Build This Garage Door Keypad

By Reinhard Metz and David Wickliff

- ***No AC required - Self-powered from operator !***
- ***More secure than commercial alternatives !***
- ***Useful for other applications as well!***

In this article we show you how to build an inexpensive keypad secure entry system for your garage door. In addition to the convenience of providing keyless home entry for your kids, you'll find it usefull in many other cirumstances - going out for a bike ride, working in the back yard, etc.

It's advantage over the commercially available units is that it is self-powered from the two wire relay circuit that already exists with most installed openers - no battery, no separate 110 volt A.C. or inconvenient outlet transformer is required! It's also designed to be more secure than conventional keyless entry systems. In addition to the added convenience of this handy device, it's another excellent opportunity to learn about the popular PIC microcontroller and serial EEPROM devices, which are at the heart of the unit.

Functional Overview

The keypad operation is extremely simple: To program an entry code, a program button is pressed inside the unit (normally out of reach inside the garage), and a key sequence (of up to 127 digits) is entered, followed by the # sign. The unit stores these, and access is allowed (by operation of a relay) when the same sequence, again terminated by the # sign, is entered. Additional security is provided by time-outs. Furthermore, only the keypad is located outside, connected to the electronics by a ribbon cable. Thus a potential intruder is prohibited from prying the unit off the wall and shorting the two wires going to the door operator to gain access, as is possible with some of the commercial units or key switches.

Design Overview

Four main components constitute the essentials of the design: A PIC 16C54 microcontroller for the logic, a 93AA46 serial EEPROM for access code storage, the keypad for input entry, and a power circuit. Figure 1 shows the schematic diagram. The PIC microcontroller scans the program button and keypad via eight I/O leads, pulled up by the resistor array R1. Key presses are detected by the

microcontroller as low input signals in a scanning process described later in the firmware section.

The PIC connects to the memory device with three leads. The serial EEPROM has a very simple and elegant operation, which consists of a sequence of serial control, addressing, and data I/O operations, as depicted in timing diagrams shown in figure 2.

The self-powering aspect of the design is provided by rectifier BR1, which rectifies what typically appears as an AC or DC voltage from 10 to 24 volts from the garage door opener. Typically, the opener circuit works similar to the partial schematic shown in figure 3. Normally, the opener is activated by shorting the control leads, which energizes a relay or circuit in the opener. The trick in this unit is to rectify and use some of the power available at the two control wires without energizing the relay. This is easy given the low power requirements of the PIC device. Then, to energize the opener, a relay in the keypad unit shorts the control leads for one second, during which time the PIC, memory, and relay circuits are operated from energy stored in filter capacitor C1. Finally, IC3 regulates the filtered power to 5 volts for the logic. While many, especially older installed garage door openers will work with this power scheme, some newer models may not. In that event, a pair of jumpers may be removed from the relay contact and the contact wired directly to the door operating circuit, and the circuit may then be powered from an auxiliary wall transformer.

Firmware Overview

The software for the garage door keypad system is written in PIC assembly language and is programmed into the on-chip EEPROM of the microcontroller. Both the source code file and a pre-assembled hex file for programming a microcontroller are available. A pre-programmed PIC 16C54 microcontroller is also available.

The main loop

Various program variables used by the software are configured immediately after the microcontroller is powered on. After this initialization the microcontroller loops in the main software loop illustrated in the flow chart of Figure 4.

The first step of the main loop is to configure the input and output port pins of the microcontroller. This port configuration was cleared due to either a power-on reset or, as we will see later, a watchdog reset.

Next the keypad is sampled for any new key presses. A key press is considered valid if the same key value is sampled on two consecutive passes of the main loop. This multiple sampling requirement ensures that only one key press is registered even though the keypad switch contacts may bounce on closure and

release. The value of a valid key press is flagged and stored for later use by the state machine software discussed in the next section.

The next step of the main loop is to decrement a keypad entry timeout count. Each pass of the main loop this counter is decremented by one. If the count ever reaches zero, the state machine software is reset to the start of comparing keypad presses. However, whenever a key is pressed this counter is re-initialized to a large value. Operated in this way, the keypad counter will clear any partial key sequences entered that are not completed within approximately 60 seconds (e.g. the small neighbor kid playing with the keypad or a basketball hit). So, the next time someone enters a valid key sequence it is correctly recognized.

After maintaining the key entry timeout, the next step of the main loop is to run the state machine software. As we will see in the next section, the state machine software is where most of keypad lock's behavior is implemented.

Finally, the on-chip watch dog timer of the microcontroller is used at the end of the main loop to reset back to the beginning of the loop. This timer will cause a reset of the microcontroller when it is allowed to expire. Normally, the software would frequently clear the watch dog timer such that it is not allowed to expire. However, at the end of the main loop a SLEEP instruction is executed and the microcontroller stops executing instructions. The watch dog timer will expire approximately 18ms later and reset the microcontroller back to the beginning of the main loop.

The software state machine

Most of the software is organized using a software programming technique called the software state machine. First you describe your programming problem with a traditional state diagram. Next you write small software code segments that perform the actions of each state of the diagram. Finally, you initialize a variable which will represent the current state of the state machine and you periodically execute the segment of software that corresponds to the current state. This technique allows you to organize and breakdown a large problem into smaller and more manageable pieces.

A variable in the PIC's RAM is used to represent the current state of the state machine. This variable is simply called "State" in the code. The main loop software uses the current value of "State" to determine which of the state machine software segments to execute on each iteration of the loop. A segment could choose to change the value of the "State" variable, thereby forcing the state machine into a different state.

Figure 5 illustrates the state diagram for the garage door keypad. A power-on reset starts the state machine off in the COMPARE DIGITS state. In the

COMPARE DIGITS state, entered keys are compared with the series of key values stored in the EEPROM. The state machine continues to stay in the COMPARE DIGITS state as long as the entered keys match the EEPROM's key values. If a "#" key is entered and it matches the next stored key value, then the user has entered the correct code and the state machine advances to the UNLOCK state. If ever an entered key does not match the next key stored in EEPROM, then the state machine advances to the LOCKOUT state.

The UNLOCK state simply activates the relay for approximately 1 second - opening or closing the garage door. The LOCKOUT state discards entered keys until a "#" is entered. In this way, the potential intruder is not alerted that the key sequence that he/she is entering has stopped matching the stored EEPROM sequence.

If the PROGRAM button is pressed in any of the other states, the STORE DIGITS state is entered. In this state, entered keys are sequentially stored into the EEPROM until a "#" key is entered.

The software segments that perform each state generally should not loop within themselves, but perform the desired action only once and exit. For example, the code for the STORE DIGITS segment stores only one digit at a time.

Listing 1 shows the segment of PIC assembly software that implements the STORE DIGITS state. Figure 6 shows the flow chart for this segment of software.

The STORE DIGITS state software segment as well as the other state software make use of a number of support subroutines for blinking the LED (BlinkLED and WinkLED), reading the EEPROM (EEPRead), writing the EEPROM (EEPWrite), and delaying for a specified period of time (Delay). See the source code file for details.

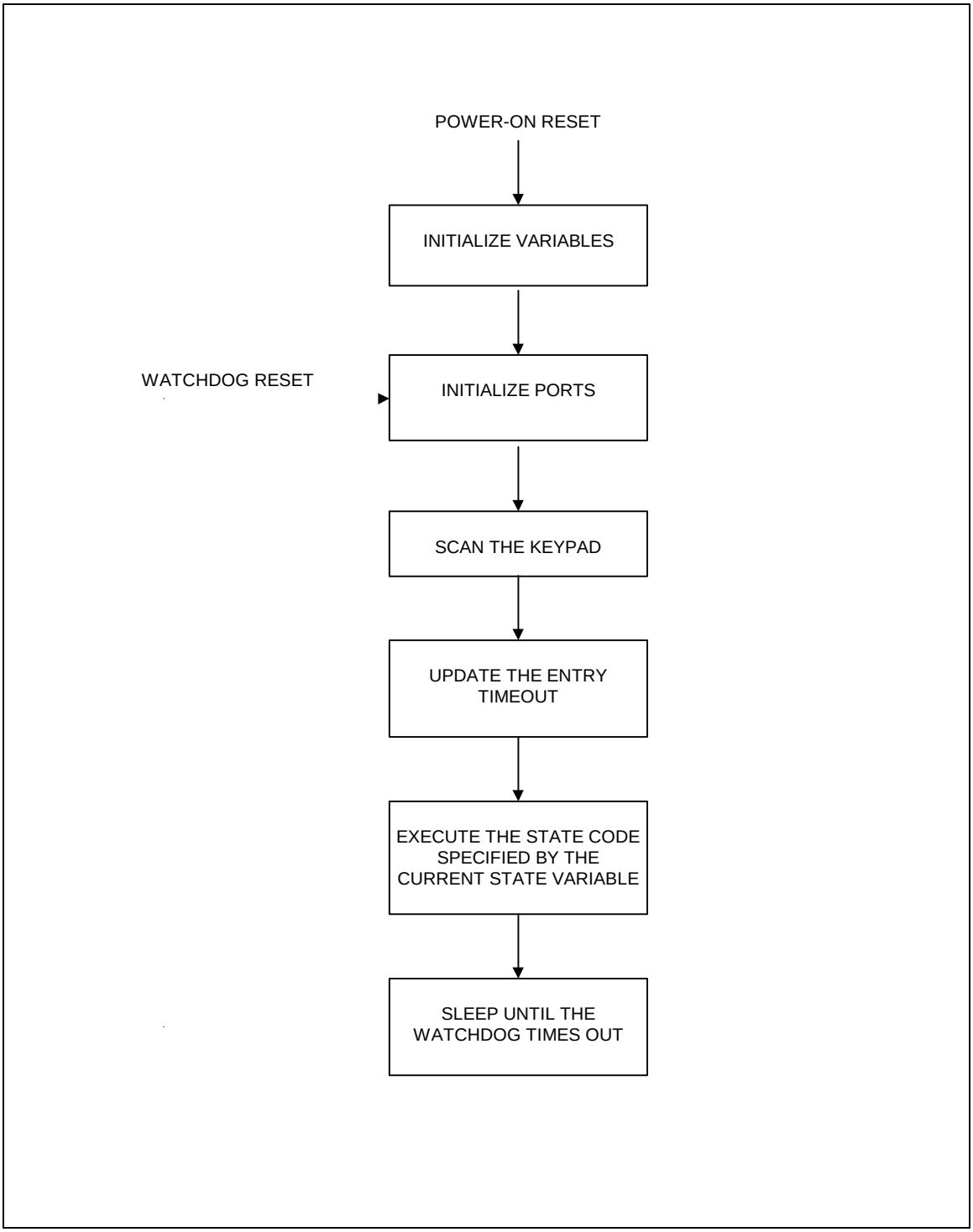

Fig. 4. The flow chart for the main loop.

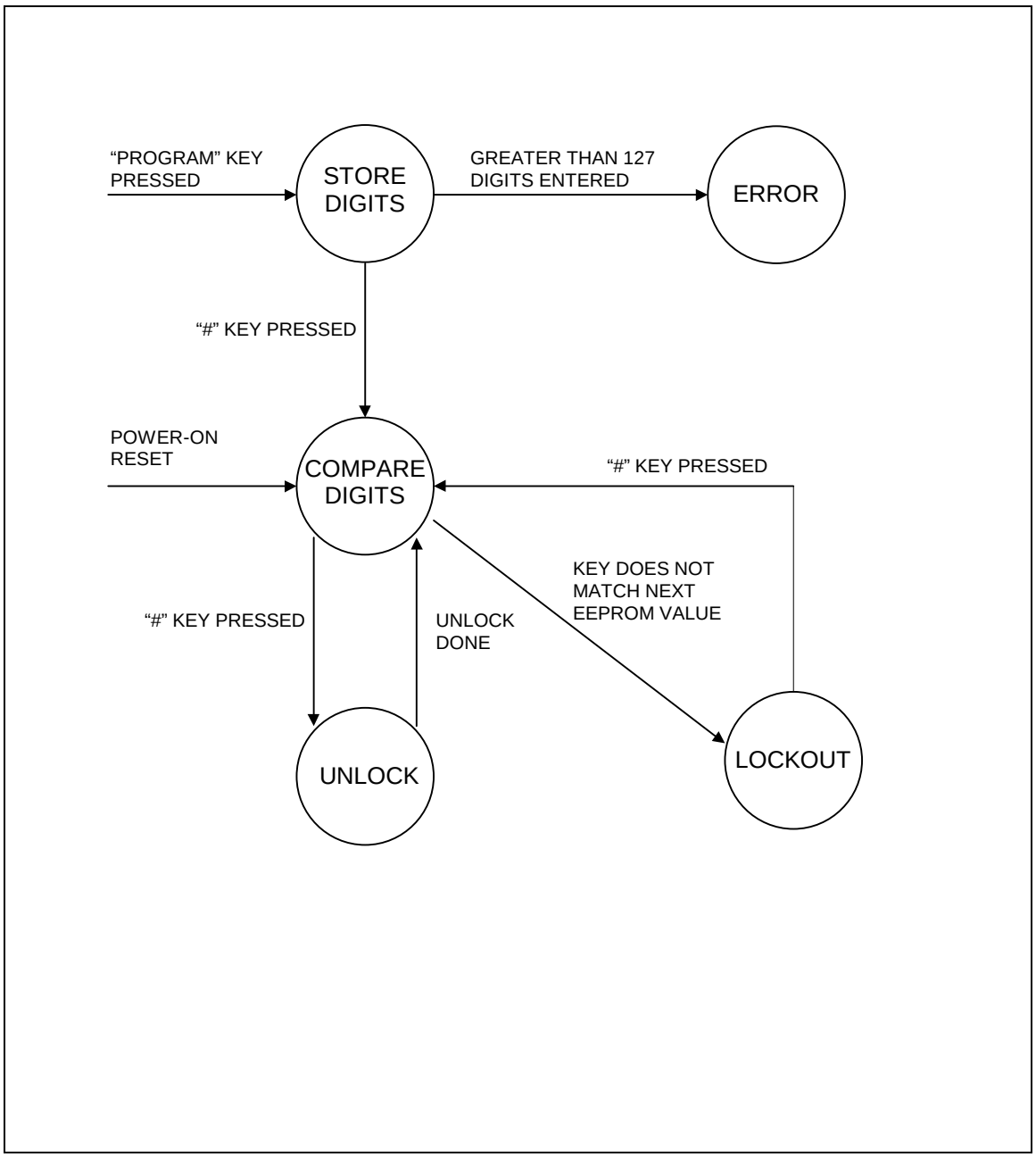

Fig. 5. The state diagram for the software state machine.

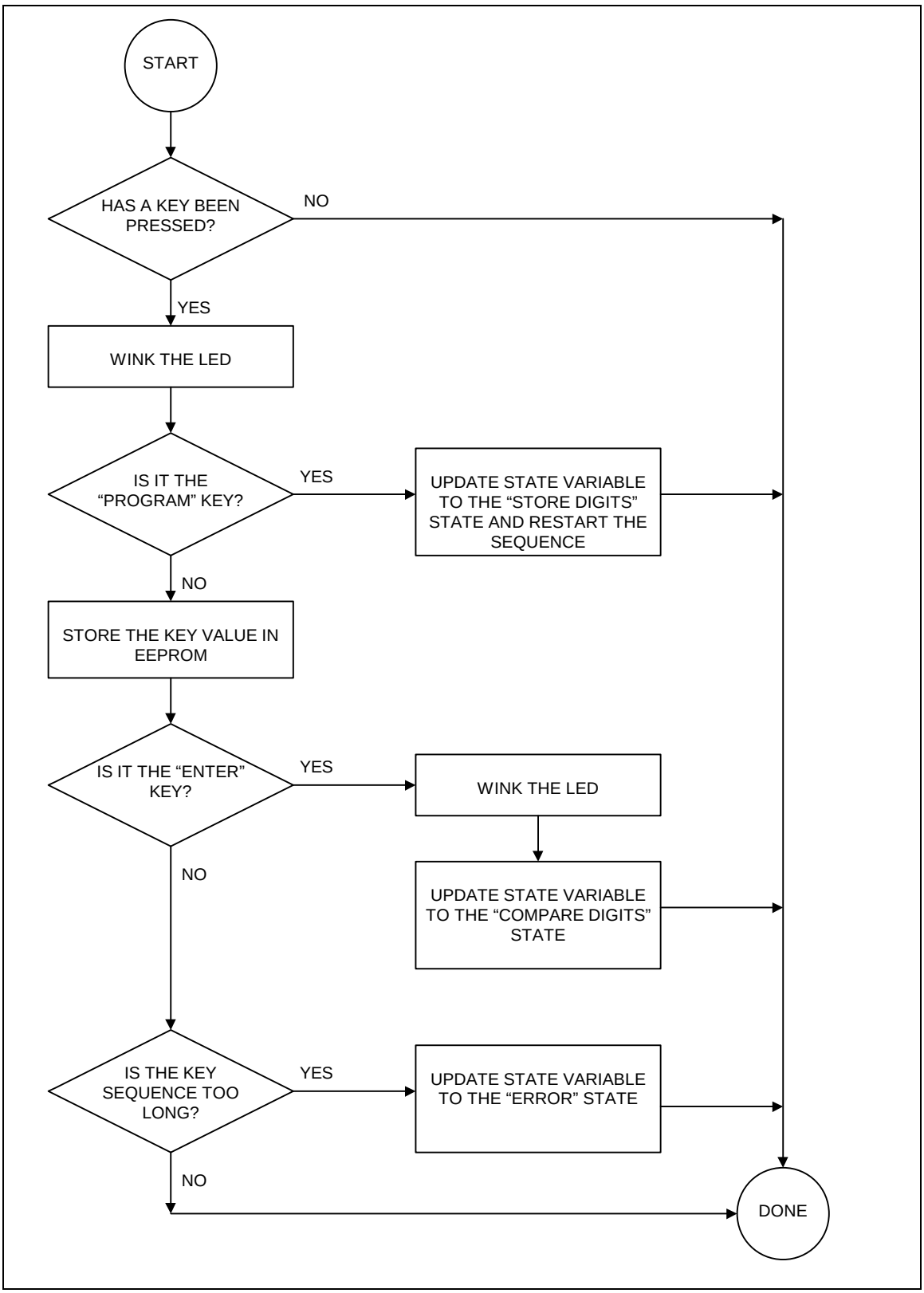

Fig. 6 - The flow chart for the STORE STATE.

LISTING 1

```
;-----------------------------------------------------------------------
StoreDigits
; State Code

      ; If there is a key press:

      btfss  KeyStatus,KeyEvent
      goto   StoreDigitsDone

      bcf    KeyStatus,KeyEvent ; Clear the flag

      call   WinkLED            ; Wink the LED

      movf   Key,W        ; If this is the "PROGRAM" key, then
      xorlw  ProgramKey
      btfss  STATUS,Z
      goto   StoreDigits0

      movlw  SStoreDigits ;   restart the StoreDigits state
      movwf  State
      movlw  MaxKeyCount+1
      movwf  KeyCount
      goto   StoreDigitsDone

StoreDigits0
      call   EEPWriteEnable     ; Enable EEPROM writes

      movf   Key,W        ; Store the key value
      movwf  EEPData
      movf   KeyCount,W
      movwf  EEPAddr
      call   EEPWrite

      call   EEPWriteDisable    ; Disable EEPROM writes

      movf   Key,W        ; If this is the "ENTER" key, then
      xorlw  EnterKey
      btfss  STATUS,Z
      goto   StoreDigits1

      call   WinkLED            ;   wink the LED again

      movlw  SCompareDigits     ;   begin and go to
      movwf  State        ;   the CompareDigits state
      movlw  MaxKeyCount+1
      movwf  KeyCount

      goto   StoreDigitsDone

StoreDigits1
      decfsz KeyCount,F   ; If there are more than the maximum
      goto   StoreDigitsDone ; number of keys entered, then

      movlw  SStoreError  ;   go to the Error state
      movwf  State

StoreDigitsDone
      sleep
      goto   Begin        ; (Not executed by 16c5x type of devices)
```

Construction

Circuit board artwork is shown in Figure 7, if you should choose to make your own PC boards. Begin assembly by installing all board mounted components, carefully checking resistor values and component orientations. Temporarily connect the keyboard, and if a bench power supply is available, apply 10 to 24 volts through a 1K resistor in series to the circuit power input. Push the program button, and enter a short digit sequence, followed by the # sign. Re-enter the same sequence, again followed by "#". You should hear the relay operate for about one second. An LED (LED1) may be optionally installed, along with R3, to indicate button presses and successful operation of the relay.

Installation

If you use the specified keyboard, prepare for installation by drilling four small corner holes matching the keyboard, and create a slot for the ribbon cable by sawing or drilling a series of holes. Install the keyboard with a bead of clear silicon rubber inside the back perimeter. Run the ribbon cable through the jam inside the garage. Connect it to the circuit board and mount the circuit board inside the garage as desired. Connect the two wires from the relay output/power input to the opener operating contacts or in parallel with the existing inside garage door operating button. If your unit requires auxiliary power, leave out the two jumpers on the board, and connect the relay contacts separately to the door operating circuit, and apply power to the board power input from a wall transformer that supplies from 10 to 24 volts, AC or DC.

Other Applications

You can easily use the keypad for keyless operation in a variety of other applications. If they are not amenable to the powered relay technique applied in the garage door case, then you can simply leave out the jumpers and power the board from a DC or AC source from 10 to 24 volts. Less than 1 milliamp of current is required.