip 3D silicon MEMS
- Low noise to 2.3mgRMS

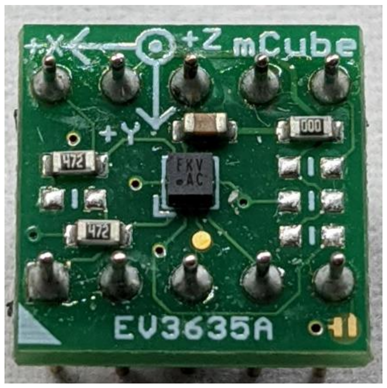


TABLE OF CONTENTS

1 General Operation 3

1.1 Pinouts 3

1.2 Power Pins 3

1.3 I2C Pins 4

1.4 SPI Pins 5

1.5 Interrupt Pins 5

2 Assembly and Test 6

2.1 I2C Interface 6

2.2 SPI Interface 7

3 Demo 8

3.1 GET the Driver from MEMSIC 8

3.2 Load the Demo 8

4 Library Reference 11

4.1 Create MEMSIC_MC36XX Object 11

4.2 Initialize and Configure Sensor 11

4.3 Set Range 11

4.4 Read Range 11

4.5 Set Resolution 11

4.6 Read Resolution 12

4.7 Set CWake Sampling Rate 12

4.8 Read CWake Sampling Rate 12

4.9 Set Sniff Sampling Rate 12

4.10 Read Sniff Sampling Rate 12

4.11 Config Sniff Mode 12

4.12 Config Interrupt Mode 13

4.13 Read Raw Count Data 13

5 Schematics 14

6 Bill of Materials 16

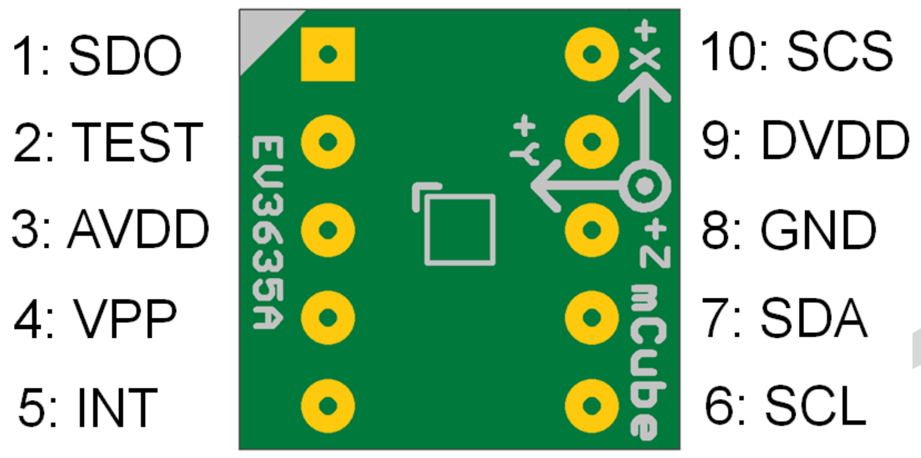
7 Fabrication Print 17

8 NOTE 18

9 Revision History 19

1 GENERAL OPERATION

1.1 PINOUTS

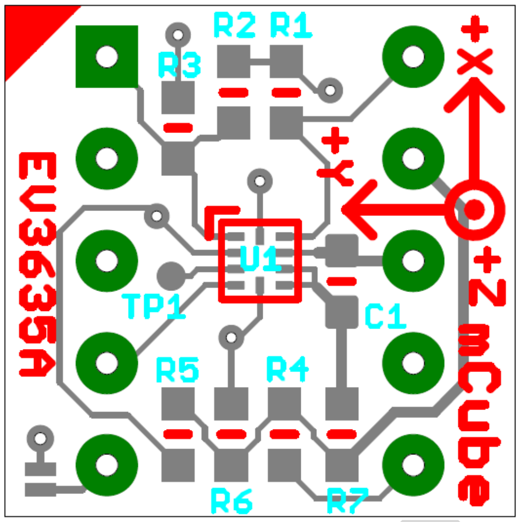


1.2 POWER PINS

- **DVDD** – 3.3V Power Supply Input
- **GND** – Ground Pin for Power and Logic
- **R7**: The current drawn 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

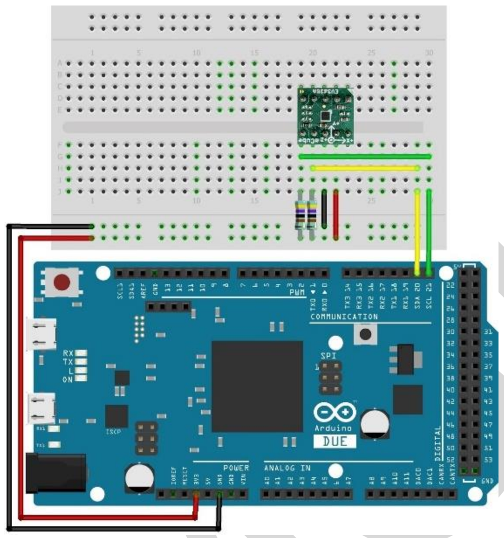
INT - HW interrupt signal pin. It will be driven by the chip when data is ready to read, or a motion event is detected by the accelerometer. (Not currently supported in the library for the interrupt pin, so please check the datasheet for the I2C commands toward related registers).

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

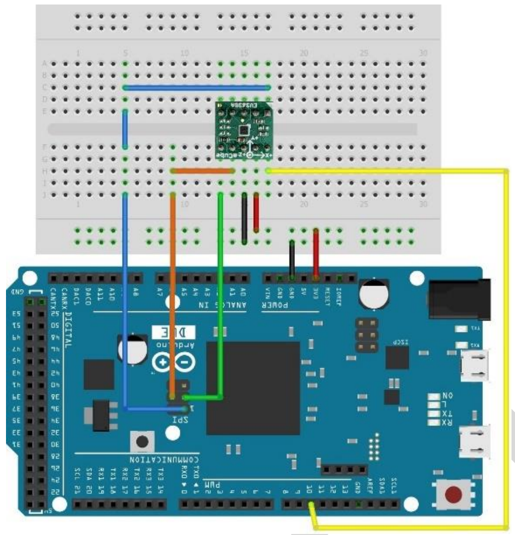


The EV3635A/B evaluation board can be easily wired to any microcontroller. This example shows a typical Arduino DUE platform. For other microcontrollers, be sure it has I2C with repeated-start support, then port the code. Please refer to the illustration below to connect the related pins.

- 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 MC3635 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 EV3635A/B 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 MC3635 Library from MEMSIC.

Rename the uncompressed folder **Accelerometer_MC36XX** and check that the Accelerometer_MC3635 folder contains **MC36XX.cpp** and **MC36XX.h**

If you need the sensor running on SPI, please configure the bus as SPI in the MC36XX.h shown as below. Otherwise, the default is I2C bus.

```
//#define MC36XX_CFG_BUS_I2C

#define MC36XX_CFG_BUS_SPI
```

SPI could support 8MHz speed if high speed mode is enabled as below.

```
#define SPI_HS
```

Place the Accelerometer_MC3635 library folder to your **Arduino_sketch_folder/libraries/** folder.

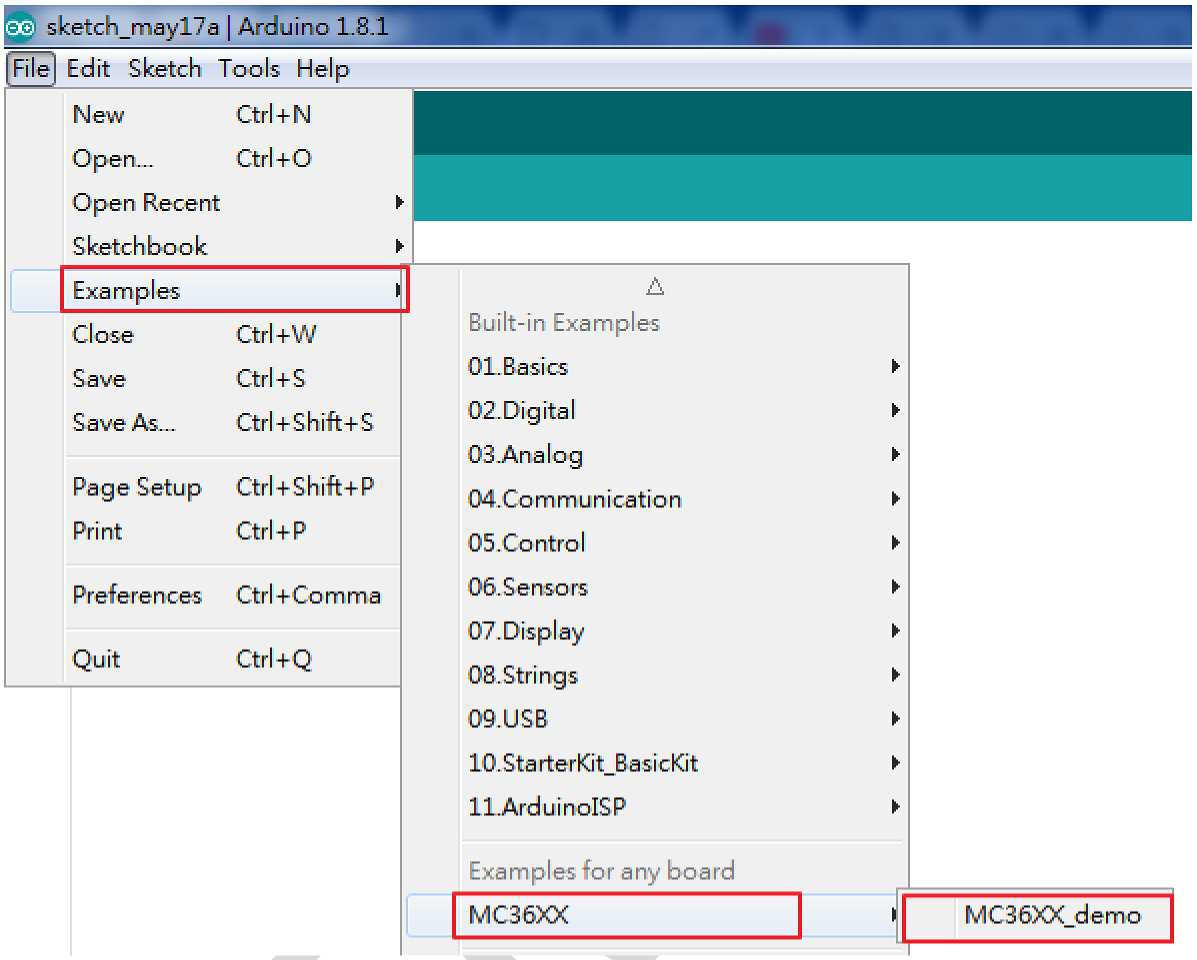
You may need to create the library subfolder if it is your first library. Then just restart the IDE.

An excellent tutorial on Arduino library installation is located at:

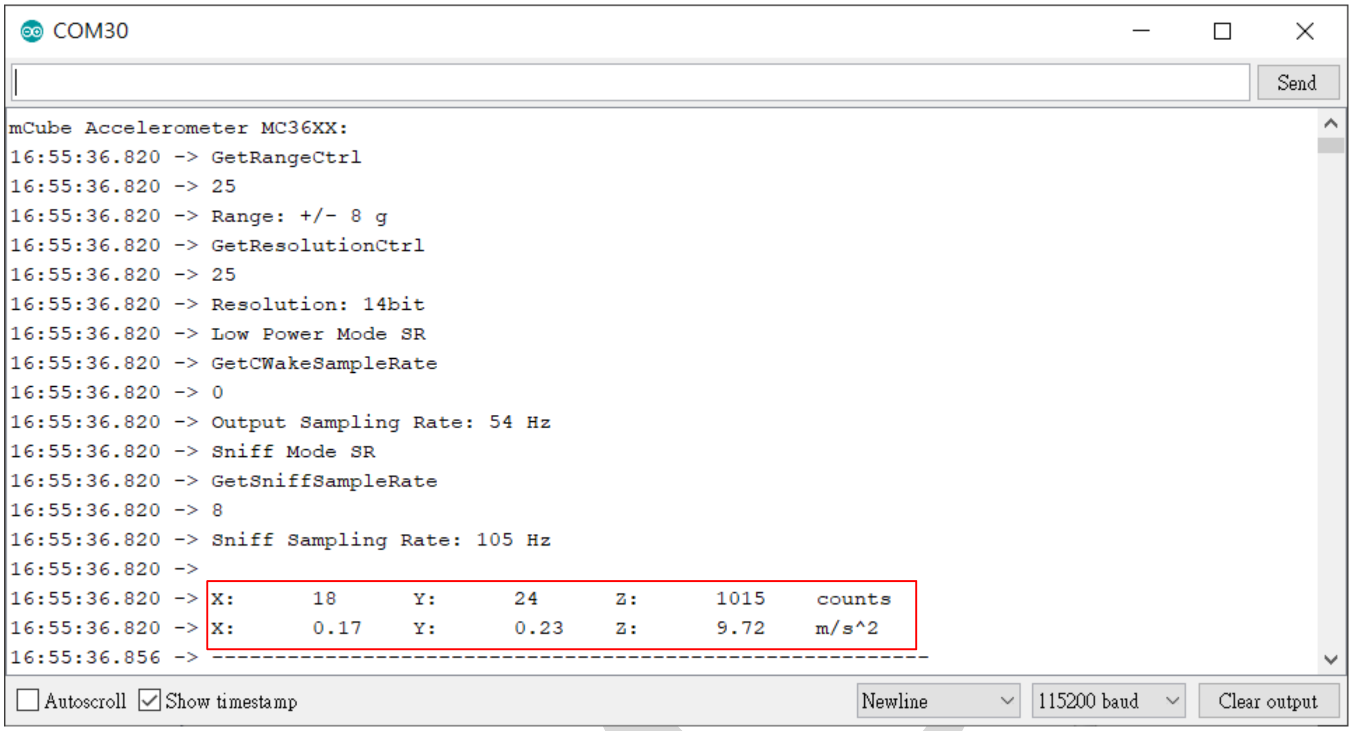
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

3.2 LOAD THE DEMO

Open up File->Examples-> MC36XX -> MC36XX_demo and upload to your Arduino wired up to the sensor



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



```
mCube Accelerometer MC36XX:
16:55:36.820 -> GetRangeCtrl
16:55:36.820 -> 25
16:55:36.820 -> Range: +/- 8 g
16:55:36.820 -> GetResolutionCtrl
16:55:36.820 -> 25
16:55:36.820 -> Resolution: 14bit
16:55:36.820 -> Low Power Mode SR
16:55:36.820 -> GetCWakeSampleRate
16:55:36.820 -> 0
16:55:36.820 -> Output Sampling Rate: 54 Hz
16:55:36.820 -> Sniff Mode SR
16:55:36.820 -> GetSniffSampleRate
16:55:36.820 -> 8
16:55:36.820 -> Sniff Sampling Rate: 105 Hz
16:55:36.820 ->
16:55:36.820 -> X:      18      Y:      24      Z:      1015    counts
16:55:36.820 -> X:      0.17    Y:      0.23    Z:      9.72    m/s^2
16:55:36.856 -> ---------------------------------------------------------
```

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: 18   Y: 24   Z: 1015   with 8G range, 14bit ADC resolution.

Line 2 indicates the SI units for measuring acceleration as X: 0.17 m/s^2   Y: 0.23 m/s^2   Z: 9.72 m/s^2.

This demo also includes the example for FIFO and Sniff interrupt mode. Those could be enabled by modify the definition below. **These two examples must be run separately**.

```
#define ENABLE_FIFO_WAKEUP          1
#define ENABLE_SNIFF_SHAKE_WAKEUP   0
```

Default input pin for interrupt is pin 8 and default FIFO threshold is 3 samples. FIFO size could be set to maximum 32 samples or just enable FIFO to FIFO_FULL mode.

```
#define INTERRUPT_PIN               8
#define FIFO_SIZE                   3
```

4 LIBRARY REFERENCE

4.1 CREATE MEMSIC_MC36XX OBJECT

You can create the MEMSIC_MC36XX object with:

```
MC36XX MC36XX_acc = MC36XX();
```

4.2 INITIALIZE AND CONFIGURE SENSOR

Initialize and configure the sensor with:

```
MC36XX_acc.start();
```

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

```
MC36XX_acc.wake();
```

Stop sensor to change setting:

```
MC36XX_acc.stop();
```

Set sensor as sniff mode:

```
MC36XX_acc.sniff ();
```

4.3 SET RANGE

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

```
MC36XX_acc.SetRangeCtrl(MC36XX_RANGE_8G);
```

4.4 READ RANGE

Read the current range with:

```
MC36XX_acc.GetRangeCtrl();
```

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

4.5 SET RESOLUTION

Set the accelerometer resolution to 6, 7, 8, 10, 12 or 14 bit.

```
MC36XX_acc.SetResolutionCtrl(MC36XX_RESOLUTION_14BIT);
```

When the FIFO is enabled, the output of the FIFO is mapped to registers 0x02 to 0x07, and the data has a maximum resolution of 12-bits.

4.6 READ RESOLUTION

Read the current resolution with:

```
MC36XX_acc.GetResolutionCtrl();
```

It returns: 0 for 6-bit | 1 for 7-bit | 2 for 8-bit | 3 for 10-bit | 4 for 12-bit | 5 for 14-bit.

4.7 SET CWAKE SAMPLING RATE

Set the accelerometer CWake mode sampling rate with:

```
MC36XX_acc.SetCWakeSampleRate(MC36XX_CWAKE_SR_DEFAULT_54Hz);
```

4.8 READ CWAKE SAMPLING RATE

Read the current CWake sampling rate with:

```
MC36XX_acc.GetCWakeSampleRate();
```

It returns sampling rate from 14 ~ 600 Hz.

4.9 SET SNIFF SAMPLING RATE

Set the accelerometer sniff mode sampling rate with:

```
MC36XX_acc.SetSniffSampleRate(MC36XX_SNIFF_SR_DEFAULT_7Hz);
```

4.10 READ SNIFF SAMPLING RATE

Read the current sniff sampling rate with:

```
MC36XX_acc.GetSniffSampleRate();
```

This returns sampling rate from 0.4 ~ 600 Hz.

4.11 CONFIG SNIFF MODE

Set the threshold values used by the SNIFF logic for activity detection:

```
MC36XX_acc.SetSniffThreshold(MC36XX_AXIS_X,5);
```

All three axes could be configured separately with different threshold value.

Set the threshold values used by the SNIFF logic for activity detection:

```
MC36XX_acc.SetSniffDetectCount(MC36XX_AXIS_X,3);
```

For each axis, a delta count is generated and compared to the threshold. When the delta count is greater than the threshold, a SNIFF wakeup event occurs. There is a unique sniff threshold for each axis, and an optional “false detection count” which requires multiple sniff detection events to occur before a wakeup condition is declared.

Configure sniff and/or mode with:

```
MC36XX_acc.SetSniffAndOrN(MC36XX_ANDORN_OR);
```

The SNIFF block supports the logical AND or OR of the X/Y/Z SNIFF wakeup flags when generating a SNIFF wakeup interrupt.

Configure sniff delta mode with:

```
MC36XX_acc.SetSniffDeltaMode(MC36XX_DELTA_MODE_C2P);
```

C2P mode: The current sample and the immediate previous sample are subtracted generate a delta

C2B mode: The current sample and the first sample captured when entering SNIFF mode are subtracted to generate a delta.

4.12 CONFIG INTERRUPT MODE

Configure the interrupt mode with:

```
MC36XX_acc.SetINTCtrl(0,0,0,0,1);
```

MC36XX have 5 interrupt modes – FIFO_THR | FIFO FULL | FIFO EMPTY | ACQ | WAKE.

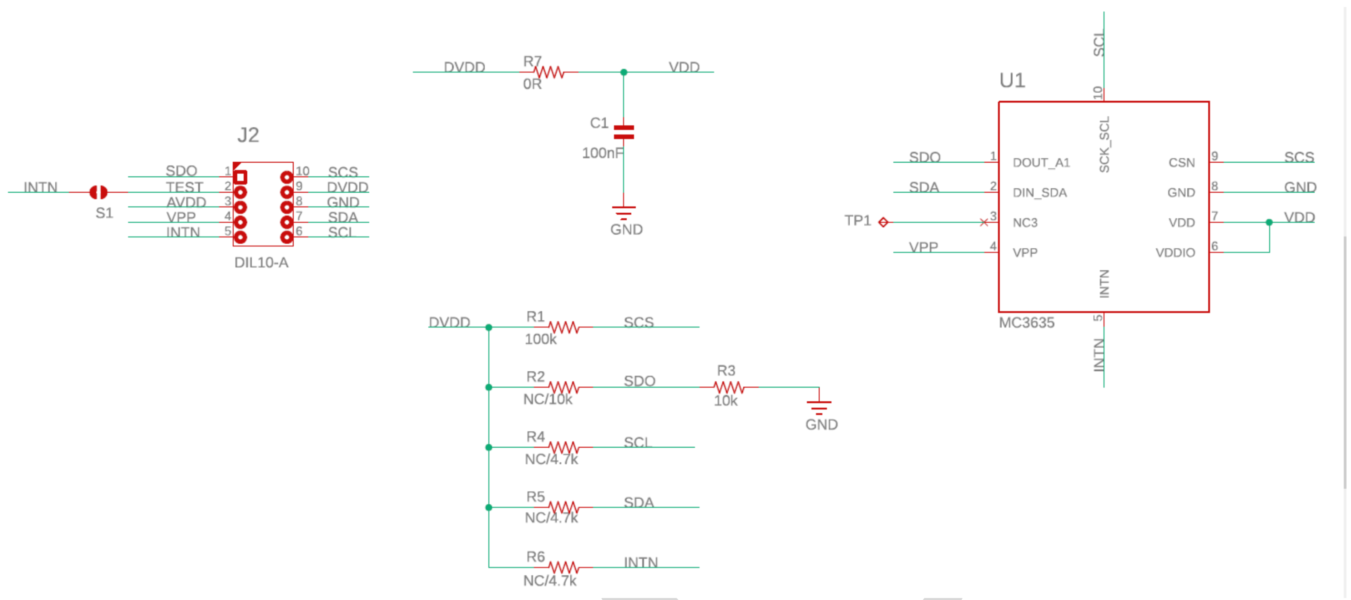
These modes can only be enabled separately.

4.13 READ RAW COUNT DATA

Read the raw count data and SI unit measurement with:

```
MC36XX_acc.readRawAccel();
```

5 SCHEMATICS



Above is a schematic on EV3635A/B. This is the factory preset when receiving the part. For other options, please refer to the following table:

Interface (EV3635A)	**R1**	**R2**	**R3**
SPI (**Factory Default**)	4.7KΩ	4.7KΩ	DNI
SPI or I2C 0x4C	100KΩ	DNI[1]	10KΩ
SPI or I2C 0x6C	100KΩ	10KΩ	DNI

Interface (EV3635B)	**R1**	**R2**	**R3**
SPI or I2C 0x4C (**Factory Default**)	100KΩ	DNI	10KΩ
SPI or I2C 0x6C	100KΩ	10KΩ	DNI

[1] DNI: Do Not Install

The difference between EV3635A and EV3635B is the resistor values for R1 and R2.

EV3635A could be worked on I2C interface by changing R1 value to 100KΩ and R3 value to 10KΩ (for 0x4C I2C address).

EV3635B could be worked on I2C interface (with address 0x4C) without any reworking.

Configure the bus in the MC36XX Driver accordingly when changing from SPI to I2C interface, and vice versa (please refer to Section 3.1).

R4, R5: Install ~4.7KΩ (if no other pull-up installed) as the pull-up for I2C interface.

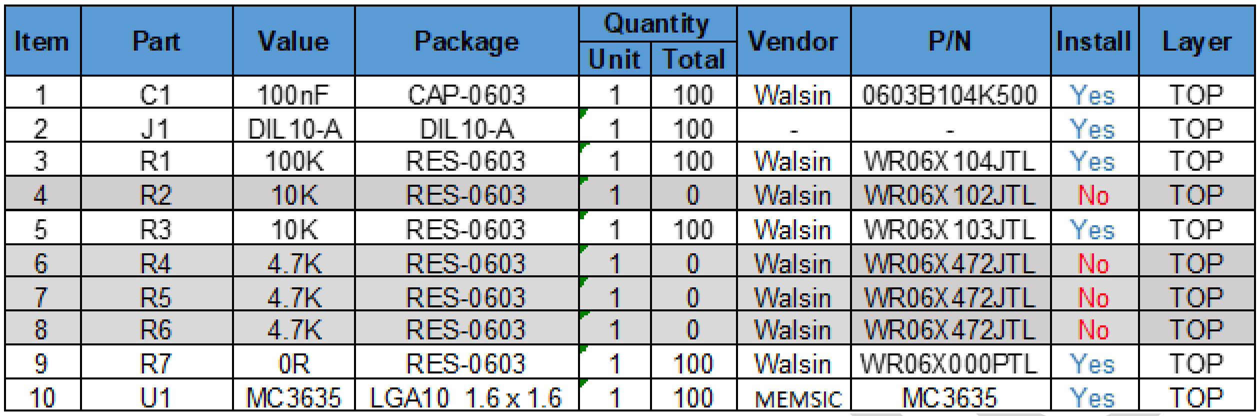
NOTE: It is recommended not to install more than one pull-up per I2C bus.

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

R7: The driving current of sensor can be measured by putting an ammeter in place of R7.

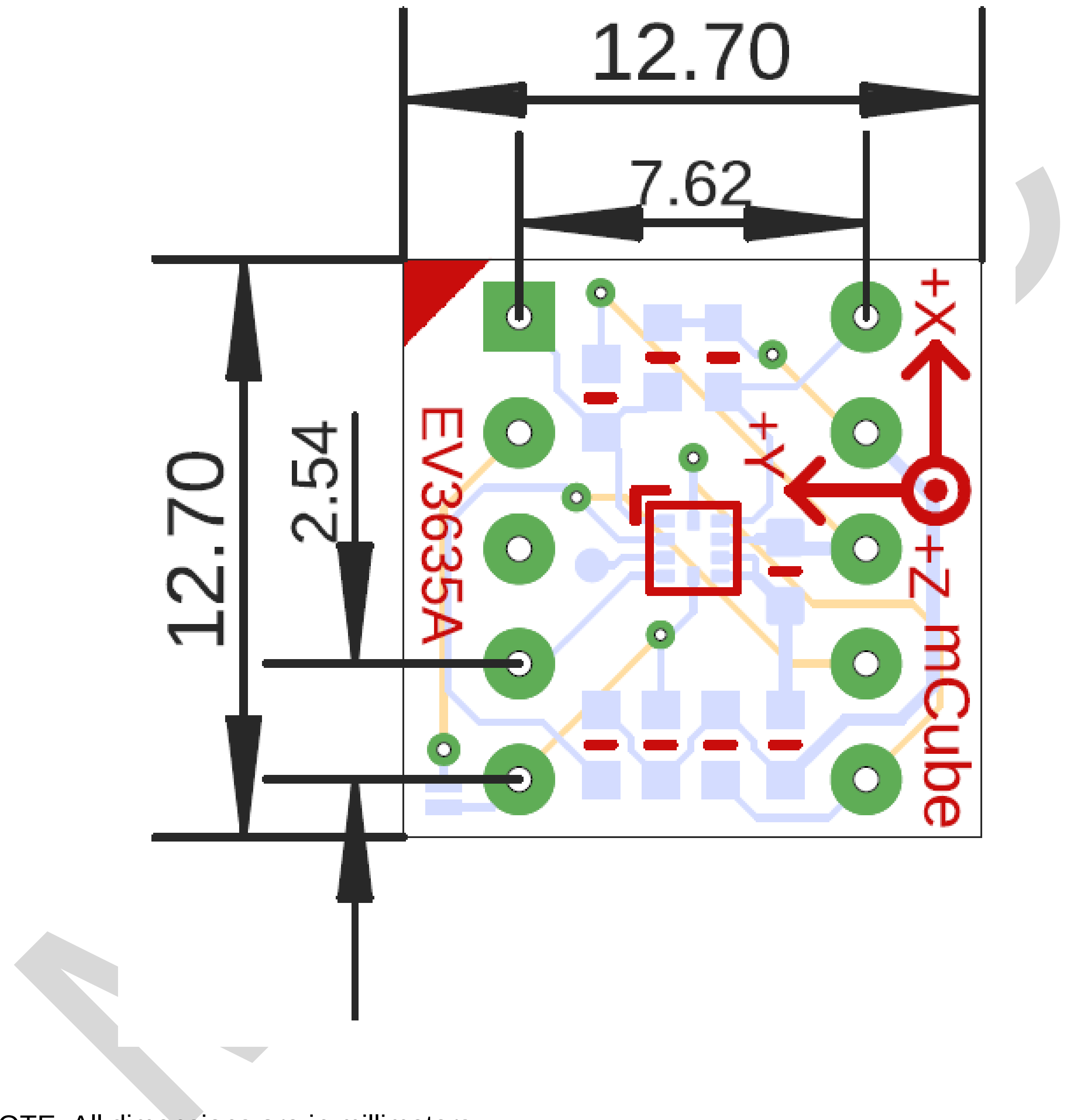
The physical location of the resistor is in the diagram in Section 1.3.

6 BILL OF MATERIALS



Item	Part	Value	Package	Quantity		Vendor	P/N	Install	Layer
				Unit	Total				
1	C1	100nF	CAP-0603	1	100	Walsin	0603B104K500	Yes	TOP
2	J1	DIL10-A	DIL10-A	1	100	-	-	Yes	TOP
3	R1	100K	RES-0603	1	100	Walsin	WR06X104JTL	Yes	TOP
4	R2	10K	RES-0603	1	0	Walsin	WR06X102JTL	No	TOP
5	R3	10K	RES-0603	1	100	Walsin	WR06X103JTL	Yes	TOP
6	R4	4.7K	RES-0603	1	0	Walsin	WR06X472JTL	No	TOP
7	R5	4.7K	RES-0603	1	0	Walsin	WR06X472JTL	No	TOP
8	R6	4.7K	RES-0603	1	0	Walsin	WR06X472JTL	No	TOP
9	R7	0R	RES-0603	1	100	Walsin	WR06X000PTL	Yes	TOP
10	U1	MC3635	LGA10 1.6 x 1.6	1	100	MEMSIC	MC3635	Yes	TOP

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**
2016-08	APS-045-0018v1.0	First release.
2018-11	APS-045-0018v1.1	Added SPI interface. Revised schematics and BOM.
2020-02	APS-045-0018v1.2	Revised schematics and BOM. Added warning in Section 1.2. Changed illustration to Arduino DUE. Revised section 3 & 4 to illustrate demo program.
2020-03	APS-045-0018v1.3	Added description for A/B versions in Section 5.
2020-07	APS-045-0018v1.4	Change to MEMSIC format based on the License Agreement with mCube