# FRDM-KW40Z Demo Software Reference Manual

## NXP Semiconductors

Rev 0.0
Sept 2016

# Contents

## Chapter 1
## Introduction

## Chapter 2
## Buzzer

## Chapter 3
## e-Compass

# Chapter 4
# Accelerometer

Section number Title Page

## Chapter 5
## IR Controller

# Chapter 6
# LED Control

# Chapter 7
# Potentiometer

## Chapter 8
## Temperature Sensor

## Chapter 9
## Input Report

# Chapter 1
# Introduction

FRDM-KW40Z Demo application enables all the available interfaces present on the FRDM-KW40Z board (Figure 1). Interfaces information is read or controlled by using a smartphone application that communicates using Bluetooth(R) Low Energy.

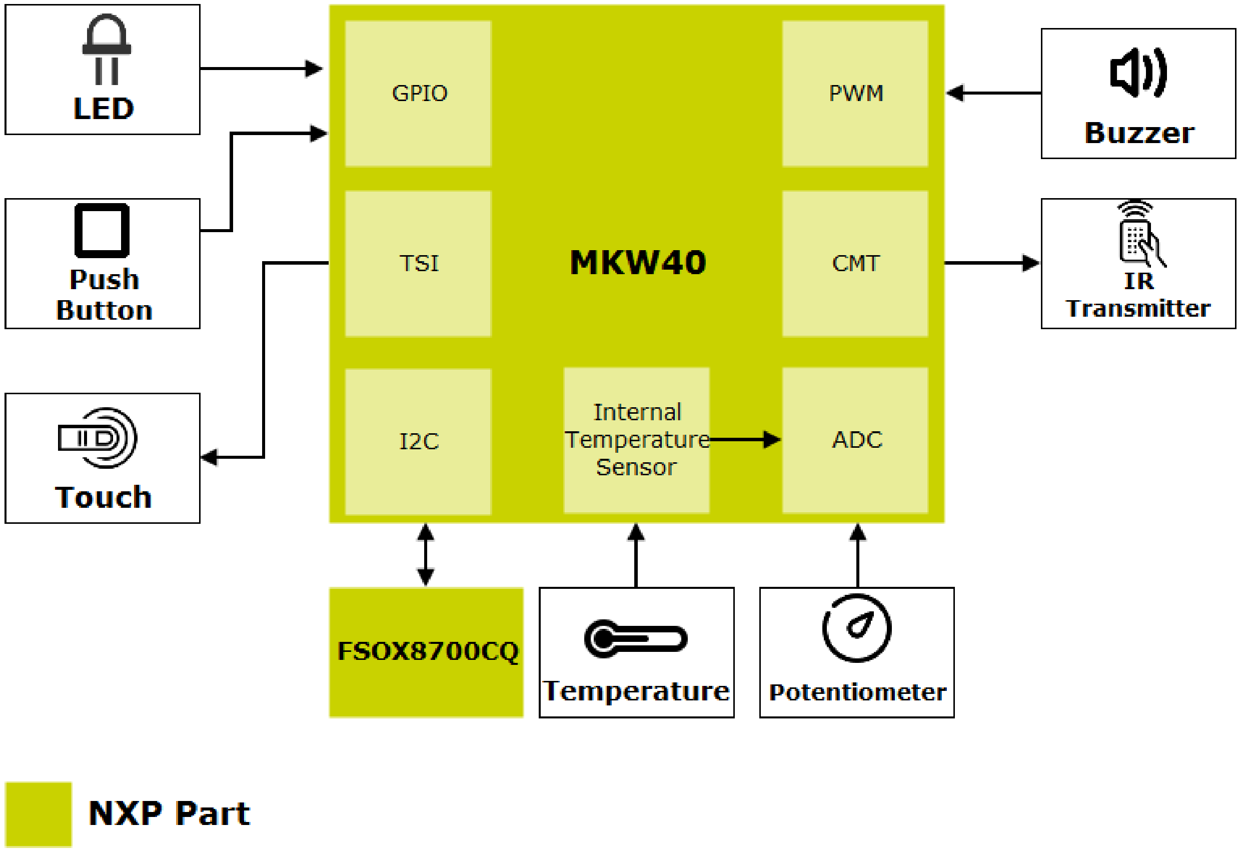

Figure 1: Hardware Block Diagram

FRDM-KW40Z Demo application includes the following interfaces:

- LED Control
- Input Report (GPIO and TSI)
- Buzzer Control
- Internal Temperature Sensor
- Potentiometer (ADC)
- Accelerometer
- e-Compass

- Remote Controller

Application software is divided in three main sections. The KSDK layer that comprises all the low level drivers for the MCU module. The Connectivity Software that includes the full connectivity stack to handle BLE connections. And the application layer that includes all the functions that handle the sensors and actuators.

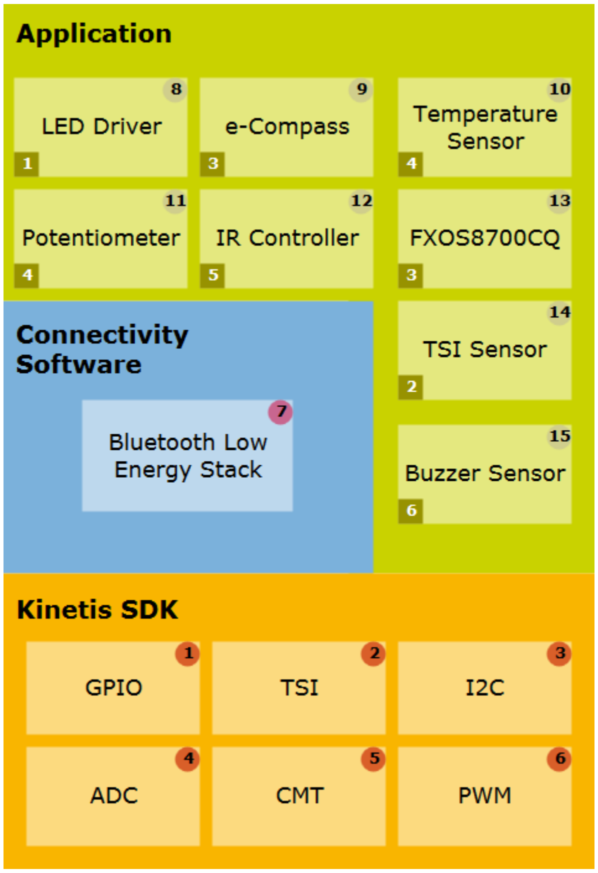

Figure 2: Software Block Diagram

This document describes the functionality of the application layer components. Please refer to the Kinetis SDK and KW40 Connectivity Software documentation for information on the KSDK and Connectivity Software layers respectively.

## 1.1 Revision history

| Version | Date | Updates |
|---|---|---|
| 0 | 08/2016 | Initial release. |

# Chapter 2
# Buzzer

## 2.1 Overview

Buzzer module implements functions to control a DC buzzer using PWM.

### Files

- file buzzer_driver.h

### Macros

- #define BUZZER_DRIVER_PWM_CHANNEL
- #define BUZZER_DRIVER_TPM_MODULE
- #define BUZZER_DRIVER_FREQUENCY_HZ

### Enumerations

- enum buzzer_driver_status_t {
  kBuzzerDriverOk,
  kBuzzerDriverInitError,
  kBuzzerPwmStartError }

### Functions

- buzzer_driver_status_t buzzer_driver_init (void)
- buzzer_driver_status_t buzzer_driver_change_buzzer_status (uint8_t buzzerStatus)

## 2.2 Macro Definition Documentation

### 2.2.1 #define BUZZER_DRIVER_PWM_CHANNEL

TPM PWM channel to use.

### 2.2.2 #define BUZZER_DRIVER_TPM_MODULE

TPM module to use.

### 2.2.3 #define BUZZER_DRIVER_FREQUENCY_HZ

Buzzer signal frequency in Hz.

## 2.3 Enumeration Type Documentation

### 2.3.1 enum buzzer_driver_status_t

Buzzer driver status.

Enumerator

| | |
|---|---|
| ***kBuzzerDriverOk*** | No error. |
| ***kBuzzerDriverInitError*** | Error during initialization. |
| ***kBuzzerPwmStartError*** | Error starting the PWM channel. |

## 2.4 Function Documentation

### 2.4.1 buzzer_driver_status_t buzzer_driver_init ( void )

Initializes the Buzzer driver

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

buzzer_driver_status_t Error status

### 2.4.2 buzzer_driver_status_t buzzer_driver_change_buzzer_status ( uint8_t *buzzerStatus* )

Change the current buzzer status

Parameters

| | | |
|---|---|---|
| `in` | *buzzerStatus* | The buzzer status to set (0: Off, 1: On) |

Returns

buzzer_driver_status_t Error status

# Chapter 3
# e-Compass

## 3.1 Overview

e-Compass module combines accelerometer and magnetometer readings to determine the magnetic north position relative to the board orientation.

This module uses APIs from the FXOS8700CQ module to obtain accelerometer and magnetometer readings. These measurements are passed to the e-Compass Driver module to calculate the current magnetic north position.

### Modules

- e-Compass Driver

### Files

- file e-compass.h

## 3.2 e-Compass Driver

### 3.2.1 Overview

e-Compass Driver implements the functions to calculate the compass heading based on the accelerometer and magnetometer data.

All functions are taken from and explained in the NXP application note AN4248 "Implementing a Tilt-Compansated eCompass using Accelerometer and Magnetometer Sensors"

Warning

> The magnetometer sensor must be calibrated before calculating the compass heading by moving the board in all directions while the magnetometer is enabled.

Version 1.0

This driver version does not include these features:

1. Calibration function

### Functions

- int16_t ecompass_calculate_heading (int16_t iBpx, int16_t iBpy, int16_t iBpz, int16_t iGpx, int16_t iGpy, int16_t iGpz)
- void ecompass_calibrate_hard_iron (int16_t xAxisOffset, int16_t yAxisOffset, int16_t zAxisOffset)

## 3.2.2 Function Documentation

### 3.2.2.1 int16_t ecompass_calculate_heading ( int16_t *iBpx*, int16_t *iBpy*, int16_t *iBpz*, int16_t *iGpx*, int16_t *iGpy*, int16_t *iGpz* )

Calculates the current compass heading (in degrees) based on the current accelerometer and magnetometer data.

Note

> The accelerometer and magnetometer input parameters must be passed using the polarity and directions in the NED coordinate system. If your sensors have a different position on the board, adjust the measurements obtained to fit the NED system.

Parameters

| | | |
|---|---|---|
| `in` | *iBpx* | Magnetometer X-axis reading. |
| `in` | *iBpy* | Magnetometer Y-Axis reading. |
| `in` | *iBpz* | Magnetometer Z-Axis reading. |
| `in` | *iGpx* | Accelerometer X-axis reading. |
| `in` | *iGpy* | Accelerometer Y-axis reading. |
| `in` | *iGpz* | Accelerometer Z-axis reading. |

Returns

> Compass heading position (in degrees, from -180 to 180).

Warning

> The magnetometer must be calibrated before using this function for accurate results.

### 3.2.2.2 void ecompass_calibrate_hard_iron ( int16_t *xAxisOffset*, int16_t *yAxisOffset*, int16_t *zAxisOffset* )

Calibrates the magnetometer by determining the current hard iron offset.

Note

> This function must be periodically called during the program execution to constantly calibrate the magnetometer sensor. It is not neccesary to call this function if the magnetometer sensor has the autocalibration feature.

Parameters

| | | |
|---|---|---|
| `in` | *xAxisOffset* | Magnetometer X-axis reading. |
| `in` | *yAxisOffset* | Magnetometer Y-Axis reading. |
| `in` | *zAxisOffset* | Magnetometer Z-Axis reading. |

Returns

void

# Chapter 4
# Accelerometer

## 4.1 Overview

Accelerometer module initializes and obtains acceleration measurements from the NXP FXOS8700CQ accelerometer plus magnetometer sensor. APIs to initialize, configure, write and read the accelerometer sensor are included in the FXOS8700CQ module.

### Modules

- FXOS8700CQ

### Files

- file FXOS8700CQ.h

## 4.2 FXOS8700CQ

### 4.2.1 Overview

The FXOS8700CQ module implements the functions to initialize, configure, and read the FXOS8700CQ accelerometer + magnetometer sensor from NXP.

The sensor registers are described in the FXOS8700CQ Registers chapter.

The current driver version is 1.0 and it does not include these features:

1. Accelerometer filter
2. Accelerometer pulse detection
3. Accelerometer motion and free-fall
4. Accelerometer acceleration transient
5. Accelerometer orientation detection
6. Accelerometer vector magnitude change
7. FXOS8700 low-power configuration
8. Magnetometer magnetic threshold
9. Magnetometer vector-magnitude
10. Magnetometer magnetic Min/Max detection
11. FIFO configuration
12. SPI transport

## Modules

- FXOS8700CQ Registers

## Files

- file FXOS8700CQ_registers.h

## Data Structures

- struct FXOS8700CQ_config_t
- struct FXOS8700CQ_interrupt_config_t
- struct FXOS8700CQ_accelerometer_config_t
- struct FXOS8700CQ_magnetometer_config_t
- struct FXOS8700CQ_output_data_t

## Macros

- #define FXOS8700CQ_TRANSPORT_I2C
- #define FXOS8700CQ_I2C_ADDRESS
- #define FXOS8700CQ_I2C_INSTANCE
- #define FXOS8700CQ_I2C_BAUDRATE_KBPS
- #define FXOS8700CQ_TIMEOUT_MS

## Typedefs

- typedef void(∗ FXOS8700CQ_callback_function_t) (uint8_t ∗dataSource, uint8_t byteCount)

## Enumerations

- enum FXOS8700CQ_status_t {
  kStatusSuccess,
  kStatusTimeOutError,
  kStatusInitializationError,
  kStatusTransportBusyError,
  kStatusCommunicationsError,
  kStatusMemoryAllocationError }

## Functions

- FXOS8700CQ_status_t FXOS8700CQ_init (FXOS8700CQ_config_t ∗pConfigStruct)
- FXOS8700CQ_status_t FXOS8700CQ_start (void)
- FXOS8700CQ_status_t FXOS8700CQ_stop (void)

- FXOS8700CQ_status_t FXOS8700CQ_communication_test (void)
- FXOS8700CQ_status_t FXOS8700CQ_interrupt_configuration (FXOS8700CQ_interrupt_config_t ∗pConfigurationParameters)
- FXOS8700CQ_status_t FXOS8700CQ_get_interrupt_status (uint8_t ∗interruptStatus)
- FXOS8700CQ_status_t FXOS8700CQ_accelerometer_configuration (FXOS8700CQ_accelerometer_config_t ∗pConfigurationParameters)
- FXOS8700CQ_status_t FXOS8700CQ_get_accelerometer_readings (FXOS8700CQ_output_data_t ∗pAccelerometerData)
- FXOS8700CQ_status_t FXOS8700CQ_magnetometer_configuration (FXOS8700CQ_magnetometer_config_t ∗pConfigurationParameters)
- FXOS8700CQ_status_t FXOS8700CQ_get_magnetometer_readings (FXOS8700CQ_output_data_t ∗pMagnetometerData)
- FXOS8700CQ_status_t FXOS8700CQ_get_hybrid_sensor_readings (FXOS8700CQ_output_data_t ∗pAccelerometerData, FXOS8700CQ_output_data_t ∗pMagnetometerData)
- FXOS8700CQ_status_t FXOS8700CQ_get_registers (uint8_t startRegisterAddress, uint8_t byteCount, FXOS8700CQ_callback_function_t onCompletionCallback)
- FXOS8700CQ_status_t FXOS8700CQ_get_registers_blocking (uint8_t startRegisterAddress, uint8_t byteCount, uint8_t ∗pOutBuffer)
- FXOS8700CQ_status_t FXOS8700CQ_set_register (uint8_t registerAddress, uint8_t ∗registerDataPtr)
- FXOS8700CQ_status_t FXOS8700CQ_set_register_blocking (uint8_t registerAddress, uint8_t ∗registerDataPtr)

## Variables

- unsigned char **FXOS8700CQ_CTRL_REG1_map_t::ctrl_reg1**
- unsigned char **FXOS8700CQ_CTRL_REG1_map_t::active**: 1
- unsigned char **FXOS8700CQ_CTRL_REG1_map_t::f_read**: 1
- unsigned char **FXOS8700CQ_CTRL_REG1_map_t::lnoise**: 1
- unsigned char **FXOS8700CQ_CTRL_REG1_map_t::dr**: 3
- unsigned char **FXOS8700CQ_CTRL_REG1_map_t::aslp_rate**: 2
- struct {

  unsigned char **active**: 1

  unsigned char **f_read**: 1

  unsigned char **lnoise**: 1

  unsigned char **dr**: 3

  unsigned char **aslp_rate**: 2

  } **FXOS8700CQ_CTRL_REG1_map_t::ctrl_reg1_map**

- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::ctrl_reg2**
- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::mods**: 2
- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::slpe**: 1
- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::smods**: 2
- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::reserved**: 1
- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::rst**: 1
- unsigned char **FXOS8700CQ_CTRL_REG2_map_t::st**: 1
- struct {

  unsigned char **mods**: 2

  unsigned char **slpe**: 1

unsigned char **smods**: 2
unsigned char **reserved**: 1
unsigned char **rst**: 1
unsigned char **st**: 1
} **FXOS8700CQ_CTRL_REG2_map_t::ctrl_reg2_map**

- unsigned char **FXOS8700CQ_XYZ_DATA_CFG_map_t::xyz_data_cfg**
- unsigned char **FXOS8700CQ_XYZ_DATA_CFG_map_t::fs**: 2
- unsigned char **FXOS8700CQ_XYZ_DATA_CFG_map_t::reserved0**: 2
- unsigned char **FXOS8700CQ_XYZ_DATA_CFG_map_t::hpf_out**: 1
- unsigned char **FXOS8700CQ_XYZ_DATA_CFG_map_t::reserved1**: 3
- struct {

unsigned char **fs**: 2
unsigned char **reserved0**: 2
unsigned char **hpf_out**: 1
unsigned char **reserved1**: 3
} **FXOS8700CQ_XYZ_DATA_CFG_map_t::xyz_data_cfg_map**

- unsigned char **FXOS8700CQ_M_CTRL_REG1_map_t::m_ctrl_reg1**
- unsigned char **FXOS8700CQ_M_CTRL_REG1_map_t::m_hms**: 2
- unsigned char **FXOS8700CQ_M_CTRL_REG1_map_t::m_os**: 3
- unsigned char **FXOS8700CQ_M_CTRL_REG1_map_t::m_ost**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG1_map_t::m_rst**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG1_map_t::m_acal**: 1
- struct {

unsigned char **m_hms**: 2
unsigned char **m_os**: 3
unsigned char **m_ost**: 1
unsigned char **m_rst**: 1
unsigned char **m_acal**: 1
} **FXOS8700CQ_M_CTRL_REG1_map_t::m_ctrl_reg1_map**

- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::m_ctrl_reg2**
- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::m_rst_cnt**: 2
- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::m_maxmin_rst**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::m_maxmin_dis_ths**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::m_maxmin_dis**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::hyb_autoinc_mode**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG2_map_t::reserved**: 2
- struct {

unsigned char **m_rst_cnt**: 2
unsigned char **m_maxmin_rst**: 1
unsigned char **m_maxmin_dis_ths**: 1
unsigned char **m_maxmin_dis**: 1
unsigned char **hyb_autoinc_mode**: 1
unsigned char **reserved**: 2
} **FXOS8700CQ_M_CTRL_REG2_map_t::m_ctrl_reg2_map**

- unsigned char **FXOS8700CQ_M_CTRL_REG3_map_t::m_ctrl_reg3**

- unsigned char **FXOS8700CQ_M_CTRL_REG3_map_t::reserved**: 3
- unsigned char **FXOS8700CQ_M_CTRL_REG3_map_t::m_ths_xys_update**: 1
- unsigned char **FXOS8700CQ_M_CTRL_REG3_map_t::m_aslp_os**: 3
- unsigned char **FXOS8700CQ_M_CTRL_REG3_map_t::m_raw**: 1
- struct {

  unsigned char **reserved**: 3

  unsigned char **m_ths_xys_update**: 1

  unsigned char **m_aslp_os**: 3

  unsigned char **m_raw**: 1

  } **FXOS8700CQ_M_CTRL_REG3_map_t::m_ctrl_reg3_map**

## 4.2.2 Data Structure Documentation

### 4.2.2.1 struct FXOS8700CQ_config_t

Configuration structure for the FXOS8700CQ initialization.

Data Fields

| | | |
|---|---|---|
| FXOS8700CQ_data_rate_hz_t | outputDataRate | Set the output data rate. |
| FXOS8700CQ_sensor_enable_t | enabledSensors | Set the sensors to enable. |

### 4.2.2.2 struct FXOS8700CQ_interrupt_config_t

Configuration structure for the FXOS8700CQ interruptions initialization.

Data Fields

| | | |
|---|---|---|
| FXOS8700CQ_interrupt_sources_t | interruptSources | Interrupt sources to enable separated by \| operator. |
| FXOS8700CQ_interrupt_pin_map_t | interruptPinMap | Interrupt sources to map to INT2 separated by \| operator (mapped to INT1 when clear) |

### 4.2.2.3 struct FXOS8700CQ_accelerometer_config_t

Configuration structure for the accelerometer sensor initialization.

Data Fields

| | | |
|---|---|---|
| FXOS8700CQ_sensitivity_t | sensitivity | Accelerometer sensitivity. |
| bool_t | fastModeEnabled | Enable fast mode; 8-bit output resolution when enabled. |
| bool_t | lowNoiseEnabled | Enable low noise; does not work with +-8g sensitivity. |
| FXOS8700CQ_oversampling_mods_t | oversamplingMod | Select the ADC oversampling mod. |

### 4.2.2.4 struct FXOS8700CQ_magnetometer_config_t

Configuration structure for the magnetometer sensor initialization.

Data Fields

| | | |
|---|---|---|
| FXOS8700CQ_magnetometer_osr_t | oversamplingRatio | Magnetometer oversampling ratio. |
| FXOS8700CQ_magnetic_sensor_reset_t | autoSensorResetFreq | Magnetic sensor reset (degaussing) frequency. |
| bool_t | autoCalibrationEnabled | Enable sensor autocalibration. |

### 4.2.2.5 struct FXOS8700CQ_output_data_t

Sensor ouput data presentation structure.

Data Fields

| | | |
|---|---|---|
| uint8_t | status | Output data status flags. |
| int16_t | xAxisData | X-Axis data in a signed 16-bit format. |
| int16_t | yAxisData | Y-Axis data in a signed 16-bit format. |
| int16_t | zAxisData | Z-Axis data in a signed 16-bit format. |

## 4.2.3 Macro Definition Documentation

### 4.2.3.1 #define FXOS8700CQ_TRANSPORT_I2C

I2C transport selected.

Select just one transport. If both are selected, I2C takes precedence

### 4.2.3.2 #define FXOS8700CQ_I2C_ADDRESS

I2C address configured for the FXOS8700CQ.

### 4.2.3.3 #define FXOS8700CQ_I2C_INSTANCE

I2C module instance to use.

### 4.2.3.4 #define FXOS8700CQ_I2C_BAUDRATE_KBPS

Baudrate (in kbit/s)

### 4.2.3.5 #define FXOS8700CQ_TIMEOUT_MS

Timeout for the blocking functions.

## 4.2.4 Typedef Documentation

### 4.2.4.1 typedef void(∗ FXOS8700CQ_callback_function_t) (uint8_t ∗dataSource, uint8_t byteCount)

FXOS8700CQ non-blocking functions callback type.

## 4.2.5 Enumeration Type Documentation

### 4.2.5.1 enum FXOS8700CQ_status_t

Status responses for the FXOS8700CQ driver functions.

Enumerator

- ***kStatusSuccess*** No error occured.
- ***kStatusTimeOutError*** Timeout error occured.

***kStatusInitializationError*** Error during initialization.
***kStatusTransportBusyError*** Transport is busy.
***kStatusCommunicationsError*** Error communicating with the FXOS8700CQ.
***kStatusMemoryAllocationError*** Error trying to allocate memory.

## 4.2.6 Function Documentation

### 4.2.6.1 FXOS8700CQ_status_t FXOS8700CQ_init ( FXOS8700CQ_config_t ∗ *pConfigStruct* )

Initializes the FXOS8700CQ module over the selected transport

Parameters

| | | |
|---|---|---|
| `in` | *pConfigStruct* | Module configuration structure (See FXOS8700CQ_config_t ) |

Returns

FXOS8700CQ_status_t Function execution status

### 4.2.6.2 FXOS8700CQ_status_t FXOS8700CQ_start ( void )

Starts the FXOS8700CQ module

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

FXOS8700CQ_status_t Function execution status

### 4.2.6.3 FXOS8700CQ_status_t FXOS8700CQ_stop ( void )

Stops the FXOS8700CQ module

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.4 FXOS8700CQ_status_t FXOS8700CQ_communication_test ( void )

Executes communications tests by reading the WHO AM I register

Parameters

| | | |
|---|---|---|
| in | *None* | |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.5 FXOS8700CQ_status_t FXOS8700CQ_interrupt_configuration ( FXOS8700CQ_interrupt_config_t ∗ *pConfigurationParameters* )

Configures the FXOS8700CQ module to generate interrupt requests using the INT1 and INT2 pins.

Parameters

| | | |
|---|---|---|
| in | *pConfigurationParameters* | Interrupt configuration parameters (See FXOS8700CQ_interrupt_config_t) |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.6 FXOS8700CQ_status_t FXOS8700CQ_get_interrupt_status ( uint8_t ∗ *interruptStatus* )

Gets the current interrupt status register

Parameters

| | | |
|---|---|---|
| out | *interruptStatus* | Pointer to the 8-bit variable where the result is to be stored |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.7 FXOS8700CQ_status_t FXOS8700CQ_accelerometer_configuration ( FXOS8700CQ_accelerometer_config_t ∗ *pConfigurationParameters* )

Configures the accelerometer sensor

Parameters

| | | |
|---|---|---|
| `in` | *pConfigurationParameters* | Configuration structure for the accelerometer (see FXOS8700CQ_accelerometer_config_t) |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.8 FXOS8700CQ_status_t FXOS8700CQ_get_accelerometer_readings ( FXOS8700CQ_output_data_t ∗ *pAccelerometerData* )

Reads the accelerometer data from the FXOS8700CQ

Parameters

| | | |
|---|---|---|
| `out` | *pAccelerometerData* | Pointer to the structure where the results are to be stored (see FXOS8700CQ_output_data_t) |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.9 FXOS8700CQ_status_t FXOS8700CQ_magnetometer_configuration ( FXOS8700CQ_magnetometer_config_t ∗ *pConfigurationParameters* )

Configures the magnetometer sensor

Parameters

| | | |
|---|---|---|
| `in` | *pConfigurationParameters* | Configuration structure for the magnetometer (see FXOS8700CQ_magnetometer_config_t) |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.10 FXOS8700CQ_status_t FXOS8700CQ_get_magnetometer_readings ( FXOS8700CQ_output_data_t ∗ *pMagnetometerData* )

Reads the magnetometer data from the FXOS8700CQ

Parameters

| | | |
|---|---|---|
| out | *pMagnetometerData* | Pointer to the structure where the results are to be stored (see FXOS8700CQ_output_data_t) |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.11 FXOS8700CQ_status_t FXOS8700CQ_get_hybrid_sensor_readings ( FXOS8700CQ_output_data_t ∗ *pAccelerometerData*, FXOS8700CQ_output_data_t ∗ *pMagnetometerData* )

Reads the accelerometer and magnetometer data in a single sequence

Parameters

| | | |
|---|---|---|
| out | *pAccelerometerData* | Pointer to the structure where the results are to be stored (see FXOS8700CQ_output_data_t) |
| out | *pMagnetometerData* | Pointer to the structure where the results are to be stored (see FXOS8700CQ_output_data_t) |

Returns

FXOS8700CQ_status_t Function execution status

Note

When using this function to read the magnetometer data, the magnetometer STATUS byte in the struct is not updated.

#### 4.2.6.12 FXOS8700CQ_status_t FXOS8700CQ_get_registers ( uint8_t *startRegisterAddress*, uint8_t *byteCount*, FXOS8700CQ_callback_function_t *onCompletionCallback* )

Reads a sequence of the FXOS8700CQ registers

Parameters

| | | |
|---|---|---|
| `in` | *startRegister↩ Address* | Address for the first register to read |
| `in` | *byteCount* | Number of registers to read |
| `in` | *on↩ Completion↩ Callback* | Callback function to execute upon read complete. This function must be compatible with the FXOS8700CQ_callback_function_t type. |

Returns

FXOS8700CQ_status_t Function execution status

### 4.2.6.13 FXOS8700CQ_status_t FXOS8700CQ_get_registers_blocking ( uint8_t *startRegisterAddress,* uint8_t *byteCount,* uint8_t ∗ *pOutBuffer* )

Reads a sequence of the FXOS8700CQ registers blocking the execution until completion.

Parameters

| | | |
|---|---|---|
| `in` | *startRegister↩ Address* | Address for the first register to read |
| `in` | *byteCount* | Number of registers to read |
| `out` | *pOutBuffer* | Pointer to the buffer where the data are to be stored |

Returns

FXOS8700CQ_status_t Function execution status

### 4.2.6.14 FXOS8700CQ_status_t FXOS8700CQ_set_register ( uint8_t *registerAddress,* uint8_t ∗ *registerDataPtr* )

Starts a FXOS8700CQ register write.

Parameters

| | | |
|---|---|---|
| `in` | *registerAddress* | Address for the register to write. |
| `in` | *registerDataPtr* | Pointer to the variable containing the data to write. |

Returns

FXOS8700CQ_status_t Function execution status

#### 4.2.6.15 FXOS8700CQ_status_t FXOS8700CQ_set_register_blocking ( uint8_t *registerAddress,* uint8_t ∗ *registerDataPtr* )

Starts a FXOS8700CQ register write and waits until it is complete.

Parameters

| | | |
|---|---|---|
| `in` | *registerAddress* | Address for the register to write. |
| `in` | *registerDataPtr* | Pointer to the variable containing the data to write. |

Returns

FXOS8700CQ_status_t Function execution status

### 4.2.7 Variable Documentation

#### 4.2.7.1 unsigned char FXOS8700CQ_CTRL_REG1_map_t::ctrl_reg1

#### 4.2.7.2 unsigned { ... } ::active

#### 4.2.7.3 unsigned { ... } ::f_read

#### 4.2.7.4 unsigned { ... } ::lnoise

#### 4.2.7.5 unsigned { ... } ::dr

#### 4.2.7.6 unsigned { ... } ::aslp_rate

#### 4.2.7.7 struct { ... } FXOS8700CQ_CTRL_REG1_map_t::ctrl_reg1_map

#### 4.2.7.8 unsigned char FXOS8700CQ_CTRL_REG2_map_t::ctrl_reg2

#### 4.2.7.9 unsigned { ... } ::mods

#### 4.2.7.10 unsigned { ... } ::slpe

#### 4.2.7.11 unsigned { ... } ::smods

#### 4.2.7.12 unsigned { ... } ::reserved

#### 4.2.7.13 unsigned { ... } ::rst

#### 4.2.7.14 unsigned { ... } ::st

#### 4.2.7.15 struct { ... } FXOS8700CQ_CTRL_REG2_map_t::ctrl_reg2_map

#### 4.2.7.16 unsigned char FXOS8700CQ_XYZ_DATA_CFG_map_t::xyz_data_cfg

#### 4.2.7.17 unsigned { ... } ::fs

#### 4.2.7.18 unsigned { ... } ::reserved0

#### 4.2.7.19 unsigned { ... } ::hpf_out

#### 4.2.7.20 unsigned { ... } ::reserved1

#### 4.2.7.21 struct { ... } FXOS8700CQ_XYZ_DATA_CFG_map_t::xyz_data_cfg_map

#### 4.2.7.22 unsigned char FXOS8700CQ_M_CTRL_REG1_map_t::m_ctrl_reg1

#### 4.2.7.23 unsigned { ... } ::m_hms

#### 4.2.7.24 unsigned { ... } ::m_os

## FXOS8700CQ

See the FXOS8700CQ datasheet for more information.

### Data Structures

- union FXOS8700CQ_CTRL_REG1_map_t
- struct FXOS8700CQ_CTRL_REG1_map_t.ctrl_reg1_map
- union FXOS8700CQ_CTRL_REG2_map_t
- struct FXOS8700CQ_CTRL_REG2_map_t.ctrl_reg2_map
- union FXOS8700CQ_XYZ_DATA_CFG_map_t
- struct FXOS8700CQ_XYZ_DATA_CFG_map_t.xyz_data_cfg_map
- union FXOS8700CQ_M_CTRL_REG1_map_t
- struct FXOS8700CQ_M_CTRL_REG1_map_t.m_ctrl_reg1_map
- union FXOS8700CQ_M_CTRL_REG2_map_t
- struct FXOS8700CQ_M_CTRL_REG2_map_t.m_ctrl_reg2_map
- union FXOS8700CQ_M_CTRL_REG3_map_t
- struct FXOS8700CQ_M_CTRL_REG3_map_t.m_ctrl_reg3_map

### Macros

- #define **FXOS8700CQ_STATUS**
- #define **FXOS8700CQ_OUT_X_MSB**
- #define **FXOS8700CQ_OUT_X_LSB**
- #define **FXOS8700CQ_OUT_Y_MSB**
- #define **FXOS8700CQ_OUT_Y_LSB**
- #define **FXOS8700CQ_OUT_Z_MSB**
- #define **FXOS8700CQ_OUT_Z_LSB**
- #define **FXOS8700CQ_F_SETUP**
- #define **FXOS8700CQ_TRIG_CFG**
- #define **FXOS8700CQ_SYSMOD**
- #define **FXOS8700CQ_INT_SOURCE**
- #define **FXOS8700CQ_WHO_AM_I**
- #define **FXOS8700CQ_XYZ_DATA_CFG**
- #define **FXOS8700CQ_HP_FILTER_CUTOFF**
- #define **FXOS8700CQ_PL_STATUS**
- #define **FXOS8700CQ_PL_CFG**
- #define **FXOS8700CQ_PL_COUNT**
- #define **FXOS8700CQ_PL_BF_ZCOMP**
- #define **FXOS8700CQ_PL_THS_REG**
- #define **FXOS8700CQ_A_FFMT_CFG**
- #define **FXOS8700CQ_A_FFMT_SRC**
- #define **FXOS8700CQ_A_FFMT_THS**
- #define **FXOS8700CQ_A_FFMT_COUNT**
- #define **FXOS8700CQ_TRANSIENT_CFG**
- #define **FXOS8700CQ_TRANSIENT_SRC**
- #define **FXOS8700CQ_TRANSIENT_THS**
- #define **FXOS8700CQ_TRANSIENT_COUNT**
- #define **FXOS8700CQ_PULSE_CFG**
- #define **FXOS8700CQ_PULSE_SRC**
- #define **FXOS8700CQ_PULSE_THSX**
- #define **FXOS8700CQ_PULSE_THSY**
- #define **FXOS8700CQ_PULSE_THSZ**
- #define **FXOS8700CQ_PULSE_TMLT**
- #define **FXOS8700CQ_PULSE_LTCY**
- #define **FXOS8700CQ_PULSE_WIND**

- #define **FXOS8700CQ_ASLP_COUNT**
- #define **FXOS8700CQ_CTRL_REG1**
- #define **FXOS8700CQ_CTRL_REG2**
- #define **FXOS8700CQ_CTRL_REG3**
- #define **FXOS8700CQ_CTRL_REG4**
- #define **FXOS8700CQ_CTRL_REG5**
- #define **FXOS8700CQ_OFF_X**
- #define **FXOS8700CQ_OFF_Y**
- #define **FXOS8700CQ_OFF_Z**
- #define **FXOS8700CQ_M_DR_STATUS**
- #define **FXOS8700CQ_M_OUT_X_MSB**
- #define **FXOS8700CQ_M_OUT_X_LSB**
- #define **FXOS8700CQ_M_OUT_Y_MSB**
- #define **FXOS8700CQ_M_OUT_Y_LSB**
- #define **FXOS8700CQ_M_OUT_Z_MSB**
- #define **FXOS8700CQ_M_OUT_Z_LSB**
- #define **FXOS8700CQ_CMP_X_MSB**
- #define **FXOS8700CQ_CMP_X_LSB**
- #define **FXOS8700CQ_CMP_Y_MSB**
- #define **FXOS8700CQ_CMP_Y_LSB**
- #define **FXOS8700CQ_CMP_Z_MSB**
- #define **FXOS8700CQ_CMP_Z_LSB**
- #define **FXOS8700CQ_M_OFF_X_MSB**
- #define **FXOS8700CQ_M_OFF_X_LSB**
- #define **FXOS8700CQ_M_OFF_Y_MSB**
- #define **FXOS8700CQ_M_OFF_Y_LSB**
- #define **FXOS8700CQ_M_OFF_Z_MSB**
- #define **FXOS8700CQ_M_OFF_Z_LSB**
- #define **FXOS8700CQ_MAX_X_MSB**
- #define **FXOS8700CQ_MAX_X_LSB**
- #define **FXOS8700CQ_MAX_Y_MSB**
- #define **FXOS8700CQ_MAX_Y_LSB**
- #define **FXOS8700CQ_MAX_Z_MSB**
- #define **FXOS8700CQ_MAX_Z_LSB**
- #define **FXOS8700CQ_MIN_X_MSB**
- #define **FXOS8700CQ_MIN_X_LSB**
- #define **FXOS8700CQ_MIN_Y_MSB**
- #define **FXOS8700CQ_MIN_Y_LSB**
- #define **FXOS8700CQ_MIN_Z_MSB**
- #define **FXOS8700CQ_MIN_Z_LSB**
- #define **FXOS8700CQ_TEMP**
- #define **FXOS8700CQ_M_THS_CFG**
- #define **FXOS8700CQ_M_THS_SRC**
- #define **FXOS8700CQ_M_THS_X_MSB**
- #define **FXOS8700CQ_M_THS_X_LSB**
- #define **FXOS8700CQ_M_THS_Y_MSB**
- #define **FXOS8700CQ_M_THS_Y_LSB**
- #define **FXOS8700CQ_M_THS_Z_MSB**
- #define **FXOS8700CQ_M_THS_Z_LSB**
- #define **FXOS8700CQ_M_THS_COUNT**
- #define **FXOS8700CQ_M_CTRL_REG1**
- #define **FXOS8700CQ_M_CTRL_REG2**
- #define **FXOS8700CQ_M_CTRL_REG3**
- #define **FXOS8700CQ_M_INT_SRC**
- #define **FXOS8700CQ_A_VECM_CFG**
- #define **FXOS8700CQ_A_VECM_THS_MSB**

- #define **FXOS8700CQ_A_VECM_THS_LSB**
- #define **FXOS8700CQ_A_VECM_CNT**
- #define **FXOS8700CQ_A_VECM_INITX_MSB**
- #define **FXOS8700CQ_A_VECM_INITX_LSB**
- #define **FXOS8700CQ_A_VECM_INITY_MSB**
- #define **FXOS8700CQ_A_VECM_INITY_LSB**
- #define **FXOS8700CQ_A_VECM_INITZ_MSB**
- #define **FXOS8700CQ_A_VECM_INITZ_LSB**
- #define **FXOS8700CQ_M_VECM_CFG**
- #define **FXOS8700CQ_M_THS_MSB**
- #define **FXOS8700CQ_M_THS_LSB**
- #define **FXOS8700CQ_M_VECM_CNT**
- #define **FXOS8700CQ_M_VECM_INITX_MSB**
- #define **FXOS8700CQ_M_VECM_INITX_LSB**
- #define **FXOS8700CQ_M_VECM_INITY_MSB**
- #define **FXOS8700CQ_M_VECM_INITY_LSB**
- #define **FXOS8700CQ_M_VECM_INITZ_MSB**
- #define **FXOS8700CQ_M_VECM_INITZ_LSB**
- #define **FXOS8700CQ_A_FFMT_THS_X_MSB**
- #define **FXOS8700CQ_A_FFMT_THS_X_LSB**
- #define **FXOS8700CQ_A_FFMT_THS_Y_MSB**
- #define **FXOS8700CQ_A_FFMT_THS_Y_LSB**
- #define **FXOS8700CQ_A_FFMT_THS_Z_MSB**
- #define **FXOS8700CQ_A_FFMT_THS_Z_LSB**

### Enumerations

- enum FXOS8700CQ_data_rate_hz_t {
  **kDataRate0**,
  **kDataRate1**,
  **kDataRate2**,
  **kDataRate3**,
  **kDataRate4**,
  **kDataRate5**,
  **kDataRate6**,
  **kDataRate7** }
- enum FXOS8700CQ_sensor_enable_t {
  kAccelerometerOnly,
  kMagnetometerOnly,
  kHybridMode }
- enum FXOS8700CQ_interrupt_sources_t {
  kInterruptSleep,
  kInterruptFIFO,
  kInterruptTransient,
  kInterruptOrientation,
  kInterruptPulse,
  kInterruptFFMT,
  kInterruptVECM,
  kInterruptDataReady,
  kInterruptDisabled }

- enum FXOS8700CQ_interrupt_pin_map_t {
  kInterruptSleepPinIsInt1,
  kInterruptFIFOPinIsInt1,
  kInterruptTransientPinIsInt1,
  kInterruptOrientationPinIsInt1,
  kInterruptPulsePinIsInt1,
  kInterruptFFMTPinIsInt1,
  kInterruptVECMPinIsInt1,
  kInterruptDataReadyPinIsInt1,
  kInterruptAllPinsAreInt2 }
- enum FXOS8700CQ_sensitivity_t {
  kSensitivity2g,
  kSensitivity4g,
  kSensitivity8g }
- enum FXOS8700CQ_oversampling_mods_t {
  kNormal,
  kLowNoiseLowPower,
  kHighResolution,
  kLowPower }
- enum FXOS8700CQ_magnetometer_osr_t {
  kMagnetometerOSR0,
  kMagnetometerOSR1,
  kMagnetometerOSR2,
  kMagnetometerOSR3,
  kMagnetometerOSR4,
  kMagnetometerOSR5,
  kMagnetometerOSR6,
  kMagnetometerOSR7 }
- enum FXOS8700CQ_magnetic_sensor_reset_t {
  kMagneticAutoReset1,
  kMagneticAutoReset16,
  kMagneticAutoReset512,
  kMagneticAutoResetDisabled }

### 4.2.8.2 Data Structure Documentation

#### 4.2.8.2.1 union FXOS8700CQ_CTRL_REG1_map_t

Data Fields

| | | |
|---|---|---|
| unsigned char | ctrl_reg1 | |

| | | |
|---|---|---|
| struct FXOS8700CQ_CTRL_REG1_map_t | ctrl_reg1_map | |

### 4.2.8.2.2 struct FXOS8700CQ_CTRL_REG1_map_t.ctrl_reg1_map

Data Fields

| | | |
|---|---|---|
| unsigned char | active: 1 | |
| unsigned char | f_read: 1 | |
| unsigned char | lnoise: 1 | |
| unsigned char | dr: 3 | |
| unsigned char | aslp_rate: 2 | |

### 4.2.8.2.3 union FXOS8700CQ_CTRL_REG2_map_t

Data Fields

| | | |
|---|---|---|
| unsigned char | ctrl_reg2 | |
| struct FXOS8700CQ_CTRL_REG2_map_t | ctrl_reg2_map | |

### 4.2.8.2.4 struct FXOS8700CQ_CTRL_REG2_map_t.ctrl_reg2_map

Data Fields

| | | |
|---|---|---|
| unsigned char | mods: 2 | |
| unsigned char | slpe: 1 | |
| unsigned char | smods: 2 | |
| unsigned char | reserved: 1 | |
| unsigned char | rst: 1 | |
| unsigned char | st: 1 | |

### 4.2.8.2.5 union FXOS8700CQ_XYZ_DATA_CFG_map_t

Data Fields

| | | |
|---|---|---|
| unsigned char | xyz_data_cfg | |
| struct FXO↩ S8700CQ_X↩ YZ_DATA_↩ CFG_map_t | xyz_data_cfg↩ _map | |

#### 4.2.8.2.6 struct FXOS8700CQ_XYZ_DATA_CFG_map_t.xyz_data_cfg_map

Data Fields

| | | |
|---|---|---|
| unsigned char | fs: 2 | |
| unsigned char | reserved0: 2 | |
| unsigned char | hpf_out: 1 | |
| unsigned char | reserved1: 3 | |

#### 4.2.8.2.7 union FXOS8700CQ_M_CTRL_REG1_map_t

Data Fields

| | | |
|---|---|---|
| unsigned char | m_ctrl_reg1 | |
| struct FXO↩ S8700CQ_M↩ _CTRL_RE↩ G1_map_t | m_ctrl_reg1_↩ map | |

#### 4.2.8.2.8 struct FXOS8700CQ_M_CTRL_REG1_map_t.m_ctrl_reg1_map

Data Fields

| | | |
|---|---|---|
| unsigned char | m_hms: 2 | |
| unsigned char | m_os: 3 | |
| unsigned char | m_ost: 1 | |
| unsigned char | m_rst: 1 | |
| unsigned char | m_acal: 1 | |

#### 4.2.8.2.9 union FXOS8700CQ_M_CTRL_REG2_map_t

Data Fields

| | | |
|---|---|---|
| unsigned char | m_ctrl_reg2 | |
| struct FXOS8700CQ_M_CTRL_REG2_map_t | m_ctrl_reg2_map | |

#### 4.2.8.2.10 struct FXOS8700CQ_M_CTRL_REG2_map_t.m_ctrl_reg2_map

Data Fields

| | | |
|---|---|---|
| unsigned char | m_rst_cnt: 2 | |
| unsigned char | m_maxmin_rst: 1 | |
| unsigned char | m_maxmin_dis_ths: 1 | |
| unsigned char | m_maxmin_dis: 1 | |
| unsigned char | hyb_autoinc_mode: 1 | |
| unsigned char | reserved: 2 | |

#### 4.2.8.2.11 union FXOS8700CQ_M_CTRL_REG3_map_t

Data Fields

| | | |
|---|---|---|
| unsigned char | m_ctrl_reg3 | |
| struct FXOS8700CQ_M_CTRL_REG3_map_t | m_ctrl_reg3_map | |

#### 4.2.8.2.12 struct FXOS8700CQ_M_CTRL_REG3_map_t.m_ctrl_reg3_map

Data Fields

| | | |
|---|---|---|
| unsigned char | reserved: 3 | |
| unsigned char | m_ths_xys_update: 1 | |

| | | |
|---|---|---|
| unsigned char | m_aslp_os: 3 | |
| unsigned char | m_raw: 1 | |

### 4.2.8.3 Enumeration Type Documentation

#### 4.2.8.3.1 enum FXOS8700CQ_data_rate_hz_t

Defines output data rate.

- Single mode: Accelerometer or Magnetometer only
- Hybrid mode: Accelerometer and Magnetometer

#### 4.2.8.3.2 enum FXOS8700CQ_sensor_enable_t

Defines enabled sensors.

Enumerator

***kAccelerometerOnly*** Only accelerometer is enabled.
***kMagnetometerOnly*** Only magnetometer is enabled.
***kHybridMode*** Accelerometer and magnetometer are enabled.

#### 4.2.8.3.3 enum FXOS8700CQ_interrupt_sources_t

Available interrupt sources.

Enumerator

***kInterruptSleep*** Sleep interrupt.
***kInterruptFIFO*** FIFO interrupt.
***kInterruptTransient*** Transient interrupt.
***kInterruptOrientation*** Orientation interrupt.
***kInterruptPulse*** Pulse interrupt.
***kInterruptFFMT*** Freefall/Motion interrupt.
***kInterruptVECM*** Acceleration vector-magnitude interrupt.
***kInterruptDataReady*** Data ready interrupt.
***kInterruptDisabled*** All interrupt sources are disabled.

#### 4.2.8.3.4 enum FXOS8700CQ_interrupt_pin_map_t

Interrupt pin mapping.

- 0: Interrupt mapped to INT1 pin

- 1: Interrupt mapped to INT2 pin

Enumerator

***kInterruptSleepPinIsInt1*** Sleep interrupt is mapped to INT1.
***kInterruptFIFOPinIsInt1*** FIFO interrupt is mapped to INT1.
***kInterruptTransientPinIsInt1*** Transient interrupt is mapped to INT1.
***kInterruptOrientationPinIsInt1*** Orientation interrupt is mapped to INT1.
***kInterruptPulsePinIsInt1*** Pulse interrupt is mapped to INT1.
***kInterruptFFMTPinIsInt1*** Freefall/Motion interrupt is mapped to INT1.
***kInterruptVECMPinIsInt1*** Acceleration vector-magnitude interrupt is mapped to INT1.
***kInterruptDataReadyPinIsInt1*** Data ready interrupt is mapped to INT1.
***kInterruptAllPinsAreInt2*** All interrupt sources are routed to INT2.

#### 4.2.8.3.5 enum FXOS8700CQ_sensitivity_t

Sensitivity options.

Enumerator

***kSensitivity2g*** +-2g Sensitivity
***kSensitivity4g*** +-4g Sensitivity
***kSensitivity8g*** +-8g Sensitivity

#### 4.2.8.3.6 enum FXOS8700CQ_oversampling_mods_t

Accelerometer Oversampling Mods.

Enumerator

***kNormal*** Normal mode.
***kLowNoiseLowPower*** Low-Noise + Low-Power mode.
***kHighResolution*** High-resolution mode.
***kLowPower*** Low-Power Mode.

#### 4.2.8.3.7 enum FXOS8700CQ_magnetometer_osr_t

Magnetometer Oversampling Ratio.

Enumerator

***kMagnetometerOSR0*** Magnetometer oversampling ratio = 0.
***kMagnetometerOSR1*** Magnetometer oversampling ratio = 1.

| | |
|---|---|
| ***kMagnetometerOSR2*** | Magnetometer oversampling ratio = 2. |
| ***kMagnetometerOSR3*** | Magnetometer oversampling ratio = 3. |
| ***kMagnetometerOSR4*** | Magnetometer oversampling ratio = 4. |
| ***kMagnetometerOSR5*** | Magnetometer oversampling ratio = 5. |
| ***kMagnetometerOSR6*** | Magnetometer oversampling ratio = 6. |
| ***kMagnetometerOSR7*** | Magnetometer oversampling ratio = 7. |

#### 4.2.8.3.8 enum FXOS8700CQ_magnetic_sensor_reset_t

Magnetic Sensor Reset Frequency.

Enumerator

| | |
|---|---|
| ***kMagneticAutoReset1*** | Automatic magnetic sensor reset every 1 ODR cycle. |
| ***kMagneticAutoReset16*** | Automatic magnetic sensor reset every 16 ODR cycles. |
| ***kMagneticAutoReset512*** | Automatic magnetic sensor reset every 512 ODR cycles. |
| ***kMagneticAutoResetDisabled*** | Automatic magnetic sensor reset disabled (must be manually reset) |

## 4.3 IR Controller Parameters

### 4.3.1 Overview

This module includes the definition of time parameters and command codes for different TVs.

### Data Structures

- struct controller_parameter_list_t

### Variables

- controller_parameter_list_t controllerParameterList [3]

### 4.3.2 Data Structure Documentation

#### 4.3.2.1 struct controller_parameter_list_t

TV Controller parameter structure.

Data Fields

| | | |
|---|---|---|
| ir_controller_↩ parameters_t | controller↩ Timing↩ Parameters | Timing parameters (see ir_controller_parameters_t). |
| uint32_t | controller↩ Commands[6] | Controller command codes for these functions:<br>1. On/Off<br>2. Mute<br>3. Volume +<br>4. Volume -<br>5. Channel +<br>6. Channel -<br>Note<br>The command codes must be left-aligned |
| uint8_t | bitsPer↩ Command | Size of the command in bytes (see controller specification) |
| uint8_t | command↩ RepeatTimes | Number of times a command must be repeated to be valid (see controller specification) |

## 4.3.3 Variable Documentation

### 4.3.3.1 controller_parameter_list_t controllerParameterList[3]

List of controller parameters of type controller_parameter_list_t supported in this driver.

# Chapter 5
# IR Controller

## 5.1 Overview

IR controller module implements functions to command different TVs using an infrared transmitter. IR Controller Driver uses CMT to modulate the IR signal. Timing parameters and controller command codes are defined using the structures on IR Controller Parameters.

### Modules

- IR Controller Parameters
- IR Controller Driver

### Files

- file controller_parameter_list.h
- file ir_controller.h

## 5.2 IR Controller Driver

### 5.2.1 Overview

This module implements the APIs to initialize the IR controller driver and send IR commands.

### Data Structures

- struct ir_controller_parameters_t

### Enumerations

- enum ir_controller_status_t {
  irControllerStatusOk,
  irControllerStatusInitError,
  irControllerStatusModuleBusy }

### Functions

- ir_controller_status_t ir_controller_init (void)
- ir_controller_status_t ir_controller_send_command (ir_controller_parameters_t ∗controllerParamPtr, uint32_t ∗ptrCommandToSend, uint8_t numberOfBits, uint8_t commandRepeatTimes)
- void ir_controller_task (void)

- ir_controller_status_t ir_controller_get_current_status (void)

## 5.2.2 Data Structure Documentation

### 5.2.2.1 struct ir_controller_parameters_t

Modulation parameters for each remote controller.

Data Fields

| | | |
|---|---|---|
| uint16_t | headerOnTime | On time for the header in us. |
| uint16_t | headerOffTime | Off time for the header in us. |
| uint16_t | logic0OnTime | On time for the logic 0 signal in us. |
| uint16_t | logic0OffTime | Off time for the logic 0 signal in us. |
| uint16_t | logic1OnTime | On time for the logic 1 signal in us. |
| uint16_t | logic1OffTime | Off time for the logic 1 signal in us. |
| uint16_t | stopTime | On time for the stop signaling in us. |
| uint16_t | gapTime | GAP time (the minimum time between one command and the next command) in us. |
| uint8_t | carrierOnTime | Signal carrier On time in the multiples of 0.125 us. Must be 0 if the carrier is not needed. |
| uint8_t | carrierOffTime | Signal carrier Off time in the multiples of 0.125 us. Must be 0 if the carrier is not needed |

## 5.2.3 Enumeration Type Documentation

### 5.2.3.1 enum ir_controller_status_t

IR Controller Status.

Enumerator

***irControllerStatusOk*** No error occurred.
***irControllerStatusInitError*** An error occurred during the initialization phase.
***irControllerStatusModuleBusy*** The IR Controller is busy and can't start a new command at this time.

## 5.2.4 Function Documentation

### 5.2.4.1 ir_controller_status_t ir_controller_init ( void )

Initializes the IR Controller required modules.

Note

The pin mux must be initialized separately.

Parameters

| | | |
|---|---|---|
| in | *None* | |

Returns

ir_controller_status_t IR Controller module status

### 5.2.4.2 ir_controller_status_t ir_controller_send_command ( ir_controller_parameters_t ∗ *controllerParamPtr,* uint32_t ∗ *ptrCommandToSend,* uint8_t *numberOfBits,* uint8_t *commandRepeatTimes* )

Sends a new IR controller command.

Parameters

| | | |
|---|---|---|
| in | *controllerParamPtr* | Controller modulation parameters (See ir_controller_parameters_t) |
| in | *ptrCommandToSend* | Pointer to the 32-bit array containing the command to send. |

Warning

Data bits must be LEFT-ALIGNED, Example: If the following 11 bits will be sent (0b0...10110001001 = 0x00000589) the array must be [0xB1, 0x20, 0x00, 0x00 = 0b1011000100100000...)

Parameters

| | | |
|---|---|---|
| in | *numberOfBits* | Number of BITS to modulate. |
| in | *commandRepeatTimes* | Number of times the command must be repeatedly transmitted. |

Returns

ir_controller_status_t IR Controller module status

### 5.2.4.3 void ir_controller_task ( void )

This task must be executed periodically for a proper module functionality.

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

None

### 5.2.4.4 ir_controller_status_t ir_controller_get_current_status ( void )

Gets the current status of the IR Controller module

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

ir_controller_status_t. irControllerStatusOk if free, irControllerStatusModuleBusy if busy

# Chapter 6
# LED Control

## 6.1 Overview

This module provides the functions to control the FRDM-KW40Z board LED functionality.

The available actions are:

- Initialize the LED module
- Turn the LEDs On/Off
- Toggle the LEDs

### Files

- file led_control.h

### Macros

- #define LED_CONTROL_NUMBER_OF_LEDS

### Enumerations

- enum led_control_status_t {
  kStatusOk,
  kStatusLedNumberOutOfRange,
  kStatusInvalidCommand,
  kStatusUnexpectedError }
- enum led_control_command_t {
  kLedControlOff,
  kLedControlOn,
  kLedControlToggle,
  kLedControlInvalidCommand }

### Functions

- void led_control_init (void)
- led_control_status_t led_control_update_led (uint8_t ledNumber, led_control_command_t command)

## 6.2 Macro Definition Documentation

### 6.2.1 #define LED_CONTROL_NUMBER_OF_LEDS

Number of LEDs on the board.

## 6.3 Enumeration Type Documentation

### 6.3.1 enum led_control_status_t

Enumeration of all possible return status for the LED Control functions.

Enumerator

| | |
|---|---|
| ***kStatusOk*** | No errors. |
| ***kStatusLedNumberOutOfRange*** | LED number is out of range. |
| ***kStatusInvalidCommand*** | Invalid Command. |
| ***kStatusUnexpectedError*** | Unexpected error. |

### 6.3.2 enum led_control_command_t

Enumeration of all available commands.

Enumerator

| | |
|---|---|
| ***kLedControlOff*** | Turn off the LED. |
| ***kLedControlOn*** | Turn on the LED. |
| ***kLedControlToggle*** | Toggle the LED. |
| ***kLedControlInvalidCommand*** | Invalid command. |

## 6.4 Function Documentation

### 6.4.1 void led_control_init ( void )

Initializes all hardware required for the LED control

Parameters

| | | |
|---|---|---|
| `in` | *Void* | |

Returns

Void

### 6.4.2 led_control_status_t led_control_update_led ( uint8_t *ledNumber,* led_control_command_t *command* )

Updates the LED with the provided data

Parameters

| | | |
|---|---|---|
| `in` | *ledNumber* | The number of LEDs to update |
| `in` | *command* | The action to apply |

Returns

led_control_status_t Error status for the operation.

# Chapter 7
# Potentiometer

## 7.1 Overview

Potentiometer module implements functions to obtain the relative position in a scale from 0% to 100%.

### Files

- file potentiometer_driver.h

### Macros

- #define POTENTIOMETER_ADC_INSTANCE
- #define POTENTIOMETER_ADC_RESOLUTION

### Enumerations

- enum potentiometer_status_t {
  kPotentiometerOk,
  kPotentiometerInitError,
  kPotentiometerConversionStartError }

### Functions

- potentiometer_status_t potentiometer_init (void)
- uint8_t potentiometer_get_position (void)

## 7.2 Macro Definition Documentation

### 7.2.1 #define POTENTIOMETER_ADC_INSTANCE

ADC instance for the potentiometer input.

### 7.2.2 #define POTENTIOMETER_ADC_RESOLUTION

Expanded ADC resolution (2^n)

## 7.3 Enumeration Type Documentation

### 7.3.1 enum potentiometer_status_t

Potentiometer driver status.

Enumerator

***kPotentiometerOk*** No error.
***kPotentiometerInitError*** Initialization error.
***kPotentiometerConversionStartError*** Error starting the channel conversion.

## 7.4 Function Documentation

### 7.4.1 potentiometer_status_t potentiometer_init ( void )

Initializes the Potentiometer driver

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

potentiometer_status_t Error status

### 7.4.2 uint8_t potentiometer_get_position ( void )

Returns the current potentiometer position in percentage

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

Potentiometer position in percentage (from 0 % to 100 %)

# Chapter 8
# Temperature Sensor

## 8.1 Overview

Temperature sensor module implements functions to obtain the internal chip temperature of the SoC by reading the temperature sensor ADC channel.

### Files

- file temperature_sensor.h

### Macros

- #define TEMPERATURE_SENSOR_ADC_INSTANCE
- #define TEMPERATURE_SENSOR_V_BANDGAP_mV
- #define TEMPERATURE_SENSOR_ADC_RESOLUTION
- #define TEMPERATURE_SENSOR_VTEMP25_mV
- #define TEMPERATURE_SENSOR_SLOPE_uV

### Enumerations

- enum temperature_sensor_status_t {
  kTemperatureSensorOk,
  kTemperatureInitError,
  kTemperatureSensorConversionStartError }

### Functions

- temperature_sensor_status_t temperature_sensor_init (void)
- int16_t temperature_sensor_get_chip_temperature (void)

## 8.2 Macro Definition Documentation

### 8.2.1 #define TEMPERATURE_SENSOR_ADC_INSTANCE

ADC module instance connected to the internal temperature sensor.

### 8.2.2 #define TEMPERATURE_SENSOR_V_BANDGAP_mV

Bandgap ADC channel voltage (in mV)

### 8.2.3 #define TEMPERATURE_SENSOR_ADC_RESOLUTION

Expanded resolution value 2^Resolution.

### 8.2.4 #define TEMPERATURE_SENSOR_VTEMP25_mV

Temperature sensor voltage @25C defined by the datasheet.

### 8.2.5 #define TEMPERATURE_SENSOR_SLOPE_uV

Temperature sensor slope (in uV) defined by the datasheet.

## 8.3 Enumeration Type Documentation

### 8.3.1 enum temperature_sensor_status_t

Temperature sensor status.

Enumerator

| | |
|---|---|
| ***kTemperatureSensorOk*** | No error. |
| ***kTemperatureInitError*** | Initialization error. |
| ***kTemperatureSensorConversionStartError*** | Error starting the channel conversion. |

## 8.4 Function Documentation

### 8.4.1 temperature_sensor_status_t temperature_sensor_init ( void )

Initializes the temperature sensor

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

temperature_sensor_status_t Error status

### 8.4.2 int16_t temperature_sensor_get_chip_temperature ( void )

Returns the current chip temperature with a 0.01 °C resolution.

Parameters

| | | |
|---|---|---|
| `in` | *None* | |

Returns

Temperature measurement with a 0.01 °C resolution or 0xFFFF if an error occurred

# Chapter 9
# Input Report

## 9.1 Overview

Input Report implements functions to acquire users input when a button is pressed (GPIO) or a capacitive sensor is touched (TSI).

Input Report functions are divided in two sub-modules. Keyboard module is part of the Connectivity Software stack and implements functions to handle GPIO inputs. TSI Sensor module includes functions to acquire user input when pressing capacitive touch sensors.

Keyboard module is explained in the Connectivity Software stack documentation. Please refer to the Connectivity Framwork Reference Manual (CONNFWRKRM) for more information on this module.

### Modules

- TSI Sensor

### Files

- file tsi_sensor.h

## 9.2 TSI Sensor

### 9.2.1 Overview

TSI Sensor implements functions to detect touch events in capacitive pads.

### Data Structures

- union tsi_sensor_electrode_flags_t
- struct tsi_sensor_electrode_flags_t.activeFlag
- struct tsi_sensor_electrode_data_t

### Macros

- #define TSI_SENSOR_THRESHOLD_ADDER

### Typedefs

- typedef void(∗ tsi_sensor_callback_t) (tsi_sensor_electrode_flags_t ∗pElectrodeFlags)

## Enumerations

- enum tsi_sensor_status_t {
  kTsiOk,
  kTsiInitError,
  kTsiStartError }

## Functions

- tsi_sensor_status_t tsi_sensor_init (tsi_sensor_callback_t pCallbackFunc)
- tsi_sensor_status_t tsi_sensor_start_single_measurement (void)

## Variables

- tsi_sensor_electrode_flags_t tsiSensorActiveElectrodeFlag

## 9.2.2 Data Structure Documentation

### 9.2.2.1 union tsi_sensor_electrode_flags_t

TSI Electrode flags array.

Data Fields

| | | |
|---|---|---|
| uint16_t | overallFlagStatus | Status of all TSI flags. |
| struct tsi_sensor_electrode_flags_t | activeFlag | Status of each TSI flag. |

### 9.2.2.2 struct tsi_sensor_electrode_flags_t.activeFlag

Data Fields

| | | |
|---|---|---|
| uint16_t | electrode1: 1 | |
| uint16_t | electrode2: 1 | |
| uint16_t | electrode3: 1 | |
| uint16_t | electrode4: 1 | |

| | | |
|---|---|---|
| uint16_t | electrode5: 1 | |
| uint16_t | electrode6: 1 | |
| uint16_t | electrode7: 1 | |
| uint16_t | electrode8: 1 | |
| uint16_t | electrode9: 1 | |
| uint16_t | electrode10: 1 | |
| uint16_t | electrode11: 1 | |
| uint16_t | electrode12: 1 | |
| uint16_t | electrode13: 1 | |
| uint16_t | electrode14: 1 | |
| uint16_t | electrode15: 1 | |
| uint16_t | electrode16: 1 | |

### 9.2.2.3 struct tsi_sensor_electrode_data_t

TSI electrode data structure.

Data Fields

| | | |
|---|---|---|
| uint8_t | channel | Electrode channel. |
| uint16_t | threshold | Electrode threshold. |

## 9.2.3 Macro Definition Documentation

### 9.2.3.1 #define TSI_SENSOR_THRESHOLD_ADDER

Threshold value to detect a touch event.

## 9.2.4 Typedef Documentation

### 9.2.4.1 typedef void(∗ tsi_sensor_callback_t) (tsi_sensor_electrode_flags_t ∗pElectrodeFlags)

TSI Sensor callback function type.

## 9.2.5 Enumeration Type Documentation

### 9.2.5.1 enum tsi_sensor_status_t

TSI Sensor return status for functions.

Enumerator

| | |
|---|---|
| ***kTsiOk*** | No error. |
| ***kTsiInitError*** | Error initializing the module. |
| ***kTsiStartError*** | Error starting the measurements. |

## 9.2.6 Function Documentation

### 9.2.6.1 tsi_sensor_status_t tsi_sensor_init ( tsi_sensor_callback_t *pCallbackFunc* )

Initializes the TSI sensor.

Parameters

| | | |
|---|---|---|
| `in` | *Function* | to execute when a TSI touch is sensed. |

Returns

tsi_sensor_status_t Error status.

### 9.2.6.2 tsi_sensor_status_t tsi_sensor_start_single_measurement ( void )

Starts a single TSI sensor measurement.

Parameters

| | | |
|---|---|---|
| `in` | *None.* | |

Returns

tsi_sensor_status_t Error status.

## 9.2.7 Variable Documentation

### 9.2.7.1 tsi_sensor_electrode_flags_t tsiSensorActiveElectrodeFlag

TSI electrode flags.

***How to Reach Us:***

**Home Page:**
nxp.com

**Web Support:**
nxp.com/support