

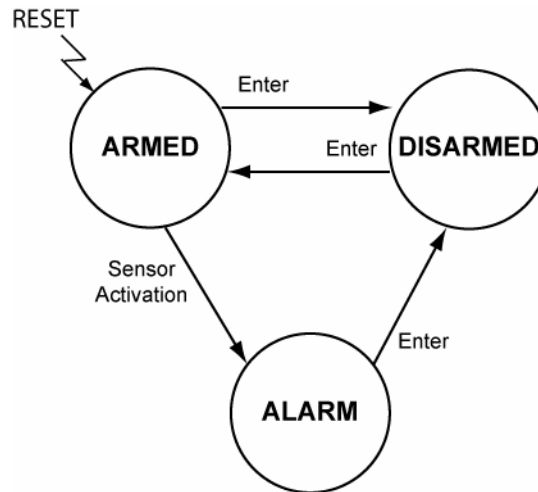
## 1 Introduction

In this experiment, you will be designing an electronic security device. It will consist of a timeslice kernel with three tasks for state control, sensor detection, and LED display. It will be a standalone final product so you will use Lab5 as a foundation.

## 2 Functional Requirements

The security system can be functionally described by the state diagram shown in Figure 1 along with the following description. The LEDs referred to are the LEDs connected to PORTA on the board and the sensors are simulated by the switches connected to PORTB. The sensor is considered activated when the switch is in the 'open' position. Each LED is an indicator for the sensor in the same bit position. For example, the LED connected to PA3 is the indicator for the sensor connected to PB3.

Figure 1 System State Diagram



### 2.1 Armed State - The system is ready to detect intruders.

- 2.1.1 This is the initial state of the system.
- 2.1.2 When this state is entered, **ARMED** should be displayed one time on the terminal.
- 2.1.3 While in this state, the sensors must be checked every 10.24ms.
- 2.1.4 The sensor LEDs should display the rotating pattern shown in Figure 2. An **Enter** key press on the terminal causes a transition to the *Disarmed* state.
- 2.1.5 Sensor activation causes a transition to the *Alarm* state.

### 2.2 Disarmed State - The system is disabled and displays the current state of the sensor inputs.

- 2.2.1 When this state is entered, **DISARMED** should be displayed one time on the terminal.
- 2.2.2 While in this state, the sensors must be checked every 10.24ms.
- 2.2.3 While in this state, the sensor LEDs display the current state of the sensors by the appropriate LED flashing as described in Figure 2. An **Enter** key press on the terminal causes a transition to the *Armed* state.
- 2.2.4 While in this state, the user can press SW6 on the board to test the buzzer. The buzzer sounds as long as the switch is pressed. Note that IC4 is the same input as PT4.

### 2.3 Alarm State - Intruder has been detected. Activated sensors are latched.

- 2.3.1 When this state is entered, **ALARM** should be displayed one time on the terminal.
- 2.3.2 While in this state, the sensors must be checked every 10.24ms.
- 2.3.3 While in this state, the sensor LEDs indicate all sensors that have been activated since the system was armed. To indicate an activated sensor the corresponding LED should flash as shown in Figure 2.
- 2.3.4 The buzzer sounds as described in section 3.3 as long as the system is in this state.
- 2.3.5 An **Enter** key press on the terminal causes a transition to the *Disarmed* state.

Figure 2 LED Display Patterns

Conditions	LED Display																						
Armed State	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">PA7</td> <td style="text-align: center;">PA0</td> <td></td> </tr> <tr> <td>Cycle 1</td> <td>● ○ ○ ○ ○ ○ ○ ○</td> <td rowspan="9" style="vertical-align: middle; padding-left: 20px;">● = On for 256ms (Cycle Period) ○ = Off</td> </tr> <tr> <td>Cycle 2</td> <td>○ ● ○ ○ ○ ○ ○ ○</td> </tr> <tr> <td>Cycle 3</td> <td>○ ○ ● ○ ○ ○ ○ ○</td> </tr> <tr> <td>Cycle 4</td> <td>○ ○ ○ ● ○ ○ ○ ○</td> </tr> <tr> <td>Cycle 5</td> <td>○ ○ ○ ○ ● ○ ○ ○</td> </tr> <tr> <td>Cycle 6</td> <td>○ ○ ○ ○ ○ ● ○ ○</td> </tr> <tr> <td>Cycle 7</td> <td>○ ○ ○ ○ ○ ○ ● ○</td> </tr> <tr> <td>Cycle 8</td> <td>○ ○ ○ ○ ○ ○ ○ ●</td> </tr> <tr> <td>Cycle 9</td> <td>● ○ ○ ○ ○ ○ ○ ○</td> </tr> </table>	PA7	PA0		Cycle 1	● ○ ○ ○ ○ ○ ○ ○	● = On for 256ms (Cycle Period) ○ = Off	Cycle 2	○ ● ○ ○ ○ ○ ○ ○	Cycle 3	○ ○ ● ○ ○ ○ ○ ○	Cycle 4	○ ○ ○ ● ○ ○ ○ ○	Cycle 5	○ ○ ○ ○ ● ○ ○ ○	Cycle 6	○ ○ ○ ○ ○ ● ○ ○	Cycle 7	○ ○ ○ ○ ○ ○ ● ○	Cycle 8	○ ○ ○ ○ ○ ○ ○ ●	Cycle 9	● ○ ○ ○ ○ ○ ○ ○
PA7	PA0																						
Cycle 1	● ○ ○ ○ ○ ○ ○ ○	● = On for 256ms (Cycle Period) ○ = Off																					
Cycle 2	○ ● ○ ○ ○ ○ ○ ○																						
Cycle 3	○ ○ ● ○ ○ ○ ○ ○																						
Cycle 4	○ ○ ○ ● ○ ○ ○ ○																						
Cycle 5	○ ○ ○ ○ ● ○ ○ ○																						
Cycle 6	○ ○ ○ ○ ○ ● ○ ○																						
Cycle 7	○ ○ ○ ○ ○ ○ ● ○																						
Cycle 8	○ ○ ○ ○ ○ ○ ○ ●																						
Cycle 9	● ○ ○ ○ ○ ○ ○ ○																						
Alarm and Disarmed States	<p>Example: Sensors 0, 2, 4, and 6 activated</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">PA7</td> <td style="text-align: center;">PA0</td> <td rowspan="2" style="vertical-align: middle; padding-left: 20px;">● = Flashing with a period of: 133ms in Alarm State 512ms in Disarmed State</td> </tr> <tr> <td>○ ● ○ ● ○ ● ○ ●</td> <td>○ ● ○ ● ○ ● ○ ●</td> </tr> </table>	PA7	PA0	● = Flashing with a period of: 133ms in Alarm State 512ms in Disarmed State	○ ● ○ ● ○ ● ○ ●	○ ● ○ ● ○ ● ○ ●																	
PA7	PA0	● = Flashing with a period of: 133ms in Alarm State 512ms in Disarmed State																					
○ ● ○ ● ○ ● ○ ●	○ ● ○ ● ○ ● ○ ●																						
Incorrect Combination 4 times	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">PA7</td> <td style="text-align: center;">PA0</td> <td rowspan="2" style="vertical-align: middle; padding-left: 20px;">● = Flashing with a period of 133ms</td> </tr> <tr> <td>● ● ● ● ● ● ● ●</td> <td>● ● ● ● ● ● ● ●</td> </tr> </table>	PA7	PA0	● = Flashing with a period of 133ms	● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ●																	
PA7	PA0	● = Flashing with a period of 133ms																					
● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ●																						

### 3 Task Design and Startup Code Requirements

#### 3.1 Startup Code

3.1.1 The startup code is identical to Lab6 including the RAM and ROM tests.

#### 3.2 Task Design Notes

There must be three tasks for this system, a sensor task, an LED display task, and a state control task. These tasks must be designed to meet the functional requirements given in Section 2 along with the following requirements:

- 3.2.1 The sensor and LED tasks are hardware driver tasks. They should ONLY deal with the hardware associated with that task and no other task may access these hardware modules without going through these tasks. For example, the sensor task should handle all accesses to/from the sensors. It should not directly write to the LEDs, send/receive data to/from the terminal or change states.
- 3.2.2 Since all tasks need to know the present state of the system, a global variable should be used to communicate the state parameters from task to task.
- 3.2.3 Since we are using a cooperative kernel, we can not use a blocking routine. So, for example, you will need to use *INPUT()* for terminal input instead of *GETCHAR()*.

#### 3.3 Alarm Buzzer

The audio transducer (buzzer) on the board is connected to PP7. This corresponds to the pulsewidth modulator, PWM7. When an alarm occurs, cycle the buzzer with the same period as the LEDs. ie. when the LED(s) are on in the alarm state, the buzzer is on. Excite the buzzer with a 3kHz pulse train that has a duty cycle between 33% and 67%. You may control the PWM (buzzer) with the LED task.

When testing the buzzer be respectful of your fellow students in the lab. Note that JP19 disconnects the buzzer.

#### 3.4 Debugging Helper Signals

There must be a helper signal for the time slice routine and each task. Each signal should be high when the CPU is running the corresponding task. These signals must use PORTK bits 0 through 3 as follows:

Task	Debug Bit
Time Slicer	PK0
Sensor Task	PK1
LED Task	PK2
State Cntl Task	PK3

### 4 Extra Credit Optional Functions

#### 4.1 Make it Green

Modify your program so the MCU goes into the WAIT mode when waiting for the next timer event. This mode is documented in the notes, in the devices guide (9S12DT128DGV2.pdf), and the Clock module (S12CRGV4.pdf). Configure the system so the system clocks and the core clock are turned off while in the wait mode. Make sure that the RTI is enabled while in the WAIT mode or you will not be able to wakeup.

#### 4.2 Add a combination for disarming.

Expand this security program by replacing the **Enter** with a four character alphanumeric combination for the *armed* → *disarmed* and *alarm* → *disarmed* transitions. Characters entered should be echoed as \*. If an incorrect combination is entered four times, the program enters the *alarm* state with the LEDs flashing as indicated in Figure 2. Wait for all four combination characters to be entered before indicating an error.

### 5 Grading and Write-up

Your operating program must be checked by the instructor before the source code due date.

The source code for your program must be sent via email by midnight on the due date. The write-up must be turned in before 5:00pm on the due date and include the following material:

**Introduction**

**Program Description**

- Including a flow diagram(s).

**Timing Analysis (Measured)**

- Peak Task Execution Times
- Peak CPU Load

**Program Listing (hardcopy)**

**Comments and Conclusions**

**Due Dates: Source – March 14, 2008, Write-up March 14, 2008**

**All Lab material is due on March 14, 2008. No late labs will be accepted after this date.**