APPLICATION NOTE

Atmel AVR3009: Driving QTouch Device with I²C Interface

Atmel QTouch

Introduction

This application note explains the communication of I²C-Compatible Master microcontroller with Atmel AT42QT1070 as a slave device. It demonstrates configuring and controlling various parameters of this device.

The host code demonstration program provided has been developed for 8-bit Atmel® megaAVR® (Atmel ATmega2560) microcontroller but can be easily adapted for other platforms.

The demonstration program is written in C and supports both GCC (Atmel Studio) and IAR™ (IAR Embedded Workbench®) compiler. The demonstration program also supports the Atmel QTouch® devices:

- AT42QT1060
- AT42QT2120
- AT42QT2160
- QT60160
- QT60240

Table of Contents

1. Overview of the Atmel AT42QT1070

AT42QT1070 is a digital burst mode charge-transfer capacitive sensor driver. The device can sense from one to seven keys, dependent on mode. It can operate in either Comms mode or Standalone mode.

In Comms mode where a host can communicate with the device through the I^{2}C-Compatible Two-Wire Interface (TWI) and up to seven keys can be configured. This allows user to configure settings for Threshold, Adjacent Key Suppression® (AKS®), Detect Integrator, Low Power (LP) mode, Guard Channel and Max On Duration for keys.

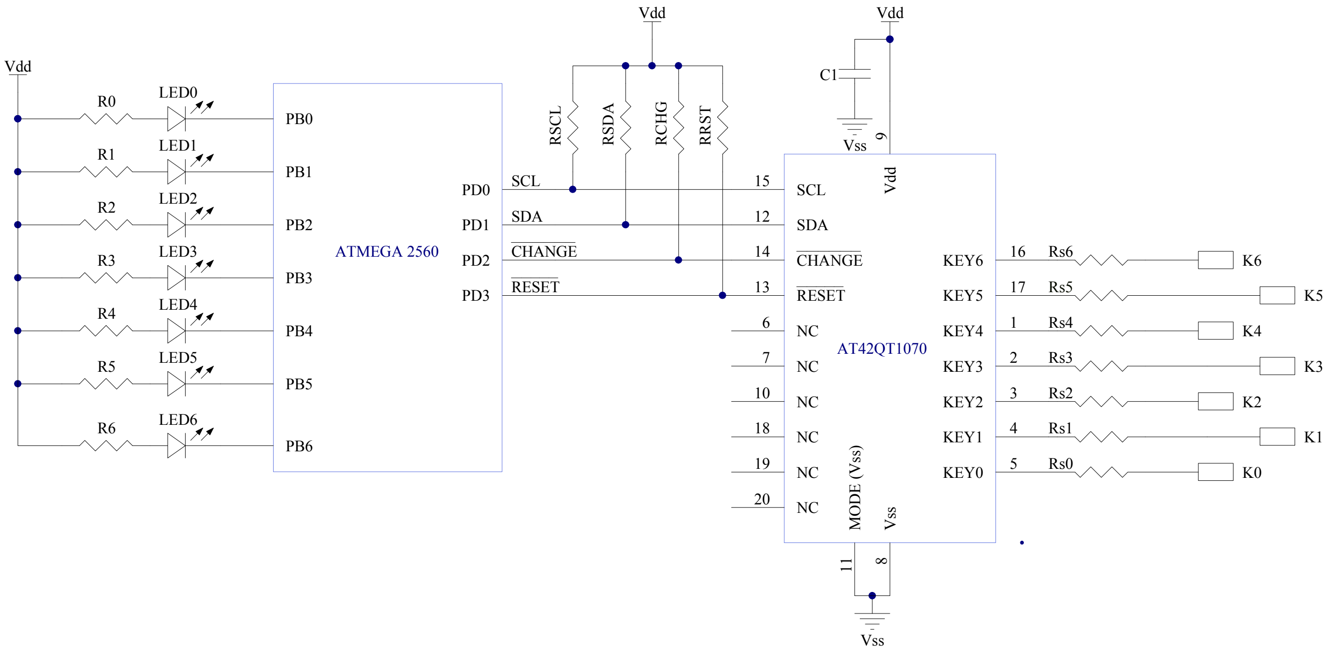
In Standalone mode, the start-up values are fixed in firmware and cannot be changed. This mode supports up to 4 keys plus a Guard Channel can be configured. The key detection is reported through their respective IOs.

The more details about the AT42QT1070 and its setup parameters can be referred in the AT42QT1070 datasheet.

2. Circuit Configuration with Host Microcontroller

Connection to the Host microcontroller uses the two I^{2}C-compatible pins (SCL and SDA), $\overline{\text{CHANGE}}$ pin and the $\overline{\text{RESET}}$ pin.

Figure 2-1. Circuit configuration with Host microcontroller.



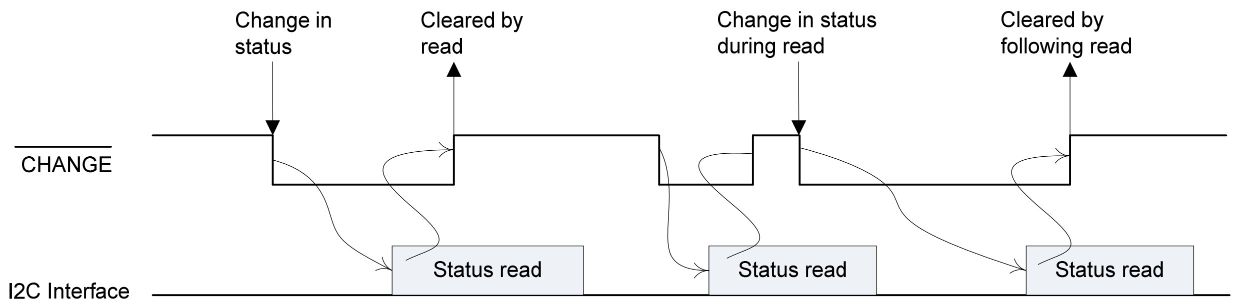
Touch keys (K0 – K6) are connected to the KEY0 to KEY6 drive signals as described in the datasheet. The demonstration program uses LEDs on host microcontroller pins PB0 to PB6 to indicate touches on keys K0 to K6.

Notes:

1. Pull-up resistors are required on the I^{2}C-compatible signals (SCL and SDA). Either discrete pull-up resistors or the host microcontroller's weak pull-ups can be used. Ensure that the pull-up resistors are connected to the same Vdd level as the AT42QT1070.
2. The $\overline{\text{CHANGE}}$ pin is open-drain and requires a 47kΩ pull-up resistor.
3. The Mode pin should be connected to Ground (Vss) to select Comms mode.

2.1 Interface timing

Figure 2-2. Atmel AT42QT1070 interface timing.

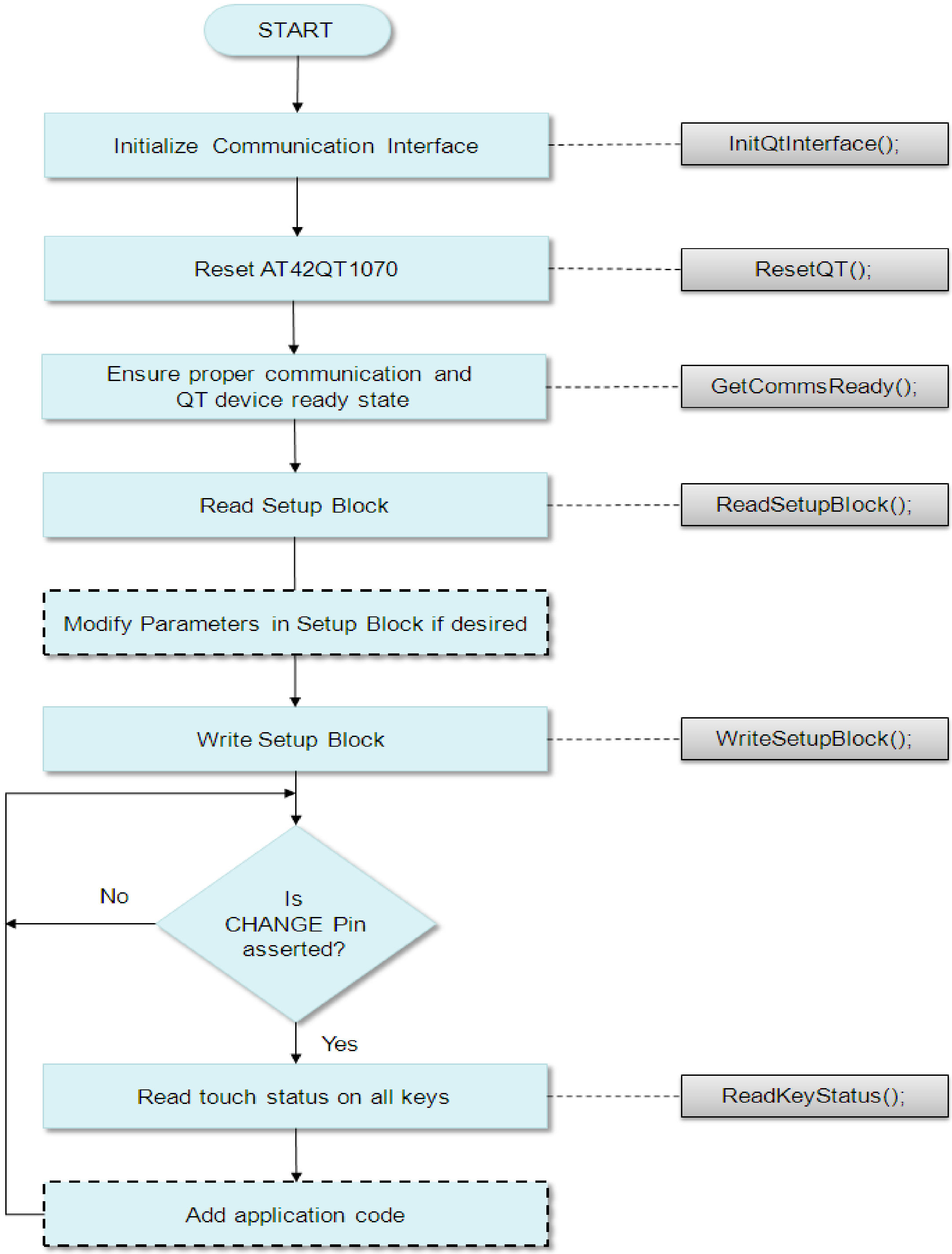


The AT42QT1070 signals any change in its sense status by driving the $\overline{\text{CHANGE}}$ pin low. Sense status includes bytes 2 and 3 of the address map. In response to the AT42QT1070 driving $\overline{\text{CHANGE}}$ low, the host reads status bytes 2 and 3. The AT42QT1070 releases the $\overline{\text{CHANGE}}$ pin when all changed status bytes have been read. Handshaking logic within the AT42QT1070 guarantees that any unread changes in any status byte will cause $\overline{\text{CHANGE}}$ to be driven low, even if this occurs while the device is being read.

3. Demonstration Program

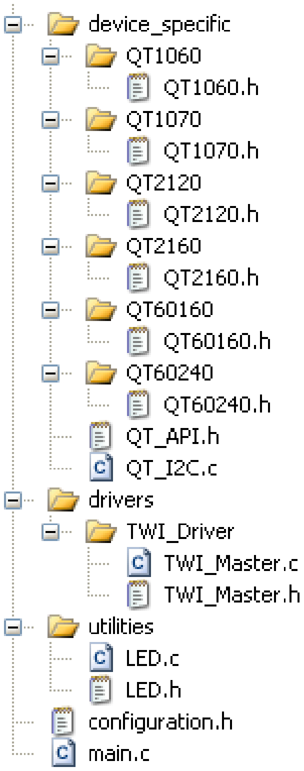
The demonstration program shows how to use the host interface to read real time touch information from QT™ devices. It also demonstrates the procedure to read and write the setup block which helps to tune the operating parameters of the device.

3.1 Program flow



3.2 Files

The folder structure for the demonstration program is shown below.



The source code consists of the following files:

File name	Description
main.c	It consists the main() function and the body of the application program
configuration.h	Device selection and port pin selections for $\overline{\text{RESET}}$ and $\overline{\text{CHANGE}}$ pins are configured here
QT_I2C.c	It consists the application interfaces to drive the QT device
QT1060.h	Header file for the Atmel AT42QT1060 device and it consists AT42QT1060 Setup Block structure and address map enums
QT1070.h	Header file for the Atmel AT42QT1070 device and it consists AT42QT1070 Setup Block structure and address map enums
QT2120.h	Header file for the Atmel AT42QT2120 device and it consists AT42QT2120 Setup Block structure and address map enums
QT2160.h	Header file for the Atmel AT42QT2160 device and it consists AT42QT2160 Setup Block structure and address map enums
QT60160.h	Header file for the Atmel QT60160 device and it consists QT60160 Setup Block structure and address map enums
QT60240.h	Header file for the Atmel QT60240 device and it consists QT60240 Setup Block structure and address map enums
TWI_Master.c	TWI driver code
TWI_Master.h	Header file for the TWI driver
LED.c	Handling LED update based on touch key status
LED.h	Header file for LED source file

3.3 Functions

Table 3-1. Application functions.

Function	Description		
`void InitQtInterface(void)`	Initialize TWI interface for 100kHz and configure change notification ($\overline{\text{CHANGE}}$) pin and $\overline{\text{RESET}}$ pin.		
	Input	none	
	Output	none	
	Return	none	
`uint8_t ReadSetupBlock(uint8_t ReadLength, uint8_t *ReadPtr)`	Read entire setup block from QT device.		
	Input	ReadLength:	Number of bytes to read
	Output	ReadPtr:	Pointer to byte array containing read-data
	Return	TRUE if read successful, else FALSE	
`uint8_t WriteSetupBlock(uint8_t WriteLength, uint8_t *WritePtr)`	Write entire setup block to QT-device.		
	Input	WriteLength: WritePtr:	Number of bytes to write Pointer to byte array containing write-data
	Output	none	
	Return	TRUE if write successful, else FALSE	
`uint8_t ReadKeyStatus(uint8_t ReadLength, uint8_t *ReadPtr)`	Read detection status of all keys from the QT device		
	Input	ReadLength:	Number of bytes to read
	Output	ReadPtr:	Pointer to byte array for read-data
	Return	TRUE if transfer completes, else FALSE	
`void GetCommsReady(void)`	Attempt to read the Device ID until it is read successfully. Then validate the chip ID is correct. If the chip ID is wrong, the program goes no further.		
	Input	none	
	Output	none	
	Return	TRUE if transfer completes, else FALSE	
`void InitTouchStatusPorts(void)`	Configure the PORTB pins for displaying touch status		
	Input	none	
	Output	none	
	Return	none	
`void UpdateLedStatus(uint8_t *QtStatusPtr)`	Update touch key status through LED indications		
	Input	QtStatusPtr:	Pointer to byte array for QTouch data
	Output	none	
	Return	none	

3.4 TWI Master driver

The demonstration program in this application note includes an interrupt-based TWI Master driver using a hardware TWI module and two port pins of the host microcontroller. The TWI communication sequences used to read and write data to the Atmel AT42QT1070 are fully described in the AT42QT1070 datasheet.

The TWI Master Driver implementation is done in the file TWI_Master.c and its header file TWI_Master.h.

Table 3-2. TWI Master driver functions.

Function	Description		
`void twi_master_initialise(uint8_t bit_rate, uint8_t pre_scalar_value)`	Initializes the TWI peripheral as Master with 100kHz speed.		
	Input	bit_rate: pre_scalar_value:	division factor for the bit rate generator (0 - 255) pre-scalar for the Bit Rate Generator
	Output	none	
	Return	none	
`uint8_t twi_transceiver_busy(void)`	This function to test if the TWI_ISR is busy in write\read.		
	Input	none	
	Output	none	
	Return	TRUE if busy, else FALSE	
`uint8_t twi_get_state_info(void)`	This function to fetch the state information of the previous operation. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation. If there was an error, then the function will return the TWI State code.		
	Input	none	
	Output	none	
	Return	Error state	
`uint8_t twi_write(uint8_t slave_address,uint8_t write_address, uint8_t write_length, uint8_t *write_ptr)`	Executes multi-byte write to QT-device. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation, then initialize the next operation, wait till write operation completion and return.		
	Input	SlaveAddress: WriteAddress: WriteLength: WritePtr:	Device address on the TWI bus Register address Number of bytes to write Pointer to byte array containing write-data
	Output	none	
	Return	TRUE if transfer completes, else FALSE	
`uint8_t twi_rewrite(void)`	This function to resend the last message. The driver will reuse the data previously put in the transceiver buffers. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation, then initialize the next operation, wait till write operation completion and return.		
	Input	none	
	Output	none	
	Return	TRUE if transfer completes, else FALSE	
`uint8_t twi_read(uint8_t slave_address,uint8_t read_address, uint8_t read_length, uint8_t *read_ptr)`	Executes multi-byte read from QT-device. The function will hold execution (loop) until the TWI_ISR has completed with the previous operation, before reading out the data and returning. If there was an error in the previous operation the function will return the TWI error code.		
	Input	SlaveAddress: ReadAddress: ReadLength:	Device address on the TWI bus Register address Number of bytes to read
	Output	ReadPtr:	Pointer to byte array containing read-data
	Return	TRUE if transfer completes, else FALSE	
`TWI_ISR`	This function is the Interrupt Service Routine (ISR), and called when the TWI interrupt is triggered; that is whenever a TWI event has occurred.		
	Input	none	
	Output	none	
	Return	none	

4. Porting Code to another Platforms

This section discusses the parts of the demonstration program which needs modification while porting to other MCUs or another I^2C compatible Atmel QTouch device which are listed in the introduction section.

- Do the QTouch device selection in the configuration.h file
- Define the IO port pins which are going to be used for $\overline{\text{CHANGE}}$ and $\overline{\text{RESET}}$ pin in the configuration.h file as below:

```
// RESET port and pin configuration
#define RESET_PORT D      // PORT
#define RESET_PIN  4      // Pin Number

// CHANGE Status port and pin configuration
#define CHANGE_STATUS_PORT      D      // PORT
#define CHANGE_STATUS_PIN       3      // Pin Number
```

- Modify the bit_rate and pre_scalar_value arguments in the function call TWI_Master_Initialise() to change the TWI clock frequency other than 100kHz

Note: The TWI driver and IO pin configuration part of this demonstration program is ONLY compatible for all ATmega devices and some of ATtiny devices.

5. References

[1]. ATmega2560 datasheet.

[2]. Refer the datasheets of the QT device which is going to interface with host controller:

- AT42QT1060
- AT42QT1070
- AT42QT2120
- AT42QT2160
- QT60160
- QT60240

6. Revision History

Doc. Rev.	Date	Comments
42064A	12/2012	Initial document release

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

© 2012 Atmel Corporation. All rights reserved. / Rev.: 42064A–AVR–12/2012

Atmel®, Atmel logo and combinations thereof, Adjacent Key Suppression®, AKS®, AVR®, Enabling Unlimited Possibilities®, megaAVR®, QTouch®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 AND 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 products 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.