Atmel QTouch Library

User Guide

Supports QTouch® and QMatrix™ acquisition for Keys, Sliders and Rotors

Rev. 8207G-AT42-12/09

Table of Contents

TABLE OF CONTENTS **2**

1 PREFACE **5**

2 INTRODUCTION **6**

3 OVERVIEW **6**

4 ABBREVIATIONS AND DEFINITIONS **7**

4.1 DEFINITIONS 7

5 QTOUCH LIBRARIES **8**

5.1 INTRODUCTION 8
5.2 ACQUISITION METHODS 9
5.2.1 QTouch acquisition method *9*
5.2.1.1 Sensor schematics for a QTouch acquisition method design 10
5.2.2 QMatrix acquisition method *11*
5.2.2.1 Sensor schematics for a QMatrix acquisition method design 12

6 QTOUCH LIBRARY API **13**

6.1 TOUCH_API.H - PUBLIC HEADER FILE 13
6.2 TYPE DEFINITIONS AND ENUMERATIONS USED IN THE LIBRARY 13
6.2.1 Typedefs *13*
6.2.2 Enumerations *13*
6.2.2.1 sensor_type_t 13
6.2.2.2 aks_group_t 14
6.2.2.3 channel_t 14
6.2.2.4 hysteresis_t 15
6.2.2.5 resolution_t 15
6.2.2.6 recal_threshold_t 15
6.3 DATA STRUCTURES 16
6.3.1 qt_touch_status_t *16*
6.3.2 qt_touch_lib_config_data_t *16*
6.3.3 qt_touch_lib_measure_data_t *17*
6.3.4 qt_burst_lengths *17*
6.3.5 tag_sensor_t *18*
6.3.6 qt_lib_siginfo_t *18*
6.4 PUBLIC FUNCTIONS 19
6.4.1 qt_set_parameters *19*
6.4.2 qt_enable_key *20*
6.4.3 qt_enable_rotor *20*
6.4.4 qt_enable_slider *21*
6.4.5 qt_init_sensing *21*
6.4.6 qt_measure_sensors *22*
6.4.7 qt_calibrate_sensing *22*
6.4.8 qt_reset_sensing *23*
6.4.9 qt_get_sensor_delta *23*
6.4.10 qt_get_library_sig *23*
6.5 OVERVIEW OF TOUCH API USAGE AND SEQUENCE 24
6.5.1 Channel Numbering *25*
6.5.1.1 Channel numbering when using QTouch acquisition method 25
6.5.1.1.1 Channel numbering when routing SNS and SNSK pins to different ports 25
6.5.1.1.2 Channel numbering when routing SNS and SNSK pins to the same port 27
6.5.1.2 Channel numbering when using QMatrix acquisition method 28

6.5.2 Sensor Numbering 30
6.5.3 Using the Sensors 31
6.5.3.1 Avoiding Cross-talk 31
6.5.3.2 Detect Integration 31
6.5.3.3 Adjacent Key Suppression 31
6.5.3.4 Multiple measurements 32
6.5.3.5 Measurement Limit 32
6.5.3.6 Filtering Signal Measurements 32
6.5.3.6.1 Example: Averaging the Last Four Signal Values 33
6.6 CONSTRAINTS 33
6.6.1 QTouch acquisition method constraints 33
6.6.2 QMatrix acquisition method API constraints 34
6.7 FREQUENCY OF OPERATION (VS) CHARGE CYCLE/DWELL CYCLE TIMES: 35
6.8 INTERRUPTS 35
6.9 INTEGRATING QTOUCH LIBRARIES IN YOUR APPLICATION 36
6.9.1 Directory structure of the library files 36
6.9.2 Integrating QTouch acquisition method libraries in your application 36
6.9.2.1 Example 37
6.9.2.2 Checklist of items for integrating QTouch acquisition method libraries 39
6.9.3 Integrating QMatrix acquisition method libraries in your application 39
6.9.3.1 Resources used by QMatrix acquisition method libraries 41
6.9.3.2 Example 41
6.9.3.3 Checklist of items for integrating QMatrix Capacitive sensing libraries 42
6.9.4 Common checklist items 43
6.9.4.1 Configuring the stack size for the application 43
6.10 EXAMPLE PROJECT FILES 44
6.10.1 Using the Sample projects 44
6.10.2 Example applications for QTouch acquisition method libraries 44
6.10.2.1 Selecting the right configuration 44
6.10.2.2 Changing the settings to match your device 46
6.10.2.2.1 Processor settings 46
6.10.2.3 Changing the library configuration parameters 47
6.10.2.4 Using the example projects 48
6.10.3 Example applications for QMatrix acquisition method libraries 49
6.10.3.1 Selecting the right configuration 50
6.10.3.2 Changing the library configuration parameters 51
6.10.3.3 Using the example projects 53
6.10.4 Allocating unused Port Pins for User Application 53
6.10.4.1 Disabling and Enabling of Pull-up for AVR devices: 54
6.10.5 Adjusting the Stack size when using IAR IDE 54
6.10.6 Optimization levels 55
6.10.7 Debug Support in Example applications 56
6.10.7.1 Debug Support in the sample applications for EVK2080 and QT600 boards 56
6.10.7.2 How to turn on the debug option 56
6.10.7.3 Debug Interface if USB Bridge board is not available 57

7 LIBRARY VARIANTS 58

7.1 QTOUCH ACQUISITION METHOD LIBRARY VARIANTS 58
7.1.1 Introduction 58
7.1.2 Support for different compiler tool chains 58
7.1.3 QTouch Acquisition method library naming conventions 58
7.1.3.1 Naming convention for libraries to be used with WinAVR GCC tool chain 58
7.1.3.2 Naming convention for libraries to be used with IAR-EWAR 59
7.1.4 QTouch acquisition method library variants 59
7.1.5 Port combinations supported for SNS and SNSK pin configurations 64
7.1.5.1 Port combinations supported for single port pair SNS and SNSK pin configurations 64
7.1.5.2 Tips on pin assignments for the sensor design using one pair of SNS/SNSK ports 65

7.1.5.3 Port combinations supported for two port pair SNS and SNSK pin configurations 66
7.1.5.3.1 Port combinations when using ports A to G on devices 66
7.1.5.3.2 Port combinations when using ports D to K on devices 67
7.1.5.4 Tips on pin assignments for the sensor design using two pairs of SNS / SNSK ports 67
7.1.6 Sample applications and Memory requirements for QTouch acquisition method libraries 68
7.1.6.1 WinAVR GCC compiler tool chain memory footprint for QTouch acquisition method libraries 68
7.1.6.2 IAR compiler tool chain memory footprint for QTouch acquisition method libraries 70
7.2 QMATRIX ACQUISITION METHOD LIBRARY VARIANTS 73
7.2.1 Introduction 73
7.2.2 Support for different compiler tool chains 73
7.2.3 QMatrix Acquisition method library naming conventions 73
7.2.4 QMatrix acquisition method library variants 75
7.2.4.1 Devices supported for QMatrix Acquisition 75
7.2.4.2 QMatrix acquisition method Library Variants for the devices 76
7.2.4.2.1 QMatrix acquisition method Library Variants for ATtiny88 76
7.2.4.2.2 QMatrix™ acquisition method Library Variants for ATtiny167 77
7.2.4.2.3 QMatrix acquisition method Library Variants for ATmega88P/ATmega88PA 77
7.2.4.2.4 QMatrix acquisition method Library Variants for ATmega8535 78
7.2.4.2.5 QMatrix acquisition method Library Variants for ATmega16/ATmega16A 79
7.2.4.2.6 QMatrix acquisition method Library Variants for ATmega164P 80
7.2.4.2.7 QMatrix acquisition method Library Variants for v3g3/avr5g3 81
(ATmega165P,ATmega325P,ATmega645) 81
7.2.4.2.8 QMatrix acquisition method Library Variants for ATmega324P 82
7.2.4.2.9 QMatrix acquisition method Library Variants for ATmega324PA 83
7.2.4.2.10 QMatrix acquisition method Library Variants for ATmega128RFA1 84
7.2.4.2.11 QMatrix acquisition method Library Variants for ATxmega64A3 85
7.2.4.2.12 QMatrix acquisition method Library Variants for ATxmega64A1 86
7.2.4.2.13 QMatrix acquisition method Library Variants for ATxmega128A3, ATxmega192A3, ... 87
ATxmega256A3, ATxmega256A3B, ... 87
7.2.4.2.14 QMatrix acquisition method Library Variants for ATxmega128A1 87
7.2.4.3 Device configurations supported by the QMatrix acquisition method library variants 88
7.2.4.3.1 Device configuration supported for ATtiny88 88
7.2.4.3.2 Device configuration supported for ATtiny167 89
7.2.4.3.3 Device configuration supported for ATmega16/ ATmega16A 89
7.2.4.3.4 Device configuration supported for ATmega8535 90
7.2.4.3.5 Device configuration supported for ATmega88P/ ATmega88PA 90
7.2.4.3.6 Device configuration supported for ATmega164P 91
7.2.4.3.7 Device configuration supported for ATmega165P 92
7.2.4.3.8 Device configuration supported for ATmega324P / ATmega324PA 95
7.2.4.3.9 Device configuration supported for ATmega128RFA1 96
7.2.4.3.10 Device configuration supported for ATxmega<xxx>A3 97
7.2.4.3.11 Device configuration supported for ATxmega<xxx>A1 97

8 CHECKLIST 98

9 MISRA COMPLIANCE REPORT 99

9.1 WHAT IS COVERED 99
9.2 TARGET ENVIRONMENT 99
9.3 DEVIATIONS FROM MISRA C STANDARDS 99
9.3.1 QTouch acquisition method libraries 99
9.3.2 QMatrix acquisition method libraries 100

10 KNOWN ISSUES 100

10.1 LINKER WARNING 101
10.1.1 SFRB Warning 101

11 REVISION HISTORY 104

DISCLAIMER 105

1 Preface

This manual contains information that enables customers to implement capacitive touch solutions on ATMEL AVR® microcontrollers using ATMEL QTouch libraries.

This guide is a functional description of the library software, its programming interface and it also describes its use on the supported reference systems.

Use of this software is bound by the Software License Agreement included with the Library.

Related documents from ATMEL
Documents related to QTouch capacitive sensing solutions from ATMEL are

- TS2080A/B data sheet.
- Release Notes for ATMEL QTouch Software libraries 3.2.
- Capacitive touch sensor design guide (ATMEL Document reference 10620E-AT42-09/09 http://www.atmel.com/dyn/resources/prod_documents/doc10620.pdf).

If you need Assistance
For assistance with QTouch capacitive sensing software libraries and related issues, contact your local ATMEL sales representative directly or send an email to *avr@atmel.com.*

2 Introduction

ATMEL's QTouch Library is a royalty free software library (available for GCC and IAR compiler tool chains) for developing touch applications on standard AVR microcontrollers. Customers can link the library into their applications in order to provide touch sensing capability in their projects. The Library can be used to develop single chip solutions for control applications which have touch sensing capabilities, or to develop standalone touch sensing solutions which interface with other host or control devices.

Features of ATMEL's QTouch Library include

- Capacitive touch sensing using patented charge-transfer signal acquisition for robust sensing.
- Support for a wide range of 8 and 32 bit AVRs.
- Support both QTouch and QMatrix acquisition methods.
- Support upto 64 touch sense channels.
- Flexible choice of touch sensing functionality (keys, sliders, wheels) in a variety of combinations.
- Includes Adjacent Key Suppression™ (AKS™) technology for the unambiguous detection of key events.
- Support for both IAR and GCC compiler tool chains.

This user guide describes the content, design and use of the QTouch Libraries.

This guide should be read in conjunction with all of the applicable documents listed below

- Device datasheet for the selected AVR device used for touch sensing.
- Data sheet for the selected evaluation board.

The intended readers of this document are Engineers, who use the QTouch Library on AVR Microcontrollers to realize capacitive touch sensing solutions.

3 Overview

This chapter gives a brief introduction to each of the chapters that make up this document

1. **Preface**
2. **Introduction** : Provides an introduction to the scope and use of the QTouch Library.
3. **Overview** : This chapter
4. **Abbreviations and Definitions** : Provides a description of the abbreviations and definitions used in this document
5. **QTouch Libraries** : Provides an overview of the QTouch libraries and the different acquisition methods
6. **QTouch Library API** : Provides details about the QTouch library Application Programming Interface.
7. **Library Variants** : Provides details on the different variants of QTouch libraries available for AVRs
8. **Checklist** : Provides tips and a list of check list items to trouble shoot common issues.
9. **Misra Compliance Report** : Provides details on the Mira compliance of the QTouch libraries
10. **Known Issues** : Provides details on known issues with the touch library.
11. **Revision History** : Provides a revision history of this document.

4 Abbreviations and Definitions

4.1 Definitions

- **AVR:** refers to a device(s) in the tinyAVR®, megaAVR®, XMEGA™ and UC3™ microcontroller family.
- **ATMEL QTouch Library**: The combination of libraries for both touch sensing technologies **(QTouch and QMatrix).**
- **QTouch Technology**: A type of capacitive touch sensing technology using self capacitance - each channel has only one electrode.
- **QMatrix Technology**: A type of capacitive touch sensing technology using mutual capacitance – each channel has an outer output electrode (X) and an inner input electrode (Y).
- **Sensor**: A channel or group of channels used to form a touch sensor. Sensors are of 3 types (keys, Rotors and Sliders).
- **KEY**: a Single Channel forms a single KEY type sensor.
- **ROTOR**, a group of channels forms a ROTOR sensor to detect angular position of touch.
    - A Rotor is composed of 3 channels for a QTouch acquisition method.
    - A Rotor can be composed of 3 to 8 channels for QMatrix acquisition method.
- **SLIDER**, a group of channels forms a SLIDER sensor to detect the linear position of touch.
    - A Slider is composed of 3 channels for a QTouch acquisition method.
    - A Slider can be composed of 3 to 8 channels for QMatrix acquisition method.
- **AKS**: Adjacent Key Suppression. See Section 6.5.3.3.
- **SNS PIN**: Sense line for capacitive measurement using the QTouch Technology - connected to Cs.
- **SNSK PIN**: Sense Key line for capacitive measurement using the QTouch Technology - connected to channel electrode through Rs.
- **X Line**: The drive electrode (or drive line) used for QMatrix Technology.
- **Y Line**: The receive electrode (or receive line) used for QMatrix Technology.
- **Port Pair**: A combination of SNS port and SNSK port to which sensors are connected for QTouch technology. The SNS and SNSK ports used in a port pair can be located in the same AVR Port (8 pins for 4 sensors), or they may be in different 2 different AVR Ports (8+8 pins for 8 sensors).
- **Charge Cycle Period**: It is the width of the charging pulse applied to the channel capacitor.
- **Dwell Cycle**: In a QMatrix acquisition method, the duration in which charge coupled from X to Y is captured.
- **Acquisition**: A single capacitive measurement process.
- **Electrode**: Electrodes are typically areas of copper on a printed circuit board but can also be areas of clear conductive indium tin oxide (ITO) on a glass or plastic touch screen.

5 QTouch Libraries

5.1 Introduction

ATMEL QTouch provides a simple to use solution to realize touch sensing solutions on a range of supported ATMEL AVR Microcontrollers.

The QTouch libraries provide support for both QTouch and QMatrix acquisition methods.

Touch sensing using QMatrix or QTouch acquisition methods can be added to an application by linking the appropriate ATMEL QTouch Library for the AVR Microcontroller and using a simple set of API to define the touch channels and sensors and then calling the touch sensing API's periodically (or based on application needs) to retrieve the channel information and determine touch sensor states.

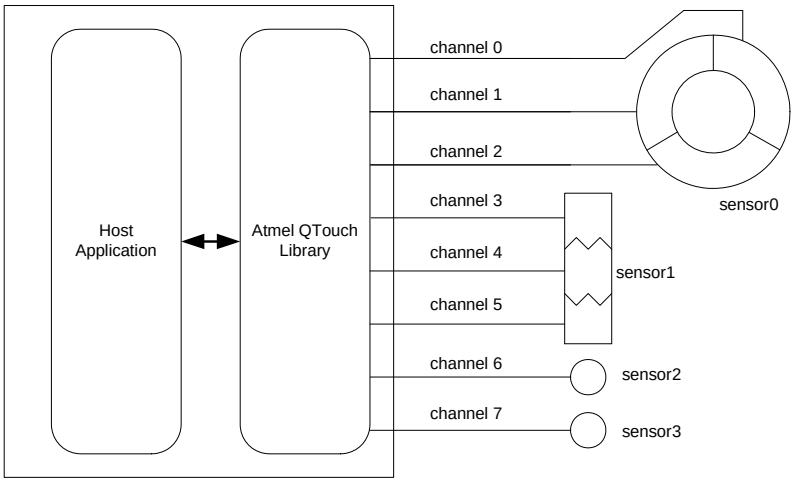


Figure 1 : Typical interface of the ATMEL QTouch library with the host application.

Figure 1 shows a typical configuration of channels when using an AVR and using the ATMEL QTouch Library. The ATMEL QTouch Library has been added to a host application running on an AVR microcontroller. The sample configuration illustrates using the library that supports eight touch channels numbered 0 to 7. The sensors are configured in the following order,

- Channels 0 to 2 have been configured as a rotor sensor.
- Channels 3 to 5 have been configured as a slider sensor.
- Channel 6 is configured as key sensor.
- Channel 7 is configured as key sensor.

The host application uses the QTouch library API's to configure these channels and sensors, and to initiate detection of a touch using capacitive measurements.

The QTouch libraries use minimal resources of the microcontroller. The sampling of the sensors is controlled by the QTouch library, while the sampling period is controlled by the application (possibly using timers, sleep periods, varying the CPU clock, external events like interrupts or communications, etc).

5.2 Acquisition Methods

There are two methods available for touch acquisition namely

1. QTouch acquisition method.
2. QMatrix acquisition method.

There are libraries available for AVR microcontrollers for each acquisition method.

5.2.1 QTouch acquisition method

The QTouch acquisition method charges an electrode of unknown capacitance to a known potential. The resulting charge is transferred into a measurement capacitor (Cs). The cycle is repeated until the voltage across Cs reaches a voltage. The signal level is the number of charge transfer cycles it took to reach the voltage. Placing a finger on the touch surface introduces external capacitance that increases the amount of charge transferred each cycle, reducing the total number of cycles required for Cs to reach the voltage. When the signal level (number of cycles) goes below the present threshold, then the sensor is reported as in detect.

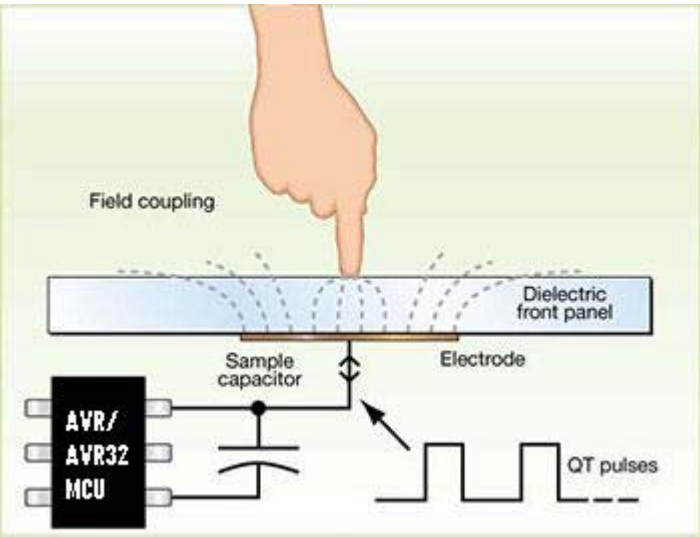


Figure 2: QTouch Acquisition

QTouch acquisition method sensors can drive single or multiple keys. Where multiple keys are used, each key can be set for an individual sensitivity level. Keys of different sizes and shapes can be used to meet both functional and aesthetic requirements.

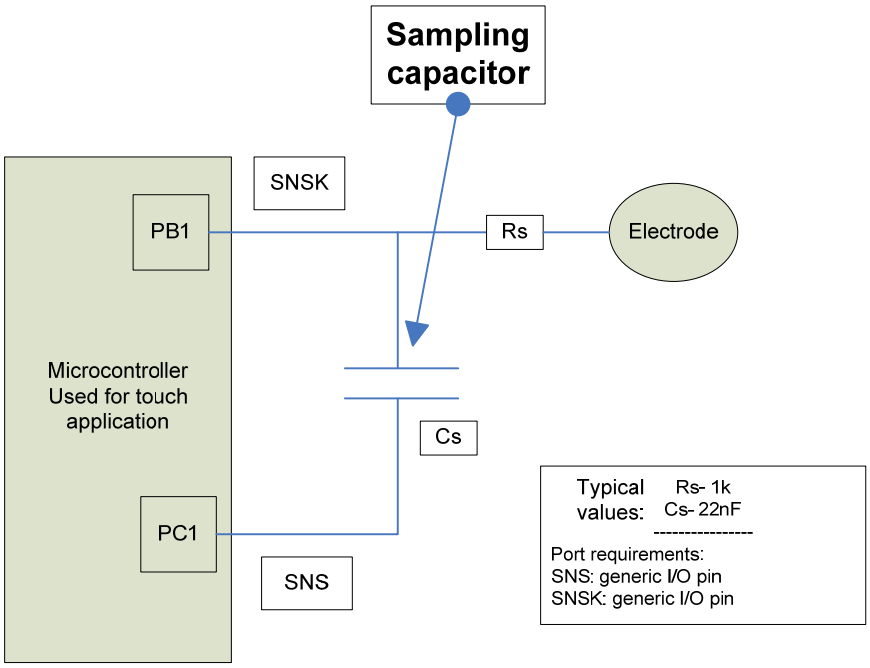
NOTE : It is recommended to keep the size of the keys larger than 6mmx6mm to ensure reliable and robust measurements, although actual key design requirements also depend on panel thickness and material. Refer to the ATMEL Touch Design Guide for details.

QTouch acquisition method can be deployed in two ways

- normal touch contact (i.e. when pressing buttons on a panel), and
- high sensitivity proximity mode (i.e. when a panel lights up before you actually contact it).

QTouch circuits offer tremendous signal-to-noise ratio, very good low power performance, and the easiest sensor layout.

5.2.1.1 Sensor schematics for a QTouch acquisition method design



Rs- Series resistor, Cs – Sample capacitor, PB1- PortB bit1, and PC1- PortC bit1

Figure 3 : Schematics for a QTouch acquisition method design

5.2.2 QMatrix acquisition method

QMatrix devices detect touch using a scanned passive matrix of electrode sets. A single QMatrix device can drive a large number of keys, enabling a very low cost-per-key to be achieved.

QMatrix uses a pair of sensing electrodes for each channel. One is an emitting electrode into which a charge consisting of logic pulses is driven in burst mode. The other is a receive electrode that couples to the emitter via the overlying panel dielectric. When a finger touches the panel the field coupling is changed, and touch is detected.

QMatrix circuits offer good immunity to moisture films, extreme levels of temperature stability, superb low power characteristics, and small IC package sizes for a given key count.

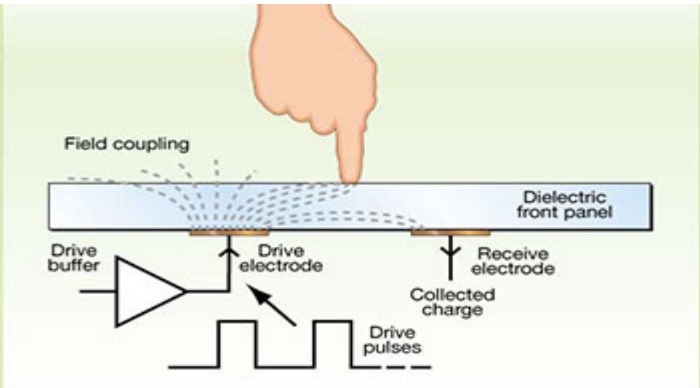


Figure 4 : QMatrix Acquisition method

The drive electrode (or drive line) used for QMatrix charge transfer is labeled as the X line.
The receiver electrode (or receive line) used for QMatrix charge transfer is labeled as the Y line.

5.2.2.1 Sensor schematics for a QMatrix acquisition method design

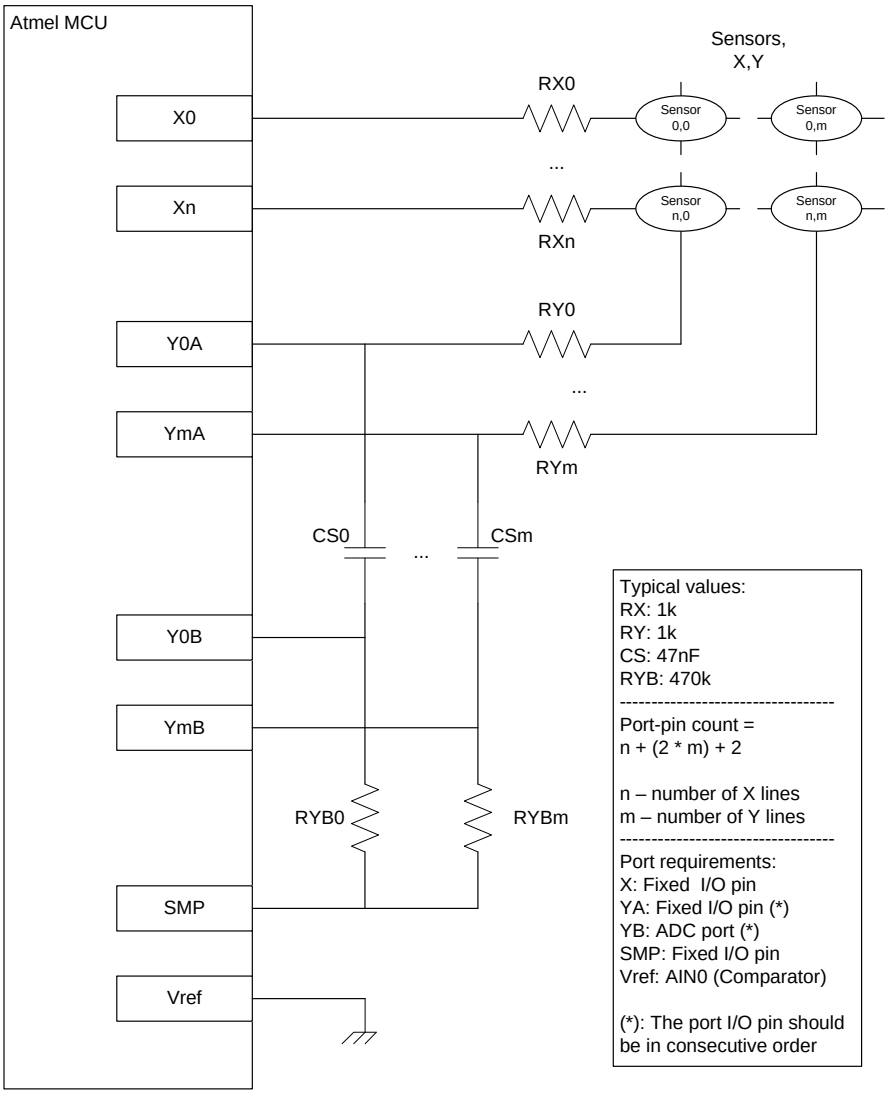


Figure 5 : Schematics for a QMatrix acquisition method design

6 QTouch Library API

This section describes the QTouch library Application Programming Interface (API) for touch sensing using QTouch and QMatrix acquisition methods on AVR microcontrollers.

Using the API, Touch sensors and the associated channels can be defined. Once touch sensing has been initiated by the user, the host application can use the API to make touch measurements and determine the status of the sensors.

6.1 touch_api.h - public header file

The *touch_api.h* header file is the public header file which needs to be included in users application and it has the type definitions and function prototypes of the API's listed in sections 6.2 , 6.3 and 6.4

The touch_api.h header file is located in the library distribution in the following directory.

- *..\Atmel_QTouch_Libraries_3.2\include*

6.2 Type Definitions and enumerations used in the library

6.2.1 Typedefs

This section lists the type definitions used in the library.

Typedef	Notes
uint8_t	unsigned 8-bit integer
int8_t	signed 8 bit integer
uint16_t	unsigned 16-bit integer
int16_t	signed 16-bit integer
uint32_t	unsigned 32 bit integer
threshold_t	unsigned 8-bit integer used for setting a sensor detection threshold

6.2.2 Enumerations

This section lists the enumerations used in the QTouch Library.

6.2.2.1 sensor_type_t

Enumeration sensor_type_t
Use Define the type of the sensor

Values	Comment
SENSOR_TYPE_UNASSIGNED	Channel is not assigned to any sensor
SENSOR_TYPE_KEY	Sensor is of type KEY
SENSOR_TYPE_ROTOR	Sensor is of type ROTOR
SENSOR_TYPE_SLIDER	Sensor is of type SLIDER

6.2.2.2 aks_group_t

Enumeration aks_group_t
Use Defines the Adjacent Key Suppression (AKS) groups each sensor may be associated with (see section 6.5.3.2)

AKS is selectable by the system designer
7 AKS groups are supported by the library

Values	Comment
NO_AKS_GROUP	NO AKS group selected for the sensor
AKS_GROUP_1	AKS Group number 1
AKS_GROUP_2	AKS Group number 2
AKS_GROUP_3	AKS Group number 3
AKS_GROUP_4	AKS Group number 4
AKS_GROUP_5	AKS Group number 5
AKS_GROUP_6	AKS Group number 6
AKS_GROUP_7	AKS Group number 7

6.2.2.3 channel_t

Enumeration channel_t
Use The channel numbers used in the library.

When using the QTouch acquisition method, the channel numbers have a one to one mapping to the pin numbers of the port being used.

When using the QMatrix acquisition method, the channel numbers are ordered in a matrix sequence

Values	Comment
CHANNEL_0	Channel number : 0
CHANNEL_1	Channel number : 1
CHANNEL_2	Channel number : 2
CHANNEL_3	Channel number : 3
.....	Channel number : ..
Upto CHANNEL (N-1)	Channel number N-1 : for an N Channel library

The maximum number of channels supported is dependent on the library variant. Possible values of N are as listed below

Acquisition method	Device type	Possible values of N (Maximum number of channels)
QTouch acquisition	8 bit	4,8,16
	32 bit	8, 16, 32
QMatrix Acquisition	8 bit	8,16,32,64

6.2.2.4 hysteresis_t

Enumeration Hysteresis_t
Use Defines the sensor detection hysteresis value. This is expressed as a percentage of the sensor detection threshold.

This is configurable per sensor.

HYST_x = hysteresis value is x percent of detection threshold value (rounded down). Note that a minimum value of 2 is used as a hard limit. Example: if detection threshold = 20, then:
HYST_50 = 10 (50 percent of 20)
HYST_25 = 5 (25 percent of 20)
HYST_12_5 = 2 (12.5 percent of 20)
HYST_6_25 = 2 (6.25 percent of 20 = 1, but set to the hard limit of 2)

Values	Comment
HYST_50	50% Hysteresis
HYST_25	25% Hysteresis
HYST_12_5	12.5% Hysteresis
HYST_6_25	6.25% Hysteresis

6.2.2.5 resolution_t

Enumeration resolution_t
Use For rotors and sliders, the resolution of the reported angle or position.
RES_x_BIT = rotor/slider reports x-bit values.
Example: if slider resolution is RES_7_BIT, then reported positions are in the range 0..127.

Values	Comment
RES_1_BIT	1 bit resolution : reported positions range 0 – 1
RES_2_BIT	2 bit resolution : reported positions range 0 – 3
RES_3_BIT	3 bit resolution : reported positions range 0 – 7
RES_4_BIT	4 bit resolution : reported positions range 0 – 15
RES_5_BIT	5 bit resolution : reported positions range 0 – 31
RES_6_BIT	6 bit resolution : reported positions range 0 – 63
RES_7_BIT	7 bit resolution : reported positions range 0 – 127
RES_8_BIT	8 bit resolution : reported positions range 0 – 255

6.2.2.6 recal_threshold_t

Enumeration recal_threshold_t
Use A sensor recalibration threshold. This is expressed as a percentage of the sensor detection threshold.

This is for automatic recovery from false conditions, such as a calibration while sensors were touched, or a significant step change in power supply voltage.

If the false condition persists the library will recalibrate according to the settings of the recalibration threshold.

This setting is applicable to all the configured sensors.

Usage :
RECAL_x = recalibration threshold is x percent of detection threshold value (rounded down).
Note: a minimum value of 4 is used.
Example: if detection threshold = 40, then:
RECAL_100 = 40 (100 percent of 40)
RECAL_50 = 20 (50 percent of 40)
RECAL_25 = 10 (25 percent of 40)
RECAL_12_5 = 5 (12.5 percent of 40)
RECAL_6_25 = 4 (6.25 percent of 40 = 2, but value is limited to 4)

Values	Comment
RECAL_100	100% recalibration threshold
RECAL_50	50% recalibration threshold
RECAL_25	25% recalibration threshold
RECAL_12_5	12.5% recalibration threshold
RECAL_6_25	6.25% recalibration threshold

6.3 Data structures

This section lists the data structures that hold sensor status, settings, and diagnostics information

6.3.1 qt_touch_status_t

structure qt_touch_status_t
Input / Output Output from the Library
Use Holds the status (On/ Off) of the sensors and the linear and angular positions of sliders and rotors respectively

Fields	Comment
sensor_states[]	For Sensor, the sensor_states. Bit “n” = state of nth sensor : Bit Value 0 - indicates the sensor is not in detect Bit Value 1 - indicates the sensor is in detect
rotor_slider_values[]	Rotors angles or slider positions if rotors and sliders are used. These values are valid when sensor states shows that the corresponding rotor or slider is in detect

6.3.2 qt_touch_lib_config_data_t

Structure qt_touch_lib_config_data_t
Input / Output Input to the library
Use Configuration data for the library.

Fields	Type	Comment
qt_recal_threshold	recal_threshold_t	Sensor recalibration threshold. Default: RECAL_50 (recalibration threshold

		= 50 percent of detection threshold
qt_di	uint8_t	Sensor detect integration (DI) limit. Default value: 4
qt_drift_hold_time	uint8_t	Sensor drift hold time in units of 200 ms. Default value: 20 (20 x 200 ms = 4s), that is hold off drifting for 4 seconds after leaving detect
qt_max_on_duration	uint8_t	Sensor maximum on duration in units of 200 ms. For example: 150 = recalibrate after 30s (150 x 200 ms). 0 = recalibration disabled Default value: 0 (recalibration disabled)
qt_neg_drift_rate	uint8_t	Sensor negative drift rate in units of 200 ms. Default value: 20 (20 x 200 ms = 4s per LSB)
qt_pos_drift_rate	uint8_t	Sensor positive drift rate in units of 200 ms. Default value: 5 (5 x 200 ms = 1s per LSB)

The QTouch library exports a variable of this type so that the user can specify the threshold parameters for the library. The API *qt_set_parameters() should be called to apply the parameters specified.*

extern qt_touch_lib_config_data_t ***qt_config_data;***

6.3.3 qt_touch_lib_measure_data_t

structure qt_touch_lib_measure_data_t
Input / Output Output from the library
Use Data structure which holds the sensor and channel states and values.

Fields	Type	Comment
channel_signals	uint16_t	The measured signal on each channel.
channel_references	uint16_t	The reference signal for each channel.
qt_touch_status	qt_touch_status_t	The state and position of the configured sensors

The QTouch library exports a variable of this type which can be accessed to retrieve the touch status of all the sensors.

extern qt_touch_lib_measure_data_t ***qt_measure_data;***

6.3.4 qt_burst_lengths

structure qt_burst_length
Input / Output Input to the library
Use **NOTE : Applicable only to the QMatrix acquisition method libraries**

This data structure is used to specify the burst lengths for each of the QMatrix channels

Fields	Type	Comment
qt_burst_length[]	uint8_t	The burst length for each of the QMatrix channel in units of pulses. Default value: 64 pulses. These values can be configured for each channel individually.

The signal gain for each sensor is controlled by circuit parameters as well as the burst length. The burst length is simply the number of times the charge-transfer ('QT') process is performed on a given sensor. Each QT process is simply the pulsing of an X line once, with a corresponding Y line enabled to capture the resulting charge passed through the sensor's capacitance Cx.

The QMatrix acquisition method library exports a variable of this type which can be accessed to set the burst length for each of the QMatrix channels

extern uint8_t ***qt_burst_lengths[QT_NUM_CHANNELS];***

6.3.5 tag_sensor_t

structure tag_sensor_t
Input / Output Output from the library
Use Data structure which holds the internal sensor state variables used by the library.

Fields	Type	Comment	
State	uint8_t	internal sensor state	
general_counter	uint8_t	general purpose counter: used for calibration, drifting, etc	
ndil_counter	uint8_t	drift Integration counter	
Threshold	uint8_t	sensor detection threshold	
type_aks_pos_hyst	uint8_t	holds information for sensor type, AKS group, positive recalibration flag, and hysteresis value	
		Bit fields	**Use**
		B1 : B0	Hysteresis
		B2	positive recalibration flag
		B5:B3	AKS group
		B7:B6	sensor type
from_channel	uint8_t	starting channel number for sensor	
to_channel	uint8_t	ending channel number for sensor	
Index	uint8_t	index for array of rotor/slider values	

6.3.6 qt_lib_siginfo_t

structure qt_lib_siginfo_t
Input / Output Output from the library
Use Data structure which holds the information about the library variant and its version information.

qt_lib_siginfo_t structure definition for a QTouch acquisition method library variant			
Fields	Type	Comment	
library_version	uint16_t	Holds the library version information.	
		Bit fields	**Use**
		B3 : B0	Patch version of the library
		B7 : B4	Minor version of the library
		B15:B8	Major version of the library
lib_sig_lword	uint16_t	Holds the general information about the library	
		Bit fields	**Use**
		B1 : B0	Library Type : 00 : QTouch acquisition method 01 : QMatrix acquisition method
		B2	Compiler tool chain used 0 – GCC 1 – IAR
		B9 : B3	Maximum number of channels supported by the library
		B10	0 – Library supports only keys

			1 – Library supports keys and rotors
		B15 : B11	Maximum number of rotors and sliders supported by the library
lib_sig_hword	uint16_t	Reserved	
Port_SNS1	int8_t	Port on which SNS Pins are configured (first pair of SNS / SNSK pins)	
Port_SNSK1	int8_t	Port on which SNSK Pins are configured (first pair of SNS / SNSK pins)	
Port_SNS2	int8_t	Port on which SNS Pins are configured (second pair of SNS / SNSK pins)	
Port_SNSK2	int8_t	Port on which SNSK pins are configured (second pair of SNS / SNSK pins	
Delay_Cycle	uint8_t	Delay Cycle time	
Port_Count	uint8_t	The number of SNS/SNSK port pairs used in the library	

qt_lib_siginfo_t structure definitions for a QMatrix acquisition method library variant			
Fields	Type	Comment	
library_version	uint16_t	Holds the library version information.	
		Bit fields	**Use**
		B3 : B0	Patch version of the library
		B7 : B4	Minor version of the library
		B15:B8	Major version of the library
lib_sig_lword	uint16_t	Holds the general information about the library	
		Bit fields	**Use**
		B1 : B0	Library Type : 00 : QTouch acquisition method 01 : QMatrix acquisition method
		B2	Compiler tool chain used 0 – GCC 1 – IAR
		B9 : B3	Maximum number of channels supported by the library
		B10	0 – Library supports only keys 1 – Library supports keys and rotors
		B15 : B11	Maximum number of rotors and sliders supported by the library
lib_sig_hword	uint16_t	Holds information about the X and Y lines for a QMatrix library variant	
		Bit fields	Use
		B4 : B0	Number of X Lines
		B8 : B5	Number of Y Lines
		B9	0 – Y lines are lower pins (pin 0 to 3) 1 – Y lines are on higher pins (Pin 4 to 7)
Port_X	int8_t	Port on which X lines are configured	
Port_YA	int8_t	Port on which YA lines are configured	
Port_YB	int8_t	Port on which YB lines are configured	
Port_SMP	int8_t	Port on which SMP line is configured	
Delay_Cycle	uint8_t	Dwell Cycle Time	
Smp_Pin	uint8_t	SMP Pin Number	

6.4 Public Functions

This section lists the public functions available in the QTouch libraries and its usage.

6.4.1 qt_set_parameters

This function is used to initialize the threshold parameters of the QTouch and QMatrix acquisition method libraries. This function should be called before configuring and using any sensors and channels.

*void qt*_set_parameters *(void)*

Arguments	Type	Comment
Void	-	This function will initialize the parameters required by the library to default values .But the default values can be changed by the user by modifying the global threshold values as defined in *qt_touch_lib_config_data_t* . See section 6.3.2 for details.

NOTE :

- This function can be called any time to apply the threshold parameters of the library as specified by modifying the global data structure *qt_config_data* exported by the library.

6.4.2 qt_enable_key

This function is used to configure a channel as a key.

void qt_enable_key (
channel_t channel , ,
aks_group_t aks_group ,
threshold_t detect_threshold ,
hysteresis_t detect_hysteresis
)

Arguments	Type	Comment
Channel	channel_t	Specifies the channel number to be configured for use as a “key”
aks_group	aks_group	Specifies the aks group associated with the sensor being configured as “key”
detect_threshold	threshold_t	Specifies the detect threshold for the sensor
detect_hysteresis	hysteresis_t	Specifies the detection hysteresis for the sensor

6.4.3 qt_enable_rotor

This function is used to configure a set of channels as a rotor.

void qt_enable_rotor (
channel_t from_channel ,
channel_t to_channel ,
aks_group_t aks_group ,
threshold_t detect_threshold ,
hysterisis_t detect_hysterisis ,
resoulution_t angle_resolution ,
uint8_t angle_hysterisis
)

Arguments	Type	Comment
from_channel	Channel_t	Specifies the starting channel number to be configured for use as a “Rotor”
to_channel	Channel_t	Specifies the end channel number to be configured for use as a “Rotor”
aks_group	aks_group	Specifies the aks group associated with the sensor being configured as “ROTOR”
detect_threshold	threshold_t	Specifies the detect threshold for the sensor
detect_hysterisis	hysterisis_t	Specifies the detection hysteresis for the sensor
angle_resolution	resolution_t	Specifies the resolution of the reported angle value
angle_hysterisis	uint8_t	Specifies the hysteresis of the reported angle value

NOTE :

- A “Rotor” sensor requires contiguous channel numbers.

- The rotor / slider number depends on the order in which the rotor or sliders are enabled. The first rotor or slider enabled will use “rotor_slider_values[0]”, the second will use “rotor_slider_values[1]”, and so on. The reported rotor value is valid when the rotor is reported as being in detect.
- In case of QMatrix acquisition method library, the channels (*from_channel and to_channel*) can be between 3 to 8 channel numbers apart (i.e can support 3 to 8 channel rotors).
- In case of QTouch acquisition method library, the channels (*from_channel and to_channel*) can only be 3 channels apart(i.e can support only 3 channel rotors).

6.4.4 qt_enable_slider

This function is used to configure a set of channels as a rotor.

void qt_enable_slider (
channel_t from_channel ,
channel_t to_channel ,
aks_group_t aks_group ,
threshold_t detect_threshold ,
hysterisis_t detect_hysterisis ,
resolution_t position_resolution ,
uint8_t position_hysteresis
)

Arguments	Type	Comment
from_channel	Channel_t	Specifies the starting channel number to be configured for use as a “Slider”
to_channel	Channel_t	Specifies the end channel number to be configured for use as a “Slider”
aks_group	aks_group	Specifies the aks group associated with the sensor being configured as “Slider”
detect_threshold	threshold_t	Specifies the detect threshold for the sensor
detect_hysterisis	hysterisis_t	Specifies the detection hysteresis for the sensor
position_resolution	resolution_t	Specifies the resolution of the reported position value
position_hysterisis	uint8_t	Specifies the hysteresis of the reported position value

NOTE :

- A “Slider” sensor requires a contiguous numbers of channels.
- The rotor / slider number depends on the order in which the rotor or sliders are enabled. The first rotor or slider enabled will use “rotor_slider_values[0]”, the second will use. “rotor_slider_values[1]”, and so on. The reported rotor value is valid when the slider is reported as being in detect.
- In case of QMatrix acquisition method library, the channels (*from_channel and to_channel*) can be between 3 to 8 channel numbers apart (i.e can support 3 to 8 channel sliders).
- In case of QTouch acquisition method library, the channels (*from_channel and to_channel*) can only be 3 channels apart(i.e can support only 3 channel sliders).

6.4.5 qt_init_sensing

This function is used to initialize the touch sensing for all enabled channels. All required sensors should be configured before calling this function.

void qt_init_sensing (void)

Arguments	Type	Comment
Void	-	-

NOTE :

- All sensors must be configured (using *qt_enable_key , qt_enable_rotor or qt_enable_slider)* before calling this function.
- This functions initializes all the configured sensors, performs calibration.

6.4.6 qt_measure_sensors

This function performs a capacitive measurement on all enabled sensors. The measured signals for each sensor are then processed to check for user touches, releases, changes in rotor angle and changes in slider position.

unit8_t qt_measure_sensors(
uint16 current_time_ms
)

Arguments	Type	Comment
current_time_ms	uint16	The current time in milliseconds

Return Value	Comment		
uint8_t	Returns the status of the Library as a combination of the following bit fields.		
	Return value	Bit definition	Comments
	QTLIB_NO_ACTIVITY	0x00	No activity detected on any of the sensors
	QTLIB_IN_DETECT	0x01	At least one sensor is in detect
	QTLIB_STATUS_CHANGE	0x02	At least one sensor has changed ON/OFF state since the last call to *qt_measure_sensor()*
	QTLIB_ROTOR_SLIDER_POS_CHANGE	0x04	At least one rotor/slider has changed position since the last call to *qt_measure_sensors()*
	QTLIB_CHANNEL_REF_CHANGE	0x08	At least one reference value has changed since last call to *qt_measure_sensors()*

NOTE :

- All sensors must be configured (using *qt_enable_key or qt_enable_rotor or qt_enable_slider)* and initialized by calling *qt_init_sensing* before calling this function.

6.4.7 qt_calibrate_sensing

This function forces a recalibration of all enabled sensors.

void qt_calibrate_sensing(
void
)

Arguments	Type	Comment
Void	-	-

NOTE :

- Recalibration may be useful if, for example, it is desired to globally recalibrate all sensors on a change in application operating mode.
- This function must be called only when the sensors have been configured and initialized.

6.4.8 qt_reset_sensing

This function disables all sensors and resets all configuration settings (for example, “qt_di”) to their default values.

void qt_reset_sensing(
void
)

Arguments	Type	Comment
Void	-	-

NOTE :

- This may be useful if it is desired to dynamically reconfigure sensing. After calling this function, any required sensors must be re-enabled, and “qt_init_sensing()” must be called before “qt_measure_sensors()” is called again.

6.4.9 qt_get_sensor_delta

This function returns the delta value for a given channel .

int16_t qt_get_sensor_delta(
uint8_t sensor_number
)

Arguments	Type	Comment
sensor_number	unit8_t	sensor id for which the delta is required

Return type	Comment
int16_t	The delta value of the sensor specified

NOTE :

- All sensors must be configured (using *qt_enable_key or qt_enable_rotor or qt_enable_slider)* and initialized by calling *qt_init_sensing* before calling this function.

6.4.10 qt_get_library_sig

This function is used to retrieve the library version and signature from the library.

void qt_get_library_sig(

*qt_lib_siginfo_t *lib_sig_ptr*

)

Arguments	Type	Comment
lib_sig_ptr	qt_lib_siginfo_t *	Pointer to the structure which needs to be updated with the library signature information

NOTE :

- The function *qt_measure_sensors()* should have been called at least once prior to calling this function.

6.5 Overview of Touch API Usage and sequence

Figure 1 illustrates the sequence of operations required to be performed to add touch to an end application. By using the simple API's as illustrated in the sequence below, the user can add touch sensing in his design.

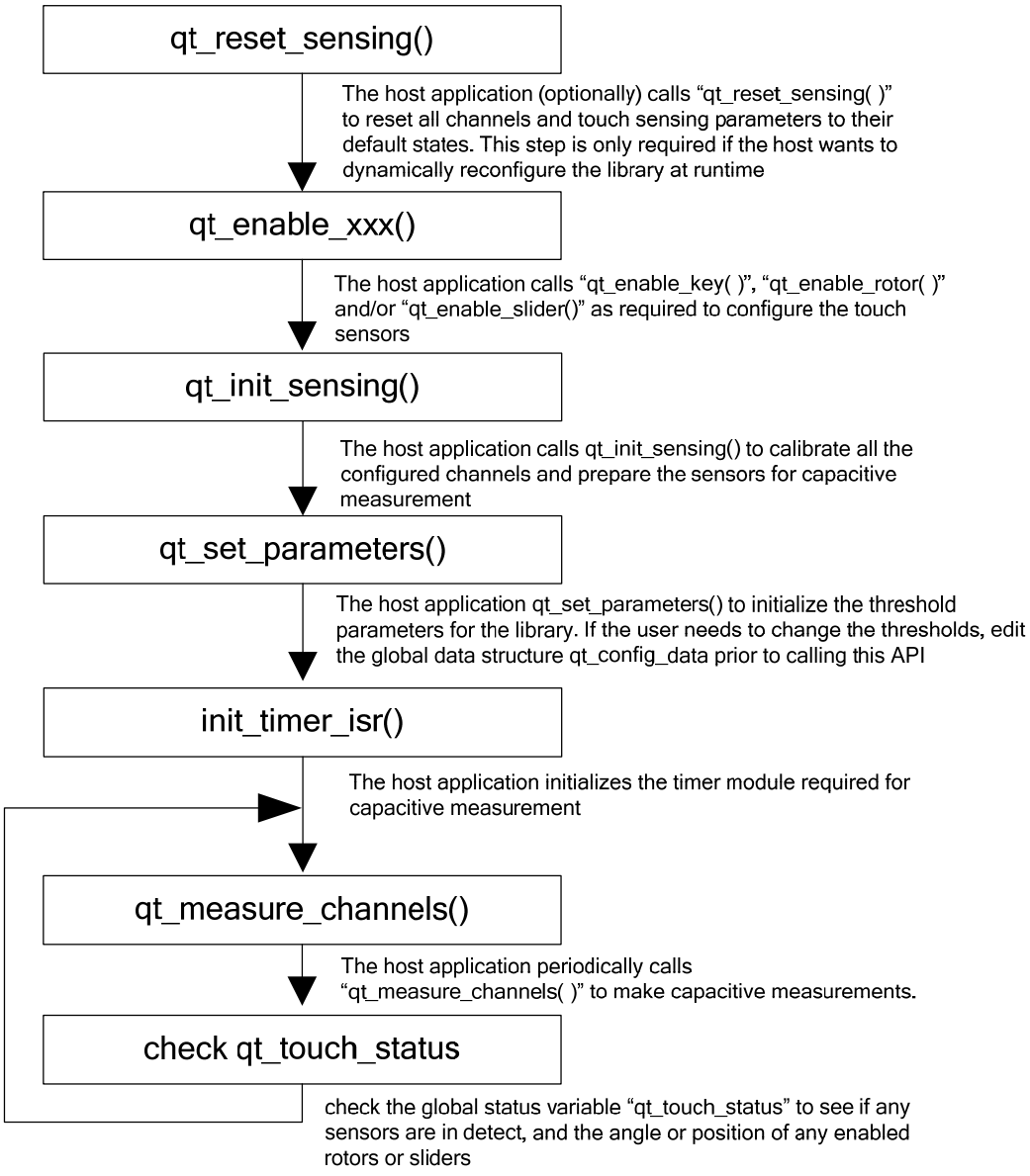


Figure 6 : Sequence of operations to add Touch capability

6.5.1 Channel Numbering

6.5.1.1 Channel numbering when using QTouch acquisition method

The choice of ports for routing the SNS and SNSK lines are determined by the availability of the QTouch acquisition method libraries which support the required port to be used to route SNS or SNSK lines.

There are two options provided for connecting the SNS and SNSK pins.

1. The SNS and SNSK pins are connected to separate ports.
2. The SNS and SNSK pins are connected to the same port.

6.5.1.1.1 Channel numbering when routing SNS and SNSK pins to different ports

Figure 7 illustrates a sample QTouch capacitive sensing solution which uses two ports on a device for routing the SNS and SNSK lines required.

When SNS and SNSK pins are connected to different ports, the channel numbering follows the pin numbering in the ports selected.

- The channel numbers follow the pin numbers starting with the LSB (pin 0 is channel 0 and pin 7 is channel 7).
- When a library and corresponding device and ports requires the use of more than two ports for SNS and SNSK pins, the channel numbers in the second set of SNS/SNSK port pair continue from the preceding pair as illustrated in Figure 7(pin 0 of next port pair is channel 8 and pin 7 of the next port pair is channel 15).
- Support for more than one pair of SNS and SNSK ports are not available for UC3™ devices.
- Since the channel numbers are fixed to the pins of the SNS and SNSK ports, if the design calls for use of a subset of the pins available in the SNS and SNSK ports, the user has to skip the channel numbers of the unused SNS and SNSK pins.
    - For example, on a 8 channel configuration using a single pair of SNS and SNSK ports, if pin 2 is not used for touch sensing (on both SNS and SNSK ports), channel number 2 is unavailable and care should be taken while configuring the channels and sensors to avoid using this channel.

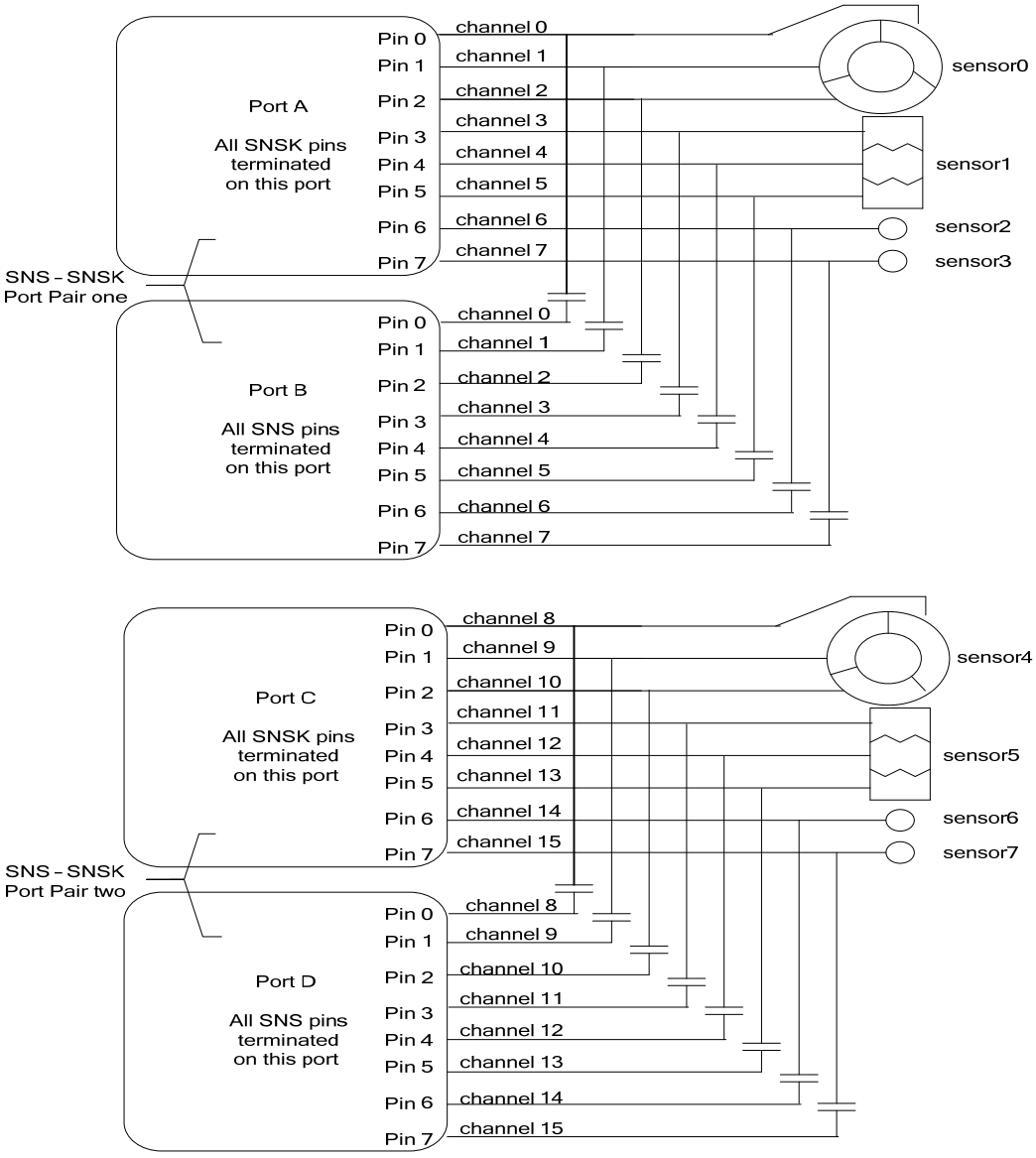


Figure 7 : channel numbering for QTouch acquisition method when the SNS and SNSK pins are connected to different ports.

6.5.1.1.2 Channel numbering when routing SNS and SNSK pins to the same port

When SNS and SNSK pins are connected to the same port, the even pin numbers will be used as SNS pins and the odd pins will be used as the SNSK pins.

- The number of channels supported will be limited 4 channels for an 8 bit device and 16 channels for a 32 bit device (e.g UC3).
- For e.g, for a 4 channel configuration where the SNS and SNSK pins are connected to Port B, the port pins 0&1 are used for channel 0.
- The channel number is derived from the position of the pins used for SNS and SNSK lines for any channel.

  channel number = floor([SNS(or SNSK) pin number] / 2)

  - For e.g, pins 4 and 5 are connected to a SNS/SNSK pair and the channel number associated with the SNS/SNSK pin is 2.

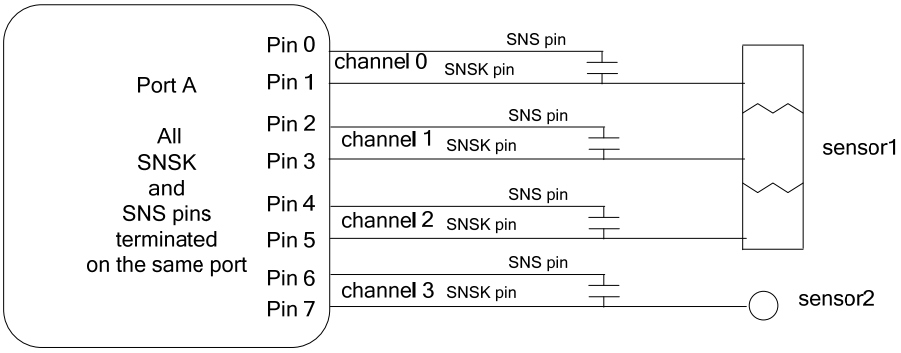


Figure 8 : Channel numbering for QTouch acquisition method when the SNS and SNSK pins are connected to the same port

6.5.1.2 Channel numbering when using QMatrix acquisition method

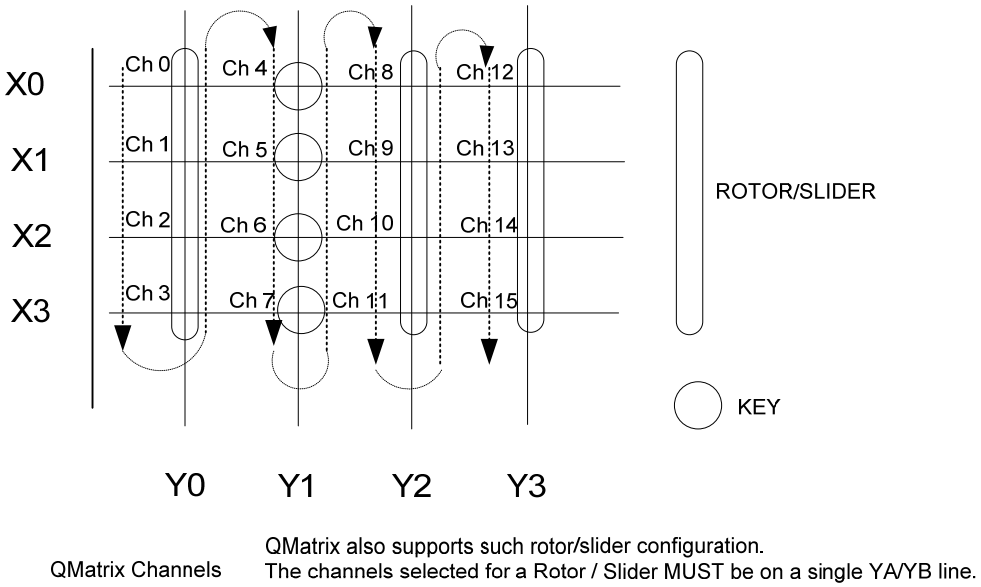


Figure 9 :Channel Numbering for QMatrix acquisition method libraries

Figure 9 illustrates a QMatrix capacitive sensing solution which uses 4 X lines and 4 Y lines thereby providing a 16 channel solution.

Note : The choice of ports for X and Y lines are determined by the availability of the QMatirx™ acquisition method libraries which support the required port to be used.

The channel numbering for QMatrix configuration follows a matrix pattern with the channel numbers starting from 0 for the matrix intersection (X0Y0) and increasing along the X lines for a given Y line (Channel 1 is X1Y0) and then moving on to the row number 0 for the next column. Table 1 lists the possible channel numbers and the associated X/Y line associations for the different configurations of QMatrix library variants.

A group of channels form a sensor and the sensor numbering is determined by the order in which the user defines the association of channels and uses them as a sensor.

The channel numbering is fixed for a specific library variant based on the number of X and Y lines used whereas the sensor numbering is determined at the time of usage based on the order in which the user defines the association of the channels to create a sensor.

NOTE:

- All channels selected for a specific rotor or slider should be on a single Y line.

Table 1 : Channel numbers for QMatrix configurations

Line label	8-channel configuration (4 x2)	16 channel Configuration (8 x 2)	16 channel Configuration (4 x 4)	32 channel configuration (8 x 4)	64 channel configuration (8 x 8)
Channel 0	X0Y0	X0Y0	X0Y0	X0Y0	X0Y0
Channel 1	X1Y0	X1Y0	X1Y0	X1Y0	X1Y0
Channel 2	X2Y0	X2Y0	X2Y0	X2Y0	X2Y0
Channel 3	X3Y0	X3Y0	X3Y0	X3Y0	X3Y0
Channel 4	X0Y1	X4Y0	X0Y1	X4Y0	X4Y0
Channel 5	X1Y1	X5Y0	X1Y1	X5Y0	X5Y0
Channel 6	X2Y1	X6Y0	X2Y1	X6Y0	X6Y0
Channel 7	X3Y1	X7Y0	X3Y1	X7Y0	X7Y0
Channel 8	N/A	X0Y1	X0Y2	X0Y1	X0Y1
Channel 9	N/A	X1Y1	X1Y2	X1Y1	X1Y1
Channel 10	N/A	X2Y1	X2Y2	X2Y1	X2Y1
Channel 11	N/A	X3Y1	X3Y2	X3Y1	X3Y1
Channel 12	N/A	X4Y1	X0Y3	X4Y1	X4Y1
Channel 13	N/A	X5Y1	X1Y3	X5Y1	X5Y1
Channel 14	N/A	X6Y1	X2Y3	X6Y1	X6Y1
Channel 15	N/A	X7Y1	X3Y3	X7Y1	X7Y1
Channel 16	N/A	N/A	N/A	X0Y2	X0Y2
Channel 17	N/A	N/A	N/A	X1Y2	X1Y2
Channel 18	N/A	N/A	N/A	X2Y2	X2Y2
Channel 19	N/A	N/A	N/A	X3Y2	X3Y2
Channel 20	N/A	N/A	N/A	X4Y2	X4Y2
Channel 21	N/A	N/A	N/A	X5Y2	X5Y2
Channel 22	N/A	N/A	N/A	X6Y2	X6Y2
Channel 23	N/A	N/A	N/A	X7Y2	X7Y2
Channel 24	N/A	N/A	N/A	X0Y3	X0Y3
Channel 25	N/A	N/A	N/A	X1Y3	X1Y3
Channel 26	N/A	N/A	N/A	X2Y3	X2Y3
Channel 27	N/A	N/A	N/A	X3Y3	X3Y3
Channel 28	N/A	N/A	N/A	X4Y3	X4Y3
Channel 29	N/A	N/A	N/A	X5Y3	X5Y3
Channel 30	N/A	N/A	N/A	X6Y3	X6Y3
Channel 31	N/A	N/A	N/A	X7Y3	X7Y3
Channel 32	N/A	N/A	N/A	N/A	X0Y4
Channel 33	N/A	N/A	N/A	N/A	X1Y4
Channel 34	N/A	N/A	N/A	N/A	X2Y4
Channel 35	N/A	N/A	N/A	N/A	X3Y4
Channel 36	N/A	N/A	N/A	N/A	X4Y4
Channel 37	N/A	N/A	N/A	N/A	X5Y4
Channel 38	N/A	N/A	N/A	N/A	X6Y4
Channel 39	N/A	N/A	N/A	N/A	X7Y4
Channel 40	N/A	N/A	N/A	N/A	X0Y5
Channel 41	N/A	N/A	N/A	N/A	X1Y5
Channel 42	N/A	N/A	N/A	N/A	X2Y5
Channel 43	N/A	N/A	N/A	N/A	X3Y5
Channel 44	N/A	N/A	N/A	N/A	X4Y5
Channel 45	N/A	N/A	N/A	N/A	X5Y5
Channel 46	N/A	N/A	N/A	N/A	X6Y5
Channel 47	N/A	N/A	N/A	N/A	X7Y5
Channel 48	N/A	N/A	N/A	N/A	X0Y6

Channel 49	N/A	N/A	N/A	N/A	X1Y6
Channel 50	N/A	N/A	N/A	N/A	X2Y6
Channel 51	N/A	N/A	N/A	N/A	X3Y6
Channel 52	N/A	N/A	N/A	N/A	X4Y6
Channel 53	N/A	N/A	N/A	N/A	X5Y6
Channel 54	N/A	N/A	N/A	N/A	X6Y6
Channel 55	N/A	N/A	N/A	N/A	X7Y6
Channel 56	N/A	N/A	N/A	N/A	X0Y7
Channel 57	N/A	N/A	N/A	N/A	X1Y7
Channel 58	N/A	N/A	N/A	N/A	X2Y7
Channel 59	N/A	N/A	N/A	N/A	X3Y7
Channel 60	N/A	N/A	N/A	N/A	X4Y7
Channel 61	N/A	N/A	N/A	N/A	X5Y7
Channel 62	N/A	N/A	N/A	N/A	X6Y7
Channel 63	N/A	N/A	N/A	N/A	X7Y7

6.5.2 Sensor Numbering

The ordering and numbering of sensors is related to the order in which the sensors are enabled. This is independent of the acquisition method (QMatrix or QTouch acquisition method libraries).

For example, consider this code snippet:

….

```
/* enable slider */
qt_enable_slider( CHANNEL_0, CHANNEL_2, AKS_GROUP_1, 16, HYST_6_25, RES_8_BIT, 0);

/* enable rotor */
qt_enable_rotor( CHANNEL_3, CHANNEL_5, AKS_GROUP_1, 16, HYST_6_25, RES_8_BIT, 0);

/* enable keys */
qt_enable_key( CHANNEL_6, AKS_GROUP_2, 10, HYST_6_25 );
qt_enable_key( CHANNEL_7, AKS_GROUP_2, 10, HYST_6_25 );
```

In the case above, the slider on channels 0 to 2 will be sensor 0, the rotor on channels 3-to-5 is sensor 1 and the keys on channels 6 and 7 are sensor numbers 3 and 4 respectively.

When the touch status is reported or queried, the corresponding sensor positions and status indicate the touch status. For example, the slider is in detect if "*qt_measure_data.qt_touch_status.sensor_states*" bit position 0 is set. Similarly, the rotor on channels 3 to 5 is sensor 1, and the keys on channels 6 and 7 are sensors 2 and 3 respectively.
However, the code could be re-arranged as follows to give a different sensor numbering.

```
/* enable rotor */
qt_enable_rotor (CHANNEL_3, CHANNEL_5, NO_AKS_GROUP, 16, HYST_6_25, RES_8_BIT,
0);

/* enable keys */
qt_enable_key (CHANNEL_6, AKS_GROUP_2, 10, HYST_6_25);
qt_enable_key (CHANNEL_7, AKS_GROUP_2, 10, HYST_6_25);

 /* enable slider */
```

qt_enable_slider (CHANNEL_0, CHANNEL_2, NO_AKS_GROUP, 16, HYST_6_25, RES_8_BIT, 0);

Now, the rotor is sensor 0, the keys are sensors 1 and 2, and the slider is sensor 3.

So, the order in which the user enables the sensors is the order in which the sensors are numbered. Depending on the user requirements, the sensors can be configured in the preferred order.

6.5.3 Using the Sensors

6.5.3.1 Avoiding Cross-talk

In ATMEL QTouch library variants that use QTouch acquisition technology, adjacent sensors are not measured at the same time. This prevents interference due to cross-talk between adjacent channels, but at the same time some sensor configurations take longer to measure than others.

For example, if an 8-channel device is configured to support 8 keys, then the library will measure the keys on channels 0, 2, 4, and 6 simultaneously, followed by keys on channels 1, 3, 5, and 7. If the same device is configured, say, to support 4 keys, putting them either on all the odd channels or on all the even channels means that they can all be measured simultaneously.

This means the library calls are faster, and the device can use less power.
So, it is recommended that the appropriate channel numbers are used when using less than the maximum number of channels available for the device to ensure optimum performance.

6.5.3.2 Detect Integration

The QTouch Libraries feature a detection integration mechanism, which acts to confirm detection in a robust fashion. A per-sensor counter is incremented each time the sensor has exceeded its threshold and stayed there for a number of acquisitions. When this counter reaches a preset limit the sensor is finally declared to be touched.

For example, if the limit value is 10, then the device has to exceed its threshold and stay there for 10 acquisitions in succession without going below the threshold level, before the sensor is declared to be touched. If on any acquisition the signal is not seen to exceed the threshold level, the counter is cleared and the process has to start from the beginning. The user can configure the DI limit for a sensor.

6.5.3.3 Adjacent Key Suppression

In designs where the sensors are close together or set for high sensitivity, multiple sensors might report detect simultaneously if touch is near them. To allow applications to determine the intended single touch, the touch library provides the user the ability to configure a certain number of sensors in an AKS™ group.

When a group of sensors are in the same AKS™ group, then only the first strongest sensor will report detection. The sensor reporting detection will continue to report detection even if another sensor's delta becomes stronger. The sensor stays in detect until its delta falls below its detection threshold, and then if any more sensors in the AKS™ group are still in detect then the strongest will report detection. So at any given time only one sensor from each AKS™ group will be reported to be in detect.

The library provides the ability to configure any sensor to be included in any one of the Adjacent Key Suppression Groups (AKS Group).

The AKS group number for a sensor is specified when the user enables the sensor using the API *qt_enable_(sensor/rotor/slider)()*

6.5.3.4 Multiple measurements

The library will automatically perform multiple measurements on a sensor in certain conditions,. These include the following situations

- When sensors are being calibrated
- When a sensor is being filtered while going into or coming out of detect (DI=Chattering/Debounce)

In such situations, multiple measurements are performed by the touch library before returning to the host application. This implies that some of the calls to perform sensor measurements might take a little bit longer.

6.5.3.5 Measurement Limit

In ATMEL QTouch library variants that use QTouch capacitive sensing, measurements on a channel are automatically stopped when they reach a value of 8192 pulses. In this case a signal level of 1 will be reported for the channel. This limit is long enough for practical uses of QTouch sensing, and traps hardware fault conditions such as shorted-out sampling capacitors.

6.5.3.6 Filtering Signal Measurements

The ATMEL QTouch Library API provides a function pointer called “qt_filter_callback”. The user can use this hook to apply filter functions to the measured signal values.
If the pointer is non-NULL, the library calls the function after library has made capacitive channel measurements, but before the library has processed the channel information and determining the sensor states.

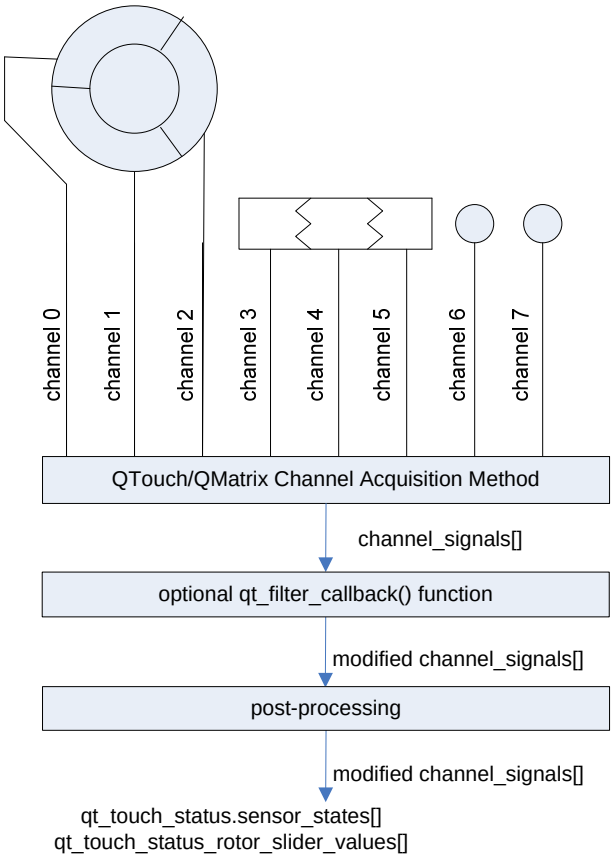


Figure 10 : Block diagram to represent usage of filter callback function

6.5.3.6.1 Example: Averaging the Last Four Signal Values

1. Add a static variable in the main module:

```
/* filter for channel signals */
static uint16_t filter[QT_NUM_CHANNELS][4];
```

2. Add a filter function prototype to the main module:

```
/* example signal filtering function */
static void filter_data_mean_4( void );
```

3. When configuring the ATMEL QTouch library, set the callback function pointer:

```
/* set callback function */
qt_filter_callback = filter_data_mean_4;
```

4. Add the filter function:

```
void filter_data_mean_4( void )
{
uint8_t i;
/*
* Shift previously stored channel signal data.
* Store new channel signal data.
* Set library channel signal data = mean of last 4 values.
*/

for( i = 0u; i < QT_NUM_CHANNELS; i++ )
        {
        filter[i][0] = filter[i][1];
        filter[i][1] = filter[i][2];
        filter[i][2] = filter[i][3];
        filter[i][3] = qt_measure_data.channel_signals[i];
        qt_measure_data.channel_signals[i] = ( (        filter[i][0] +
        filter[i][1] +
        filter[i][2] +
        filter[i][3] ) / 4u );
        }
}
```

The signal values processed by the ATMEL QTouch Library are now the mean of the last four actual signal values.

6.6 Constraints

6.6.1 QTouch acquisition method constraints

QTouch acquisition method libraries are available for a set of AVRs and port combinations.

These port combinations have fixed port assignments for SNS and SNSK pins required.

Some of the key constraints are

- Rotors/sliders have to be connected on three adjacent channels. (e.g. (1,2,3) or (3,4,5) …) within the same port. Possible combinations are (0,1,2), (1,2,3) for a configuration which supports 4 channels. Possible combinations (0,1,2), (1,2,3), (2,3,4), (3,4,5), (4,5,6), (5,6,7) for a configuration which supports 8 channels.

- If two port pairs are used for the design, all the channels for a sensor have to be connected on a single port pair. Combining channels from multiple ports is not possible when designing sensors. e.g It is not possible to have a rotor with channel numbers (7,8,9) on a 16 channel library variant which uses two port-pairs.

6.6.2 QMatrix acquisition method API constraints

QMatrix acquisition method libraries are available for a set of AVRs and port combinations.

These port combinations have fixed port and pin assignments for X, Ya, Yb and SMP / Discharge pin lines.

Some of the key constraints are

- The QMatrix acquisition method libraries internally use TIMER1 for the operation, TIMER1 will not be available for critical sections of the code where the library is called. But resources are available to the host application when the normal user's application is running.
- In case of XMEGA™ devices, the resources are used internal to the library and hence cannot be used by the host application
    - o Timer/Counter 1 on PORTC (TCC1)
    - o Analog Comparator on PORTA (ACA)
    - o Event System Channel0 (EVSYS_CH0)
- The sensor channel number and the relation with X and Y lines strictly follows from the table provided in the section Table 1.
- A rotor /slider sensor can be configured with 3 to 8 channels per rotor or slider depending on the requirement of the application subject to the total number of channels available in the library variant selected as listed below.

Number of channels	X x Y	Maximum Channels per ROTOR_SLIDER
8	4 x 2	4
16	4 x 4	4
16	8 x 2	8
32	8 x 4	8
64	8 x 8	8

- For example, 16 channel library with 4X and 4Y lines supports maximum of 4 channels per Rotor/Slider. But, a 16 channel with 8X and 2Y lines supports maximum of 8 channels per Rotor/Slider.
- If the lines of the Drive and Receive electrode (X lines or the Y lines) share the same lines with the JTAG, JTAG needs to be disabled. Please check the data sheet to ensure that there are no conflicts between the X/Y lines and JTAG lines used for the device.
- YB line for a particular device cannot be changed and it has to be the configured to be the ADC port of the selected device.
- The AIN0 pin of the device needs to be connected to the GND.
- In case of XMEGA devices, the reference pin for input to analog comparator is Pin7 of PORTA with all the combinations of libraries supported. Hence, this needs to be connected to GND
- Proper grounding should be taken care when the controller board and touch sensing board are different.
- The channels used for an individual rotor or slider should all be on the same Y line.
- The maximum number of Rotors / Sliders supported by the QMatrix acquisition method depends on the configuration. See section 7.2.4.2 for details.

6.7 Frequency of operation (Vs) Charge cycle/dwell cycle times:

The library needs different charge/ dwell cycles based on the operation and design. The recommended range of charge/dwell cycle times that the user must select based on the operating clock frequency of the Microcontroller is provided in the table below.

Fine tuning of the cycle times to match the sensor design may be done by monitoring the reference levels, and finding the shortest cycle time where the reference level has reached >98% of maximum reference value seen with a much longer cycle time. If the cycle time is too short the design may experience temperature sensitivity.

Example: When operating at 4 MHz, 1~10 cycle charge times are recommended (0.125us to 1.25us). So, when operating at 4MHz the user can update delay cycles with any of 1,2,3,4,5,10.

Table 2 : Frequency of operation

Frequency of Microcontroller (MHz))	microcontroller Cycle time (us)	Suitable Charge Cycle times (or) Suitable Dwell Cycle times (us)
1	1	1~2 cycles (1us to 2us)
2	0.5	1~5 cycles (0.5us to 2.5us)
4	0.25	1~10 cycles (0.25us to 2.5us)
8	0.125	1~10 cycles (0.125us to 1.25us)
10	0.1	2~25 cycles (0.2us to 2.5us)
16	0.0625	2~25 cycles (0.125us to 1.5625us)
20	0.05	3~50 cycles (0.15us to 2.5us)

Note :

- For UC3 devices,
  - 1 charge cycle delay time is not supported when using IAR IDE
  - 1 & 2 charge cycle delay times are not supported when using GCC compiler tool chain

If the AVR is only used for Touch detection then running at the lowest frequency possible for the desired touch response may provide the best power and EMC performance. If the AVR is also used for other functions then running at a higher frequency may be necessary. In some power critical applications it may be worth switching the frequency on the fly, such as lowering the frequency during touch detect API instead of using long cycle times, and then switching to a higher frequency for non-touch code. It is necessary to carefully design timer operation when change frequencies

6.8 Interrupts

The library disables interrupts for time-critical periods during touch sensing. These periods are generally only a few cycles long, and so host application interrupts should remain responsive during touch sensing. However, any interrupt service routines (ISRs) during touch sensing should be as short as possible to avoid affecting the touch measurements or the application responsiveness. As a rule of thumb, the combined durations of any ISRs during a capacitive measurement should be less than 1 ms. This can be tested during system development by checking the burst duration on the touch channels on an oscilloscope. If the total burst duration for any channel varies by more than 1ms while the user is not touching any sensors, then ISRs could adversely affect the measurements.
Please note that none of the API functions should be called from a user interrupt.

6.9 Integrating QTouch libraries in your application

This section illustrates the key steps required in integrating the QTouch library in your application.

6.9.1 Directory structure of the library files

The QTouch library directory structure is as listed below

What	Where			Comments
Root installation	Default directory is C:\Program Files\Atmel\Atmel_QTouch_Libraries_3.2\			
Header file	..\include			touch_api.h is located in this directory
Library files	QTouch acquisition method libraries	8-bit devices	..\Atmel_QTouch_Libraries_3.2\AVR_Tiny_Mega_XMega\QTouch\library_files	
		UC3	..\Atmel_QTouch_Libraries_3.2\32bit_AVR\UC3\QTouch\library_files	
	QMatrix acquisition method libraries	..\Atmel_QTouch_Libraries_3.2\AVR_Tiny_Mega_XMega\QMatrix\library_files		
Example Projects	QTouch acquisition method libraries	8-bit devices	..\Atmel_QTouch_Libraries_3.2\AVR_Tiny_Mega_XMega\QTouch\example_projects	
		UC3	..\Atmel_QTouch_Libraries_3.2\32bit_AVR\UC3\QTouch\example_projects	
	QMatrix acquisition method libraries	..\Atmel_QTouch_Libraries_3.2\AVR_Tiny_Mega_XMega\QMatrix\example_projects		

6.9.2 Integrating QTouch acquisition method libraries in your application

The following steps illustrate how to add QTouch acquisition method support in your application.

1) QTouch acquisition method library Variants are offered for IAR and AVR-GCC tool chains. First step is to select the compiler tool chain for which the libraries are required. The list of supported compiler tool chains can be found in 7.1.2.
2) Select the QTouch acquisition method library variant required for the device.
    a. There are specific library variants distributed for each microcontroller. You would need the following parameters to identify the right library variant to be used in your application
        i. The microcontroller to be used for the application.
        ii. The number of channels you need for the application.
        iii. The Ports on which the SNS and SNSK Pins for QTouch acquisition will be used in the selected microcontroller.
        iv. Whether Rotor and/or Slider support required in the application.
        v. The charge cycle duration required for the QTouch acquisition method.
    b. There are specific variants of the library which is pre-built with a specific configuration set supported. Section 7.1.4 lists the different variants of the QTouch acquisition method library variants. Select the library variant which matches the requirement above from the list of supported library variants specified in section 7.1.4.
3) Define the constants and symbol names required
    a. The next step is to define certain constants and symbols required in the host application files where the touch API is going to be used. These values are derived from the parameters defined in step 2-a for your application and the selected library variant in step 2-b above.
    b. The constant/symbol names are as listed in the table below.
    c. The constant/symbol definitions can be placed in any of the following.
        i. In the user's 'C' file prior to include touch_api.h in the file.

ii. Defined user's project options.
iii. Placed in a separate header file and include this prior to touch_api.h in the application file where you need to use the touch APIs.

Table 3 : Constant and symbol name definitions required to use the QTouch acquisition method libraries

Symbol / Constant name	Range of values	Comments
QTOUCH	This macro has to be defined in order to use QTouch libraries.	
SNS & SNSK	As defined in the list of supported ports for the device selected in section 7.1.5.3.1.	To be used if only single port pair is needed for the design.
SNS1 – SNSK1 & SNS2 – SNSK2	As defined in the list of supported ports for the device selected in section 7.1.5.3.	To be used if two port pairs are needed for the design.
QT_NUM_CHANNELS	4, 8, 16 for tinyAVR, megaAVR and XMEGA device libraries and 8, 16, 32 for UC3 device libraries.	
_ROTOR_SLIDER_	Rotor / slider can be added to the design, if this macro is enabled.	A library with rotor / slider functionality already available needs to be selected if this macro is to be enabled.
QT_DELAY_CYCLES	1, 2, 3, 4, 5, 10, 25, 50	Please refer to section 6.7.

4) Using QTouch API's in your application to add touch functionality
   a. The clock, host application and other peripherals needed by the host application needs to be initialized.
   b. In your application, create, initialize and configure the sensors.
      i. The APIs of interest are qt_enable_key/rotor/slider().see sections 6.4.2, 6.4.3 and 6.4.4.
   c. The channel configuration parameters need to be set by calling the *qt_set_parameters()* (see section 6.4.1.
   d. Once the sensors are configured, qt_init_sensing() has to be called to trigger the initialization of the sensors with the configuration defined in steps above.
   d. Provide timing for the QTouch libraries to operate. i,e the QTouch libraries do not use any timer resources of the microcontroller. The Host application has to provide the required timing and also call the API's at the appropriate intervals to perform touch sense detect operations.
   e. NOTE : The example applications provided with the libraries illustrate the usage for the evaluation kits supported by the library. Please refer to the source files for reference.
5) General application notes
   - The clock, host application and other peripherals needed by the host application needs to be initialized.
   - Ensure that there are no conflicts between the resources used by the touch library and the host application.
   - Ensure that the stack size for your application is adjusted to factor in the stack depth required for the operation of the touch libraries.

6.9.2.1 Example

The example below will explain in detail the steps to follow for library selection.

Criteria	Selection		Notes
Microcontroller	ATMega1281		
IDE and compiler tool chain used	AVR STUDIO® IDE and GNU compiler		The GCC compiled variant of the libraries for the device selected needs to be used.
Number of Keys required for the application	3		Each key requires 1 QTouch acquisition channel
Rotors and sliders required	Yes		
Number of Rotors and Sliders required	3		Each rotor / slider will require 3 channels.
Number of Channels required for the application (should be the sum of all channels required for all the keys ,rotors and sliders used in the design)	12		3 Keys + (3 rotors x 3 channels per rotor/slider) → 12 channels
Charge cycle time required for the design	5 cycles		Assuming the device is configured with a clock frequency of 8Mhz
Number of SNS/SNSK port pairs needed	2 pairs		This is determined based on the number of channels required and the routing required for the channels SNS and SNSK pins to the ports For this design, since more than 8 pins are required, we need 2 pairs of ports to support the required number of ports
Choice of ports available for the design	SNS/SNSK Pair1 port	SNS1 Port : D	The choice of ports for the port pairs is limited and can be found in the section 7.1.5
		SNSK1 Port : B	
	SNS/SNSK Pair 2 port	SNS2 Port : C	
		SNSK2 Port : A	

Given the above requirements for the applications, the first step is to select the right library variant required.

Step 1 : Selecting the right library variant

By following the guidelines listed for library selection in 7.1.4, we see that there are a few variants of libraries supported for ATmega1281. Since the application requires 12 channels and rotor slider support, one has to select a library variant which supports at least 12 channels or more along with Rotors/Slider functionality. Hence we select the 16 channel library variant for GCC complier which supports the required number of sensors/channels. This works out to be `libavr51g2_16qt_k_4rs.a`

Step 2 : Defining the constants / symbols in the project space
In the host application file (say main.c), define the following constants and symbols

```
#define _QTOUCH_
#define QT_NUM_CHANNELS     16
#define SNSK1               B
#define SNS1                D
#define SNSK2               A
#define SNS2                C
```

#define QT_DELAY_CYCLES 1

NOTE : The above definitions should be placed before including "touch_api.h" in your files.

Alternatively, you can define these in your IDE's project options or have them defined in a separate header file.

Step3 : Usage of library API's

Now, you can use the touch API's to create, initialize and perform touch sensing. Please refer to the sample applications in section 6.10.2 for reference. These sample applications illustrate the usage of the API's and the sequence of operation.

6.9.2.2 Checklist of items for integrating QTouch acquisition method libraries

The following is a checklist of items which needs to be ensured when integrating QTouch acquisition method libraries

- The clock prescaler register (CLKPR, XDIV) needs to be configured correctly based on the device selected. Some devices have clock frequency selection based on fuses. It has to be ensured the fuses are set correctly in such cases.
- It is recommended to disable PULL-UP resistor on all port pins used for touch sensing on the device selected (e.g. PUD bit in MCUCR, SFIOR for a few of the tinyAVR and megaAVR devices Please refer to the Data sheet of the selected device).
- The 16 bit timer in each device has been used for performing touch measurements periodically. The datasheet for all the devices have to be checked to ensure that the correct timer peripheral and its registers are used (file: main.c).
- The interrupt vector macro may also change from device to device and this needs to be verified in the datasheet for the device used.
- Check if the timer is configured correctly to support the measurement period needed (e.g. 25msec or 50 msec).
- The sample applications for the evaluation kits and supported devices illustrate the proper initialization sequence and usage of the timer resources (file: main.c). Please use this as a reference for your application design.

The host application must provide the current time to the library. This information is passed to the library as an argument to the function qt_measure_sensors()". This is used for time-based library operations such as drift compensation.

6.9.3 Integrating QMatrix acquisition method libraries in your application

Based on your application design needs, the user needs to select the right library variant and the configuration to be used along with the variant. This section illustrates the steps required to select the right QMatrix acquisition method library variant and configuration for your application.

For your design, you would need the following information to select the correct library variant

a. Device to be used for the design
b. Maximum number of channels required for the design
c. Number of X lines to be used in the design
    a. The ports on which your design permits to have the X lines
d. Number of Y lines to be used in the design
    a. The ports on which your design permits to have the Y lines
    b. Can you have the Y line pins routed to lower pin numbers (pins 0 to 3) or higher pin numbers (pins 4 to 7) on the selected port
e. Do you need support for Rotors and/or Sliders in your design
    a. If yes, How many
f. Which compiler platform you intend to use to integrate the libraries

Follow the steps listed below to arrive at the right library variant

1) Select the device from the list of supported devices listed in 7.2.4.1
2) Select the right library variant for the device selected in step 1 by using the tables listed in section 7.2.4.2. Each variant supports
    a. a specific number of channels,
    b. Supports a specific configuration of X x Y matrix pins (eg 4 X pins x 2 Y pins)
    c. Can support the Y lines on specific pin numbers
    d. has support for Rotor / Slider (either supported or not)
    e. support is available for IAR and/or GCC compiler tool chain
    f. support for specific number of rotors sliders.
    g. supports from 3 to 8 channels per rotor/slider.
3) Once the library variant is selected, you need to specify the right configuration to specify when using the library as the library variant can support a few configurations. Select the right configuration which matches your needs from the list of supported configurations for each device in section 7.2.4.3.
4) Define the constants and symbol names required
    a. The next step is to define certain constants and symbols required in the host application files where the touch APIs are going to be used.
    b. The constant/symbol names are as listed in the table below
    c. The constant/symbol definitions can be placed in any of the following
        i. In the user's 'C' file prior to include touch_api.h in the file
        ii. Defined user's project options
        iii. Placed in a separate header file and include this prior to touch_api.h in the application file where you need to use the touch APIs.

Table 4 :List of configurable parameters for touch library usage.

Symbol / Constant name	Range of values	Comments
QMATRIX	Symbol defined to indicate QMatrix acquisition method is required	Define this symbol to indicate QMatrix acquisition method is required
PORT_X	As defined in the port configurations supported for the device selected in section 7.2.4.3	Drive electrode for touch sensing using QMatrix acquisition
PORT_YA	As defined in the port configurations supported for the device selected in section 7.2.4.3	Drive electrode for touch sensing using QMatrix acquisition
PORT_YB	As defined in the port configurations supported for the device selected in section 7.2.4.3	Drive electrode for touch sensing using QMatrix acquisition
PORT_SMP	As defined in the port configurations supported for the device selected in section 7.2.4.3	Drive electrode for touch sensing using QMatrix acquisition
SMP_BIT	As defined in the port configurations supported for the device selected in section 7.2.4.3	Drive electrode for touch sensing using QMatrix acquisition
_ROTOR_SLIDER_	Symbol defined if Rotor and/or slider is required	Needs to be added in case user needs to configure ROTOR/SLIDER Needs to be removed for ALL KEYS configuration
DELAY_CYCLES	1,2,3,4,5,10,25,50	Please refer to section 6.7
QT_MAX_NUM_ROTORS_SLIDERS	0,2,4,8	Subject to support for rotors/sliders in the library selected. See section 7.2.4.2

ATXMEGA	Symbol defined if an XMEGA Device is used for QMatrix sensing technology	Needs to be added if the device to be supported is ATxmegaxxxx

Once you have selected the right library variant and configuration parameters for the application, follow the steps outlined below to integrate the library variant in your application.

5) Copy the library variant that was selected in step one to your project's working directory or update your project to point to the library selected.
6) Include the "touch_api.h" header file from the QTouch library in your application. The touch_api.h can be found at in the release package
7) Initialize/create and use the touch api's in your application
    a. In your application, create, initialize and configure the sensors
    b. configure the global thresholds for the library
    c. providing timing for the QTouch libraries to operate. i,e the QTouch libraries do not use any timer resources of the microcontroller. The Host application has to provide the required timing and also call the API's at the appropriate intervals to perform touch sense detect operations
8) General application notes
    a. The QMatrix acquisition method libraries internally use TIMER1 for their operation.
    b. Ensure that there are no conflicts between the resources used by the touch library and the host application
    c. Ensure that the stack size is adjusted to factor in the stack depth required for the operation of the touch libraries.

6.9.3.1 Resources used by QMatrix acquisition method libraries

The following additional resources are used by the QMatrix acquisition method libraries.

- One Analog Comparator
- One internal Timer (Usually Timer1 depending on the availability on particular microcontroller)
- One ADC Multiplexer(The critical section of the touch sensing library disables the use of ADC as conversion unit and enables the same ADC as a multiplexer, but the user can use the ADC for conversion in rest of his application code)

In case of XMega devices, the resources are used internal to the library and hence cannot be used by the host application

- Analog Comparator0 on PORTA (AC0 on PORTA)
- Timer/Counter1 on PORTC (TCC1)
- Event System Channel0 (EVSYS_CH0)

6.9.3.2 Example

The example below will explain in detail the steps to follow for library selection.

Criteria	Selection	Notes
Microcontroller	ATTiny88	List of supported devices can be found at 7.2.4.1
Number of channels required for the application	6	number channels available for a Tiny88 is listed in section 7.2.4.2.1
Rotors and sliders required and Number of ROTOR/SLIDERS	Yes 2	Library variants supported for ATTiny88 is listed in section 7.2.4.2.1
Y_LINES on Lower pins or higher	On lower pins of port	Supported configurations for ATtiny88 is listed in

pins of port	(YL_LO_NIB)	section 7.2.4.3.1
Compiler tool chain	IAR	Supported compiler tool chains listed in 7.2.2
Choice of ports available for the design	X Lines on PORTB	Supported device configurations for X/Y and SMP ports is listed in section 7.2.4.3.1
	YA Line on PORTD	
	YB Line on PORTC	
	SMP Pin on PORTD pin 7	
	DELAY_CYCLES of 4	

Given the above requirements for the applications, the first step is to select the right library variant required.

Step 1 :
By following the guidelines listed for library selection in 7.2.4.2.1, we see that there are a few variants of libraries supported for AT Tiny device. Since the application requires 6 channels and rotor slider support, one has to select a library variant which supports atleast 6 channels or more. Hence we select the 8 channel library which supports the required Port combination and the delay cycle preferred which works out to be the variant

- lib_t88_8qm_4x_2y_krs_2rs_YL_LO_NIB_.r90

Step 2 : Defining the constants / symbols in the project space
In the host application file (say main.c), define the following constants and symbols

```
#define _QMATRIX_
#define QT_NUM_CHANNELS          8
#define PORT_X                   B
#define PORT_YA                  D
#define PORT_YB                  C
#define PORT_SMP                 D
#define SMP_BIT                  7
#define  DELAY_CYCLES            4
#define _ROTOR_SLIDER_
#define QT_MAX_NUM_ROTORS_SLIDERS    2
```

NOTE : 1.The above definitions should be placed before including “touch_api.h” in your files.
Alternatively, you can define these in your IDE’s project options or have them defined in a separate header

2. In case XMEGA device is used for QMatrix the symbol __*ATXMEGA*_ has to be included in the Project space along with the symbols mentioned above.

Step3 : Usage of libraries
Now, you can use the touch API’s to create, initialize and perform touch sensing. Please refer to the sample applications in section 6.10.3 for reference. These sample applications illustrate the usage of the API’s and the sequence of operation

6.9.3.3 Checklist of items for integrating QMatrix Capacitive sensing libraries

When integrating QMatrix acquisition method libraries, ensure the following

- Check that the CLKPR register is available for the selected device. If not remove the CLKPR statements.
- MCUCR register is available and if so disable pullups
- Check if the timer registers and bitfields used are correct and change them if necessary.
- The above settings can be modified by the user by changing the API's that are available to the user. The API's include
    - *qt_set_parameters ( )*
- The host application must provide the current time.
  This information is passed to the library as an argument to the function *qt_measure_sensors().*
  This is used for time-based library operations such as drift compensation.
- The GPIO internal pull-ups must be disabled for all port pins used for touch sensing when calling the library.
  For 8-bit AVR devices, this can be done by
    a. Setting the "PUD" bit in the "MCUCR" register or
    b. Setting the "PUD" bit in the "SFIOR" register
    c. Setting the JTD bit in the "MCUCR" register to disable JTAG (if available). This can be done only when the JTAG lines are in conflict with the touch sensing lines.
- The library must be called often enough to provide a reasonable response time to user touches. The typical time to call the library is from 25 ms to 50 ms.

6.9.4 Common checklist items

6.9.4.1 Configuring the stack size for the application

The stack requirements for the QTouch library should be accounted for and the stack size adjusted in the user's project for proper operation of the software when using the IAR IDE. This section lists the stack usage for the different variants of the QTouch and QMatrix acquisition method libraries applicable to the IAR compiler tool chain.

Note : When using the IAR IDE / compiler tool chain, the map file generated for the application will list total CSTACK & RSTACK requirements. Please adjust the total CSTACK and RSTACK values in the IAR project options to be greater than the values listed in the map file. Refer to section 6.10.4 which illustrates how to change the settings in IAR IDE.

Table 5 : Stack requirements of the QTouch capacitive sensing libraries when using IAR IDE projects

QTouch Acquisition method Libraries : Stack usage for IAR compiler tool chain		
Configuration	**CSTACK size**	**RSTACK size**
Single port pair - only keys (4 / 8 channels)	0x30	0x28
Single port pair – keys/ rotors/ sliders (4/8 channel)	0x40	0x2C
Two port pairs - only keys keys (16 channel)	0x50	0x28
Two port pairs – keys/ rotors/ sliders (16 channel)	0x60	0x2C

Table 6 : Stack requirements of the QMatrix capacitive sensing libraries when using IAR IDE projects

QMatrix Acquisition method Libraries : Stack usage for IAR compiler tool chain			
Number of channels	**Configuration**	**CSTACK size**	**RSTACK size**
8	ONLY KEYS	0x25	0x20
8	KEYS/ROTOR/SLIDER	0x35	0x20
16	ONLY KEYS	0x30	0x20
16	KEYS/ROTOR/SLIDER	0x40	0x20
32	ONLY KEYS	0x35	0x25
32	KEYS/ROTOR/SLIDER	0x45	0x25
64	ONLY KEYS	0x45	0x25
64	KEYS/ROTOR/SLIDER	0x55	0x25

6.10 Example project files

The QTouch library is shipped with various example projects to illustrate the usage of the touch API's to add touch sensing to an application across various devices

Sample applications are also provided for the following kits
1 TS2080A : QTouch Technology evaluation Kit
2 TS2080B : QMatrix Technology evaluation Kit

Note: Example projects must be built in the installed folder, and if moved/copied elsewhere then paths must be edited appropriately.

6.10.1 Using the Sample projects

The sample applications are shipped with the complete set of files required to configure, build and download the application for both IAR-workbench and AVR Studio IDE.

Since more than one device may use the same library (applicable for QTouch acquisition method libraries), example project files and applications have been provided only for select devices which use these libraries.

6.10.2 Example applications for QTouch acquisition method libraries

6.10.2.1 Selecting the right configuration

Each example project for a device can support multiple configurations (i.e a. keys only, b. with rotors and sliders c.16 channel etc.) . The configuration sets determine the configuration options for using the library and also the right library variant to link with the project.

The configuration sets for IAR IDE are named according to the convention listed below

Configuration set for IAR IDE
Naming convention : <**vP**>g<**Q**>_<**CH**>qt_k_<**RS**>rs

Field Name	Values	Comments
vP	v1, v3, xmega, uc3a, uc3b	VersionP of the core AVR device supported by this library variant
Q	1 to 6	GroupQ of the core AVR device supported by this library variant
CH	4, 8, 16, 32	Total number of channels supported by each library.
RS	1, 2, 4, 8	Total number of rotors / sliders supported for the respective channel counts mentioned in previous row.

The configuration sets for AVR Studio IDE are named according to the convention listed below

Configuration set for AVR Studio IDE <**avrP**>g<**Q**>_<**CH**>qt_k_<**RS**>rs		
Field Name	**Values**	**Comments**
avrP	avr25, avr4, avr 51, avr5, xmega, uc3a, uc3b	VersionP of the core AVR device supported by this library variant
Q	1 to 6	GroupQ of the core AVR device supported by this library variant
CH	4, 8, 16, 32	Total number of channels supported by each library.
RS	1, 2, 4, 8	Total number of rotors / sliders supported for the respective channel counts mentioned in previous row.

Depending on your need, you need to select the right configuration required and build the project.

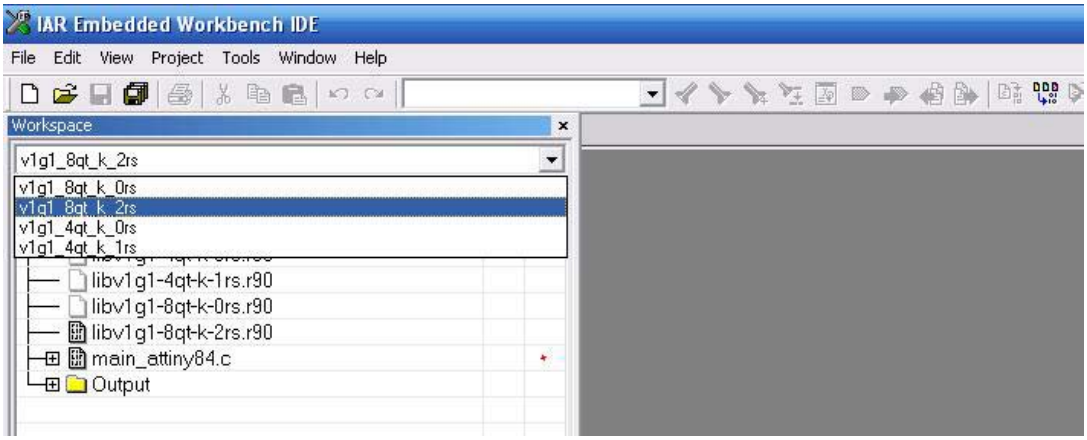


Figure 11: Selecting the right configuration in the QTouch acquisition method example applications in IAR –IDE

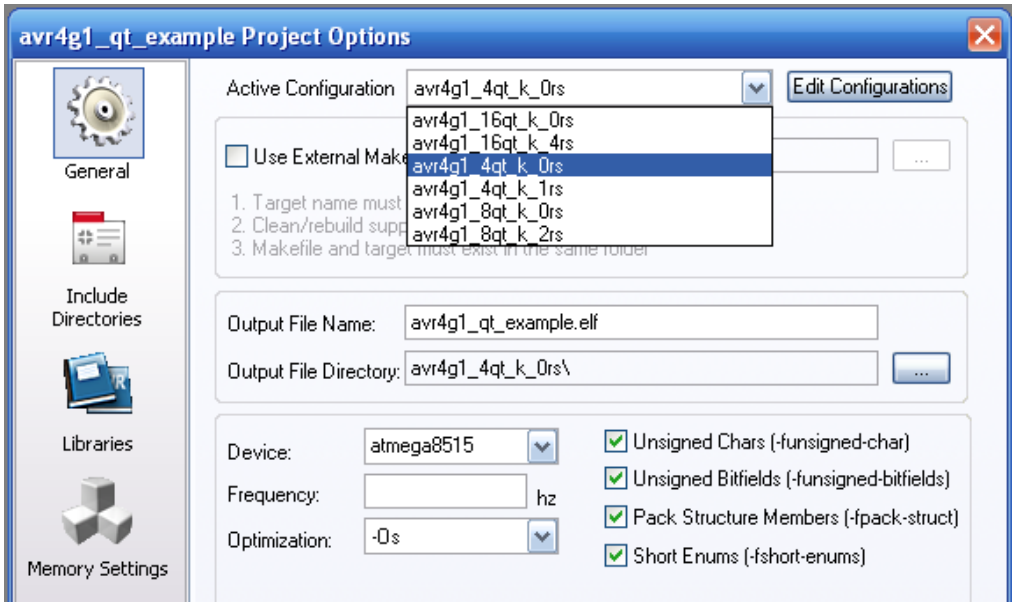


Figure 12 : Selecting the right configuration in QTouch acquisition method example applications in AVR-4 IDE

6.10.2.2 Changing the settings to match your device

6.10.2.2.1 Processor settings

Once you have selected the appropriate example project and the configuration, you need to ensure that the settings in the project are configured to reflect the correct device. The settings include

- Device type (CPU type) for the project

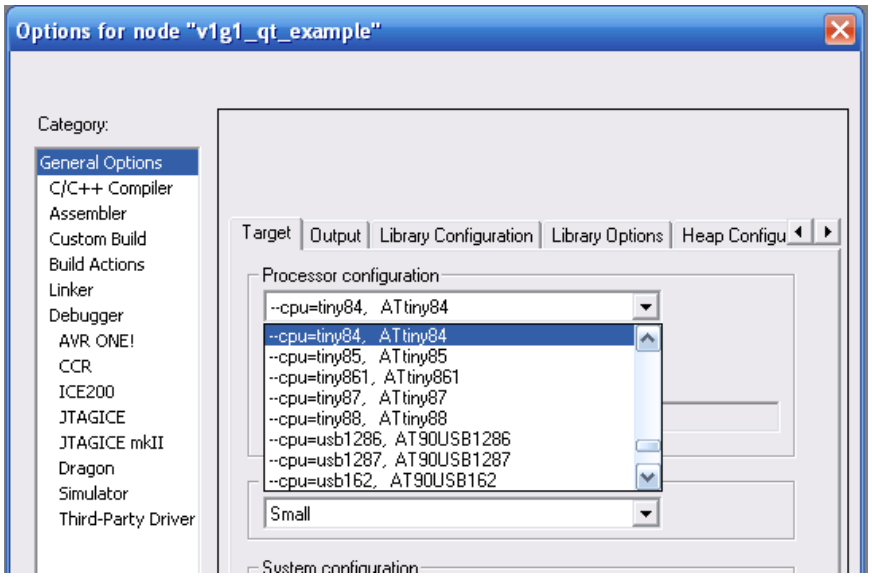


Figure 13 : Changing the processor settings for the examples in IAR IDE

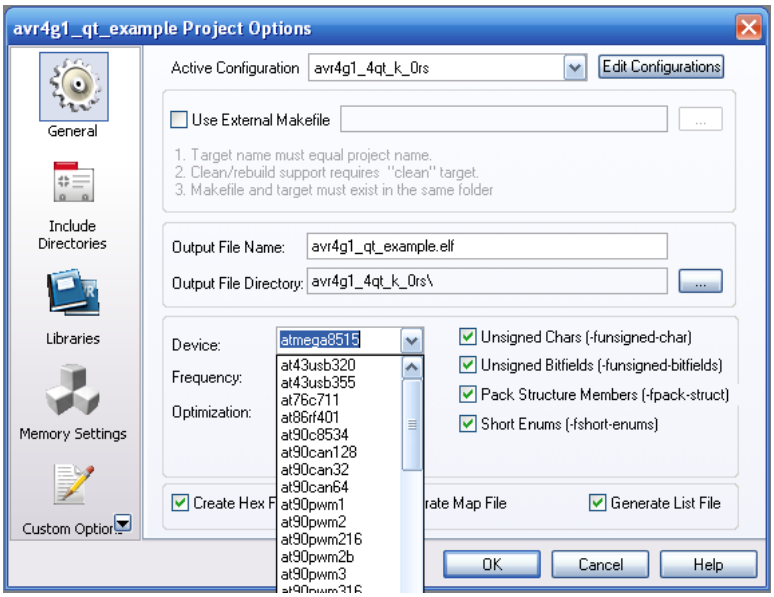


Figure 14 : Changing the processor settings for the examples in AVR-Studio 4

6.10.2.3 Changing the library configuration parameters

The configuration parameters required for the library are specified in the project options of the examples. They are as listed below.

Symbol / Constant name	Range of values	Comments
QTOUCH	This macro has to be defined in order to use QTouch libraries.	
SNS & SNSK	Section 7.1.5.1provides details on the range of values allowed.	To be used if only single port pair is needed for the design
SNS1 – SNSK1 & SNS2 – SNSK2	Section 7.1.5.3 has details on the range of values allowed	To be used if two port pairs are needed for the design
QT_NUM_CHANNELS	4, 8, 16 for tinyAVR, megaAVR and XMEGA device libraries and 8, 16, 32 for UC3 device libraries.	
_ROTOR_SLIDER_	Rotor / slider can be added to the design, if this symbol is defined	A library with rotor / slider functionality already available needs to be selected if this macro is to be enabled
_DEBUG_INTERFACE_	The debug interface code in the example application will be enabled if this macro is enabled.	This will enable the application to output QTouch measurement values to GPIO pins, which can be used by a USB bridge to view the output on Hawkeye or QTouch Studio. This feature is currently supported by EVK/TS 2080A and QT600 boards.
DELAY_CYCLES	1, 2, 3, 4, 5, 10, 25, 50	Please refer to section 5.6

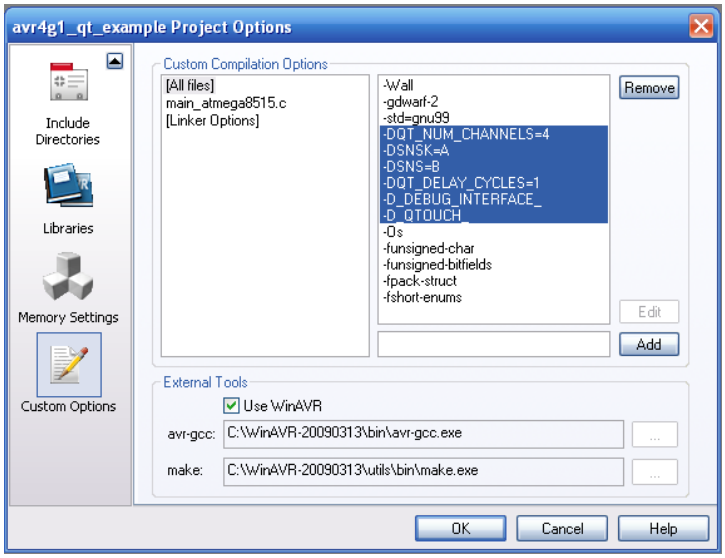


Figure 15 : Specifying the QTouch acquisition method library configuration parameters in IAR IDE

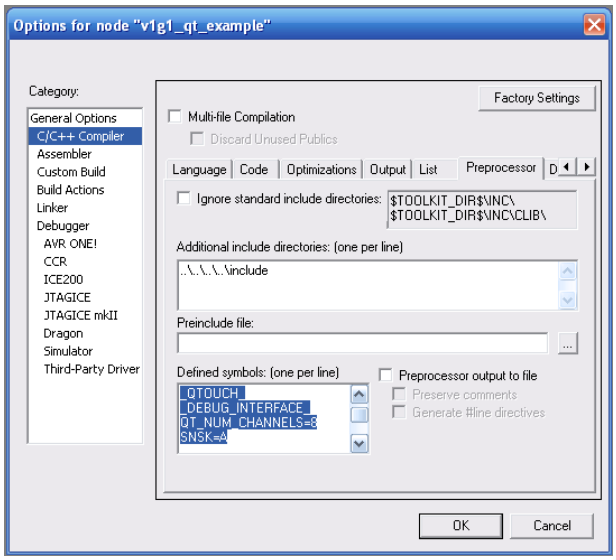


Figure 16 : Specifying the QTouch acquisition method library configuration parameters in AVR Studio IDE

6.10.2.4 Using the example projects

The sample applications are shipped with the complete set of files required to configure, build, execute and test the application for both IAR-workbench and AVR Studio IDEs.

The sample applications are provided for the evaluation kits and a few configurations for select devices.

The user can use the sample applications as a reference or baseline to configure different configurations. Please ensure to change the configuration settings in the project options to match the device selected.

To change the configuration settings of the sample applications,

1. Select the configuration from the list of configurations available.
2. If the user wishes to have a new name for the configuration to be used, a new configuration can be added to the project.
3. If a different variant of the library needs to be used, remove the existing library in that particular configuration and add the library variant that you require. Please refer to 7.1.4 for details on the different library variants. Update the linker options to specify the library to be linked.
4. Specify the tunable configuration parameters for the project as illustrated in sections 6.10.2.2 and 6.10.2.3.

6.10.3 Example applications for QMatrix acquisition method libraries

The QMatrix acquisition method libraries include example projects for some of the supported devices.

- Example projects for both IAR IDE and AVR Studio IDE along with example applications are provided for select devices using the QMatrix acquisition libraries.
- These sample applications demonstrate the usage of the touch API's to add touch sensing to an application.
- The list of devices and the supported configuration flavors for IAR IDE and AVR Studio IDE are listed below.

Device	Example Project for IAR	Project Configuration flavors supported
Example projects provided for QMatrix acquisition method libraries for IAR IDE.		
ATtiny88	t88_qm_example_iar	t88_qm_16ch_8x_2y_k_0rs_Ylow t88_qm_16ch_8x_2y_krs_4rs_Ylow
ATmega88PA	m88pa_qm_example_iar	m88_qm_32ch_8x_4y_k_0rs_Ylow m88_qm_32ch_8x_4y_krs_4rs_Ylow
ATmega16A	m16a_qm_example_iar	m16_qm_16ch_8x_2y_k_0rs_Yhigh m16_qm_16ch_8x_2y_krs_4rs_Yhigh
ATmega164P	m164p_qm_example_iar	m164p_qm_32ch_8x_4y_k_0rs_Ylow m164p_qm_32ch_8x_4y_krs_4rs_Ylow
ATmega165P	m165p_qm_example_iar	m165p_qm_32ch_8x_4y_k_0rs_Ylow m165p_qm_32ch_8x_4y_krs_4rs_Ylow
ATmega324P	m324p_qm_example_iar	m324p_qm_64ch_8x_8y_k_0rs_Ylow m324p_qm_64ch_8x_8y_krs_4rs_Ylow m324p_qm_64ch_8x_8y_krs_8rs_Ylow
ATmega8535	m8535_qm_example_iar	m8535_qm_16ch_4x_4y_k_0rs_Yhigh m8535_qm_16ch_4x_4y_krs_4rs_Yhigh
TS2080B (Tiny88 Device Evaluation Kit)	TS2080B_qm_example_iar	t88_qm_8ch_4x_2y_k_0rs t88_qm_8ch_4x_2y_krs_2rs
ATmega128RFA1	m128rfa1_qm_example_iar	m128rfa1_qm_64ch_8x_8y_k_0rs_Ylow m128rfa1_qm_64ch_8x_8y_krs_4rs_Ylow m128rfa1_qm_64ch_8x_8y_krs_8rs_Ylow
ATxmega128A1	xmega_qm_example_iar	libv6xm_32qm_8x_4y_k_0rs_YL_LO_NIB libv6xm_32qm_8x_4y_krs_4rs_YL_LO_NIB
Example projects provided for QMatrix acquisition method libraries for AVR Studio IDE with GCC toolchain		
ATtiny88	t88_qm_example_gnu	t88_qm_8ch_4x_2y_k_0rs_Ylow t88_qm_8ch_4x_2y_krs_2rs_Ylow
ATmega88p	m88p_qm_example_gnu	m88_qm_32ch_8x_4y_k_0rs_Ylow m88_qm_32ch_8x_4y_krs_4rs_Ylow
ATmega16	m16_qm_example_gnu	m16_qm_16ch_8x_2y_k_0rs_Yhigh

		m16_qm_16ch_8x_2y_krs_4rs_Yhigh
ATmega164P	m164p_qm_example_gnu	m164p_qm_32ch_8x_4y_k_0rs_Ylow m164p_qm_32ch_8x_4y_krs_4rs_Ylow
ATmega165P	m165p_qm_example_gnu	m165p_qm_32ch_8x_4y_k_0rs_Ylow m165p_qm_32ch_8x_4y_krs_4rs_Ylow
ATmega324P	m324p_qm_example_gnu	m324p_qm_64ch_8x_8y_k_0rs_Ylow m324p_qm_64ch_8x_8y_krs_4rs_Ylow m324p_qm_64ch_8x_8y_krs_8rs_Ylow
ATmega8535	m8535_qm_example_gnu	m8535_qm_16ch_4x_4y_k_0rs_Yhigh m8535_qm_16ch_4x_4y_krs_4rs_Yhigh
TS2080B(Tiny88 Device Evaluation Kit)	TS2080B_qm_example_gnu	t88_qm_8ch_4x_2y_k_0rs t88_qm_8ch_4x_2y_krs_2rs
ATmega128RFA1	m128rfa1_qm_example_gnu	m128rfa1_qm_64ch_8x_8y_k_0rs_Ylow m128rfa1_qm_64ch_8x_8y_krs_4rs_Ylow m128rfa1_qm_64ch_8x_8y_krs_8rs_Ylow
ATxmega128A1	xmega_qm_example_gnu	libv6xm_32qm_8x_4y_k_0rs_YL_LO_NIB libv6xm_32qm_8x_4y_krs_4rs_YL_LO_NIB

6.10.3.1 Selecting the right configuration

The sample application is built to support a maximum channel support configuration available for that particular device for both IAR & AVR IDEs.

Internally there are two configurations for each device.

- ALL KEYS configuration : Supports only keys
- KEYS/ROTORS/SLIDERS configuration : Supports keys or rotors or sliders concurrently

These configurations enable a set of stored options and a specific library to be selected in order to build application using the specific library.

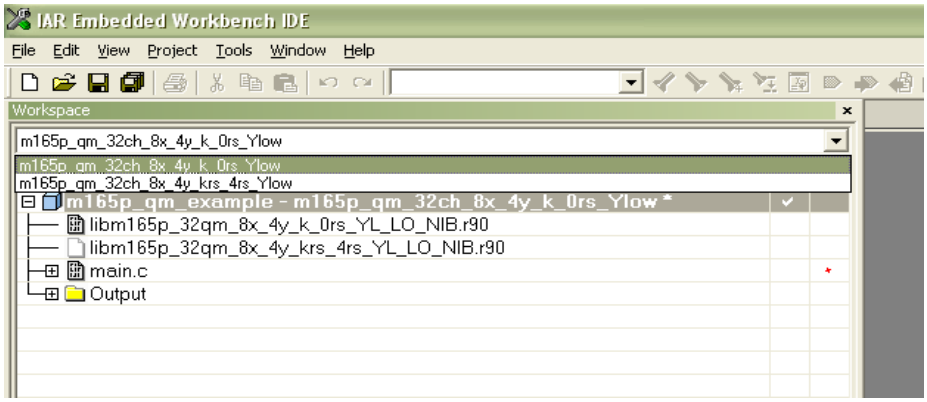


Figure 17 : Selecting the right configuration in the QMatrix acquisition method example applications in IAR –IDE

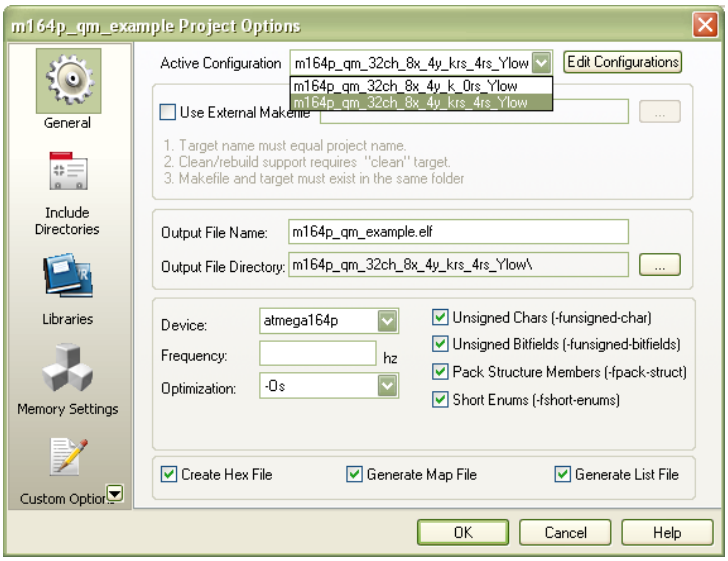


Figure 18 : Selecting the right configuration in the QMatrix acquisition method example applications in AVR Studio IDE

6.10.3.2 Changing the library configuration parameters

The configuration parameters required for the library are specified in the project options of the examples. They are as listed below.

Symbol	Range of values	Comments
QMATRIX		This macro should be defined to use QMatrix acquisition method API e.g #define _QMATRIX_
QT_NUM_CHANNELS	8,16,32,64.	Specifies the Maximum Number of channels supported by the library variant selected. section 7.2.4 *e.g #define QT_NUM_CHANNELS 8*
QT_MAX_NUM_ROTORS_SLIDERS	0,2,4,8	Specifies the Maximum Number of Rotors/Sliders supported by the library variant selected. section 7.2.4 *e.g #define QT_NUM_CHANNELS 8*
PORT_X	See device configurations supported in section 7.2.4.3	Drive electrode for touch sensing using QMatrix acquisition.
PORT_YA	See device configurations supported in section 7.2.4.3	Receive electrode for touch sensing using QMatrix acquisition
PORT_YB	See device configurations supported in section 7.2.4.3	Receive electrode for touch sensing using QMatrix acquisition
PORT_SMP	See device configurations supported in section 7.2.4.3	SMP Port used for touch sensing using QMatrix acquisition

SMP_BIT	See device configurations supported in section 7.2.4.3	SMP Pin used for touch sensing using QMatrix acquisition
_ROTOR_SLIDER_	Symbol defined if Rotor and/or slider is required	Needs to be added in case user needs to configure ROTOR/SLIDER Needs to be removed for ALL KEYS configuration
DELAY_CYCLES	1, 2, 3, 4, 5, 10, 25, 50.	The Dwell Cycle time used for the Qmatrix acquisition method.
_DEBUG_INTERFACE_		Specify this if debug data should be output to be viewed with QTouch Studio on a PC connected. See section 6.10.7. This feature is currently supported by EVK/TS 2080B and QT600 boards.

The above parameters can be customized in the project configuration settings.

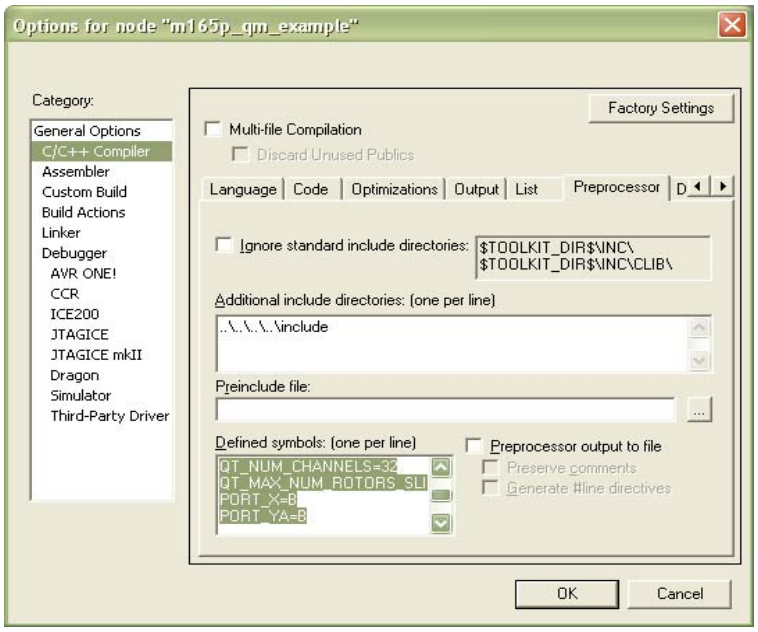


Figure 19 : Specifying QMatrix acquisition library parameters in IAR IDE project

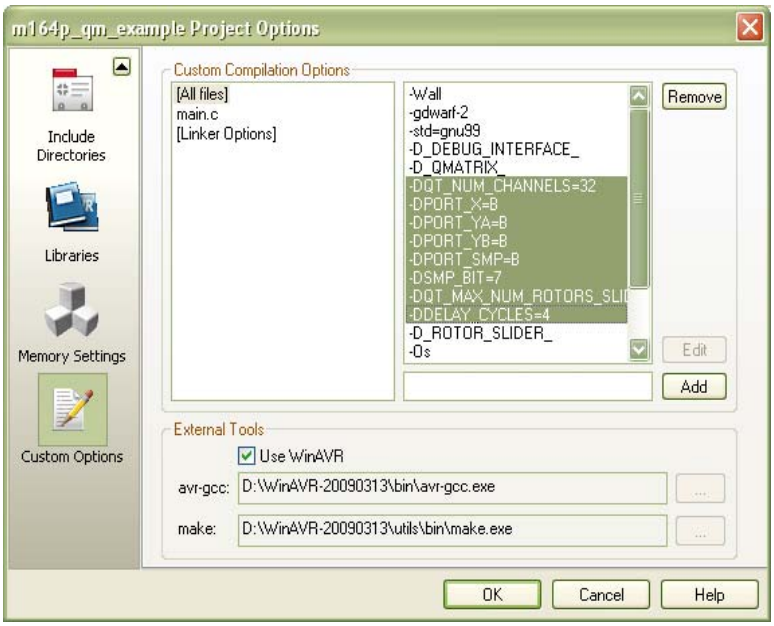


Figure 20 : Specifying QMatrix acquisition library parameters in AVR Studio IDE project

6.10.3.3 Using the example projects

The sample applications are shipped with the complete set of files required to configure, build, execute and test the application for both IAR-workbench and AVR Studio IDEs.

The sample applications are provided for the evaluation kits and a few configurations for select devices.

The user can use the sample applications as a reference or baseline to configure different configurations. Please ensure to change the configuration settings in the project options to match the device selected

To change the configuration settings of the sample applications,

1) Select the configuration from the list of configurations available
2) If the user wishes to have a new name for the configuration to be used, a new configuration can be added to the project
3) If a different variant of the library needs to be used, remove the existing library in that particular configuration and add the library variant that you require. Please refer to 7.2.4.2 for details on the different library variants. Update the linker options to specify the library to be linked
4) Specify the tunable configuration parameters for the project as illustrated in 6.10.3.2
5) For QMatrix on XMEGA devices, Please check if the pre-processor symbol *_ATXMEGA_* is added in the project space or not.

6.10.4 Allocating unused Port Pins for User Application

The GPIO pins within a port that are not used for QTouch or QMatrix acquisition methods can be used for user application. The usage of pins for QTouch is based on the channels that are being configured while enabling the sensors (keys/rotors/sliders).

The example below configuring 4 keys, a rotor and a slider shows how the pin configurability is achieved by configuring the sensor channels. The code snippet configures a specific 10 channels of a 16 channel library based on the GPIO port pins available for QTouch™.

Port Configuration:

```
#define SNSK1        C
```

```
#define SNS1        D
#define SNSK2       A
#define SNS2        B
```

Channel/Pin Configuration:

```
/*  enable a key on channel 0    */
qt_enable_key( CHANNEL_0, AKS_GROUP_2, 10u, HYST_6_25 );

/*  enable a slider on channels 2 to 4  */
qt_enable_slider( CHANNEL_2, CHANNEL_4, AKS_GROUP_1, 16u, HYST_6_25, RES_8_BIT, 0u  );

/*  enable a key on channel 6 */
qt_enable_key( CHANNEL_6, AKS_GROUP_2, 10u, HYST_6_25 );

/*  enable a key on channel 7 */
qt_enable_key( CHANNEL_7, AKS_GROUP_2, 10u, HYST_6_25 );

/*  enable a rotor on channels 12 to14   */
qt_enable_rotor( CHANNEL_12, CHANNEL_14, AKS_GROUP_1, 16u, HYST_6_25, RES_8_BIT, 0u );

/*  enable a key on channel 15 */
qt_enable_key( CHANNEL_15, AKS_GROUP_2, 10u, HYST_6_25 );
```

The channel numbers 0,2,3,4,6,7 are allocated to pins 0,2,3,4,6,7 of (D,C) port pair respectively. Pins 1 and 5 of ports C and D can be used for user application. Similarly the channel numbers 12,13,14,15 are allocated to pins 4,5,6,7 of (B,A) port pair respectively. Pins 1, 2, 3 and 4 of ports B and A are again unused by the QTouch library and can be used for user application.

6.10.4.1 Disabling and Enabling of Pull-up for AVR devices:

The Pull-up circuit available (in AVR devices) for each GPIO pin has to be disabled before QTouch acquisition is performed. For tinyAVR and megaAVR devices the Pull-up circuit for all GPIO port pins are enabled and disabled together. When user needs to configure the pins that are not used by QTouch library for his application, he may enable the Pull-up circuit after QTouch measurements are performed and disable them before the touch acquisition starts once again (as shown in the code snippet below).

```
/* Disable pull-ups for all pins */
MCUCR       |=      (1u << PUD);      //MCUCR_PUD = 1u;

/* Perform QTouch measurements */
qt_measure_sensors ( current_time_ms_touch );

/* Enable pull-ups for all pins */
MCUCR       &=      ~ (1u << PUD);    //MCUCR_PUD = 0u;
```

For XMEGA devices the Pull-up circuit for each individual GPIO port pins can be configured individually, by writing to the PINnCTRL register of the ports being used.

6.10.5 Adjusting the Stack size when using IAR IDE

The example projects for IAR IDE, the CSTACK and RSTACK values are configured to account for the requirements of the QTouch libraries and the included main.c file which illustrates the usage of the touch API.

- Adjust the CSTACK and RSTACK values appropriately based on additional software integrated or added to the examples.

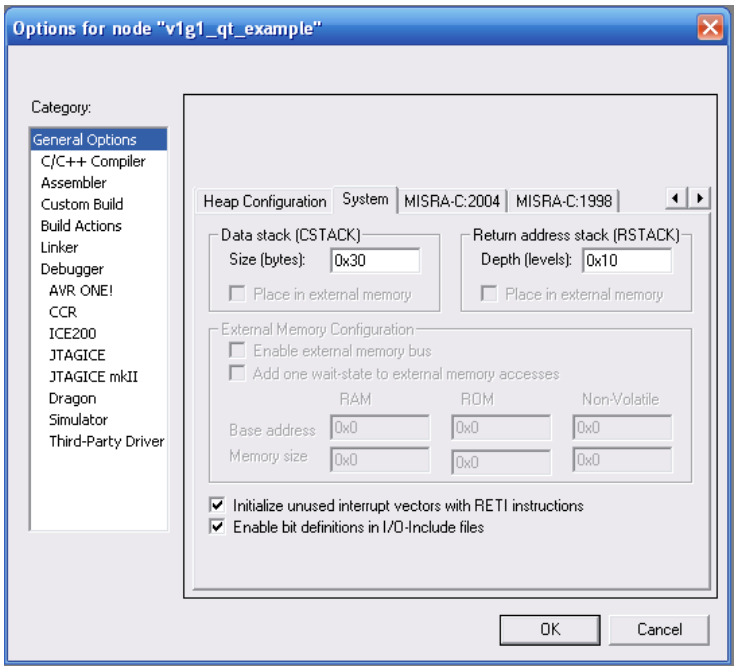


Figure 21 : Modifying the stack size in IAR IDE

6.10.6 Optimization levels

The default configuration settings in sample projects which ship with the library are set to the highest level of optimization for IAR and GCC variants of the libraries. The user might be required to change this setting for debugging purposes

- In case of IAR, The optimizations tab in project configuration options specifies High.
- In case of GCC, the libraries are compiled with the –Os which signifies that the Optimization for generating the library is maximum.

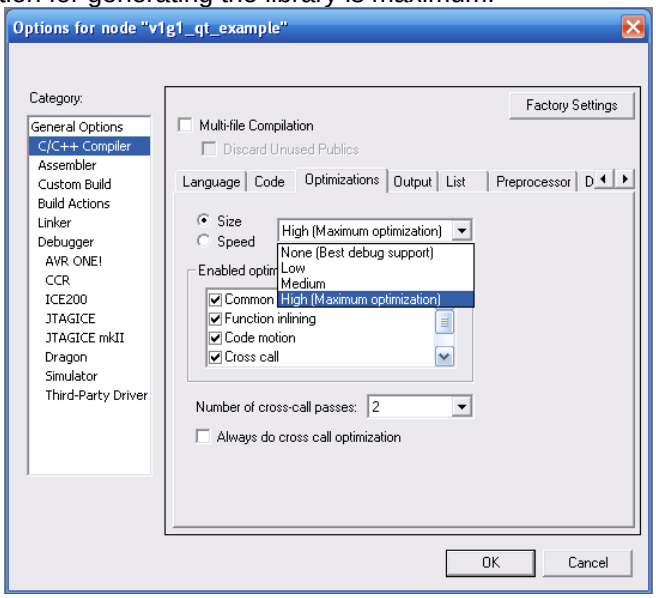


Figure 22 : Specifying the optimization level in IAR IDE

6.10.7 Debug Support in Example applications

The EVK2080 and QT600 applications provide output debug information on standard GPIO pins through the USB bridge IC to PC software for display by AVR QTouch Studio.

For ATMEL devices that are not supported through EVK or QT600 kits, the output measurement values cannot be viewed through AVR QTouch Studio. Please refer to section 6.10.7.3 for more information on observing the output touch measurement data without the use of a USB bridge or AVR QTouch Studio.

6.10.7.1 Debug Support in the sample applications for EVK2080 and QT600 boards

The sample applications provided for the EVK2080 boards and the QT600 boards output debug information which are captured by a USB bridge chip and then routed to the PC software (QTouch Studio) for display.

6.10.7.2 How to turn on the debug option

In the project options, the symbol definition _DEBUG_INTERFACE_ is used to enable reporting the debug data. Based on the IDE used, you can do the following to enable the debug feature

- ***IAR-EWAVR:***
  In the project options -> C/C++ compiler -> Preprocessor Tab
  Add the Directive _DEBUG_INTERFACE_

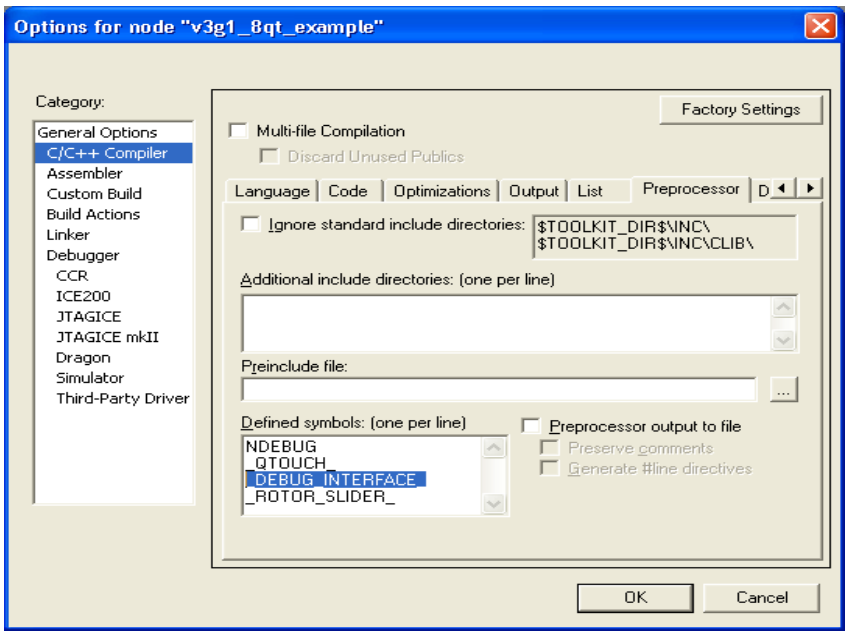


Figure 23 : Enabling Debug Support for the library in IAR IDE

- ***WINAVR- GCC:***
  In the Project configuration Options -> Custom Options->
  Add the Directive –D_DEBUG_INTERFACE_

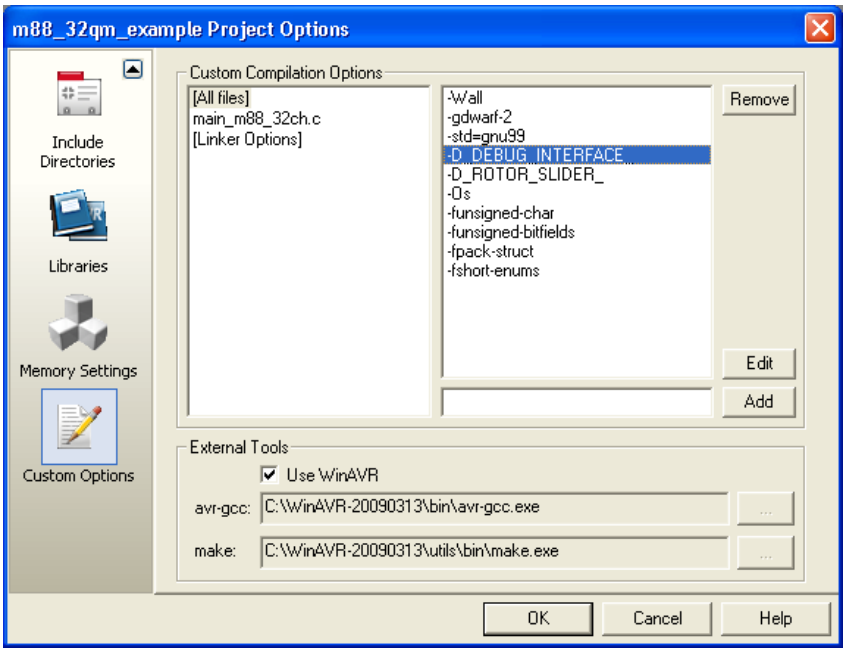


Figure 24 : Enabling Debug support for the library in AVR Studio IDE

6.10.7.3 Debug Interface if USB Bridge board is not available

For the sample applications using the devices that are not supported on EVK2080 and QT600 the debug interface code is not provided. This is because a separate USB bridge board is required to read the data and display it on QTouch studio. However in this case the output touch measurement data can still be viewed using the IAR or AVR Studio IDE when running the code in debug mode using debug wire or emulator.

extern qt_touch_lib_measure_data_t **qt_measure_data**;

The **qt_measure_data** global variable contains the output touch measurement data. Refer to section 6.3.3 for more information on the data type.

For GCC generated libraries the output touch measurement data can be observed on the watch window through the pointer variable **pqt_measure_data.**

qt_touch_lib_measure_data_t ***pqt_measure_data** = &qt_measure_data;

7 Library Variants

7.1 QTouch Acquisition method library variants

7.1.1 Introduction

Variants of the ATMEL QTouch Library based on QTouch Technology are available for a range of ATMEL Microcontrollers. This section lists the different variants available. By following a simple series of steps, the user can identify the right library variant to use in his application.

7.1.2 Support for different compiler tool chains

The QTouch acquisition method libraries are supported for the following compiler tool chains.

Table 7 Compiler tool chains supported for QTouch acquisition method libraries

Tool	Version
IAR Compiler	5.30.6
IAR Embedded Workbench	5.3.6.927
GCC – AVRStudio	4.18 build 673
IAR32 Compiler	3.20A/W32
WinAVR	20090313
GCC – AVR32 Studio	2.3.0
GCC – AVR GCC GNU Tool Chain	avr32-gnu-toolchain-2.1.0

7.1.3 QTouch Acquisition method library naming conventions

The libraries are named according the convention listed below

7.1.3.1 Naming convention for libraries to be used with WinAVR GCC tool chain

lib<**avrP**>g<**Q**>_<**CH**>qt_k_<**RS**>rs.a

Field name	Possible values	Comments
avrP	avr25 avr 35 avr 4 avr 51 avr 5 avrxmega2 avrxmega3 avrxmega4 avrxmega5 avrxmega6 avrxmega7	VersionP of the core AVR device supported by this library variant for tinyAVR and megaAVR devices. Also supports XMEGA and UC3 cores.

	uc3a uc3b	
Q	1 to 6	group Q of the core AVR device supported by this library variant
CH	4, 8, 16, 32	Total number of channels supported by each library.
RS	1, 2, 4, 8	Total number of rotors / sliders supported for the respective channel counts mentioned in previous row.

For example, the library variant *"libavr25g1_16qt_k_4rs.a"* supports the following configuration

- Device : tinyAVR or megaAVR device belonging to core group avr25
- Belongs to a set of devices of group 1 supported by this library
- Support a maximum of 16 channels
- Supports a maximum of up to 4 rotors / sliders.

7.1.3.2 Naming convention for libraries to be used with IAR-EWAR

The libraries are named according the naming convention listed below

lib<**vP**>g<**Q**>_<**CH**>qt_k_<**RS**>rs.r90

Field name	Possible values	Comments
vP	v1 v3 v3xmsf v3xm v4xm v5xm v6xm uc3a uc3b	VersionP of the core AVR device supported by this library variant variant for tinyAVR and megaAVR devices. Also supports Xmega and UC3 cores.
Q	1 to 6	GroupQ of the core AVR device supported by this library variant
CH	4, 8, 16, 32	Total number of channels supported by each library.
RS	1, 2, 4, 8	Total number of rotors / sliders supported for the respective channel counts mentioned in previous row.

For example, the library variant *"libv3g5_4qt_k_1rs.r90"* supports the following configuration

- Device : tinyAVR or megaAVR device belonging to core group v3
- Belongs to a set of devices of group 3 supported by this library
- Support a maximum of 4 channels

7.1.4 QTouch acquisition method library variants

Table 8 lists the different QTouch acquisition method library variants supported for AVRs. Use this table to select the correct library variant to be used in your application. Each row in the table below indicates

- the AVR for which the library is supported
- the corresponding Ports available for SNS and SNSK pins
- Compilers used for generating the libraries
- The library names to be selected for the requirements

Note: The libraries that are supported as listed in the table are only supported provided the device memory requirements are also satisfied.

Naming convention of the library

<ch>	Maximum channels supported by the library.

	Device	Range
	tinyAVR, megaAVR, XMEGA	4,8,16
	UC3	8,16,32
<RS>	Maximum number of rotor / sliders supported	

NOTE :

- For 8 bit devices, ports which have less than 8 pins cannot be used by the QTouch acquisition method libraries. Check the data sheet to determine the number of pins supported for each port

Table 8 : QTouch acquisition method library variants

Device	Ports Supported	Compiler	Library to be used (K) Supports only keys	Library to be used (KRS) Supports keys, rotors and sliders
Attiny43u	A, B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny44A	A, B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny45	B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny461	A, B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny84	A, B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny85	B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny861	A, B	IAR	libv1g1-<CH>qt-k-0rs.r90	libv1g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g1-<CH>qt-k-0rs.a	libavr25g1-<CH>qt-k-<RS>rs.a
Attiny48	A, B, C, D	IAR	libv1g2-<CH>qt-k-0rs.r90	libv1g2-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g2-<CH>qt-k-0rs.a	libavr25g2-<CH>qt-k-<RS>rs.a
Attiny88	A, B, C, D	IAR	libv1g2-<CH>qt-k-0rs.r90	libv1g2-<CH>qt-k-<RS>rs.r90
		GCC	libavr25g2-<CH>qt-k-0rs.a	libavr25g2-<CH>qt-k-<RS>rs.a
Attiny167	A, B	IAR	libv3g6-<CH>qt-k-0rs.r90	libv3g6-<CH>qt-k-<RS>rs.r90
		GCC	libavr35g1-<CH>qt-k-0rs.a	libavr35g1-<CH>qt-k-<RS>rs.a
Atmega8515	A, B, C, D, E	IAR	libv1g3-<CH>qt-k-0rs.r90	libv1g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g1-<CH>qt-k-0rs.a	libavr4g1-<CH>qt-k-<RS>rs.a
Atmega8535	A, B, C, D	IAR	libv1g3-<CH>qt-k-0rs.r90	libv1g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g1-<CH>qt-k-0rs.a	libavr4g1-<CH>qt-k-<RS>rs.a
Atmega8A	B, C, D	IAR	libv1g3-<CH>qt-k-0rs.r90	libv1g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g1-<CH>qt-k-0rs.a (use Atmega8 when compiling)	libavr4g1-<CH>qt-k-<RS>rs.a (use Atmega8 when compiling)
Atmega48PA	B, C, D	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a	libavr4g2-<CH>qt-k-<RS>rs.a
Atmega88PA	B, C, D	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a (use Atmega88P when compiling)	libavr4g2-<CH>qt-k-<RS>rs.a (use Atmega88P when compiling)
Atmega8HVA	A, B, C	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a	libavr4g2-<CH>qt-k-<RS>rs.a
ATPWM2	B, C, D, E	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a	libavr4g2-<CH>qt-k-<RS>rs.a
ATPWM2B	B, C, D, E	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a	libavr4g2-<CH>qt-k-<RS>rs.a

ATPWM3	B, C, D, E	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a	libavr4g2-<CH>qt-k-<RS>rs.a
ATPWM3B	B, C, D, E	IAR	libv1g4-<CH>qt-k-0rs.r90	libv1g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr4g2-<CH>qt-k-0rs.a	libavr4g2-<CH>qt-k-<RS>rs.a
Atmega64A	A, B, C, D, E, F, G	IAR	libv3g1-<CH>qt-k-0rs.r90	libv3g1-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g1-<CH>qt-k-0rs.a	libavr5g1-<CH>qt-k-<RS>rs.a
Atmega162	A, B, C, D, E	IAR	libv3g2-<CH>qt-k-0rs.r90	libv3g2-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g2-<CH>qt-k-0rs.a	libavr5g2-<CH>qt-k-<RS>rs.a
Atmega16A	A, B, C, D	IAR	libv3g2-<CH>qt-k-0rs.r90	libv3g2-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g2-<CH>qt-k-0rs.a (use Atmega16 when compiling)	libavr5g2-<CH>qt-k-<RS>rs.a (use Atmega16 when compiling)
Atmega32A	A, B, C, D	IAR	libv3g2-<CH>qt-k-0rs.r90	libv3g2-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g2-<CH>qt-k-0rs.a (use Atmega32 when compiling)	libavr5g2-<CH>qt-k-<RS>rs.a (use Atmega32 when compiling))
AT90CAN32	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
AT90CAN64	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega164P	A, B, C, D	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega165P	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega168PA	B, C, D	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a (use Atmega168P when compiling)	libavr5g3-<CH>qt-k-<RS>rs.a (use Atmega168P when compiling)
Atmega169P	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega16HVA	A, B, C	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega16U4	B, C, D, E, F	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega324PA	A, B, C, D	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a (use Atmega324P when compiling)	libavr5g3-<CH>qt-k-<RS>rs.a (use Atmega324P when compiling)
Atmega325P	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega3250P	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega328P	B, C, D	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega329P	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega3290P	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega32C1	B, C, D, E	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega32HVB	A, B, C	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90

		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega32M1	B, C, D, E	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega32U4	B, C, D, E, F	IAR	libv5g3-<CH>qt-k-0rs.a	libv5g3-<CH>qt-k-<RS>rs.a
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega32U6	A, B, C, D, E, F	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega406	A, B, C, D	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega640	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega644P	A, B, C, D	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega645	A, B, C, D, E, F, G,	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega6450	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega649	A, B, C, D, E, F, G,	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega6490	A, B, C, D, E, F, G	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega64C1	B, C, D, E	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega64M1	B, C, D, E	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
ATPWM216	B, C, D, E	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
ATPWM316	B, C, D, E	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
AT90USB646	A, B, C, D, E, F,	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
AT90USB647	A, B, C, D, E, F,	IAR	libv3g3-<CH>qt-k-0rs.r90	libv3g3-<CH>qt-k-<RS>rs.r90
		GCC	libavr5g3-<CH>qt-k-0rs.a	libavr5g3-<CH>qt-k-<RS>rs.a
Atmega128A	A, B, C, D, E, F, G	IAR	libv3g4-<CH>qt-k-0rs.r90	libv3g4-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g1-<CH>qt-k-0rs.a	libavr51g1-<CH>qt-k-<RS>rs.a
Atmega1280	A, B, C, D, E, F, G	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a
Atmega128rfa1	A, B, C, D, E, F, G	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a
Atmega1281	A, B, C, D, E, F, G	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a
Atmega1284P	A, B, C, D	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a
AT90CAN128	A, B, C, D, E, F, G	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a
AT90USB1287	A, B, C, D, E, F	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a

AT90USB1286	A, B, C, D, E, F	IAR	libv3g5-<CH>qt-k-0rs.r90	libv3g5-<CH>qt-k-<RS>rs.r90
		GCC	libavr51g2-<CH>qt-k-0rs.a	libavr51g2-<CH>qt-k-<RS>rs.a
AT32UC3Axxx	A, B	IAR	libuc3a-<CH>qt-k-0rs.r82	libuc3a-<CH>qt-k-<RS>rs.r82
		GCC	libuc3a-<CH>qt-k-0rs.a	libuc3a-<CH>qt-k-<RS>rs.a
AT32UC3Bxxx	A, B	IAR	libuc3b-<CH>qt-k-0rs.r82	libuc3b-<CH>qt-k-<RS>rs.r82
		GCC	libuc3b-<CH>qt-k-0rs.a	libuc3b-<CH>qt-k-<RS>rs.a
ATxmega128a1	A, B, C, D, E, F	IAR	libv6xmg1-<CH>qt-k-0rs.r90	libv6xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega7g1-<CH>qt-k-0rs.a	libavrxmega7g1-<CH>qt-k-<RS>rs.a
ATxmega128a1	D, E, F, J, H, K	IAR	libv6xmg2-<CH>qt-k-0rs.r90	libv6xmg2-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega7g2-<CH>qt-k-0rs.a	libavrxmega7g2-<CH>qt-k-<RS>rs.a
ATxmega64a1	A, B, C, D, E, F	IAR	libv4xmg1-<CH>qt-k-0rs.r90	libv4xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega5g1-<CH>qt-k-0rs.a	libavrxmega5g1-<CH>qt-k-<RS>rs.a
ATxmega64a1	D, E, F, J, H, K	IAR	libv4xmg2-<CH>qt-k-0rs.r90	libv6xmg2-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega5g2-<CH>qt-k-0rs.a	libavrxmega5g2-<CH>qt-k-<RS>rs.a
ATxmega256a3	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega256a3b	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega256d3	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega192a3	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega192d3	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega128a3	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega128d3	A, B, C, D, E, F	IAR	libv5xmg1-<CH>qt-k-0rs.r90	libv5xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega6g1-<CH>qt-k-0rs.a	libavrxmega6g1-<CH>qt-k-<RS>rs.a
ATxmega64a3	A, B, C, D, E, F	IAR	libv3xmg1-<CH>qt-k-0rs.r90	libv3xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega4g1-<CH>qt-k-0rs.a	libavrxmega4g1-<CH>qt-k-<RS>rs.a
ATxmega64d3	A, B, C, D, E, F	IAR	libv3xmg1-<CH>qt-k-0rs.r90	libv3xmg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega4g1-<CH>qt-k-0rs.a	libavrxmega4g1-<CH>qt-k-<RS>rs.a
ATxmega32a4	A, B, C, D, E	IAR	libv3xmsfg1-<CH>qt-k-0rs.r90	libv3xmsfg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega3g1-<CH>qt-k-0rs.a	libavrxmega3g1-<CH>qt-k-<RS>rs.a
ATxmega32d4	A, B, C, D, E	IAR	libv3xmsfg1-<CH>qt-k-0rs.r90	libv3xmsfg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega2g1-<CH>qt-k-0rs.a	libavrxmega2g1-<CH>qt-k-<RS>rs.a
ATxmega16a4	A, B, C, D, E	IAR	libv3xmsfg1-<CH>qt-k-0rs.r90	libv3xmsfg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega2g1-<CH>qt-k-0rs.a	libavrxmega2g1-<CH>qt-k-<RS>rs.a
ATxmega16d4	A, B, C, D, E	IAR	libv3xmsfg1-<CH>qt-k-0rs.r90	libv3xmsfg1-<CH>qt-k-<RS>rs.r90
		GCC	libavrxmega2g1-<CH>qt-k-0rs.a	libavrxmega2g1-<CH>qt-k-<RS>rs.a

7.1.5 Port combinations supported for SNS and SNSK pin configurations

AVRs have a variety of GPIO ports available and they can be used for QTouch acquisition method touch sensing by having the SNS and SNSK pins connected on these GPIO ports.

NOTE : If the SNS and SNSK pins are assigned on the same port for tinyAVR, megaAVR and XMEGA devices, then the maximum number of channels available on a single port is 4. For such cases, the 8-channel library for the specific device has to be used.
NOTE : If SNS and SNSK pins are to be assigned within the same port(for tinyAVR, megaAVR and XMEGA devices) and more than 4 channels are needed, then two ports have to be used. For such cases, the 16-channel library for the specific device has to be used.

7.1.5.1 Port combinations supported for single port pair SNS and SNSK pin configurations

This section lists the possible single port pair combinations supported for the various touch library variants. This section can be used as a reference when using the

- 4 & 8 channels libraries for tinyAVR, megaAVR and XMEGA devices
- 8, 16 and 32 channel libraries for UC3 devices.

NOTE : Port pair convention followed in the tables below :
MN : M – Port used for SNSK N – Port used for SNS

If a device supports a number of ports, not all combination of ports will be supported to be used as SNS and SNSK ports. The valid SNSK-SNS port combinations for libraries supporting a subset of the ports A, B, C, D, E, F & G are given in Table 9

Table 9 : Valid single port combinations for ports A, B, C, D, E, F & G supported by libraries

SNS \ SNSK	A	B	C	D	E	F	G
A	AA	BA	CA	DA	EA	FA	GA
B	AB	BB	CB	DB	EB	FB	GB
C	AC	BC	CC	DC	EC	FC	GC
D	AD	BD	CD	DD	ED	FD	GD
E	AE	BE	CE	DE	EE	FE	GE
F	AF	BF	CF	DF	EF	FF	GF
G	AG	BG	CG	DG	EG	FG	GG

= Not supported
= SNSK PORT
= SNS PORT
= Supported combination of ports

The valid SNSK-SNS port combinations for libraries supporting a subset of the ports D, E, F, J, H & K are given in Table 10

Table 10 : Valid single port combinations for ports D, E, F, J, H & K supported by libraries

SNS \ SNSK	D	E	F	J	H	K
D	DD	ED	FD	JD	HD	KD
E	DE	EE	FE	JE	HE	KE
F	DF	EF	FF	JF	HF	KF
J	DJ	EJ	FJ	JJ	HJ	KJ
H	DH	EH	FH	JH	HH	KH
K	DK	EK	FK	JK	HK	KK

= Not supported
= SNSK PORT
= SNS PORT
= Supported combination of ports

7.1.5.2 Tips on pin assignments for the sensor design using one pair of SNS/SNSK ports

This section lists tips on selecting the pin assignments when using a single port pair for the SNS and SNSK Pins.

Design choice for the sensor	Example Port configuration with pin assignments
SNSK & SNS pins are on different ports, number of channels = 4	• If the SNS(C) and SNSK(B) pins are on two different ports, the user should mount the sensors onto the corresponding pins such as (PC0,PB0), (PC1,PB1), (PC2,PB2) and (PC3,PB3)
SNSK & SNS pins are on different ports, number of channels = 8	• If the SNS(C) and SNSK(B) pins are on two different ports, the user should mount the sensors onto the corresponding pins such as (PC0,PB0), (PC1,PB1), (PC2,PB2).. • In this case channel 0 will be on (PC0, PB0) pins, channel 1 will be on (PC1, PB1) pins and so on up to channel 7 will be on (PC7, PB7) pins
SNSK & SNS pins are on different ports, number of channels = 16 when using UC3 device	• If the SNS(B) and SNSK(A) pins are on two different ports, the user should mount the sensors onto the corresponding pins such as (PB0,PA0), (PB1,PA1), (PB2,PA2).. • In this case channel 0 will be on (PB0, PA0) pins, channel 1 will be on (PB1, PA1) pins and so on up to channel 15 will be on (PB15, PA15) pins. • An exception to this is the case of configuring touch channels for QTouch which would have the SNS and SNSK pins on the same port. When using the 4, 8, 16 channel libraries (16 channel only in the case of UC3) the channels effectively available are reduced by half to 2, 4 and 8 respectively
SNSK & SNS pins are on the same port, number of channels = 2	• If the use of SNS(A) and SNSK(A) pins are on the same port, the user should always have the configuration (PA0, PA1) & (PA2, PA3). • In this case channel 0 will be on (PA0, PA1) pins, channel 1 will be on (PA2, PA3) pins. The even pins of the port are used as SNS pins and odd pins of the port are used as SNSK pins
SNSK & SNS pins are on the same port, number of channels = 4	• If the use of SNS(A) and SNSK(A) pins are on the same port, the user should always have the configuration (PA0, PA1), (PA2, PA3), (PA4, PA5) & (PA6, PA7). • In this case channel 0 will be on (PA0, PA1) pins, channel 1 will be on (PA2, PA3) pins and so on up to channel 4 will be on (PA6, PA7) pins. The even pins of the port are used as SNS pins and odd pins of the port are used as SNSK pins
SNSK & SNS pins are on the same port, number of channels = 8 *(Available only for UC3 devices if more than 4 channels are to be used on a single port. For tinyAVR,*	• This configuration is available only for UC3 library variants. • In the use of SNS(A) and SNSK(A) pins are on the same port, the user should always have the configuration (PA0, PA1), (PA2, PA3), (PA4, PA5) & so on. In this case channel 0 will be on (PA0, PA1) pins, channel 1 will be on (PA2, PA3) pins and so on up to channel 8 will be on (PA14,

megaAVR, XMEGA devices upto 8 channels with SNS and SNSK on same ports refer to section 7.1.5.3)	PA15) pins. The even pins of the port are used as SNS pins and odd pins of the port are used as SNSK pins

7.1.5.3 Port combinations supported for two port pair SNS and SNSK pin configurations

This section lists the possible two port pair combinations supported for the various touch library variants. This section can be used as a reference when using the 16 channel libraries for tinyAVR, megaAVR and XMEGA devices.If a device supports a number of ports, not all combination of ports will be supported to be used as SNSK1-SNS1 & SNSK2-SNS2 ports

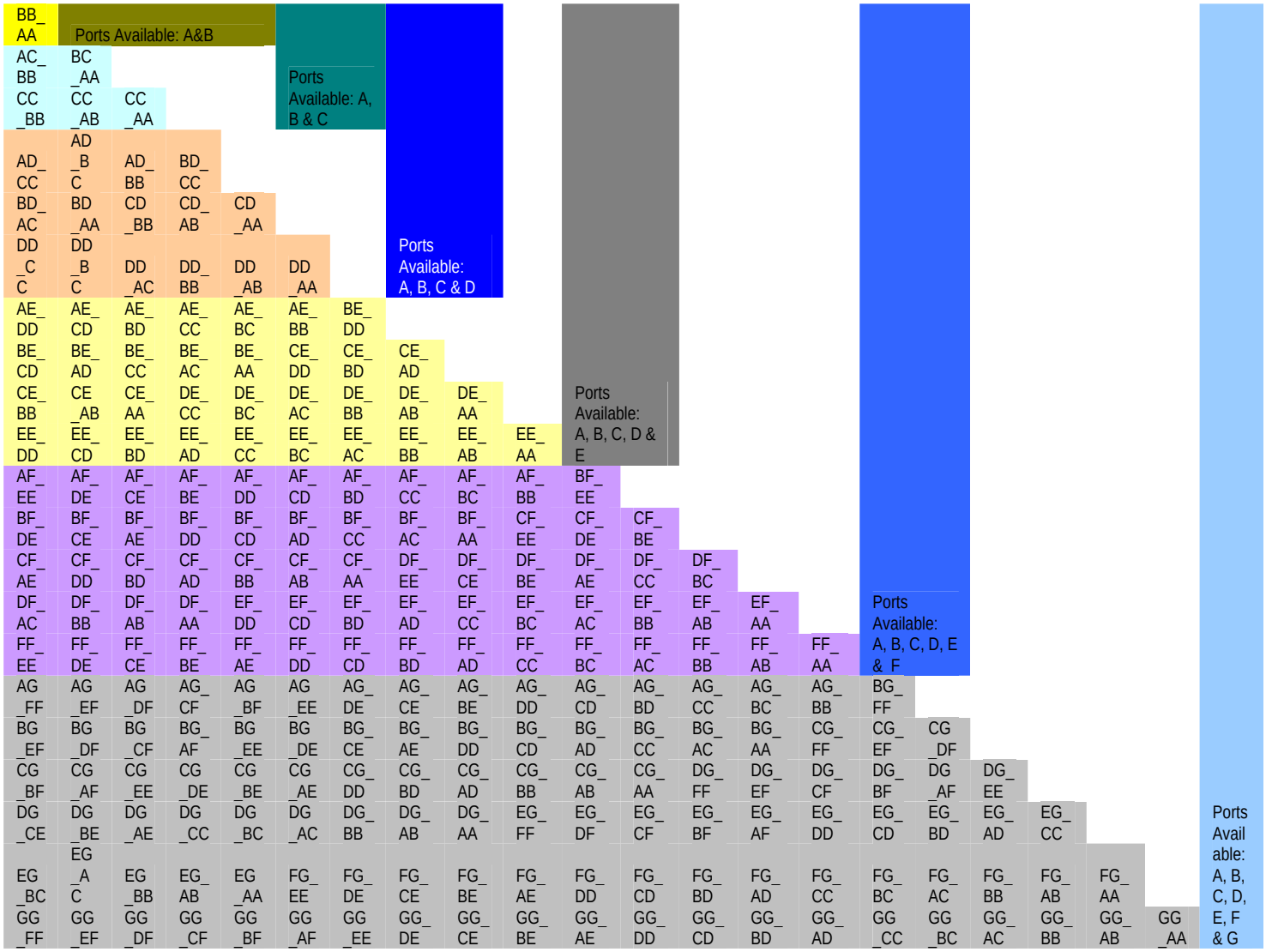
7.1.5.3.1 Port combinations when using ports A to G on devices

The valid SNSK1-SNS1 & SNSK2-SNS2 port combinations for libraries supporting a subset of the ports A, B, C, D, E, F & G are given below

NOTE : Port pair convention followed in the table below : ***MN_OP***

M – Port used for SNSK in pair 1 *N – Port used for SNS in pair 1*

O– Port used for SNSK in pair 2 *P – Port used for SNS in pair 2*



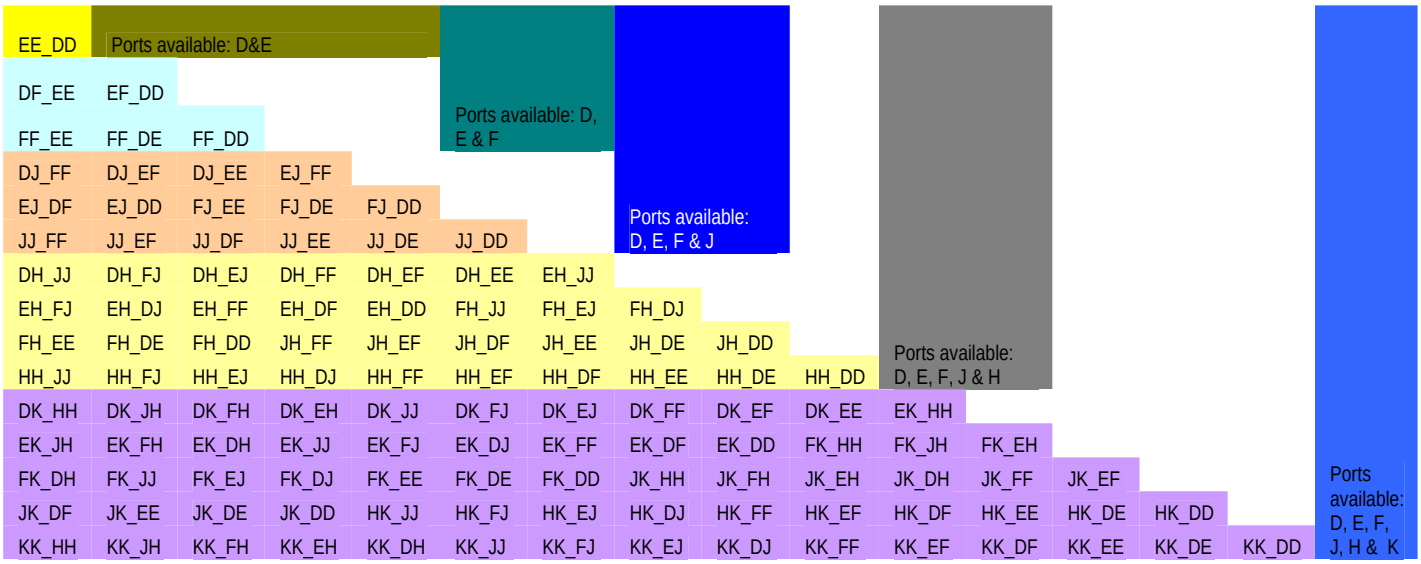
7.1.5.3.2 Port combinations when using ports D to K on devices

The valid SNSK1-SNS1 & SNSK2-SNS2 port combinations for libraries supporting a subset of the ports A, B, C, D, E, F & G are given below

NOTE : Port pair convention followed in the table below : ***MN_OP***

M – Port used for SNSK in pair 1 *N – Port used for SNS in pair 1*

O– Port used for SNSK in pair 2 *P – Port used for SNS in pair 2*



7.1.5.4 Tips on pin assignments for the sensor design using two pairs of SNS / SNSK ports

This section lists tips on selecting the pin assignments when using a single port pair for the SNS and SNSK Pins.

Design choice for the sensor	Example Port configuration with pin assignments
SNSK1-SNS1 & SNSK2-SNS2 pins are all on different ports, number of channels = 16	• e.g, SNS1(D), SNSK1(B) & SNS2(C), SNSK2(A) • Recommended configuration: (PD0, PB0), (PD1, PB1),..(PD7, PB7), (PC0,PA0).. to (PC7, PA7). • In this case channel 0 will be on (PD0, PB0) pins, channel 1 will be on (PD1, PB1) pins, channel 8 will be on (PC0, PA0), channel 9 will be on (PC1, PA1) and so on up to channel 15 will be on (PC7, PA7) pins
SNSK1-SNS1 are on same port & SNSK2-SNS2 pins are on same port, number of channels = 8	• E.g SNS1(K), SNSK1(K) & SNS2(H), SNSK2(H) on same ports, • Recommended configuration: (PK0, PK1), (PK2, PK3),..(PK6, PB7), (PH0,PH1).. to (PH6, PH7). • In this case channel 0 will be on (PK0, PK1) pins, channel 1 will be on (PK2, PK3) pins, channel 4 will be on (PH0, PH1), channel 5 will be on (PH2, PH3) and so on up to channel 7 will be on (PH6, PH7) pins. The even pins of the port are used as SNS pins and odd pins of the port are used as SNSK pins
SNSK1-SNS1 are on different ports & SNSK2-SNS2 pins are on same port, number of channels = 12	• E.g SNS1(H), SNSK1(F) on different ports & SNS2(E), SNSK2(E) on same ports. • Recommended configuration : (PH0, PF0), (PH1, PF1),..(PH7, PF7), (PE0,PE1).. to (PE6, PE7). In this case channel 0 will be on (PH0, PF0) pins, channel 1 will be on (PH1, PF1) pins... channel 8 will be on

	(PE0,PE1), channel 9 will be on (PE2,PE3) and so on up to channel 11 will be on (PH6, PH7) pins. The even pins of the port E are used as SNS pins and odd pins of the port E are used as SNSK pins.
SNSK1-SNS1 are on same port & SNSK2-SNS2 pins are on different ports, number of channels = 12	• E.g SNS1(G), SNSK1(G) on different ports & SNS2(B), SNSK2(D) on same ports • Recommended configuration: (PG0, PG1), (PG2, PG3),..(PG6, PG7), (PB0,PD0)... to (PB7, PD7). • In this case channel 0 will be on (PG0, PG1) pins, channel 1 will be on (PG2, PG3) pins... channel 3 will be on (PG6, PG7), channel 4 will be on (PB0,PD0) and so on up to channel 11 will be on (PB7, PD7) pins. The even pins of the port G are used as SNS pins and odd pins of the port G are used as SNSK pins

7.1.6 Sample applications and Memory requirements for QTouch acquisition method libraries

The memory footprint of the QTouch acquisition method libraries is listed in this section. The section also indicates if a sample project is supplied for the library variant.

7.1.6.1 WinAVR GCC compiler tool chain memory footprint for QTouch acquisition method libraries

Library Variant	Data memory used	Code Memory used	Example Project
libavr25g1_4qt_k_0rs.a	79	3572	avr25g1_qt_example
libavr25g1_4qt_k_1rs.a	100	4518	
libavr25g1_8qt_k_0rs.a	122	3572	
libavr25g1_8qt_k_2rs.a	156	4666	
libavr25g2_4qt_k_0rs.a	79	3572	avr25g2_qt_example
libavr25g2_4qt_k_1rs.a	99	4518	
libavr25g2_8qt_k_0rs.a	122	3572	
libavr25g2_8qt_k_2rs.a	98	4666	
libavr25g2_16qt_k_0rs.a	212	4246	
libavr25g2_16qt_k_4rs.a	271	5348	
libavr35g1_4qt_k_0rs.a	79	3718	avr35g1_qt_example
libavr35g1_4qt_k_1rs.a	99	4582	
libavr35g1_8qt_k_0rs.a	118	3718	
libavr35g1_8qt_k_2rs.a	156	4726	
libavr4g1_4qt_k_0rs.a	79	2982	avr4g1_qt_example
libavr4g1_4qt_k_1rs.a	102	4538	
libavr4g1_8qt_k_0rs.a	123	2982	
libavr4g1_8qt_k_2rs.a	156	4686	
libavr4g1_16qt_k_0rs.a	212	3652	
libavr4g1_16qt_k_4rs.a	271	5368	
libavr4g2_4qt_k_0rs.a	79	2982	avr4g2_qt_example
libavr4g2_4qt_k_1rs.a	99	4538	
libavr4g2_8qt_k_0rs.a	123	2982	
libavr4g2_8qt_k_2rs.a	156	4686	
libavr4g2_16qt_k_0rs.a	212	3652	
libavr4g2_16qt_k_4rs.a	271	5368	

libavr51g1_4qt_k_0rs.a	79	3010	avr51g1_qt_example
libavr51g1_4qt_k_1rs.a	99	4598	
libavr51g1_8qt_k_0rs.a	123	3010	
libavr51g1_8qt_k_2rs.a	156	4742	
libavr51g1_16qt_k_0rs.a	212	3682	
libavr51g1_16qt_k_4rs.a	271	5430	
libavr51g2_4qt_k_0rs.a	79	3010	avr51g2_qt_example
libavr51g2_4qt_k_1rs.a	99	4598	
libavr51g2_8qt_k_0rs.a	123	3010	
libavr51g2_8qt_k_2rs.a	156	4742	
libavr51g2_16qt_k_0rs.a	212	3682	
libavr51g2_16qt_k_4rs.a	271	5430	
libavr5g1_4qt_k_0rs.a	79	3010	avr5g1_qt_example
libavr5g1_4qt_k_1rs.a	99	4598	
libavr5g1_8qt_k_0rs.a	123	3010	
libavr5g1_8qt_k_2rs.a	156	4742	
libavr5g1_16qt_k_0rs.a	212	3682	
libavr5g1_16qt_k_4rs.a	271	5430	
libavr5g2_4qt_k_0rs.a	79	3010	avr5g2_qt_example
libavr5g2_4qt_k_1rs.a	99	4598	
libavr5g2_8qt_k_0rs.a	123	3010	
libavr5g2_8qt_k_2rs.a	156	4742	
libavr5g2_16qt_k_0rs.a	212	3682	
libavr5g2_16qt_k_4rs.a	271	5430	
libavr5g3_4qt_k_0rs.a	79	3010	avr5g3_qt_example
libavr5g3_4qt_k_1rs.a	99	4598	
libavr5g3_8qt_k_0rs.a	123	3010	
libavr5g3_8qt_k_2rs.a	156	4742	
libavr5g3_16qt_k_0rs.a	212	3682	
libavr5g3_16qt_k_4rs.a	271	5430	
libuc3a_8qt_k_0rs.a	130	3846	uc3a_gnu_qt_example
libuc3a_8qt_k_2rs.a	167	5704	
libuc3a_16qt_k_0rs.a	230	3850	
libuc3a_16qt_k_4rs.a	293	5726	
libuc3a_32qt_k_0rs.a	422	3868	
libuc3a_32qt_k_8rs.a	541	5732	
libuc3b_8qt_k_0rs.a	130	3834	uc3b_gnu _qt_example
libuc3b_8qt_k_2rs.a	167	5690	
libuc3b_16qt_k_0rs.a	230	3840	
libuc3b_16qt_k_4rs.a	293	5712	
libuc3b_32qt_k_0rs.a	422	3856	
libuc3b_32qt_k_8rs.a	541	5718	
libXMEGA1_4qt_k_0rs.a	79	3010	xmegag1_gnu_qt_example
libXMEGA1_4qt_k_1rs.a	99	4598	
libXMEGA1_8qt_k_0rs.a	122	3010	
libXMEGA1_8qt_k_2rs.a	156	4742	
libXMEGA1_16qt_k_0rs.a	200	3552	
libXMEGA1_16qt_k_4rs.a	271	5300	
libXMEGA2_4qt_k_0rs.a	79	3010	Xmegag2_gnu_qt_example
libXMEGA2_4qt_k_1rs.a	99	4598	

libXMEGA2_8qt_k_0rs.a	122	3010	
libXMEGA2_8qt_k_2rs.a	156	4742	
libXMEGA2_16qt_k_0rs.a	200	3552	
libXMEGA2_16qt_k_4rs.a	271	5300	

XMEGA1 = avrxmega2g1, avrxmega3g1, avrxmega4g1, avrxmega5g1, avrxmega6g1, avrxmega7g1;
XMEGA2 = avrxmega5g2, avrxmega7g2;

For XMEGA devices, the AVR Studio IDE example project files and applications have been provided only for the ATxmega128A1 device. For other XMEGA devices that are supported: the project files, along with the libraries linked to each project file and the example application needs to be modified accordingly.

7.1.6.2 IAR compiler tool chain memory footprint for QTouch acquisition method libraries

Library	Data memory used	Code memory used	Example Project
libv1g1_4qt_k_0rs.r90	88	2414	v1g1_qt_example
libv1g1_4qt_k_1rs.r90	108	3684	
libv1g1_8qt_k_0rs.r90	132	2424	
libv1g1_8qt_k_2rs.r90	165	3678	
libv1g2_4qt_k_0rs.r90	88	2424	v1g2_qt_example
libv1g2_4qt_k_1rs.r90	108	3684	
libv1g2_8qt_k_0rs.r90	132	2424	
libv1g2_8qt_k_2rs.r90	165	3678	
libv1g2_16qt_k_0rs.r90	226	3034	
libv1g2_16qt_k_4rs.r90	271	4316	
libv1g3_4qt_k_0rs.r90	93	2426	v1g3_qt_example
libv1g3_4qt_k_1rs.r90	113	3680	
libv1g3_8qt_k_0rs.r90	137	2516	
libv1g3_8qt_k_2rs.r90	170	3678	
libv1g3_16qt_k_0rs.r90	226	3010	
libv1g3_16qt_k_4rs.r90	285	4292	
libv1g4_4qt_k_0rs.r90	93	2426	v1g4_qt_example
libv1g4_4qt_k_1rs.r90	113	3680	
libv1g4_8qt_k_0rs.r90	137	2516	
libv1g4_8qt_k_2rs.r90	170	3678	
libv1g4_16qt_k_0rs.r90	226	3010	
libv1g4_16qt_k_4rs.r90	285	4292	
libv3g1_4qt_k_0rs.r90	93	2494	v3g1_qt_example
libv3g1_4qt_k_1rs.r90	113	3794	
libv3g1_8qt_k_0rs.r90	137	2484	
libv3g1_8qt_k_2rs.r90	170	3778	
libv3g1_16qt_k_0rs.r90	226	3076	
libv3g1_16qt_k_4rs.r90	285	4398	
libv3g2_4qt_k_0rs.r90	93	2494	v3g2_qt_example
libv3g2_4qt_k_1rs.r90	113	3794	
libv3g2_8qt_k_0rs.r90	137	2484	
libv3g2_8qt_k_2rs.r90	170	3778	

libv3g2_16qt_k_0rs.r90	226	3076	
libv3g2_16qt_k_4rs.r90	285	4398	
libv3g3_4qt_k_0rs.r90	93	2494	v3g3_qt_example
libv3g3_4qt_k_1rs.r90	113	3794	
libv3g3_8qt_k_0rs.r90	137	2484	
libv3g3_8qt_k_2rs.r90	170	3778	
libv3g3_16qt_k_0rs.r90	226	3076	
libv3g3_16qt_k_4rs.r90	285	4398	
libv3g4_4qt_k_0rs.r90	93	2499	v3g4_qt_example
libv3g4_4qt_k_1rs.r90	113	3800	
libv3g4_8qt_k_0rs.r90	137	2489	
libv3g4_8qt_k_2rs.r90	170	3784	
libv3g4_16qt_k_0rs.r90	226	3080	
libv3g4_16qt_k_4rs.r90	285	4403	
libv3g5_4qt_k_0rs.r90	93	2499	v3g5_qt_example
libv3g5_4qt_k_1rs.r90	113	3800	
libv3g5_8qt_k_0rs.r90	137	2489	
libv3g5_8qt_k_2rs.r90	170	3784	
libv3g5_16qt_k_0rs.r90	226	3080	
libv3g5_16qt_k_4rs.r90	285	4403	
libv3g6_4qt_k_0rs.r90	88	2498	v3g6_qt_example
libv3g6_4qt_k_1rs.r90	108	3810	
libv3g6_8qt_k_0rs.r90	132	2484	
libv3g6_8qt_k_2rs.r90	165	3802	
libuc3a_8qt_k_0rs.r82	139	2518	uc3a_iar_qt_example
libuc3a_8qt_k_2rs.r82	176	3948	
libuc3a_16qt_k_0rs.r82	235	2526	
libuc3a_16qt_k_4rs.r82	300	3956	
libuc3a_32qt_k_0rs.r82	427	2542	
libuc3a_32qt_k_8rs.r82	548	3962	
libuc3b_8qt_k_0rs.r82	139	2518	uc3b_iar_qt_example
libuc3b_8qt_k_2rs.r82	176	3948	
libuc3b_16qt_k_0rs.r82	235	2526	
libuc3b_16qt_k_4rs.r82	300	3956	
libuc3b_32qt_k_0rs.r82	427	2542	
libuc3b_32qt_k_8rs.r82	548	3962	
libv3xmsfg1_4qt_k_0rs.r90	87	2584	xmegag1_iar_qt_example
libv3xmsfg1_4qt_k_1rs.r90	107	3976	
libv3xmsfg1_8qt_k_0rs.r90	131	2575	
libv3xmsfg1_8qt_k_2rs.r90	164	3968	
libv3xmsfg1_16qt_k_0rs.r90	220	3034	
libv3xmsfg1_16qt_k_4rs.r90	279	4467	
libv3xmg1_4qt_k_0rs.r90	87	2450	xmegag1_iar_qt_example

libv3xmg1_4qt_k_1rs.r90	107	3852	
libv3xmg1_8qt_k_0rs.r90	131	2441	
libv3xmg1_8qt_k_2rs.r90	164	3744	
libv3xmg1_16qt_k_0rs.r90	220	2900	
libv3xmg1_16qt_k_4rs.r90	279	4242	
libv4xmg1_4qt_k_0rs.r90	85	2594	xmegag1_iar_qt_example
libv4xmg 1_4qt_k_1rs.r90	105	3986	
libv4xmg 1_8qt_k_0rs.r90	129	2585	
libv4xmg 1_8qt_k_2rs.r90	162	3978	
libv4xmg 1_16qt_k_0rs.r90	218	3044	
libv4xmg 1_16qt_k_4rs.r90	277	4477	
libv5xmg 1_4qt_k_0rs.r90	87	2450	xmegag1_iar_qt_example
libv5xmg 1_4qt_k_1rs.r90	107	3852	
libv5xmg 1_8qt_k_0rs.r90	131	2441	
libv5xmg 1_8qt_k_2rs.r90	164	3744	
libv5xmg 1_16qt_k_0rs.r90	220	2900	
libv5xmg 1_16qt_k_4rs.r90	279	4243	
libv6xmg 1_4qt_k_0rs.r90	87	2594	xmegag1_iar_qt_example
libv6xmg 1_4qt_k_1rs.r90	107	3986	
libv6xmg 1_8qt_k_0rs.r90	131	2585	
libv6xmg 1_8qt_k_2rs.r90	164	3978	
libv6xmg 1_16qt_k_0rs.r90	220	3044	
libv6xmg 1_16qt_k_4rs.r90	279	4477	
libv4xmg1_4qt_k_0rs.r90	85	2594	Xmegag2_iar_qt_example
libv4xmg 1_4qt_k_1rs.r90	105	3986	
libv4xmg 1_8qt_k_0rs.r90	129	2585	
libv4xmg 1_8qt_k_2rs.r90	162	3978	
libv4xmg 1_16qt_k_0rs.r90	218	3044	
libv4xmg 1_16qt_k_4rs.r90	277	4477	
libv6xmg 1_4qt_k_0rs.r90	87	2594	Xmegag2_iar_qt_example
libv6xmg 1_4qt_k_1rs.r90	107	3986	
libv6xmg 1_8qt_k_0rs.r90	131	2585	
libv6xmg 1_8qt_k_2rs.r90	164	3978	
libv6xmg 1_16qt_k_0rs.r90	220	3044	
libv6xmg 1_16qt_k_4rs.r90	279	4477	

For XMEGA devices, the IAR IDE example project files and applications have been provided only for the ATxmega128A1 device. For other XMEGA devices that are supported: the project files, along with the libraries linked to each project file and the example application needs to be modified accordingly.

7.2 QMatrix acquisition method library variants

7.2.1 Introduction

Variants of the ATMEL QTouch Library based on Matrix™ acquisition technology are available for a range of ATMEL Microcontrollers. This section lists the different variants available. By following a simple series of steps, the user can identify the right library variant to use in his application.

7.2.2 Support for different compiler tool chains

The QMatrix acquisition method libraries are supported for the following compiler tool chains.

Tool	Version
IAR Compiler	5.30.6
IAR Embedded Workbench	5.3.6.927
GCC – AVR Studio	4.18 build 673
WinAVR	20090313

7.2.3 QMatrix Acquisition method library naming conventions

The libraries are named according the naming convention listed below

Tool Chain	Naming convention
GCC Tool Chain	lib<***D***>_<***NC***>qm_<***NX***>x_<***NY***>y_<***CFG***>_<***NRS***>rs_<*Y*_***LINES***>.a
IAR –EWAR	lib<***D***>_<***NC***>qm_<***NX***>x_<***NY***>y_<***CFG***>_<***NRS***>rs_< *Y*_***LINES*** >.r90

Field name	Possible values	Comments
D	**Common for IAR & GCC:** t88, t167, m164p, m324p, m128rfa1, m8535 **Specific to IAR:** m16a m88pa m324pa v3xm (ATxmega64A3) v4xm(ATxmega64A1) v5xm(ATxmega128A3, ATxmega192A3, ATxmega256A3, ATxmega256A3B) v6xm(ATxmega128A1) v3g3 (ATmega165P,	Indicates the device / core group name in short form. For XMEGA Devices, Core groups are taken which follows As below for both GCC and IAR Supported XMEGA Devices ATxmega64A1 ATxmega128A1 ATxmega64A3 ATxmega128A3, ATxmega192A3, ATxmega256A3, ATxmega256A3B)

	ATmega325P, ATmega645) **Specific to GCC:** m16 m88p avrxmega4 (ATxmega64A3) avrxmega5(ATxmega64A1) avrxmega6(ATxmega128A3, ATxmega192A3, ATxmega256A3, ATxmega256A3B) avrxmega7(ATxmega128A1) avr5g3 (ATmega165P, ATmega325P, ATmega645)	
NC	8,16,32,64	Indicates the maximum number of channels that the library supports
NX	4,8	Indicates the number of X-Lines that the library needs for supporting the listed number of channels
NY	2,4,8	Indicates the number of Y-Lines that the library needs for supporting the listed number of channels
CFG	k krs	k – library variant supports only keys krs – library variant supports keys, Rotors and Sliders
NRS	0,2,4,8	Maximum number of rotor sliders that the library supports.
Y_LINES	YL_LO_NIB YL_HI_NIB	Indicates the beginning bit position of the ports on which Y-Lines are supported by the library variant **If Y_LINES is YL_LO_NIB** - • If 2 Y-lines are used, Y-Lines are on pins 0,1 • If 4 Y-lines are used, Y-Lines are on pins 0,1,2,3. • If 8 Y-lines are used, Y-Lines are on pins from 0 to 7. **If Y_LINES is YL_HI_NIB** - • If 2 Y-lines are used, Y-Lines are on pins 4,5 • If 4 Y-lines are used, Y-Lines are on pins 4,5,6,7 • If 8 Y-lines are used, the libraries are named as YL_LO_NIB as suffix since all the 8 port pins would be used.

The table below provides a few examples of the naming convention.

Example Library name	Configuration supported
libm164p_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	• Compiler tool chain : GCC • Device : ATMega164P • 8 Channels • 4 X lines • 2 Y lines • Supports Keys, Rotors and Sliders (krs)

	• 2 Rotors and Sliders • YL-HI_NB : Indicates that Yline pins 4 and 5 will be used
libt88_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	• Compiler tool chain : IAR • Device : ATTiny88 • 16 Channels • 8 X lines • 2 Y lines • Supports only keys (k) • 0 Rotors and Sliders • YL-LO_NB : Indicates that Yline pins 0 and 1 will be used

7.2.4 QMatrix acquisition method library variants

7.2.4.1 Devices supported for QMatrix Acquisition

The following table lists all the devices for which QMatrix acquisition method libraries are provided

Support is provided for both IAR and GCC compiler tool chains unless explicitly noted otherwise.

ATMEL Device Name For which QMatrix Acquisition is supported	**Supported Channels**	**Comments**
ATtiny88	8,16	
ATtiny167	8,16	
ATmega88PA (IAR),ATmega88P (GCC)	8,16,32	
ATmega8535	8,16	
ATmega16A (IAR) ,ATmega16 (GCC)	8,16,32	
ATmega164P	8,16,32	
ATmega165P	8,16,32	
ATmega325P	8,16,32	
ATmega645	8,16,32	
ATmega324P	8,16,32,64	
ATmega324PA	8,16,32,64	Support provided only for IAR compiler
ATmega128RFA1	8,16,32,64	
ATxmega64A1 ATxmega128A1 ATxmega64A3 ATxmega128A3 ATxmega192A3	8,16,32	Possible Y- Lines only on 0,1 (NUM_Y_LINES=2) or 0,1,2,3 (NUM_Y_LINES=4) pins of the port assigned based on number of Y-Lines

ATxmega256A3 ATxmega256A3B		

7.2.4.2 QMatrix acquisition method Library Variants for the devices

The following QMatrix acquisition method library variants are supported along with the code and data memory footprint for the devices listed for QTouch library release 3.2

7.2.4.2.1 QMatrix acquisition method Library Variants for ATtiny88

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATtiny88							
8	4x2	0	0,1	IAR	libt88_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2162	195
				GCC	libt88_8qm_4x_2y_k_0rs_YL_LO_NIB.a	3360	146
8	4x2	0	4,5	IAR	libt88_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2162	195
				GCC	libt88_8qm_4x_2y_k_0rs_YL_HI_NIB.a	3360	146
8	4x2	2	0,1	IAR	libt88_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3450	236
				GCC	libt88_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4332	194
8	4x2	2	4,5	IAR	libt88_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3450	236
				GCC	libt88_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4332	194
16	4x4	0	0,1,2,3	IAR	libt88_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2210	246
				GCC	libt88_16qm_4x_4y_k_0rs_YL_LO_NIB.a	3258	240
16	4x4	2	0,1,2,3	IAR	libt88_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3220	292
				GCC	libt88_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4340	309
16	4x4	4	0,1,2,3	IAR	libt88_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3220	316
				GCC	libt88_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4340	339
16	8x2	0	0,1	IAR	libt88_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2220	260
				GCC	libt88_16qm_8x_2y_k_0rs_YL_LO_NIB.a	3254	254
16	8x2	0	4,5	IAR	libt88_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2220	260
				GCC	libt88_16qm_8x_2y_k_0rs_YL_HI_NIB.a	3254	254
16	8x2	2	0,1	IAR	libt88_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3500	308
				GCC	libt88_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4340	329
16	8x2	2	4,5	IAR	libt88_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3500	308
				GCC	libt88_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4340	329
16	8x2	4	0,1	IAR	libt88_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3500	334
				GCC	libt88_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4340	349
16	8x2	4	4,5	IAR	libt88_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3500	334
				GCC	libt88_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4340	349

7.2.4.2.2 QMatrix™ acquisition method Library Variants for ATtiny167

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATtiny167							
8	4x2	0	0,1	IAR	libt167_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	160	2164
				GCC	libt167_8qm_4x_2y_k_0rs_YL_HI_NIB.a	140	3326
8	4x2	0	4,5	IAR	libt167_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	202	3535
				GCC	libt167_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	195	4342

7.2.4.2.3 QMatrix acquisition method Library Variants for ATmega88P/ATmega88PA

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATmega88P/ATmega88PA							
8	4x2	0	0,1	IAR	libm88pa_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2130	195
				GCC	libm88p_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2760	146
8	4x2	0	4,5	IAR	libm88pa_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2130	195
				GCC	libm88p_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2760	146
8	4x2	2	0,1	IAR	libm88pa_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3410	236
				GCC	libm88p_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4332	194
8	4x2	2	4,5	IAR	libm88pa_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3410	236
				GCC	libm88p_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4332	194
16	4x4	0	0,1,2,3	IAR	libm88pa_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2160	246
				GCC	libm88p_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2758	240
16	4x4	2	0,1,2,3	IAR	libm88pa_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3180	292
				GCC	libm88p_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4340	309
16	4x4	4	0,1,2,3	IAR	libm88pa_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3180	316
				GCC	libm88p_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4340	339
16	8x2	0	0,1	IAR	libm88pa_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2180	260
				GCC	libm88p_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2754	254
16	8x2	0	4,5	IAR	libm88pa_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2180	260
				GCC	libm88p_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2754	254
16	8x2	2	0,1	IAR	libm88pa_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3460	308
				GCC	libm88p_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4340	329
16	8x2	2	4,5	IAR	libm88pa_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3460	308
				GCC	libm88p_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4340	329
16	8x2	4	0,1	IAR	libm88pa_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3460	334

				GCC	libm88p_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4340	349
16	8x2	4	4,5	IAR	libm88pa_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3460	334
				GCC	libm88p_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4340	349
32	8x4	0	0,1,2,3	IAR	libm88pa_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2240	566
				GCC	libm88p_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2790	456
32	8x4	4	0,1,2,3	IAR	libm88pa_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3510	696
				GCC	libm88p_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4394	571

7.2.4.2.4 QMatrix acquisition method Library Variants for ATmega8535

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATmega8535							
8	4x2	0	0,1	IAR	libm8535_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2110	195
				GCC	libm8535_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2740	146
8	4x2	0	4,5	IAR	libm8535_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2110	195
				GCC	libm8535_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2740	146
8	4x2	2	0,1	IAR	libm8535_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3390	236
				GCC	libm8535_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4312	194
8	4x2	2	4,5	IAR	libm8535_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3390	236
				GCC	libm8535_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4312	194
16	4x4	0	0,1,2,3	IAR	libm8535_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2140	246
				GCC	libm8535_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2738	240
16	4x4	0	4,5,6,7	IAR	libm8535_16qm_4x_4y_k_0rs_YL_HI_NIB.r90	2140	246
				GCC	libm8535_16qm_4x_4y_k_0rs_YL_HI_NIB.a	2738	240
16	4x4	2	0,1,2,3	IAR	libm8535_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3160	292
				GCC	libm8535_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4320	309
16	4x4	2	4,5,6,7	IAR	libm8535_16qm_4x_4y_krs_2rs_YL_HI_NIB.r90	3160	292
				GCC	libm8535_16qm_4x_4y_krs_2rs_YL_HI_NIB.a	4320	309
16	4x4	4	0,1,2,3	IAR	libm8535_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3160	316
				GCC	libm8535_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4320	339
16	4x4	4	4,5,6,7	IAR	libm8535_16qm_4x_4y_krs_4rs_YL_HI_NIB.r90	3160	316
				GCC	libm8535_16qm_4x_4y_krs_4rs_YL_HI_NIB.a	4320	339
16	8x2	0	0,1	IAR	libm8535_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2160	260
				GCC	libm8535_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2734	254
16	8x2	0	4,5	IAR	libm8535_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2160	260
				GCC	libm8535_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2734	254
16	8x2	2	0,1	IAR	libm8535_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3440	308
				GCC	libm8535_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4320	329
16	8x2	2	4,5	IAR	libm8535_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3440	308
				GCC	libm8535_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4320	329
16	8x2	4	0,1	IAR	libm8535_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3440	334
				GCC	libm8535_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4320	349
16	8x2	4	4,5	IAR	libm8535_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3440	334
				GCC	libm8535_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4320	349

7.2.4.2.5 QMatrix acquisition method Library Variants for ATmega16/ATmega16A

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATmega16/ATmega16A							
8	4x2	0	0,1	IAR	libm16a_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2170	195
				GCC	libm16_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2770	146
8	4x2	0	4,5	IAR	libm16a_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2170	195
				GCC	libm16_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2770	146
8	4x2	2	0,1	IAR	libm16a_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3490	236
				GCC	libm16_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4382	194
8	4x2	2	4,5	IAR	libm16a_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3490	236
				GCC	libm16_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4382	194
16	4x4	0	0,1,2,3	IAR	libm16a_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2200	246
				GCC	libm16_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2768	240
16	4x4	0	4,5,6,7	IAR	libm16a_16qm_4x_4y_k_0rs_YL_HI_NIB.r90	2200	246
				GCC	libm16_16qm_4x_4y_k_0rs_YL_HI_NIB.a	2768	240
16	4x4	2	0,1,2,3	IAR	libm16a_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3520	292
				GCC	libm16_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4390	309
16	4x4	2	4,5,6,7	IAR	libm16a_16qm_4x_4y_krs_2rs_YL_HI_NIB.r90	3520	292
				GCC	libm16_16qm_4x_4y_krs_2rs_YL_HI_NIB.a	4390	309
16	4x4	4	0,1,2,3	IAR	libm16a_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3520	316
				GCC	libm16_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4390	339
16	4x4	4	4,5,6,7	IAR	libm16a_16qm_4x_4y_krs_4rs_YL_HI_NIB.r90	3520	316
				GCC	libm16_16qm_4x_4y_krs_4rs_YL_HI_NIB.a	4390	339
16	8x2	0	0,1	IAR	libm16a_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2220	260
				GCC	libm16_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2764	254
16	8x2	0	4,5	IAR	libm16a_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2220	260
				GCC	libm16_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2764	254
16	8x2	2	0,1	IAR	libm16a_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3540	308
				GCC	libm16_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4390	329
16	8x2	2	4,5	IAR	libm16a_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3540	308
				GCC	libm16_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4390	329
16	8x2	4	0,1	IAR	libm16a_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3540	334
				GCC	libm16_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4390	349
16	8x2	4	4,5	IAR	libm16a_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3520	334
				GCC	libm16_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4390	349
32	8x4	0	0,1,2,3	IAR	libm16a_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2280	566
				GCC	libm16_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2800	456
32	8x4	0	4,5,6,7	IAR	libm16a_32qm_8x_4y_k_0rs_YL_HI_NIB.r90	2280	566
				GCC	libm16_32qm_8x_4y_k_0rs_YL_HI_NIB.a	2800	456
32	8x4	4	0,1,2,3	IAR	libm16a_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3590	696
				GCC	libm16_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4434	571
32	8x4	4	4,5,6,7	IAR	libm16a_32qm_8x_4y_krs_4rs_YL_HI_NIB.r90	3590	696
				GCC	libm16_32qm_8x_4y_krs_4rs_YL_HI_NIB.a	4434	571

7.2.4.2.6 QMatrix acquisition method Library Variants for ATmega164P

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATmega164P							
8	4x2	0	0,1	IAR	libm164p_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2170	195
				GCC	libm164p_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2795	146
8	4x2	0	4,5	IAR	libm164p_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2170	195
				GCC	libm164p_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2795	146
8	4x2	2	0,1	IAR	libm164p_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3490	236
				GCC	libm164p_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4407	194
8	4x2	2	4,5	IAR	libm164p_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3490	236
				GCC	libm164p_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4407	194
16	4x4	0	0,1,2,3	IAR	libm164p_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2200	246
				GCC	libm164p_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2793	240
16	4x4	0	4,5,6,7	IAR	libm164p_16qm_4x_4y_k_0rs_YL_HI_NIB.r90	2200	246
				GCC	libm164p_16qm_4x_4y_k_0rs_YL_HI_NIB.a	2793	240
16	4x4	2	0,1,2,3	IAR	libm164p_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3520	292
				GCC	libm164p_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4415	309
16	4x4	2	4,5,6,7	IAR	libm164p_16qm_4x_4y_krs_2rs_YL_HI_NIB.r90	3520	292
				GCC	libm164p_16qm_4x_4y_krs_2rs_YL_HI_NIB.a	4415	309
16	4x4	4	0,1,2,3	IAR	libm164p_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3520	316
				GCC	libm164p_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4415	339
16	4x4	4	4,5,6,7	IAR	libm164p_16qm_4x_4y_krs_4rs_YL_HI_NIB.r90	3520	316
				GCC	libm164p_16qm_4x_4y_krs_4rs_YL_HI_NIB.a	4415	339
16	8x2	0	0,1	IAR	libm164p_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2220	260
				GCC	libm164p_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2790	254
16	8x2	0	4,5	IAR	libm164p_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2220	260
				GCC	libm164p_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2790	254
16	8x2	2	0,1	IAR	libm164p_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3540	308
				GCC	libm164p_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4415	329
16	8x2	2	4,5	IAR	libm164p_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3540	308
				GCC	libm164p_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4415	329
16	8x2	4	0,1	IAR	libm164p_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3540	334
				GCC	libm164p_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4415	349
16	8x2	4	4,5	IAR	libm164p_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3520	334
				GCC	libm164p_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4415	349
32	8x4	0	0,1,2,3	IAR	libm164p_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2280	566
				GCC	libm164p_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2825	456
32	8x4	0	4,5,6,7	IAR	libm164p_32qm_8x_4y_k_0rs_YL_HI_NIB.r90	2280	566
				GCC	libm164p_32qm_8x_4y_k_0rs_YL_HI_NIB.a	2825	456
32	8x4	4	0,1,2,3	IAR	libm164p_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3590	696
				GCC	libm164p_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4470	571
32	8x4	4	4,5,6,7	IAR	libm164p_32qm_8x_4y_krs_4rs_YL_HI_NIB.r90	3590	696
				GCC	libm164p_32qm_8x_4y_krs_4rs_YL_HI_NIB.a	4470	571

7.2.4.2.7 QMatrix acquisition method Library Variants for v3g3/avr5g3 (ATmega165P,ATmega325P,ATmega645)

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATMega165P/ ATmega325P/ATmega645							
8	4x2	0	0,1	IAR	libv3g3_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2170	195
				GCC	libavr5g3_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2795	146
8	4x2	0	4,5	IAR	libv3g3_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2170	195
				GCC	libavr5g3_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2795	146
8	4x2	2	0,1	IAR	libv3g3_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3490	236
				GCC	libavr5g3_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4407	194
8	4x2	2	4,5	IAR	libv3g3_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3490	236
				GCC	libavr5g3_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4407	194
16	4x4	0	0,1,2,3	IAR	libv3g3_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2200	246
				GCC	libavr5g3_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2793	240
16	4x4	0	4,5,6,7	IAR	libv3g3_16qm_4x_4y_k_0rs_YL_HI_NIB.r90	2200	246
				GCC	libavr5g3_16qm_4x_4y_k_0rs_YL_HI_NIB.a	2793	240
16	4x4	2	0,1,2,3	IAR	libv3g3_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3520	292
				GCC	libavr5g3_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4415	309
16	4x4	2	4,5,6,7	IAR	libv3g3_16qm_4x_4y_krs_2rs_YL_HI_NIB.r90	3520	292
				GCC	libavr5g3_16qm_4x_4y_krs_2rs_YL_HI_NIB.a	4415	309
16	4x4	4	0,1,2,3	IAR	libv3g3_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3520	316
				GCC	libavr5g3_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4415	339
16	4x4	4	4,5,6,7	IAR	libv3g3_16qm_4x_4y_krs_4rs_YL_HI_NIB.r90	3520	316
				GCC	libavr5g3_16qm_4x_4y_krs_4rs_YL_HI_NIB.a	4415	339
16	8x2	0	0,1	IAR	libv3g3_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2220	260
				GCC	libavr5g3_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2790	254
16	8x2	0	4,5	IAR	libv3g3_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2220	260
				GCC	libavr5g3_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2790	254
16	8x2	2	0,1	IAR	libv3g3_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3540	308
				GCC	libavr5g3_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4415	329
16	8x2	2	4,5	IAR	libv3g3_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3540	308
				GCC	libavr5g3_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4415	329
16	8x2	4	0,1	IAR	libv3g3 _16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3540	334
				GCC	libavr5g3_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4415	349
16	8x2	4	4,5	IAR	libv3g3 _16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3520	334
				GCC	libavr5g3_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4415	349
32	8x4	0	0,1,2,3	IAR	libv3g3 _32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2280	566
				GCC	libavr5g3_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2825	456
32	8x4	0	4,5,6,7	IAR	libv3g3 _32qm_8x_4y_k_0rs_YL_HI_NIB.r90	2280	566
				GCC	libavr5g3_32qm_8x_4y_k_0rs_YL_HI_NIB.a	2825	456
32	8x4	4	0,1,2,3	IAR	libv3g3 _32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3590	696
				GCC	libavr5g3_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4470	571
32	8x4	4	4,5,6,7	IAR	libv3g3 _32qm_8x_4y_krs_4rs_YL_HI_NIB.r90	3590	696
				GCC	libavr5g3_32qm_8x_4y_krs_4rs_YL_HI_NIB.a	4470	571
64	8x8	0	0,1,2,3,	IAR	libv3g3_64qm_8x_8y_k_0rs_YL_LO_NIB.r90	2280	966

			4,5,6,7	GCC	libavr5g3_64qm_8x_8y_k_0rs_YL_LO_NIB.a	2825	856
64	8x8	4	0,1,2,3, 4,5,6,7	IAR	libv3g3_64qm_8x_8y_krs_4rs_YL_LO_NIB.r90	3590	1150
				GCC	libavr5g3_64qm_8x_8y_krs_4rs_YL_LO_NIB.a	4470	1020
64	8x8	8	0,1,2,3, 4,5,6,7	IAR	libv3g3_64qm_8x_8y_krs_8rs_YL_LO_NIB.r90	3590	1182
				GCC	libavr5g3_64qm_8x_8y_krs_8rs_YL_LO_NIB.a	4470	1052

7.2.4.2.8 QMatrix acquisition method Library Variants for ATmega324P

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATMega324P							
8	4x2	0	0,1	IAR	libm324p_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2170	195
				GCC	libm324p_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2795	146
8	4x2	0	4,5	IAR	libm324p_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2170	195
				GCC	libm324p_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2795	146
8	4x2	2	0,1	IAR	libm324p_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3490	236
				GCC	libm324p_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4407	194
8	4x2	2	4,5	IAR	libm324p_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3490	236
				GCC	libm324p_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4407	194
16	4x4	0	0,1,2,3	IAR	libm324p_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2200	246
				GCC	libm324p_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2793	240
16	4x4	0	4,5,6,7	IAR	libm324p_16qm_4x_4y_k_0rs_YL_HI_NIB.r90	2200	246
				GCC	libm324p_16qm_4x_4y_k_0rs_YL_HI_NIB.a	2793	240
16	4x4	2	0,1,2,3	IAR	libm324p_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3520	292
				GCC	libm324p_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4415	309
16	4x4	2	4,5,6,7	IAR	libm324p_16qm_4x_4y_krs_2rs_YL_HI_NIB.r90	3520	292
				GCC	libm324p_16qm_4x_4y_krs_2rs_YL_HI_NIB.a	4415	309
16	4x4	4	0,1,2,3	IAR	libm324p_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3520	316
				GCC	libm324p_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4415	339
16	4x4	4	4,5,6,7	IAR	libm324p_16qm_4x_4y_krs_4rs_YL_HI_NIB.r90	3520	316
				GCC	libm324p_16qm_4x_4y_krs_4rs_YL_HI_NIB.a	4415	339
16	8x2	0	0,1	IAR	libm324p_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2220	260
				GCC	libm324p_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2790	254
16	8x2	0	4,5	IAR	libm324p_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2220	260
				GCC	libm324p_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2790	254
16	8x2	2	0,1	IAR	libm324p_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3540	308
				GCC	libm324p_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4415	329
16	8x2	2	4,5	IAR	libm324p_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3540	308
				GCC	libm324p_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4415	329
16	8x2	4	0,1	IAR	libm324p_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3540	334
				GCC	libm324p_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4415	349
16	8x2	4	4,5	IAR	libm324p_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3520	334
				GCC	libm324p_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4415	349
32	8x4	0	0,1,2,3	IAR	libm324p_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2280	566
				GCC	libm324p_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2825	456
32	8x4	0	4,5,6,7	IAR	libm324p_32qm_8x_4y_k_0rs_YL_HI_NIB.r90	2280	566

				GCC	libm324p_32qm_8x_4y_k_0rs_YL_HI_NIB.a	2825	456
32	8x4	4	0,1,2,3	IAR	libm324p_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3590	696
				GCC	libm324p_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4470	571
32	8x4	4	4,5,6,7	IAR	libm324p_32qm_8x_4y_krs_4rs_YL_HI_NIB.r90	3590	696
				GCC	libm324p_32qm_8x_4y_krs_4rs_YL_HI_NIB.a	4470	571
64	8x8	0	0,1,2,3,4, 5,6,7	IAR	libm324p_64qm_8x_8y_k_0rs_YL_LO_NIB.r90	2320	966
				GCC	libm324p_64qm_8x_8y_k_0rs_YL_LO_NIB.a	2825	856
64	8x8	4	0,1,2,3, 4,5,6,7	IAR	libm324p_64qm_8x_8y_krs_4rs_YL_LO_NIB.r90	3640	1156
				GCC	libm324p_64qm_8x_8y_krs_4rs_YL_LO_NIB.a	4470	1020
64	8x8	8	0,1,2,3, 4,5,6,7	IAR	libm324p_64qm_8x_8y_krs_8rs_YL_LO_NIB.r90	3640	1206
				GCC	libm324p_64qm_8x_8y_krs_8rs_YL_LO_NIB.a	4470	1080

7.2.4.2.9 QMatrix acquisition method Library Variants for ATmega324PA

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATmega324PA							
8	4x2	0	0,1	IAR	libm324pa_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2170	195
8	4x2	0	4,5	IAR	libm324pa_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2170	195
8	4x2	2	0,1	IAR	libm324pa_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3490	236
8	4x2	2	4,5	IAR	libm324pa_8qm_4x_2y_krs_2rs_YL_HI_NIB.r90	3490	236
16	4x4	0	0,1,2,3	IAR	libm324pa_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2200	246
16	4x4	0	4,5,6,7	IAR	libm324pa_16qm_4x_4y_k_0rs_YL_HI_NIB.r90	2200	246
16	4x4	2	0,1,2,3	IAR	libm324pa_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3520	292
16	4x4	2	4,5,6,7	IAR	libm324pa_16qm_4x_4y_krs_2rs_YL_HI_NIB.r90	3520	292
16	4x4	4	0,1,2,3	IAR	libm324pa_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3520	316
16	4x4	4	4,5,6,7	IAR	libm324pa_16qm_4x_4y_krs_4rs_YL_HI_NIB.r90	3520	316
16	8x2	0	0,1	IAR	libm324pa_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2220	260
16	8x2	0	4,5	IAR	libm324pa_16qm_8x_2y_k_0rs_YL_HI_NIB.r90	2220	260
16	8x2	2	0,1	IAR	libm324pa_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3540	308
16	8x2	2	4,5	IAR	libm324pa_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3540	308
16	8x2	4	0,1	IAR	libm324pa_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3540	334
16	8x2	4	4,5	IAR	libm324pa_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3520	334
32	8x4	0	0,1,2,3	IAR	libm324pa_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2280	566
32	8x4	0	4,5,6,7	IAR	libm324pa_32qm_8x_4y_k_0rs_YL_HI_NIB.r90	2280	566
32	8x4	4	0,1,2,3	IAR	libm324pa_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3590	696
32	8x4	4	4,5,6,7	IAR	libm324pa_32qm_8x_4y_krs_4rs_YL_HI_NIB.r90	3590	696
64	8x8	0	0,1,2,3, 4,5,6,7	IAR	libm324pa_64qm_8x_8y_k_0rs_YL_LO_NIB.r90	2320	966

64	8x8	4	0,1,2,3, 4,5,6,7	IAR	libm324pa_64qm_8x_8y_krs_4rs_YL_LO_NIB.r90	3640	1156
64	8x8	8	0,1,2,3, 4,5,6,7	IAR	libm324pa_64qm_8x_8y_krs_8rs_YL_LO_NIB.r90	3640	1206

7.2.4.2.10 QMatrix acquisition method Library Variants for ATmega128RFA1

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATMega128RFA1							
8	4x2	0	0,1	IAR	libm128rfa1_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2126	145
				GCC	libm128rfa1_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2790	141
8	4x2	0	4,5	IAR	libm128rfa1_8qm_4x_2y_k_0rs_YL_HI_NIB.r90	2126	145
				GCC	libm128rfa1_8qm_4x_2y_k_0rs_YL_HI_NIB.a	2790	141
8	4x2	2	0,1	IAR	libm128rfa1_8qm_4x_2y_krs_2rs_YL_LO_NIB.r 90	3468	198
				GC C	libm128rfa1_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4402	194
8	4x2	2	4,5	IAR	libm128rfa1_8qm_4x_2y_krs_2rs_YL_HI_NIB.r9 0	3468	198
				GCC	libm128rfa1_8qm_4x_2y_krs_2rs_YL_HI_NIB.a	4402	194
16	4x4	0	0,1,2,3	IAR	libm128rfa1_16qm_4x_4y_k_0rs_YL_LO_NIB.r9 0	2146	242
				GCC	libm128rfa1_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2794	240
16	4x4	0	4,5,6,7	IAR	libm128rfa1_16qm_4x_4y_k_0rs_YL_HI_NIB.r9 0	2146	242
				GCC	libm128rfa1_16qm_4x_4y_k_0rs_YL_HI_NIB.a	2794	240
16	4x4	2	0,1,2,3	IAR	libm128rfa1_16qm_4x_4y_krs_2rs_YL_LO_NIB. r90	3488	311
				GCC	libm128rfa1_16qm_4x_4y_krs_2rs_YL_LO_NIB. a	4406	309
16	4x4	2	4,5,6,7	IAR	libm128rfa1_16qm_4x_4y_krs_2rs_YL_HI_NIB.r 90	3488	311
				GCC	libm128rfa1_16qm_4x_4y_krs_2rs_YL_HI_NIB.a	4406	309
16	4x4	4	0,1,2,3	IAR	libm128rfa1_16qm_4x_4y_krs_4rs_YL_LO_NIB. r90	3488	331
				GCC	libm128rfa1_16qm_4x_4y_krs_4rs_YL_LO_NIB. a	4406	329
16	4x4	4	4,5,6,7	IAR	libm128rfa1_16qm_4x_4y_krs_4rs_YL_HI_NIB.r 90	3488	331
				GCC	libm128rfa1_16qm_4x_4y_krs_4rs_YL_HI_NIB.a	4406	329
16	8x2	0	0,1	IAR	libm128rfa1_16qm_8x_2y_k_0rs_YL_LO_NIB.r9 0	2166	258
				GCC	libm128rfa1_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2786	254
16	8x2	0	4,5	IAR	libm128rfa1_16qm_8x_2y_k_0rs_YL_HI_NIB.r9 0	2166	258
				GCC	libm128rfa1_16qm_8x_2y_k_0rs_YL_HI_NIB.a	2786	254
16	8x2	2	0,1	IAR	libm128rfa1_16qm_8x_2y_krs_2rs_YL_LO_NIB.	3508	327

					r90		
				GCC	libm128rfa1_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4396	323
16	8x2	2	4,5	IAR	libm128rfa1_16qm_8x_2y_krs_2rs_YL_HI_NIB.r90	3508	327
				GCC	libm128rfa1_16qm_8x_2y_krs_2rs_YL_HI_NIB.a	4396	323
16	8x2	4	0,1	IAR	libm128rfa1_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3508	347
				GCC	libm128rfa1_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4396	343
16	8x2	4	4,5	IAR	libm128rfa1_16qm_8x_2y_krs_4rs_YL_HI_NIB.r90	3508	347
				GCC	libm128rfa1_16qm_8x_2y_krs_4rs_YL_HI_NIB.a	4396	343
32	8x4	0	0,1,2,3	IAR	libm128rfa1_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2186	452
				GCC	libm128rfa1_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2820	450
32	8x4	0	4,5,6,7	IAR	libm128rfa1_32qm_8x_4y_k_0rs_YL_HI_NIB.r90	2186	452
				GCC	libm128rfa1_32qm_8x_4y_k_0rs_YL_HI_NIB.a	2820	450
32	8x4	4	0,1,2,3	IAR	libm128rfa1_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3528	573
				GCC	libm128rfa1_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4440	571
32	8x4	4	4,5,6,7	IAR	libm128rfa1_32qm_8x_4y_krs_4rs_YL_HI_NIB.r90	3528	573
				GCC	libm128rfa1_32qm_8x_4y_krs_4rs_YL_HI_NIB.a	4440	571
64	8x8	0	0,1,2,3,4,5,6,7	IAR	libm128rfa1_64qm_8x_8y_k_0rs_YL_LO_NIB.r90	2218	842
				GCC	libm128rfa1_64qm_8x_8y_k_0rs_YL_LO_NIB.a	2830	840
64	8x8	4	0,1,2,3,4,5,6,7	IAR	libm128rfa1_64qm_8x_8y_krs_4rs_YL_LO_NIB.r90	3560	1025
				GCC	libm128rfa1_64qm_8x_8y_krs_4rs_YL_LO_NIB.a	4438	1027
64	8x8	8	0,1,2,3,4,5,6,7	IAR	libm128rfa1_64qm_8x_8y_krs_8rs_YL_LO_NIB.r90	3560	1067
				GCC	libm128rfa1_64qm_8x_8y_krs_8rs_YL_LO_NIB.a	4438	1065

7.2.4.2.11 QMatrix acquisition method Library Variants for ATxmega64A3

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATxmega64A3 (IAR – v3xm, GCC - avrxmega4)							
8	4x2	0	0,1	IAR	libv3xm_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2271	147
				GCC	libavrxmega4_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2906	141
8	4x2	2	0,1	IAR	libv3xm_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3610	200
				GCC	libavrxmega4_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4530	194

16	4x4	0	0,1,2,3	IAR	libv3xm_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2291	246
				GCC	libavrxmega4_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2910	240
16	4x4	2	0,1,2,3	IAR	libv3xm_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3638	246
				GCC	libavrxmega4_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4534	309
16	4x4	4	0,1,2,3	IAR	libv3xm_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3642	266
				GCC	libavrxmega4_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4534	329
16	8x2	0	0,1	IAR	libv3xm_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2311	260
				GCC	libavrxmega4_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2902	254
16	8x2	2	0,1	IAR	libv3xm_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3658	260
				GCC	libavrxmega4_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4524	323
16	8x2	4	0,1	IAR	libv3xm_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3662	280
				GCC	libavrxmega4_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4524	343
32	8x4	0	0,1,2,3	IAR	libv3xm_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2311	456
				GCC	libavrxmega4_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2936	460
32	8x4	4	0,1,2,3	IAR	libv3xm_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3682	578
				GCC	libavrxmega4_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4566	575

7.2.4.2.12 QMatrix acquisition method Library Variants for ATxmega64A1

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATxmega64A1 (IAR – v4xm, GCC – avrxmega5)							
8	4x2	0	0,1	IAR	libv4xm_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2402	147
				GCC	libavrxmega5_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2906	141
8	4x2	2	0,1	IAR	libv4xm_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3806	200
				GCC	libavrxmega5_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4530	194
16	4x4	0	0,1,2,3	IAR	libv4xm_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2426	246
				GCC	libavrxmega5_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2910	240
16	4x4	2	0,1,2,3	IAR	libv4xm_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3838	246
				GCC	libavrxmega5_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4534	309
16	4x4	4	0,1,2,3	IAR	libv4xm_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3838	266
				GCC	libavrxmega5_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4534	329
16	8x2	0	0,1	IAR	libv4xm_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2466	260
				GCC	libavrxmega5_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2902	254
16	8x2	2	0,1	IAR	libv4xm_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3858	260
				GCC	libavrxmega5_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4524	323
16	8x2	4	0,1	IAR	libv4xm_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3858	280
				GCC	libavrxmega5_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4524	343
32	8x4	0	0,1,2,3	IAR	libv4xm_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2482	456
				GCC	libavrxmega5_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2936	460
32	8x4	4	0,1,2,3	IAR	libv4xm_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3895	578
				GCC	libavrxmega5_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4566	575

7.2.4.2.13 QMatrix acquisition method Library Variants for ATxmega128A3, ATxmega192A3, ATxmega256A3, ATxmega256A3B,

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATxmega128A3, ATxmega192A3, ATxmega256A3, ATxmega256A3B **(IAR – v5xm, GCC – avrxmega6)**							
8	4x2	0	0,1	IAR	libv5xm_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2301	167
				GCC	libavrxmega6_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2906	141
8	4x2	2	0,1	IAR	libv5xm_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3644	210
				GCC	libavrxmega6_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4530	194
16	4x4	0	0,1,2,3	IAR	libv5xm_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2321	266
				GCC	libavrxmega6_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2910	240
16	4x4	2	0,1,2,3	IAR	libv5xm_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3662	256
				GCC	libavrxmega6_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4534	309
16	4x4	4	0,1,2,3	IAR	libv5xm_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3676	256
				GCC	libavrxmega6_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4534	329
16	8x2	0	0,1	IAR	libv5xm_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2341	280
				GCC	libavrxmega6_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2902	254
16	8x2	2	0,1	IAR	libv5xm_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3692	270
				GCC	libavrxmega6_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4524	323
16	8x2	4	0,1	IAR	libv5xm_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3700	270
				GCC	libavrxmega6_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4524	343
32	8x4	0	0,1,2,3	IAR	libv5xm_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2341	476
				GCC	libavrxmega6_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2936	460
32	8x4	4	0,1,2,3	IAR	libv5xm_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3712	588
				GCC	libavrxmega6_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4566	575

7.2.4.2.14 QMatrix acquisition method Library Variants for ATxmega128A1

Legend :
NC – Number of channels
Nx x NY – Number of X x Y channels
NRS – Number of Rotors & Sliders

NC	NX x NY	NRS	Y-line pins Y_LINES	Compiler	Library to be used	Code memory	Data Memory
ATxmega128A1 **(IAR – v6xm, GCC – avrxmega7)**							
8	4x2	0	0,1	IAR	libv6xm_8qm_4x_2y_k_0rs_YL_LO_NIB.r90	2432	167
				GCC	libavrxmega7_8qm_4x_2y_k_0rs_YL_LO_NIB.a	2906	141
8	4x2	2	0,1	IAR	libv6xm_8qm_4x_2y_krs_2rs_YL_LO_NIB.r90	3840	220
				GCC	libavrxmega7_8qm_4x_2y_krs_2rs_YL_LO_NIB.a	4530	194
16	4x4	0	0,1,2,3	IAR	libv6xm_16qm_4x_4y_k_0rs_YL_LO_NIB.r90	2456	266
				GCC	libavrxmega7_16qm_4x_4y_k_0rs_YL_LO_NIB.a	2910	240
16	4x4	2	0,1,2,3	IAR	libv6xm_16qm_4x_4y_krs_2rs_YL_LO_NIB.r90	3872	266

				GCC	libavrxmega7_16qm_4x_4y_krs_2rs_YL_LO_NIB.a	4534	309
16	4x4	4	0,1,2,3	IAR	libv6xm_16qm_4x_4y_krs_4rs_YL_LO_NIB.r90	3872	286
				GCC	libavrxmega7_16qm_4x_4y_krs_4rs_YL_LO_NIB.a	4534	329
16	8x2	0	0,1	IAR	libv6xm_16qm_8x_2y_k_0rs_YL_LO_NIB.r90	2496	280
				GCC	libavrxmega7_16qm_8x_2y_k_0rs_YL_LO_NIB.a	2902	254
16	8x2	2	0,1	IAR	libv6xm_16qm_8x_2y_krs_2rs_YL_LO_NIB.r90	3892	280
				GCC	libavrxmega7_16qm_8x_2y_krs_2rs_YL_LO_NIB.a	4524	323
16	8x2	4	0,1	IAR	libv6xm_16qm_8x_2y_krs_4rs_YL_LO_NIB.r90	3892	300
				GCC	libavrxmega7_16qm_8x_2y_krs_4rs_YL_LO_NIB.a	4524	343
32	8x4	0	0,1,2,3	IAR	libv6xm_32qm_8x_4y_k_0rs_YL_LO_NIB.r90	2512	476
				GCC	libavrxmega7_32qm_8x_4y_k_0rs_YL_LO_NIB.a	2936	460
32	8x4	4	0,1,2,3	IAR	libv6xm_32qm_8x_4y_krs_4rs_YL_LO_NIB.r90	3930	598
				GCC	libavrxmega7_32qm_8x_4y_krs_4rs_YL_LO_NIB.a	4566	575

7.2.4.3 Device configurations supported by the QMatrix acquisition method library variants

This section lists the different configurations of X, Y and SMP pins supported for each of the devices for which QMatrix Library variants are used.

The following sections list the various combinations possible. Each table row lists the possible configuration of the Matrix combination, the X, Y and SMP line ports and the SMP pin position supported. The corresponding configuration information will be required to be provided when using the library.

7.2.4.3.1 Device configuration supported for ATtiny88

Device : ATtiny88 NOTE : AREF is PD6 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	(4 x2)	YL_LO_NIB	B	D	C	D	7
8 channel	(4 x2)	YL_LO_NIB	D	B	C	B	7
8 channel	(4 x2)	YL_HI_NIB	B	D	C	D	0
8 channel	(4 x2)	YL_HI_NIB	D	B	C	B	0
16 channel	(4 x4)	YL_LO_NIB	B	D	C	D	7
16 channel	(4x4)	YL_LO_NIB	D	B	C	B	7
16 channel	(8 x2)	YL_LO_NIB	B	D	C	D	7
16 channel	(8 x2)	YL_LO_NIB	D	B	C	B	7
16 channel	(8 x2)	YL_HI_NIB	B	D	C	D	0
16 channel	(8 x2)	YL_HI_NIB	D	B	C	B	0

7.2.4.3.2 Device configuration supported for ATtiny167

Device : ATtiny167 NOTE : AREF is PA6 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	(4 x2)	YL_HI_NIB	B	B	A	B	6
8 channel	(4 x2)	YL_HI_NIB	B	B	A	A	7
8 channel	(4 x2)	YL_HI_NIB	B	B	A	A	2
8 channel	(4 x2)	YL_HI_NIB	B	B	A	A	3

7.2.4.3.3 Device configuration supported for ATmega16/ ATmega16A

Device : ATmega16/ ATmega16A NOTE : AREF is PB2 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	(4 x2)	YL_LO_NIB	C	B	A	A	7
8 channel	(4 x2)	YL_LO_NIB	D	B	A	A	7
8 channel	(4 x2)	YL_LO_NIB	D	C	A	A	7
8 channel	(4 x2)	YL_LO_NIB	C	D	A	A	7
8 channel	(4 x2)	YL_HI_NIB	C	B	A	A	0
8 channel	(4 x2)	YL_HI_NIB	D	B	A	A	0
8 channel	(4 x2)	YL_HI_NIB	D	C	A	A	0
8 channel	(4 x2)	YL_HI_NIB	C	D	A	A	0
16 channel	(4 x 4)	YL_LO_NIB	D	C	A	A	7
16 channel	(4 x 4)	YL_LO_NIB	C	D	A	A	7
16 channel	(4 x 4)	YL_HI_NIB	D	C	A	A	0
16 channel	(4 x 4)	YL_HI_NIB	C	D	A	A	0
16 channel	(8 x 2)	YL_LO_NIB	C	B	A	A	7
16 channel	(8 x 2)	YL_LO_NIB	D	B	A	A	7
16 channel	(8 x 2)	YL_LO_NIB	D	C	A	A	7
16 channel	(8 x 2)	YL_LO_NIB	C	D	A	A	7
16 channel	(8 x 2)	YL_HI_NIB	C	B	A	A	0
16 channel	(8 x 2)	YL_HI_NIB	D	B	A	A	0
16 channel	(8 x 2)	YL_HI_NIB	D	C	A	A	0
16 channel	(8 x 2)	YL_HI_NIB	C	D	A	A	0
32 channel	(8 x 4)	YL_LO_NIB	D	C	A	A	7
32 channel	(8 x 4)	YL_LO_NIB	C	D	A	A	7
32 channel	(8 x 4)	YL_HI_NIB	D	C	A	A	0
32 channel	(8 x 4)	YL_HI_NIB	C	D	A	A	0

7.2.4.3.4 Device configuration supported for ATmega8535

Device : ATmega8535 NOTE : AREF is PB2 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	(4 x 2)	YL_LO_NIB	C	B	A	A	7
8 channel	(4 x 2)	YL_LO_NIB	D	B	A	A	7
8 channel	(4 x 2)	YL_LO_NIB	D	C	A	A	7
8 channel	(4 x 2)	YL_LO_NIB	C	D	A	A	7
8 channel	(4 x 2)	YL_HI_NIB	C	B	A	A	0
8 channel	(4 x 2)	YL_HI_NIB	D	B	A	A	0
8 channel	(4 x 2)	YL_HI_NIB	D	C	A	A	0
8 channel	(4 x 2)	YL_HI_NIB	C	D	A	A	0
16 channel	(4 x 4)	YL_LO_NIB	D	C	A	A	7
16 channel	(4 x 4)	YL_LO_NIB	C	D	A	A	7
16 channel	(4 x 4)	YL_HI_NIB	D	C	A	A	0
16 channel	(4 x 4)	YL_HI_NIB	C	D	A	A	0
16 channel	(8 x 2)	YL_LO_NIB	C	B	A	A	7
16 channel	(8 x 2)	YL_LO_NIB	D	B	A	A	7
16 channel	(8 x 2)	YL_LO_NIB	D	C	A	A	7
16 channel	(8 x 2)	YL_LO_NIB	C	D	A	A	7
16 channel	(8 x 2)	YL_HI_NIB	C	B	A	A	0
16 channel	(8 x 2)	YL_HI_NIB	D	B	A	A	0
16 channel	(8 x 2)	YL_HI_NIB	D	C	A	A	0
16 channel	(8 x 2)	YL_HI_NIB	C	D	A	A	0

7.2.4.3.5 Device configuration supported for ATmega88P/ ATmega88PA

Device : ATmega88P/ ATmega88PA NOTE : AREF is PD6 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	(4x2)	YL_LO_NIB	B	D	C	D	7
8 channel	(4x2)	YL_LO_NIB	D	B	C	D	7
8 channel	(4x2)	YL_HI_NIB	B	D	C	D	0
8 channel	(4x2)	YL_HI_NIB	D	B	C	D	0
16 channel	(4 x 4)	YL_LO_NIB:	B	D	C	D	7
16 channel	(4 x 4)	YL_LO_NIB:	D	B	C	D	7

16 channel	(8 x 2)	YL_LO_NIB:	B	D	C	D	7
16 channel	(8 x 2)	YL_HI_NIB	B	D	C	D	0
32 channel	8 x 4	YL_LO_NIB:	B	D	C	D	7

7.2.4.3.6 Device configuration supported for ATmega164P

Device : ATmega164P NOTE : AREF is PB2 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	4 x 2	YL_LO_NIB:	C	B	A	A	7
8 channel	4 x 2	YL_LO_NIB:	D	B	A	A	7
8 channel	4 x 2	YL_LO_NIB:	D	C	A	A	7
8 channel	4 x 2	YL_LO_NIB:	C	D	A	A	7
8 channel	4 x 2	YL_HI_NIB	C	B	A	A	0
8 channel	4 x 2	YL_HI_NIB	D	B	A	A	0
8 channel	4 x 2	YL_HI_NIB	D	C	A	A	0
8 channel	4 x 2	YL_HI_NIB	C	D	A	A	0
16 channel	4 x 4	YL_LO_NIB:	D	C	A	A	7
16 channel	4 x 4	YL_LO_NIB:	C	D	A	A	7
16 channel	4 x 4	YL_HI_NIB	D	C	A	A	0
16 channel	4 x 4	YL_HI_NIB	C	D	A	A	0
16 channel	4 x 4	YL_HI_NIB	D	B	A	A	0
16 channel	4 x 4	YL_HI_NIB	C	B	A	A	0
16 channel	8 x 2	YL_LO_NIB:	C	B	A	A	7
16 channel	8 x 2	YL_LO_NIB:	D	B	A	A	7
16 channel	8 x 2	YL_LO_NIB:	D	C	A	A	7
16 channel	8 x 2	YL_LO_NIB:	C	D	A	A	7
16 channel	8 x 2	YL_HI_NIB:	C	B	A	A	0
16 channel	8 x 2	YL_HI_NIB:	D	B	A	A	0
16 channel	8 x 2	YL_HI_NIB:	D	C	A	A	0
16 channel	8 x 2	YL_HI_NIB:	C	D	A	A	0
32 channel	8 x 4	YL_LO_NIB:	D	C	A	A	7
32 channel	8 x 4	YL_LO_NIB:	C	D	A	A	7
32 channel	8 x 4	YL_HI_NIB:	D	C	A	A	0
32 channel	8 x 4	YL_HI_NIB:	C	D	A	A	0
32 channel	8 x 4	YL_HI_NIB:	D	B	A	A	0
32 channel	8 x 4	YL_HI_NIB:	C	B	A	A	0

7.2.4.3.7 Device configuration supported for ATmega165P

Device : Atmega165P NOTE : AREF is PE2 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	4 x 2:	YL_LO_NIB:	A	B	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	A	C	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	A	C	F	G	1
8 channel	4 x 2:	YL_LO_NIB:	A	C	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	A	D	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	A	E	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	B	A	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	C	A	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	D	A	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	B	C	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	B	D	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	B	E	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	C	B	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	D	B	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	C	D	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	C	E	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	D	C	F	G	0
8 channel	4 x 2:	YL_LO_NIB:	D	E	F	G	0
8 channel	4 x 2:	YL_HI_NIB:	A	B	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	A	C	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	A	D	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	A	E	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	B	A	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	C	A	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	D	A	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	B	C	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	B	D	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	B	E	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	C	B	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	D	B	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	C	D	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	C	E	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	D	C	F	F	0
8 channel	4 x 2:	YL_HI_NIB:	D	E	F	F	0
16 channel	4 x 4:	YL_LO_NIB:	A	B	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	A	C	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	A	C	F	G	1
16 channel	4 x 4:	YL_LO_NIB:	A	C	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	A	D	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	B	A	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	C	A	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	D	A	F	G	0

16 channel	4 x 4:	YL_LO_NIB:	B	C	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	B	D	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	C	B	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	D	B	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	C	D	F	G	0
16 channel	4 x 4:	YL_LO_NIB:	D	C	F	G	0
16 channel	4 x 4:	YL_HI_NIB:	A	B	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	A	C	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	A	D	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	B	A	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	C	A	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	D	A	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	B	C	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	B	D	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	C	B	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	D	B	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	C	D	F	F	0
16 channel	4 x 4:	YL_HI_NIB:	D	C	F	F	0
16 channel	8 x 2:	YL_LO_NIB	A	B	F	G	0
16 channel	8 x 2:	YL_LO_NIB	A	C	F	G	0
16 channel	8 x 2:	YL_LO_NIB	A	C	F	G	1
16 channel	8 x 2:	YL_LO_NIB	A	C	F	G	0
16 channel	8 x 2:	YL_LO_NIB	A	D	F	G	0
16 channel	8 x 2:	YL_LO_NIB	B	A	F	G	0
16 channel	8 x 2:	YL_LO_NIB	C	A	F	G	0
16 channel	8 x 2:	YL_LO_NIB	D	A	F	G	0
16 channel	8 x 2:	YL_LO_NIB	B	C	F	G	0
16 channel	8 x 2:	YL_LO_NIB	B	D	F	G	0
16 channel	8 x 2:	YL_LO_NIB	B	E	F	G	0
16 channel	8 x 2:	YL_LO_NIB	C	B	F	G	0
16 channel	8 x 2:	YL_LO_NIB	D	B	F	G	0
16 channel	8 x 2:	YL_LO_NIB	C	D	F	G	0
16 channel	8 x 2:	YL_LO_NIB	C	E	F	G	0
16 channel	8 x 2:	YL_LO_NIB	D	C	F	G	0
16 channel	8 x 2:	YL_LO_NIB	D	E	F	G	0
16 channel	8 x 2:	YL_HI_NIB	A	B	F	F	0
16 channel	8 x 2:	YL_HI_NIB	A	C	F	F	0
16 channel	8 x 2:	YL_HI_NIB	A	D	F	F	0
16 channel	8 x 2:	YL_HI_NIB	A	E	F	F	0
16 channel	8 x 2:	YL_HI_NIB	B	A	F	F	0
16 channel	8 x 2:	YL_HI_NIB	C	A	F	F	0
16 channel	8 x 2:	YL_HI_NIB	D	A	F	F	0
16 channel	8 x 2:	YL_HI_NIB	B	C	F	F	0
16 channel	8 x 2:	YL_HI_NIB	B	D	F	F	0
16 channel	8 x 2:	YL_HI_NIB	B	E	F	F	0
16 channel	8 x 2:	YL_HI_NIB	C	B	F	F	0
16 channel	8 x 2:	YL_HI_NIB	D	B	F	F	0
16 channel	8 x 2:	YL_HI_NIB	C	D	F	F	0
16 channel	8 x 2:	YL_HI_NIB	C	E	F	F	0
16 channel	8 x 2:	YL_HI_NIB	D	C	F	F	0
16 channel	8 x 2:	YL_HI_NIB	D	E	F	F	0

32 channel	8 x 4:	YL_LO_NIB	A	B	F	G	0
32 channel	8 x 4:	YL_LO_NIB	A	C	F	G	0
32 channel	8 x 4:	YL_LO_NIB	A	C	F	G	1
32 channel	8 x 4:	YL_LO_NIB	A	C	F	G	0
32 channel	8 x 4:	YL_LO_NIB	A	D	F	G	0
32 channel	8 x 4:	YL_LO_NIB	B	A	F	G	0
32 channel	8 x 4:	YL_LO_NIB	C	A	F	G	0
32 channel	8 x 4:	YL_LO_NIB	D	A	F	G	0
32 channel	8 x 4:	YL_LO_NIB	B	C	F	G	0
32 channel	8 x 4:	YL_LO_NIB	B	D	F	G	0
32 channel	8 x 4:	YL_LO_NIB	C	B	F	G	0
32 channel	8 x 4:	YL_LO_NIB	D	B	F	G	0
32 channel	8 x 4:	YL_LO_NIB	C	D	F	G	0
32 channel	8 x 4:	YL_LO_NIB	D	C	F	G	0
32 channel	8 x 4:	YL_HI_NIB	A	B	F	F	0
32 channel	8 x 4:	YL_HI_NIB	A	C	F	F	0
32 channel	8 x 4:	YL_HI_NIB	A	D	F	F	0
32 channel	8 x 4:	YL_HI_NIB	B	A	F	F	0
32 channel	8 x 4:	YL_HI_NIB	C	A	F	F	0
32 channel	8 x 4:	YL_HI_NIB	D	A	F	F	0
32 channel	8 x 4:	YL_HI_NIB	B	C	F	F	0
32 channel	8 x 4:	YL_HI_NIB	B	D	F	F	0
32 channel	8 x 4:	YL_HI_NIB	C	B	F	F	0
32 channel	8 x 4:	YL_HI_NIB	D	B	F	F	0
32 channel	8 x 4:	YL_HI_NIB	C	D	F	F	0
32 channel	8 x 4:	YL_HI_NIB	D	C	F	F	0
64 channel	8 x 4:	YL_LO_NIB	A	B	F	G	0
64 channel	8 x 4:	YL_LO_NIB	A	C	F	G	0
64 channel	8 x 4:	YL_LO_NIB	A	C	F	G	1
64 channel	8 x 4:	YL_LO_NIB	A	C	F	G	0
64 channel	8 x 4:	YL_LO_NIB	A	D	F	G	0
64 channel	8 x 4:	YL_LO_NIB	B	A	F	G	0
64 channel	8 x 4:	YL_LO_NIB	C	A	F	G	0
64 channel	8 x 4:	YL_LO_NIB	D	A	F	G	0
64 channel	8 x 4:	YL_LO_NIB	B	C	F	G	0
64 channel	8 x 4:	YL_LO_NIB	B	D	F	G	0
64 channel	8 x 4:	YL_LO_NIB	C	B	F	G	0
64 channel	8 x 4:	YL_LO_NIB	D	B	F	G	0
64 channel	8 x 4:	YL_LO_NIB	C	D	F	G	0
64 channel	8 x 4:	YL_LO_NIB	D	C	F	G	0

7.2.4.3.8 Device configuration supported for ATmega324P / ATmega324PA

Device : ATmega324P / Atmega324PA NOTE : AREF is PB2 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	4 x 2 :	YL_LO_NIB:	C	B	A	A	7
8 channel	4 x 2 :	YL_LO_NIB:	D	B	A	A	7
8 channel	4 x 2 :	YL_LO_NIB:	D	C	A	A	7
8 channel	4 x 2 :	YL_LO_NIB:	C	D	A	A	7
8 channel	4 x 2 :	YL_HI_NIB	C	B	A	A	0
8 channel	4 x 2 :	YL_HI_NIB	D	B	A	A	0
8 channel	4 x 2 :	YL_HI_NIB	D	C	A	A	0
8 channel	4 x 2 :	YL_HI_NIB	C	D	A	A	0
16 channel	4 x 4:	YL_LO_NIB:	D	C	A	A	7
16 channel	4 x 4:	YL_LO_NIB:	C	D	A	A	7
16 channel	4 x 4:	YL_HI_NIB	D	C	A	A	0
16 channel	4 x 4:	YL_HI_NIB	C	D	A	A	0
16 channel	4 x 4:	YL_HI_NIB	D	B	A	A	0
16 channel	4 x 4:	YL_HI_NIB	C	B	A	A	0
16 channel	8 x 2:	YL_LO_NIB:	C	B	A	A	7
16 channel	8 x 2:	YL_LO_NIB:	D	B	A	A	7
16 channel	8 x 2:	YL_LO_NIB:	D	C	A	A	7
16 channel	8 x 2:	YL_LO_NIB:	C	D	A	A	7
16 channel	8 x 2:	YL_HI_NIB:	C	B	A	A	0
16 channel	8 x 2:	YL_HI_NIB:	D	B	A	A	0
16 channel	8 x 2:	YL_HI_NIB:	D	C	A	A	0
16 channel	8 x 2:	YL_HI_NIB:	C	D	A	A	0
32 channel	8 x 4:	YL_LO_NIB:	D	C	A	A	7
32 channel	8 x 4:	YL_LO_NIB:	C	D	A	A	7
32 channel	8 x 4:	YL_HI_NIB:	D	C	A	A	0
32 channel	8 x 4:	YL_HI_NIB:	C	D	A	A	0
32 channel	8 x 4:	YL_HI_NIB:	D	B	A	A	0
32 channel	8 x 4:	YL_HI_NIB:	C	B	A	A	0
64 channel	8x8	YL_LO_NIB	D	C	A	B	0
64 channel	8x8	YL_LO_NIB	D	C	A	B	3
64 channel	8x8	YL_LO_NIB	D	C	A	B	7
64 channel	8x8	YL_LO_NIB	C	D	A	B	0
64 channel	8x8	YL_LO_NIB	C	D	A	B	3
64 channel	8x8	YL_LO_NIB	C	D	A	B	7

7.2.4.3.9 Device configuration supported for ATmega128RFA1

Device : ATmega128RFA1 NOTE : AREF is PE2 and needs to be connected to GND							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_ SMP**	**SMP_BIT**
8 channel	4 x 2 :	YL_LO_NIB:	D	B	F	G	4
8 channel	4 x 2 :	YL_LO_NIB:	D	E	F	G	4
8 channel	4 x 2 :	YL_LO_NIB:	D	B	F	G	5
8 channel	4 x 2 :	YL_LO_NIB:	D	E	F	G	5
8 channel	4 x 2 :	YL_HI_NIB	D	B	F	G	4
8 channel	4 x 2 :	YL_HI_NIB	D	E	F	G	4
8 channel	4 x 2 :	YL_HI_NIB	D	B	F	G	5
8 channel	4 x 2 :	YL_HI_NIB	D	E	F	G	5
16 channel	4 x 4:	YL_LO_NIB:	D	B	F	G	4
16 channel	4 x 4:	YL_LO_NIB:	D	B	F	G	5
16 channel	4 x 4:	YL_HI_NIB	D	B	F	G	4
16 channel	4 x 4:	YL_HI_NIB	D	E	F	G	4
16 channel	4 x 4:	YL_HI_NIB	D	B	F	G	5
16 channel	4 x 4:	YL_HI_NIB	D	E	F	G	5
16 channel	8 x 2:	YL_LO_NIB:	D	B	F	G	4
16 channel	8 x 2:	YL_LO_NIB:	D	E	F	G	4
16 channel	8 x 2:	YL_LO_NIB:	D	B	F	G	5
16 channel	8 x 2:	YL_LO_NIB:	D	E	F	G	5
16 channel	8 x 2:	YL_HI_NIB:	D	B	F	G	4
16 channel	8 x 2:	YL_HI_NIB:	D	E	F	G	4
16 channel	8 x 2:	YL_HI_NIB:	D	B	F	G	5
16 channel	8 x 2:	YL_HI_NIB:	D	E	F	G	5
32 channel	8 x 4:	YL_LO_NIB:	D	B	F	G	4
32 channel	8 x 4:	YL_LO_NIB:	D	B	F	G	5
32 channel	8 x 4:	YL_HI_NIB:	D	B	F	G	4
32 channel	8 x 4:	YL_HI_NIB:	D	E	F	G	4
32 channel	8 x 4:	YL_HI_NIB:	D	B	F	G	5
32 channel	8 x 4:	YL_HI_NIB:	D	E	F	G	5
64 channel	8x8	YL_LO_NIB	D	B	F	G	4
64 channel	8x8	YL_LO_NIB	D	B	F	G	5

7.2.4.3.10 Device configuration supported for ATxmega<xxx>A3

Device : ATxmega<xxx>A3 <xxx>- ATxmega64A3/ATxmega128A3/ ATxmega192A3/ATxmega256A3/ ATxmega256A3B **NOTE : AREF is PA7 and needs to be connected to GND**							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_SMP**	**SMP_ BIT**
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	C	A	C	7

7.2.4.3.11 Device configuration supported for ATxmega<xxx>A1

Device : ATxmega<xxx>A1 <xxx>- ATxmega64A1/ATxmega128A1 **NOTE : AREF is PA7 and needs to be connected to GND**							
Number of Channels supported by the library variant	**X x Y configuration**	**Y_LINES**	**PORT_X**	**PORT_YA**	**PORT_YB**	**PORT_SMP**	**SMP_ BIT**
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	H	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	J	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	K	C	A	C	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	H	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	J	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	K	D	A	D	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	E	A	E	7

8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	H	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	J	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	K	E	A	E	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	H	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	J	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	K	F	A	F	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	H	A	H	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	H	A	H	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	H	A	H	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	H	A	H	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	J	H	A	H	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	K	H	A	H	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	J	A	J	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	J	A	J	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	J	A	J	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	J	A	J	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	H	J	A	J	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	K	J	A	J	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	C	K	A	K	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	D	K	A	K	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	E	K	A	K	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	F	K	A	K	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	H	K	A	K	7
8/16/32 channel	4x2/4x4/8x2/8x4	YL_LO_NIB	J	K	A	K	7

8 Checklist

This section lists troubleshooting tips and common configuration tips.

Symptom	**Cause**	**Action**
Sensors do not go into detect or have unknown results	Multiplexing pins used by QTouch libraries in your design	Check the Pins used for QTouch or QMatrix acquisition methods do not overlap with the applications usage of the ports
Signal values report arbitrary values	Stray capacitance	Check the sensor design and minimize stray capacitance interference in your design
Waveforms of charging / discharging of channels do not show up properly in oscilloscopes	JTAG ICE connected to the board	Try disconnecting the JTAG ICE completely from the kit
When using the example applications, the debug values for some of the channels does not display appropriate values	JTAG Pins are enabled in the target.	JTAG Pins are explicitly needs to be disabled in the main.c file /* disable JTAG pins */ MCUCR |= (1u << JTD); MCUCR |= (1u << JTD)

9 MISRA Compliance Report

This section lists the compliance and deviations for MISRA standards of coding practice for the QTouch and QMatrix acquisition method libraries.

9.1 What is covered

The QTouch and QMatrix acquisition method libraries adhere to the MISRA standards. The additional reference code provided in the form of sample applications is not guaranteed to be MISRA compliant.

9.2 Target Environment

Development Environment	IAR Embedded Workbench V5.3
MISRA Checking software	The MISRA C Compliance has been performed for the library using MISRA C 2004 Rules in IAR WorkBench V5.3 development environment.
MISRA Rule set applied	MISRAC 2004 Rule Set

9.3 Deviations from MISRA C Standards

9.3.1 QTouch acquisition method libraries

The QTouch acquisition method libraries were subject to the above mentioned MISRA compliance rules. The following exceptions have not been fixed as they are required for the implementation of the library.

Applicable Release	QTouch libraries ver 3.2	
Rule No	**Rule Description**	**Exception noted / How it is addressed**
1.1	Rule states that all code shall conform to ISO 9899 standard C, with no extensions permitted.	This Rule is not supported as the library implementation requires IAR extensions like __interrupt. These intrinsic functions relate to device hardware functionality, and cannot practically be avoided.
10.1	Rule states that implicit conversion from Underlying long to unsigned long	The library uses macros to combine symbol definitions to form a unique expanded symbol name and in this, the usage of unsigned qualifiers for numeric constants (e.g. 98u) causes name mangling. This is the only occurrence of this error in the library.
10.6	This Rule says that a 'U' suffix shall be applied to all constants of 'unsigned' type	The library uses macros to combine symbol definitions to form a unique expanded symbol name and in this, the usage of unsigned qualifiers for numeric constants (e.g. 98u) causes name mangling. This is the only occurrence of this error in the library.
14.4	Rule states that go-to statement should not be used.	The library uses conditional jump instructions to reduce the code footprint at a few locations and this is localized to small snippets of code. Hence this rule is not supported.
19.10	Rule states that In the definition of a function-like macro, each instance of a parameter shall be enclosed in parenthesis	There is one instance where the library breaks this rule where two macro definitions are combined to form a different symbol name. Usage of parenthesis cannot be used in this scenario.
19.12	Rule states that there shall be at most one occurrence of the # or ## preprocessor operator in a single macro definition	There is one instance in the library where this rule is violated where the library concatenates two macro definitions to arrive at a different definition.

9.3.2 QMatrix acquisition method libraries

The QMatrix acquisition method software was subject to the above mentioned MISRA compliance rules. The following exceptions have not been fixed as they are required for the implementation of the library.

Applicable release	**QTouch libraries ver 3.2**	
Rule No	**Rule Description**	**Exceptions Reason**
1.1	Rule states that all code shall conform to ISO 9899 standard C, with no extensions permitted.	This Rule is not supported as the library implementation requires IAR extensions like __interrupt. These intrinsic functions relates to device hardware functionality, and cannot practically be avoided
10.1	Rule states that Illegal implicit conversion from Underlying long to unsigned long	The library uses macros to combine symbol definitions to form a unique expanded symbol name and in this, the usage of unsigned qualifiers for numeric constants (e.g. 98u) causes name mangling. This is the only occurrence of this error in the library.
10.6	This Rule says that a 'U' suffix shall be applied to all constants of 'unsigned' type	The library uses macros to combine symbol definitions to form a unique expanded symbol name and in this, the usage of unsigned qualifiers for numeric constants (e.g 98u) causes name mangling. This is the only occurrence of this error in the library.
19.10	Rule states that In the definition of a function-like macro, each instance of a parameter shall be enclosed in parenthesis	There is one instance where the library breaks this rule where two macro definitions are combined to form a different symbol name. Usage of parenthesis cannot be used in this scenario.
19.12	Rule states that there shall be at most one occurrence of the # or ## preprocessor operator in a single macro definition	There is one instance in the library where this rule is violated where the library concatenates two macro definitions to arrive at a different definition.

10 Known Issues

Issue	**Cause**	**Remedy / workaround**
QMatrix Touch Sensing on XMEGA AVRs The Pins PE6,PE7 and PF5 of ATxmega<xxx>A3 devices do not work for touch sensing when tested along with STK600		Suggested to use combinations with other ports
GCC compiler Optimisation issue with QMatrix Libraries on ATtiny167 In case of QMatrix GCC library for ATtiny167, few channels produces low signal and reference value compared to other channels, when tested with STK600.		Suggested to use the default Optimization level of –O0 for ATtiny167 in case of GCC.
GCC Compiler Issue for ATmega128RFA1 device The Latest GCC WinAVR-20090313 is having		In the linker options for the GCC project of ATmega128RFA1 device , the following options have to be

problems with ATmega128RFA1 device. It places the .data section to the memory location 0x800100 instead of 0x800200.		added: `-Wl,--section-start=.data=0x800200`

10.1 Linker warning

10.1.1 SFRB Warning

When using IAR workbench to integrate the touch libraries, the linker would generate a warning indicating that there is a type conflict for the bit definitions of the PORT registers.

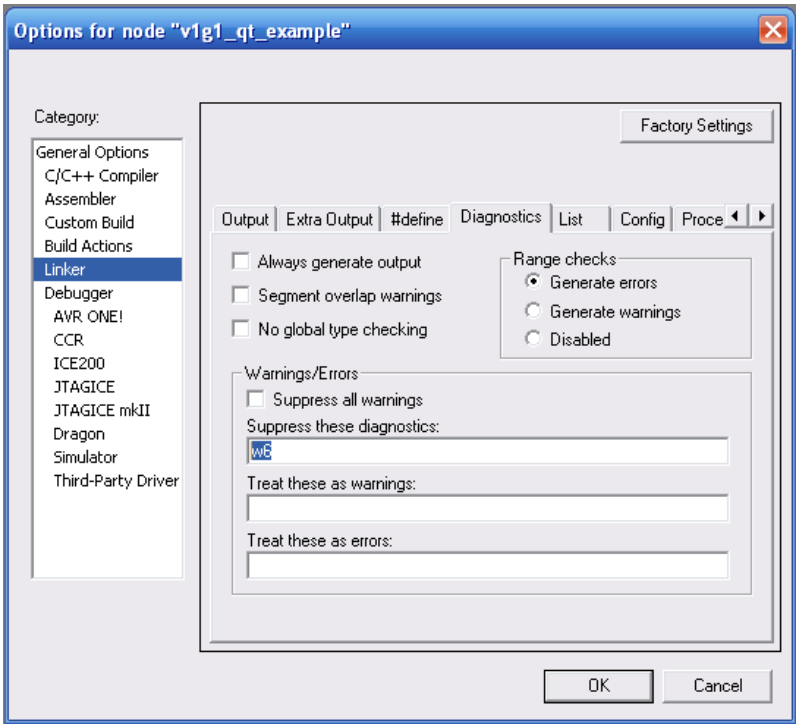
The QTouch libraries are built using a definition of the PORT registers which has a superset of all the bit definitions for the applicable devices in a device-core group. The IAR compiler uses a subset of this definition as applicable to the device selected which does not have all the bit fields defined for the selected device and this results in this warning being emitted.

These warnings should be ignored as they do not cause any malfunction or side effects. The following code snippet shows the linker warnings

Warning[w6]: Type conflict for external/entry "_A_DDRC", in module burst_10_BC against external/entry in module main;
class/struct/union field/base
types do not match for field/base "; class/struct/union field names do not match: DDRC_DDC7 vs DDRC_Dummy7
/* In module burst_10_BC: */
union /* Elements: 3, Bytes: 1 */
/* First seen in burst_10_BC */
{
unsigned char DDRC;
struct /* Elements: 8, Bytes: 1 */
/* First seen in main */
{
unsigned char DDRC_Bit0 : 1 /* disp: 0 */;
unsigned char DDRC_Bit1 : 1 /* disp: 1 */;
unsigned char DDRC_Bit2 : 1 /* disp: 2 */;
unsigned char DDRC_Bit3 : 1 /* disp: 3 */;
unsigned char DDRC_Bit4 : 1 /* disp: 4 */;
unsigned char DDRC_Bit5 : 1 /* disp: 5 */;
unsigned char DDRC_Bit6 : 1 /* disp: 6 */;
unsigned char DDRC_Bit7 : 1 /* disp: 7 */;
} ;
struct /* Elements: 8, Bytes: 1 */
/* First seen in burst_10_BC */
{
unsigned char DDRC_DDC0 : 1 /* disp: 0 */;
unsigned char DDRC_DDC1 : 1 /* disp: 1 */;
unsigned char DDRC_DDC2 : 1 /* disp: 2 */;
unsigned char DDRC_DDC3 : 1 /* disp: 3 */;
unsigned char DDRC_DDC4 : 1 /* disp: 4 */;
unsigned char DDRC_DDC5 : 1 /* disp: 5 */;
unsigned char DDRC_DDC6 : 1 /* disp: 6 */;
unsigned char DDRC_DDC7 : 1 /* disp: 7 */;

```
} ;
} __io volatile _A_DDRC;
/* In module main: */
union /* Elements: 3, Bytes: 1 */
/* First seen in main */
{
unsigned char DDRC;
struct /* Elements: 8, Bytes: 1 */
/* First seen in main */
{
unsigned char DDRC_Bit0 : 1 /* disp: 0 */;
unsigned char DDRC_Bit1 : 1 /* disp: 1 */;
unsigned char DDRC_Bit2 : 1 /* disp: 2 */;
unsigned char DDRC_Bit3 : 1 /* disp: 3 */;
unsigned char DDRC_Bit4 : 1 /* disp: 4 */;
unsigned char DDRC_Bit5 : 1 /* disp: 5 */;
unsigned char DDRC_Bit6 : 1 /* disp: 6 */;
unsigned char DDRC_Bit7 : 1 /* disp: 7 */;
} ;
struct /* Elements: 8, Bytes: 1 */
/* First seen in main */
{
unsigned char DDRC_DDC0 : 1 /* disp: 0 */;
unsigned char DDRC_DDC1 : 1 /* disp: 1 */;
unsigned char DDRC_DDC2 : 1 /* disp: 2 */;
unsigned char DDRC_DDC3 : 1 /* disp: 3 */;
unsigned char DDRC_DDC4 : 1 /* disp: 4 */;
unsigned char DDRC_DDC5 : 1 /* disp: 5 */;
unsigned char DDRC_DDC6 : 1 /* disp: 6 */;
unsigned char DDRC_Dummy7 : 1 /* disp: 7 */;
} ;
} __io volatile _A_DDRC;
```

If you prefer to suppress the warnings, you can configure the linker options in your project to disable them. In the IAR IDE, open the project options and in the Linker group, click on the “diagnostics” tab. In the field labeled “Suppress these diagnostics” add W6. This directs the IAR linker to ignore the SFR linker warnings.



11 Revision History

The table below lists the revision history for chapters in the user guide.

QTouch Library User guide Revision History		
Date/Version	**Chapter**	**Change notes**
May 2009 Ver2.0	All	2nd release of QTouch library users guide
Sep 2009 Ver. 3.0	All	Re-structured user guide with new and expanded sections
Nov 2009 Ver. 3.1	6.3, 6.9, 6.10, 7, 10	• Updated API changes • Updated new libraries and device support information • Updated debug interface information supported by the QTouch libraries • Updated known issues table
Dec 2009 Ver. 3.2	6.10.4, 7.1.2, 7.1.5, 7.1.6, 10, 7.2.4.2.2, 7.2.4.3.7, 7.2.4.3.2, 7.2.4.3.5, 7.2.4.3.7,	• Added section about configuring unused pins in user application • Added more information to some sections to clear ambiguity • Updated Memory footprint information for IAR and GCC compiled QTouch libraries. • Updated known issues table • Added the device support, port combinations, memory requirements QMatrix IAR and GCC libraries.to support ATmega325P,ATmega645, and ATtiny167. • Modified port combinations for the 165P for QMatrix libraries. • Few Port combinations added in case of ATmega88 libraries.

Headquarters

ATMEL Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

ATMEL Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

ATMEL Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

ATMEL Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site	***Technical Support***	***Sales Contact***
http://www.atmel.com/	avr@atmel.com	www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with ATMEL products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of ATMEL products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** ATMEL makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. ATMEL does not make any commitment to update the information contained herein. Unless specifically provided otherwise, ATMEL products are not suitable for, and shall not be used in, automotive applications. ATMEL's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2009 ATMEL Corporation. All rights reserved. ATMEL®, ATMEL logo and combinations thereof, AVR®, AVR Studio®, megaAVR® , tinyAVR®, QTouch®, and others are registered trademarks, QMatrix™, XMEGA™, UC3™ and others are trademarks of ATMEL Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Rev. 8207G-AT42-12/09