

The MLX90609

Appnote by Colin O'Flynn

The Melexis MLX90609 single-axis gyroscope has a built-in ADC, which makes interfacing with a microcontroller especially easy. This example will show you how to interface the MLX90609 to an Atmel AVR microcontroller. The code is written in high-level C, and with only a few changes can be adapted to any microcontroller for which you have a C compiler.

This appnote describes the supplied files `mlx90609.c` and `mlx90609.h`.

Hardware Considerations

The MLX90609's external requirements are specified in the datasheet with regards to powering and decoupling. The MLX90609 requires only 5 volts externally. From this a higher-voltage supply is generated, which is used for both the EEPROM and rate gyro sensor.

If you are planning on exclusively using the SPI interface, you should pull the self-test pin high. It could potentially conflict with the SPI settings.

The AVR uses the SPI lines for programming. When programming the AVR, you want the Melexis device to ignore the programming commands for the AVR device. To do this put a pull-up on CS line of the gyro device. During programming the AVR's pins will be high-impedance, so the pull-up will keep the output line from the gyroscope from conflicting.

Figure 1 shows the example hookup used for the code. The example code will be described in more detail later.

Setting up the MLX90609

Before you can work with the MLX90609, you must initialize it. This simply involves turning the ADC on, which is accomplished with the `MLX90609_init()` function.

Reading the ADC

Reading from the gyro requires two operations. The first is to tell the internal ADC to start a conversion with the `mlx90609_startadc()` function, the second is to read the value with the `mlx90609_readadc()` function which returns the result. The `mlx90609_startadc()` function is passed either `CH_GYRO` to read the gyro, or `CH_TEMP` to read the temperature sensor.

Note that the `mlx90609_readadc()` function waits for the conversion to be complete. Calling the read function immediately after starting a conversion will result in a delay as the converter completes. Instead you should stagger the functions in your program so the time spent performing the conversion is not wasted.

The example code uses an interrupt to accomplish this. The ADC is read at the start of the interrupt, and then told to start another conversion. This means that by the time the interrupt occurs again, the ADC will be ready.

Using the EEPROM

The MLX90609 has 128 bytes of EEPROM. Of that 16 bytes are user-accessible, the remainder are used by Melexis to store calibration information.

Before you attempt to use the EEPROM, there are two very important considerations you need to remember. The first is that the EEPROM and rate gyro share the same internal high-voltage source. However only one of them can be powered up at a time, which means you need to remember to switch the voltage source in your code!

The second is that it is possible to lock the user EEPROM so **no further erases or writes can occur**. This can be very useful to lock serial numbers or similar into a device. The AVR's EEPROM for instance could be used for the same purpose, but it would be possible to accidentally overwrite the information. However with the data stored in the MLX90609, it could not be corrupted in the same manner.

To lock the EEPROM, the three least-significant bits at address 0x70 must be written to '1'. At power-up the bits are checked, and the MEMLOCK flag is active. After this occurs, attempts to erase or write to the EEPROM will fail. Note that the bits are not active until power is cycled – which means you should always verify data written to bank 0, as you still do have a chance to save it.

The example code includes three routines to use the EEPROM. They are the erase, write, and read routines.

The user-accessible EEPROM is split into two banks, each 8 bytes long. You must erase and write an entire bank at once, however read operations operate at the byte level.

<i>Address</i>	<i>Name</i>
0x00 - 0x6F	Calibration Data
0x70 - 0x77	User Bank 0
0x78 - 0x7F	User Bank 1

For example to write bank 1, you would use the following code:

```
/* Write some parameters to EEPROM */
//steal high voltage
mlx90609_mode(ST_EEPROM);

/* Erase bank 1 */
mlx90609_erase(1);

/* Write data to bank 1 */
mlx90609_write(1, data);
```

This assumes you have an array of unsigned characters called “data”, with data[0] through data[7]

containing the data to write to locations 0x78 through to 0x7F. Note that all EEPROM routines return a value, which will be 0 if they completed successfully.

To read the data from location 0x78, you could then use:

```
mlx90609_read(0x78, &databyte);
```

Where databyte is a variable of type unsigned character.

Self-Test

The self-test bits or the external self-test line can be used to force the gyro into a self-test mode. If using the SPI interface, it is recommended to hold the self-test input at a logic high state.

The `mlx90609_mode()` command can be used to force a self-test. The four options you can pass to this routine are “ST_NONE” for normal operation, “ST_POS” to simulate a positive rate, “ST_NEG” to simulate a negative rate, and “ST_EEPROM” which is used for EEPROM writes.

When passing the ST_EEPROM option, it switches the internal charge-pump generator from powering the gyroscope sensor to powering the EEPROM write voltage. This means that the gyroscope will not be oscillating, tripping the continuous self-test function. This can be used in your system to simulate a failure of the gyroscope, to ensure your processor either reads the error values correctly or monitors the error output.

Error Detection

The software routines in this app-note check the status of the error bit when reading from the ADC. If an error is detected the global variable `mlx90609_error` will be non-zero. This will stay non-zero until you clear it.

The routines could be easily modified to flag the error in a more proactive way, as appropriate in your system. As well the ERROR output line can be connected to an interrupt in the processor. This ensures that even if the device is not currently being used it's failure will be noted.

The MLX90609 does not flag a saturated sensor as an error condition. If there is a chance you may exceed 75 deg/s of angular rate, you must consider an output of either extreme (0 for -75 deg/s, or 2048 for 75 deg/s) as an indication the sensor cannot report the rate correctly.

Using the Example Code

The `example.c` code provides an example driver. This also integrates the rate provided by the device, to obtain an angle.

On power-up, the code attempts to discover the null value of the gyro. It is essential that on power-on the device is held completely stationary, or there will be a incorrect offset. The act of throwing a power switch on your circuit board may introduce a small angular rate that will affect the value. You may want to add a settling period of a few seconds immediately after power-on.

This null-value code is fairly accurate. As a test on one particular device at room temperature, it powers

up with a null-value of either 1004 or 1005 using the example code. A histogram of 400000 samples over 1.5 hours is shown in figure 2, which has an average value of 1004.7. This is extremely close to the calculated value using simple integer arithmetic over only 1024 samples. Note this "null" value is a local null of any rotations occurring, not necessarily the sensor zero-rate output.

The example schematic needs to be powered by 5.0 volts, and will need a conversion between the logic level serial port and RS232 for the computer. This is handled well on the STK500 development board, which can provide the power and the level conversion for the serial port.

Once running, the code communicates at 115200 baud rate and will continuously output data. If you wish to use the STK500's on-board oscillator instead of the special crystal frequency used, the Makefile will have to be changed to reflect the new frequency. Note this will also change the timing of the interrupt, which will require a change to the integration constant.

Summary

The Melexis MLX90609 has a very easy interface to almost microcontroller, using only the SPI bus. The driver routines provided here further simplify that by providing a high level interface to the MLX90609.

As well a part library is provided in CadSoft Eagle format. The PCB footprint in this library is suitable for hand-soldering, provided you have a small iron and a steady hand!