AVR245: Code Lock with 4x4 Keypad and I2C™ LCD

8-bit AVR® Microcontrollers

Application Note

Features

- **Application example for code lock**
  - **Ideal for low pin count AVRs**
- **Uses I/O pins to read 4x4 keypad**
- **Uses Timer/Counter to control piezoelectric buzzer**
- **Uses USI in TWI mode to communicate with I2C™ LCD**
- **Firmware written entirely in C language**

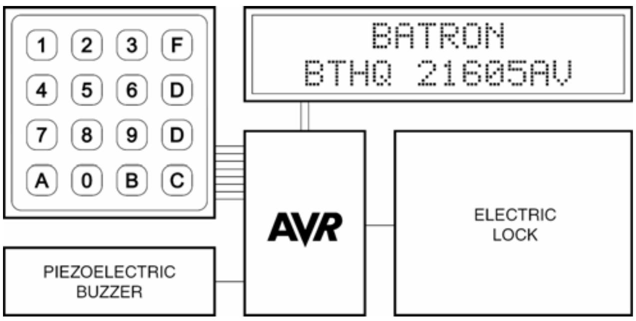
1 Introduction

Code locks make the task of access management more flexible and easy to configure. Mechanical locks require new keys to be manufactured every time a user with access privileges is added. The lock has to be reconfigured or replaced to change access privileges of a single user, and it may also have to be replaced if any of the keys is lost. Situations like these may occur when people move from one household to another, or when people enter and leave as employees of a company. Code locks are easy to reprogram and can contain unique codes for each user.

This application note describes how to build a code lock with an AVR and a handful of components. The code lock uses a 4x4 keypad for user input, a piezoelectric buzzer for audible feedback and an LCD for informational output.

The design is based on ATtiny24/44/84 devices, but can easily be migrated to other AVR microcontrollers.

Figure 1-1. Block Schematic of Code Lock.



Rev. 8013A-AVR-10/05

2 Theory of Operation

The code lock described in this application note consists of a tinyAVR, a keypad, an LCD, a piezoelectric buzzer and the actual (door) lock. The AVR constantly scans the keypad for user input and will release the lock when the correct number sequence has been entered. An LCD is used for feedback and status information. A small piezoelectric buzzer is used for giving audible feedback on keystrokes and lock status.

2.1 Keypad

The input device is a standard 4x4 matrix keypad. The keypad contains 16 keys, symmetrically arranged in four rows with four keys, each. Each column and row of the keypad is connected to an I/O pin.

The microcontroller continually scans the keypad by setting all row pins except one to high and reading all column pins. All column inputs have pull-up resistors enables so when a key is pressed the microcontroller will record a low level on one of the column inputs.

2.2 Piezoelectric Buzzer

Piezoelectric sound components produce clear, penetrating tones, free of harmonics. The devices have high acoustic output and low power requirements, making them ideal for microcontroller applications.

The sound source of a piezoelectric buzzer is the diaphragm. Applying a DC voltage to the device causes mechanical distortion to the diaphragm and applying an AC voltage will move the diaphragm in a repeated bending motion, creating sound waves. The tone of the piezoelectric buzzer is directly proportional to the frequency of the AC signal applied.

A timer/counter of the AVR is well suited for generating the piezoelectric control signal.

2.3 LCD

The display used in this design is a 2x16-character LCD with an integrated controller. The special feature of this display is the I^2C™ interface on the LCD controller. The I^2C interface is compatible with the Two-Wire Interface (TWI) found on AVR controllers, and since ATtiny24 is equipped with a Universal Serial Interface (USI) capable of operating in TWI mode this means the LCD can be controlled using only two wires and pins.

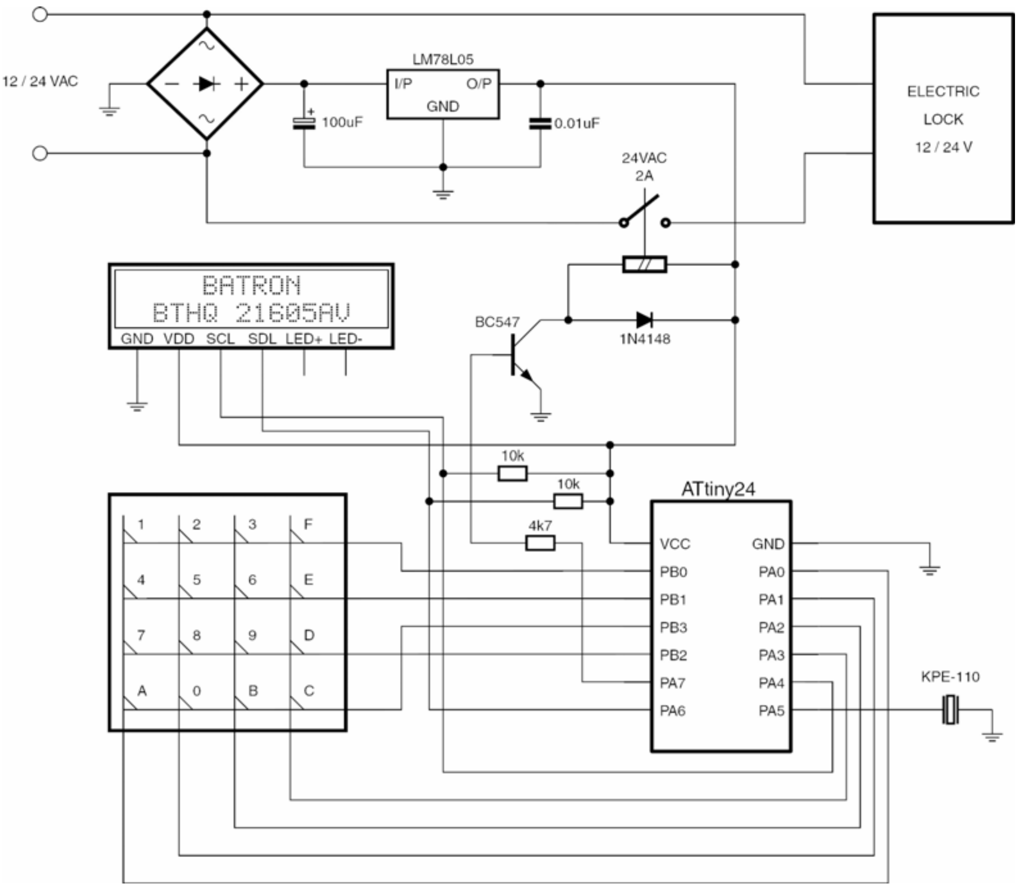
3 Implementation

The hardware implementation is rather straightforward. A prototype can be built easily using an STK500, an STK505, some jumper cables and the essential keypad and LCD.

3.1 Hardware

The schematic is illustrated in the figure below.

Figure 2-1. Schematic.



The electric lock is an application specific part and can be an electromechanical striker, a magnetic lock, or anything relay-driven. Lock manufacturers include ASSA ABLOY and ESMI, for example.

3.2 Firmware

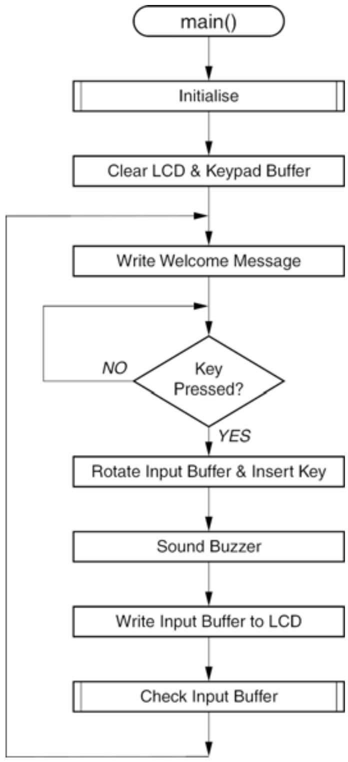
The firmware is written entirely in C language. This is to make the design easy to understand and to make further development of the design as straightforward as possible.

The software is fully documented in the doxygen documentation provided with the source code (see `readme.html`). Compiler requirements and info are also listed in the file.

3.2.1 Overview

The flow chart of the main program is shown in the figure below.

Figure 2-2. Flow Chart of Main Program.



3.2.2 TWI Driver

The TWI driver handles the serial interface protocol. It is based on application note AVR310 and uses the Universal Serial Interface (USI) in Two-Wire Mode (TWI). The driver provides a communication layer between the LCD driver and the LCD.

Associated files:

- USI_TWI_Master.c
- USI_TWI_Master.h

3.2.3 LCD Driver

The LCD driver provides a user-friendly interface between the main program and the LCD. It offers basic functions such as initialization and clearing of the LCD. It also offers functions for writing a single character to any location and writing a full row of text to the display.

Associated files:

- LCD.c
- LCD.h

3.2.4 Keypad Driver

The keypad driver uses timer/counter 0 to time the scanning sequence. After a timer/counter overflow the active keypad row is increased and wrapped around, when necessary. A key push is registered and decoded via the Pin Change Interrupts.

Associated files:

- KPD.c
- KPD.h

Note: debugWIRE requires one of the pins used for the keypad interface. When the symbol DEBUGWIRE in KPD.h is defined this pin will be freed for debugging, but the last row of the keypad cannot be used.

3.2.5 Event Timer

The design uses the watchdog timer as an event window handler. Each key press refreshes the watchdog timer and if no key presses are detected within a predefined time the watchdog interrupt will trigger. In ATtiny24/44/84 it is possible to choose between a watchdog and interrupt and a hardware reset.

The watchdog interrupt service routine clears the keypad buffer and erases the line of LCD where keystrokes have been recorded.

3.2.6 Buzzer Driver

The piezoelectric sound element is connected to the output of timer/counter 1.

The buzzer control function uses the input parameter to directly set the output frequency. This means the frequency response of the buzzer is inversely proportional to the function parameter. In order words, increasing the value of the calling parameter decreases the frequency of the buzzer.

A more user-friendly approach would be to take the inverse of the calling parameter and use the result to program the timer/counter. Alas, this requires a lot more code space due to the inclusion of a divide operation.

4 Quick Start

Power up the application. The firmware will display the welcome message, as follows:

```
Tiny24 CODE LOCK
```

The second row is empty at this time. Push key number one on the keypad. The display will now update to show the keypad buffer, as follows:

```
Tiny24 CODE LOCK
    .......1
```

If no other keys are pushed in a few seconds time the keypad buffer and the second line of the display will be cleared.

Enter the key sequence 1234. The application will accept this as the access code and proceed to open the lock. A progress bar is shown to indicate how long the lock will remain open, as follows:

```
   LOCK  OPEN
■■■■■■■■■■■■■■
```

When the progress bar has decreased to zero the lock will close.

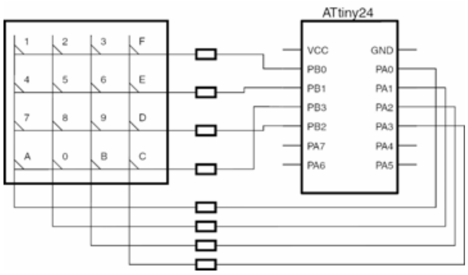
5 Further Development

Here are outlined some ideas for improvement and further development of the design.

5.1 ESD Protection

In some applications it may be necessary to protect the internal electronics from external ESD. This is readily done by adding current limit resistors in series between keypad column & row pins and the microcontroller. This is shown in the figure below.

Figure 5-1. ESD Protection.



ESD protection resistors are typically in the order of 1 kohm, or less.

5.2 Freeing Up I/O Pins

This application uses all available I/O pins of the 14-pin microcontroller. For applications where more I/O is required, it is possible to free up some of the I/O pins, hence removing the need to migrate to a microcontroller with higher pin count. At the end of the day, even 8-pin AVR microcontrollers can be used for this application.

The following approaches are suggested.

5.2.1 Smaller Keypad

Consider replacing the 4x4 keypad with a 4x3 keypad. Typically, only numerical keys and one or two special keys are required. This approach saves one I/O pin.

5.2.2 Keypad and LCD Multiplexing

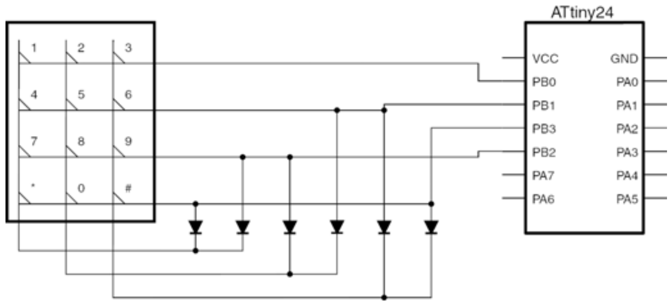
Consider multiplexing the keypad and the LCD. It is not required to write to the LCD and read the keypad at the same time. These two functions can therefore be multiplexed, freeing up two I/O pins.

5.2.3 Compact Keypad Interface

Consider using diodes to reduce the number of pins required to read the keypad. It is possible to use only four I/O pins to control a 4x3 keypad. This enhancement saves up to four I/O pins.

To interface a 3x4 keypad using only four I/O pins requires the use of six diodes, as shown in the figure below.

Figure 5-2. Using Four I/O Pins to Control a 3x4 Keypad.



The keypad driver in this configuration is similar to the default configuration; only the return codes from I/O pins are different. There is no longer a one-to-one relationship between key pressed and code returned. A single key press may toggle more than one I/O pin.

The algorithm for scanning the keypad is as follows. First, set all four I/O pins to inputs and enable pull-up resistors. Then, set one I/O pin to output, low and read the three other I/O pins. The three-bit information will indicate if a key in the corresponding row was pressed or not. Repeat scanning for all I/O pins.

5.3 Additional I2C Devices

The firmware already contains a driver for I2C communication, which means that it is a very simple task to add other devices with I2C interface on the same bus. A few examples of such devices are:

- Card Readers
- Fingerprint Sensors
- Remote Sensors (such as RF)

5.4 Multiple Access Codes

It is a simple task to add multiple access codes to the design. The only limit is the amount of memory available. The default firmware fits into the memory of an ATtiny24 without code compression techniques. Applying code compression (by setting IAR compiler to Release Mode) frees up some space for additional access codes. More space is naturally available in pin and functionally compatible devices, such as ATtiny44 and ATtiny84.

6 Table of Contents

Atmel Corporation

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

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

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

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

8013A-AVR-10/05