Designing a Working Prototype using MQX™ RTOS and Tower System for Kinetis Microcontrollers

Kinetis Intelligent Sprinkler System

by: Maclain Lobdell, Ron Kim, Anthony Huereca
Microcontroller Solutions Group

1 Introduction

Designing a Microcontroller-based system can consume a significant amount of time and resources. Hardware and software must be quickly and carefully developed to avoid problems that can delay time to market. Fortunately, Freescale provides software and hardware tools that accelerate development time and allow for fast prototyping. The ability to quickly prototype a system can lower the overall project risk and increase the likelihood of a successful product. The purpose of this application note is to provide a working example of a Kinetis Microcontroller based system developed with Freescale software and hardware tools.

The following Software and Hardware tools have been used:

- Freescale MQX™ Real Time Operating System (RTOS)
- Freescale Tower hardware prototyping platform

Additionally, this application note provides examples of using several powerful features of Freescale Kinetis ARM Cortex-M4 Microcontrollers.

Following are the Kinetis ARM Cortex-M4 Microcontroller features and capabilities that have been used in this example:

- Segment LCD
- Real-time Clock
- FlexMemory
- SPI interface to Wi-Fi system
- FlexBus external bus interface to graphic LCD display
2 Intelligent Home Irrigation (Sprinkler) System

Resource conservation is important for minimizing human impact on the environment. As population growth continues and resources become more scarce, societies and individuals must find ways to do more with less. Many communities have home irrigation sprinkler systems installed at residences and commercial properties that supplement rainfall and keep lawns and gardens lush and green. Unfortunately, many of the existing systems are inefficient and wasteful. With the growth of ubiquitous wireless networking, it is now possible to create intelligent network attached efficient systems. These systems can communicate with the network to obtain weather data to use along with its local sensors to determine the most effective watering schedule. Freescale provides Microcontroller tools and software to create important products such as this that can positively impact the world.

2.1 Intelligent Home Irrigation (Sprinkler) System

We chose to prototype a home irrigation system because of the following:
• It is a simple and recognizable application
• The concepts can be expanded to many more complex and demanding applications
• It showcases a complete solution
• It really works

2.2 System Requirements

The following are the system requirements:
• Practical, Capable, and Efficient
  • Small size
  • Low cost, Low power
  • Fast development time
  • Flexible and easy to expand
  • Connect to common Wi-Fi or wired Ethernet home network

• Intuitive User Interface
  • Touch screen graphical user interface
  • Smart phone web browser interface or smart phone application
  • Segment LCD to display information

• Robust System
  • Power loss recovery
  • Segment LCD fault detection
  • Data logging for analysis of system performance and weather data
• The system will be located indoors (for example, garage)

2.3 Development Requirements

• Use Freescale complementary Microcontroller software.
• Develop with IAR Embedded Workbench® for ARM or Freescale CodeWarrior Development Studio for Microcontrollers.
• Use Freescale Tower modules for low cost hardware prototyping. No debug hardware pod required.
• Six week development schedule with three engineers working on hardware/software development, integration, testing, and debug.
• Additional costs may be incurred for MQX RTOS paid support if required.

3 System Overview

This application note presents two system implementations that each highlight different features and capabilities of Kinetis Microcontrollers.

System 1: Kinetis K60N512-based Home Irrigation System Controller

![System Block Diagram](image)

**Figure 1. Kinetis K60N512 System Block Diagram**

Features:

• SPI interface to Wi-Fi system
• FlexBus external bus interface to graphic LCD display
• ADC connection to touch screen overlay
• Ethernet Controller
System Overview

- Hardware touch sensor interface (TSI)
- Real-time Clock
- I2C communication with external Freescale sensors
- Low power modes with Low-leakage wakeup unit
- General-purpose input/output (GPIO)

System 2: Kinetis K40X256-based Home Irrigation System Controller

Figure 2. Kinetis K40X256 System Block Diagram

Features:

- Segment LCD
- FlexMemory
- Real-time Clock
- I2C communication with external Freescale sensors
- Low power modes with Low-leakage wakeup unit
- General-purpose input/output

3.1 Custom Relay Module for the Tower System

The Tower system is an excellent platform for prototyping because it is easy for anyone to build modules to interface to the system. Freescale and its partners provide many Tower modules to prototype a system with, however many applications require additional application-specific circuitry. This can be accomplished by wiring circuits on the TWR-Proto prototyping module or by designing and building a custom Tower module. Tower modules are inexpensive because they do not require a costly connector interface to other boards. The card edges are all that are required to access the multitude of signals running through the Tower elevator modules.
A custom module was designed and built for this prototype system. The module holds relay circuits capable of switching 24 V of AC power that is required by typical home irrigation systems. The relay circuits are controlled by GPIO signals on the Microcontroller module. The GPIO signals pass from the Microcontroller module down through the Tower Elevator module to the relay module to control the relays.

See the reference section of this application note for more information on the relay module.

![Figure 3. Relay Module](image)

### 3.2 Kinetis K60 system

The implementation of the intelligent sprinkler system on the Kinetis K60 system showcases the following capabilities:

- Graphical, resistive touch-based LCD to display system status and receive input from the user
- HTTP Web Server for remote status and control. Available over Wi-Fi or wired network.
- Accurate time of day using Real Time Clock
- Low power state when the system is inactive to save power
- TSI to wake up system from low power state

### 3.2.1 Prototyping the design with the Tower System

#### 3.2.1.1 Hardware modules used

- Kinetis K40 MCU Module (TWR-K40X256)
- Tower Sensor Module (TWR-SENSOR-PAK)
- Tower Elevator Modules (TWR-ELEV)
- Custom Relay module

All modules are available from [https://www.freescale.com](https://www.freescale.com) except the custom relay module, which was created specifically for this prototyping exercise.
3.2.2 Setting up the system

See the Setup Guide located with source code for details on hardware and software setup.

3.2.3 Design implementations for each key feature

For additional information, see the reference section of this application note.

Graphical LCD

The graphical LCD used in this project provides an eye-catching and easy-to-use interface to the system. In addition to the graphic display capability, it has a resistive touch interface to provide touch screen input. The LCD can be used with almost any MCU with a SPI or external bus and analog-to-digital converters (ADC). Smart LCD screens, such as the one used on the TWR-LCD, are panels with integrated display RAM and serial or parallel bus interface. The evolution of LCD panels integrating the display RAM and LCD controller means that LCD systems can easily be implemented without the need for a conventional microprocessor (MPU) with graphical controller on the chip.

The MCU module communicates with the TWR-LCD either by a 8- or 16-bit external bus, or by SPI. For this application, the FlexBus external bus interface in 16-bit mode is used. The screen resolution is 320 x 240. The screen is refreshed every 100 ms or 10 times per second.

Enabling the LCD is Freescale’s free eGUI software. This software implements a graphical user interface that drives the smart LCD. The graphics used for the system are created specifically for this application, and the eGUI Image Converter is used to bring them into the eGUI software. For more information about eGUI see the Reference section.
**FlexBus interface to the TWR-LCD**

For this application, the FlexBus is used to send screen image data to the TWR-LCD module. The FlexBus is configured for 16-bit wide non-multiplexed mode and runs at 12 MHz. Screen image data is sent to the TWR-LCD module to refresh the screen every 100 ms. Refer to eGUI reference manual available at https://www.freescale.com for information about the driver to interface to the TWR-LCD.

**Resistive Touch Overlay support on the TWR-LCD**

The analog output of the TWR-LCD resistive touch overlay is measured by four ADC channels that are connected to the Kinetis MCU. The measurement result determines the x-y position of touch events. In this application, the result of the ADC conversions is polled every 100 ms.

**Wi-Fi/Wired Network**

The system is network-connected to enable remote monitoring and control. Additionally, the network connectivity provides the capability of using advanced water efficiency techniques to conserve water, such as communicating with a weather server. The system is designed with basic network capability for remote monitoring and control. Additional software framework will be required to be developed to enable the system to interact with a weather server and adapt its watering schedule.

The system uses DHCP to obtain an IP address from the network. The IP address is displayed on the graphical interface in the “Web” details page. There is also the option to use a static IP address if so desired.
The MQX Real-Time Communication Suite (RTCS) is used to manage the TCP/IP networking. The RTCS networking features function identically over Wi-Fi and wired networking interfaces.

**Wi-Fi Interface Details**

The system uses a Redpine RS9110-N-11-21 Wi-Fi module to provide wireless connectivity over 802.11b/g/n networks. The system communicates to the Wi-Fi module via a SPI interface. The SPI data rate is 1 MHz. For more information, refer to the TWR-WIFI-RS2101 documentation. See the Reference section for more details.

**Wired Network Details**

Wired Ethernet connection is possible using the 10/100 Mbps Ethernet MAC (ENET) built-in Ethernet controller available on Kinetis K60 devices. The Ethernet controller connects to an Ethernet PHY device via Reduced Media Independent Interface (RMII). The TWR-SER serial interface module provides the Ethernet PHY and RJ-45 jack used to connect the system to a wired network.

**HTTP Web Server**

The system uses a HTTP web server to interface to a web browser on a laptop, tablet, or smart phone. The web server functionality is provided by MQX RTCS.

HTML web pages are stored in the internal memory of the system and served to a web browser when requested. During software development, the HTML web pages are converted to trivial file system data using the mktfs utility provided with MQX. The web page data is then compiled into the application when the software application is compiled.

The AJAX technique is used to create interactive web pages. The HTML pages contain JavaScript which make requests to the server to execute Common Gateway Interface (CGI) functions. The functions either provide information back to the browser or perform some action in the system. The forms and data on the web pages interact with the functions in the application to enable the web browser interface for monitoring and controlling the system. For more details, see the httpsrv software example found in the /rtcs/examples folder of MQX.

![Figure 5. Web Page Interface](image)

**Time of Day**

The current date and time is calculated with the Real Time Clock (RTC), which is backed up by the battery on the underside of the K60 Tower module. MQX can calculate and store the current date and time with API calls, simplifying the code that users have to implement.

**Touch Sense Interface**

The TSI is used to provide more options for interfacing with the system. In this case, it can turn on and off the TWR-LCD screen. Since it uses capacitive touch, there is no need for ugly and protruding buttons. The TSI module in Kinetis automatically samples the inputs to determine when a touch occurs, and all the relevant touch detection occurs in the TSI interrupt after a scan in completed.

**Sensor data collection**
Temperature, humidity, and barometric pressure data is collected by the TWR-SENSOR-PAK module connected in the Tower system. The Sensor module contains a MC9S08QE96 MCU, which samples the various sensors over I2C. The Kinetis system communicates with the MC9S08QE96 over I2C to get the results of those sensor readings. Additionally, the jumper settings on the TWR-Sensor can be configured for the Kinetis to directly get the sensor data over I2C. However, in this application the MC9S08QE96 is used to aggregate the data.

**GPIO**

General purpose output signals are used to control relays that switch on/off sprinkler valves that can be attached to the system. The relays are located on the custom module that was designed for this application.

### 3.3 Kinetis K40 System

The implementation of the intelligent sprinkler system on the Kinetis K40 system showcases the following capabilities:

- Segment LCD to display system status with detection of faults in the display segments
- FlexMemory for data logging
- FlexMemory for fast system recovery after power interruption
- Low power states when sprinkler system is inactive to save power

#### 3.3.1 Prototyping the design with the Tower System

##### 3.3.1.1 Hardware modules used

- Kinetis K40 MCU Module (TWR-K40X256)
- Tower Sensor Module (TWR-SENSOR-PAK)
- Tower Elevator Modules (TWR-ELEV)
- Custom Relay module

All modules are available from [https://www.freescale.com](https://www.freescale.com) except the custom relay module, which was created specifically for this prototyping exercise.

#### 3.3.2 Setting up the system

See the Setup Guide located with source code for details on hardware and software setup.

#### 3.3.3 Design implementations of each key feature

For additional information, see the reference section of this application note.

**Segment LCD**

The use of segment LCD displays is a relatively simple and inexpensive way to display information to the user.

The segment LCD is used in this application to display the current zone being watered and the time remaining for the zone being watered. When the system is not actively watering a particular zone, the LCD is used to display the time of day.
See Kinetis Quick Reference User’s Guide and Kinetis example code for details on the sLCD initialization and usage.

**Segment Fault Detection**

If a defect exists in one or more of the LCD segments, the information displayed on the LCD can lead the user to make an incorrect decision based on this faulty information. The presentation of accurate information on a segment LCD is particularly critical in industrial and medical applications, where health and safety is at risk.

In most applications that employ a segment LCD, an initialization routine that executes upon system startup can perform a diagnostic check of each LCD segment to ensure that it can turn on and off as expected.

If, however, a segment fault occurs after system initialization but the type of fault still allows a valid character or digit to be displayed, the user may not be aware that the information displayed on the segment LCD may have been compromised.

Consider the following example where the center LCD segment of the second digit does not properly illuminate due to a short circuit to ground in one of the LCD backplanes. A fault in one of the LCD segments causes the actual temperature (28°C) to be displayed incorrectly (20°C).

![Figure 7. Fault on Segment LCD Display](image)

The segment LCD module that is built into the Kinetis K40 MCU boasts a feature that allows segment faults to be detected in real time.

The fault detection circuitry works by applying a pull-up voltage on the LCD pins and monitoring the capacitive characteristics on the segment matrix. An open-circuit or short-circuit condition connection changes the capacitive characteristics of the segment matrix, so any significant difference between the measured capacitance and a user-defined reference value can allow the segment LCD controller to generate an interrupt to signal a segment fault error condition.

Segment LCD fault detection is implemented in the intelligent sprinkler system design by allowing the user to enable/disable fault detection through a pushbutton on the K40 Tower Board. When fault detection is enabled, an error message will be displayed on the terminal window whenever a fault condition exists in the LCD display, display connector, or board connections between the MCU and the display.

**Data Logging with FlexMemory**
In many systems, data logging a certain set of parameters in non-volatile memory is helpful to track a system’s performance over time. The number of parameters, as well the length of the parameter value history log, is, of course, limited to the non-volatile memory’s size and write endurance.

The Kinetis family of MCUs offers integrated non-volatile memory with a feature known as FlexMemory. FlexMemory is an extremely versatile and powerful solution that enables designers to configure non-volatile memory as byte-writable/erasable EEPROM, data flash (D-Flash), or a combination of both, depending on the needs of the application.

![FlexMemory Blocks](image_url)

**Figure 8. Kinetis Flash Memory blocks**

Depending on the needs of the application, a system designer can configure FlexMemory to specify the D-Flash size, EEPROM size and space allocated for EEPROM backup. Example configurations are shown below:

**Table 1. FlexMemory Configuration options**

<table>
<thead>
<tr>
<th>FlexNVM Configuration</th>
<th>FlexRAM Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 KB data flash</td>
<td>0 KB for EEPROM backup</td>
</tr>
<tr>
<td>224 KB data flash</td>
<td>32 KB for EEPROM backup</td>
</tr>
<tr>
<td>192 KB data flash</td>
<td>64 KB for EEPROM backup</td>
</tr>
<tr>
<td>128 KB data flash</td>
<td>128 KB for EEPROM backup</td>
</tr>
<tr>
<td>64 KB data flash</td>
<td>192 KB for EEPROM backup</td>
</tr>
<tr>
<td>32 KB data flash</td>
<td>224 KB for EEPROM backup</td>
</tr>
<tr>
<td>0 KB data flash</td>
<td>256 KB for EEPROM backup</td>
</tr>
</tbody>
</table>

EEPROM endurance depends on the specified EEPROM size and the space allocated for EEPROM backup.
If data flash size is of lower importance, then allocating FlexNVM for larger EEPROM backup sizes will result in higher EEPROM write endurance.

For example, if a system designer needs the full 4 KB of EEPROM and allocates all 256 KB of FlexNVM as EEPROM backup, then expected EEPROM endurance can reach 310,000 cycles.

Or, if a system designer only needs 32 bytes of EEPROM data and allocates all 256 KB of FlexNVM as EEPROM backup, then expected EEPROM endurance can reach over 10 million cycles.

See Application Note AN4282 available from https://www.freescale.com for additional details on endurance calculations.

In this particular application, the K40 FlexMemory has been configured to use the FlexRAM as 4 KB of EEPROM and the entire FlexNVM (256 KB) as EEPROM backup.

The intelligent sprinkler system’s sensors send temperature, percent humidity, barometric pressure, active sprinkler zone, sprinkler zone watering duration and sprinkler mode data over the I2C bus to the K40 MCU so that they can be stored in EEPROM with a time and date stamp every 90 seconds:

<table>
<thead>
<tr>
<th>Date and Time Stamp</th>
<th>Temp</th>
<th>Humidity</th>
<th>Pressure</th>
<th>ActZone</th>
<th>Runtime</th>
<th>SprMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011/02/17 19:57:10</td>
<td>24.36</td>
<td>43.82</td>
<td>98.86</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>2011/02/17 19:55:40</td>
<td>24.17</td>
<td>43.13</td>
<td>98.67</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>2011/02/17 19:54:09</td>
<td>24.15</td>
<td>44.21</td>
<td>98.73</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2011/02/17 19:52:39</td>
<td>23.91</td>
<td>43.65</td>
<td>98.73</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2011/02/17 19:51:08</td>
<td>23.80</td>
<td>43.48</td>
<td>98.73</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2011/02/17 19:49:38</td>
<td>23.53</td>
<td>44.54</td>
<td>98.67</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2011/02/17 19:48:07</td>
<td>23.13</td>
<td>45.40</td>
<td>98.86</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2011/02/17 19:46:37</td>
<td>22.70</td>
<td>46.18</td>
<td>98.79</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>2011/02/17 19:45:06</td>
<td>23.31</td>
<td>45.21</td>
<td>98.62</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
</tbody>
</table>

Each entry of the data log stores several different parameters:

- Date and time when the log entry was recorded
- Temperature is recorded in degrees Celsius
- Humidity is stored as a percentage
- Pressure is stored as kPa
- “ActZone” is the sprinkler zone that is currently watering at the time the log entry is recorded. If the sprinkler system is not watering when a log entry is recorded, the active zone is 0.
- “Runtime” is the programmed watering duration (in minutes) at the time the log entry is recorded. If the sprinkler system is not watering when a log entry is recorded, the run time is 0.
- “SprMode” is the sprinkler mode at the time the log entry is recorded. If the sprinkler system is not watering when a log entry is recorded, the sprinkler mode is 0 (off). If the sprinkler system is in the process of watering all zones sequentially when a log entry is recorded, the sprinkler mode is 1 (sequence). If the sprinkler system is in the process of watering a single zone when a log entry is recorded, the sprinkler mode is 2 (single).
Each log entry is 32 bytes. Since about 3328 bytes of the EEPROM have been allocated for data log storage, this means that a history of 104 data log entries can be recorded. When 3328 bytes have been written, the log begins re-writing the oldest data first. With each data log entry being stored every 90 seconds, this means that a history from the most recent 156 minutes can be accessed within the EEPROM. The expected endurance for the EEPROM for this FlexMemory configuration is 310,000 cycles.

**Power Loss Recovery with FlexMemory**

When FlexMemory is configured as EEPROM write times are as fast as SRAM—approximately 100 µs to write to a pre-erased location. The EEPROM state machine ensures the integrity of EEPROM writes, even in the event of an unexpected system interruption to the MCU clocks or to power.

In this application, the software regularly writes to EEPROM with the state of several operating parameters. These parameters include:

- programmed sprinkler start time
- programmed watering durations for each zone
- zone currently being watered
- time remaining for a particular zone that is currently being watered

The system recovers from a power interruption by reading the system state from the EEPROM when the power is restored. The system then resumes its operation in the same state as before the power loss. If a power interruption occurs during an active watering event, the system will continue watering the active zone for the time remaining for that zone (rounded to the minute). The user need not re-program the watering schedule or restart the watering process manually.

If a power loss occurs at the exact moment that an EEPROM write is in progress and the EEPROM write did not complete successfully, the EEPROM data will revert to the previous good value. Therefore, the data will not be corrupted.

The fast EEPROM write times also make it possible for the system to quickly save its state when the low voltage detect (LVD) has detected the power dropping. However, this feature is not implemented in the sprinkler system application.

**Low Power States When Sprinkler System Is Inactive**

Every MCU in the Kinetis family offers ten power modes that allow a system designer to control MCU operation with a fine degree of granularity to balance performance and power conservation needs.

In this application, the MCU enters the low leakage stop (LLS) mode when the sprinkler mode is not in active operation. The MCU enters a low leakage mode by reducing the voltage to internal logic while internal logic states are retained. Because recovery from this power mode does not require executing through the reset flow, wakeup times are around 6 µs.

**Sensor data collection**

Temperature, humidity, and barometric pressure data is collected by the TWR-SENSOR-PAK module connected in the Tower system. The Sensor module contains a MC9S08QE96 MCU, which samples the various sensors over I2C. The Kinetis system communicates with the MC9S08QE96 over I2C to get the results of those sensor readings. The jumper settings on the TWR-Sensor can also be configured for the Kinetis to directly get the sensor data over I2C, however in this application the MC9S08QE96 is used to aggregate the data.

**GPIO**

General purpose output signals are used to control relays that switch on/off sprinkler valves that can be attached to the system. The relays are located on the custom module that is designed for this application.

### 4 Software

This section is an overview of the software. The source code is available for download as a supplement to this Application note. For additional details, refer to the setup guide located with the source code.
4.1 Software Overview

The system is configured to run at 48 MHz to conserve power. This is approximately half of the maximum frequency that Kinetis is capable of. Even when running at this frequency, the application only consumes 7% of the CPU cycles. Therefore this application represents only a small fraction of what Kinetis MCUs are capable of.

The total code size for the K60 project for the release configuration (highest compiler optimization settings) is 166 KB. For the K40 project, the code size is 48 KB.

**NOTE**

Code size will vary depending on the application. Additional optimization is possible by removing optional features.

The approximate break down of the code is as follows:

<table>
<thead>
<tr>
<th>Table 2. K60 Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images for graphical interface</td>
</tr>
<tr>
<td>Fonts for graphical interface</td>
</tr>
<tr>
<td>Board support package</td>
</tr>
<tr>
<td>Processor support package</td>
</tr>
<tr>
<td>Web pages for web interface</td>
</tr>
<tr>
<td>Sprinkler system application and other</td>
</tr>
<tr>
<td>Real Time Communication Suite (Networking Protocol)</td>
</tr>
<tr>
<td>Total Size of K60 project</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3. K40 Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board support package</td>
</tr>
<tr>
<td>Processor support package</td>
</tr>
<tr>
<td>Sprinkler system application and other</td>
</tr>
<tr>
<td>Total Size of K40 project</td>
</tr>
</tbody>
</table>

4.1.1 Overview of Tasks

The sprinkler system uses the Freescale MQX RTOS to provide task scheduling and peripheral drivers. The project is broken into several tasks. The MQX task list is located in Sprinkler_Init.c.

The tasks are:

- INIT_TASK: Runs at boot-up and kick-starts the other tasks
- SPR_AUTO_TASK: Controls when the sprinkler will being watering on a schedule
- SPR_TASK: Controls the actual sprinkler system, turning on and off the relays for the set amount of time
- SENSOR_TASK: Reads the sensor data. FlexMemory data logging is also done by this task (K40 Version).
- SHELL_TASK: Implements the command-line interface
- D4D_TASK: eGUI task to control the LCD (K60 Version)
• SLCD_TASK: Controls the segment LCD display (K40 Version)
• PWR_RECOV_LOG_TASK: Used to recover the log data from FlexMemory in the event of a power loss. (K40 Version)

4.1.2 Summary of each task

INIT_TASK (Sprinkler_Init.c)

The init task is the only auto-start task in the system. Therefore, it is the only task that starts when the system boots up. The init task creates and initializes system variables and starts the other tasks of the system. The init task starts SENSOR_TASK, SLCD_TASK, SPR_TASK, SPR_AUTO_TASK. The init task also calls the function to initialize the networking protocol (RTCS).

SENSOR_TASK (Sensor.c)

The sensor task uses the MQX I2C driver to read the current status of the temperature, humidity, and pressure sensors from the TWR-SENSOR-PAK module. It sets up the I2C driver and then polls the MC9S08QE96 on the TWR-SENSOR-PAK module to get the current sensor status.

In the K40 System, these values, as well as current date and time, active sprinkler mode and sprinkler run time, are stored in FlexMemory at an interval specified in LOG_FREQUENCY_MILLSEC to serve as a data log. This task is located in Sensors.c, which can be found in the Sensors folder.

SLCD_TASK (sLCD_Task.c)

The SLCD task drives the segment LCD Tower Plug-In module (TWRPI). After initializing the segment LCD screen, the SLCD task displays the current time of day in its default mode. If a sprinkler zone is active, then the active sprinkler zone being watered is displayed along with the time remaining for the zone that is being watered.

SPR_TASK (Sprinkler_tasks.c)

The sprinkler tasks controls the running of the sprinkler zones. The task waits for an event signal to be received which tells it to start turning on sprinkler valves. The event can be triggered by the SPR_AUTO_TASK, D4D_TASK (via the graphical interface screen), and the web server (via cgi functions).

SPR_AUTO_TASK (Sprinkler_tasks.c)

The sprinkler auto task periodically checks to see if it is time to start the sprinkler schedule. It compares the desired start time with the current time. Once those times match up, it begins running all the zones for the times pre-configured by the user.

PWR_RECOV_LOG_TASK (Sensor.c)

The sprinkler power recovery log task periodically saves the current active sprinkler zone, sprinkler mode and remaining watering time to FlexMemory so that, in the event of a power loss, these parameters can be recovered when power is restored and allow the sprinkler system watering activity to resume where it left off when power was lost. This task is located in Sensors.c which can be found in the Sensors folder.

D4D_TASK (Sprinkler_D4D_task.c)

The D4D task drives the TWR-LCD screen. The D4D task calibrates the touch screen, enables periodic timers to refresh the values on the LCD screen, and then periodically polls the eGUI software for changes.

You can see the files that create the LCD screen in the D4D_Application folder.

images.c contains the byte arrays of the images used by the project, that are created by the eGUI Image Convertor. There are files (screen_xxxxx.c) for each of the four screens used by the sprinkler system. In each file, label declarations are for the text on the screen:

D4D_DECLARE_STD_LABEL(lbl_zone, str_zone, 80, 80, 5, 5, FONT_ARIAL16_BOLD)

the picture and button declarations for the graphics:

D4D_DECLARE_STD_PICTURE(picBanner, 0, 0, &bmpKinetisBanner)
D4D_DECLARE_BUTTON(btn_web, NULL, 210, 180, 0, 0, BTN_FLAGS, &bmpWebButton, &bmpWebButton, NULL, 0, NULL, OnClick_BtnWeb, NULL)

as well as the screen declaration which has all the objects displayed on the screen listed:

D4D_DECLARE_STD_SCREEN_BEGIN(screen_main, ScreenMain_

The screen is refreshed every 100 ms to detect touches that occur. There are also MQX periodic timers that are enabled to refresh certain items at slower intervals to avoid flickering affects.

Shell task (Sprinkler_Shell_Commands.c)

The Shell task implements the command line interface to the sprinkler system. The list of commands and the code associated with each command are located in Sprinkler_Shell_Commands.c which can be found in the Shell folder.

HTTP web Server Task

When the HTTP web Server is initialized and started, RTCS creates a task to manage the web server. The networking initialization functions are located in Sprinkler_network.c, which can be found in the web folder.

TSI (tsi.c)

The Touch Sense Interface is initialized by INIT_TASK. The TSI continuously scans the capacitance on the touch sense electrodes. At the end of each scan, an interrupt routine is launched which evaluates the TSI scan. When touch events are detected in the interrupt routine, the code in the service routine performs the desired actions on the system, such as turning a led on. Therefore, the TSI does not require its own task to process touch sense information. Code for the TSI is located in tsi.c which can be found in the TSI folder.

4.1.3 Software flow

After boot up and MQX initialization, the INIT_TASK executes first because it is set up as the only auto-start task. The INIT_TASK enables many of the other tasks of the system. These tasks are then able to execute based on their priority. A task will run for a certain amount of time and then give up control, allowing other tasks to run. The mechanism that is typically used to give up control to other tasks is the _time_delay() function. For example, the SENSOR_TASK will run in an endless loop while it collects data. At the end of each loop it will execute _time_delay(500), which will give up control to other tasks for 500 ms until it is time to collect more sensor data.

4.1.4 Task Priorities

The tasks of the system are configured to be priority based preemptive (non time slice). Therefore, the task with the highest priority that is ready to run will preempt other lower priority tasks of the system. Each task generally performs some operations and then gives up control allowing other tasks to run. If no tasks require running the system idle task will run and wait for another task to be ready. Each task has a unique priority which represents its importance in the system. Note that higher priority tasks have a lower priority number and vice versa. The HTTP web Server task that is created by RTCS has a higher priority than all other tasks. The task priorities are pre-configured in the MQX task list located in Sprinkler_Init.c.

4.1.5 Communication between tasks

There are several ways that tasks can communicate and pass data between each other within MQX. The requirements of the intelligent sprinkler system are relatively simple, so only a few simple techniques are used. Global variables without semaphore protection are used for non-critical data that is updated by one task and consumed by another task. If the data is not interdependent and will not be corrupted if a task is preempted while reading or writing it, then this method is acceptable. Also, light weight events are used to trigger actions from one task to another. For instance, either the graphical interface or web interface tasks can signal to the sprinkler task with a light weight event to tell it to run the sprinkler zones.
5 References

Following are the search keywords that can be used to quickly find the documents at https://www.freescale.com.

5.1 Intelligent Sprinkler System additional documents

Following additional documents are available with the source code that accompanies this application note:

Sprinkler Demo Setup Guide: Hardware and software setup instructions including board modifications
K40 Sprinkler Demo Users Guide: Detailed information about K40 system operation
K60 Sprinkler Demo Users Guide: Detailed information about K60 system operation

5.2 Custom Relay Module

900-76126_TWR-MECHDRW: Tower Mechanical Drawing

5.3 Kinetis ARM Cortex-M4 Microcontrollers

Kinetis resources are available at https://www.freescale.com

KQRUG: Kinetis Peripheral Module Quick Reference
KINETIS512_SC: Bare-metal example code projects for Kinetis
AN4282: Using the Kinetis Family Enhanced EEPROM Functional
K60P144M100SF2RM: K60 Sub-Family Reference Manual for 100 MHz devices in 144 pin packages
K60P144M100SF2: K60 Sub-Family Data Sheet for 100 MHz devices in 144 pin packages
K40P144M100SF2RM: K40 Sub-Family Reference Manual for 100 MHz devices in 144 pin packages
K40P144M100SF2: K40 Sub-Family Data Sheet for 100 MHz devices in 144 pin packages

5.4 Freescale MQX™ Software Solutions

MQX™ resources are available at https://www.freescale.com/mqx

MQXUG: Freescale MQX™ User Guide
AN3905: Writing First MQX™ Application
MQXIOUG: Freescale MQX™ I/O Drivers User Guide
MQXRTCSUG: Freescale MQX™ RTCS User Guide
5.5 Freescale eGUI: Graphical LCD Driver for MCUs and MPUs

Freescale eGUI resources are available at https://www.freescale.com

DRM116: Freescale Embedded GUI (D4D) Reference Manual

EGUIPRE: Freescale eGUI Introduction Presentation

AN4263: Driver for char Displays

5.6 Freescale Sensors

Information on Freescale sensors is available at https://www.freescale.com/sensors

5.7 Freescale Tower System

Tower Systems resources are available at https://www.freescale.com.

Note: All tower modules described in this application note are available for purchase from Freescale distributors or directly from Freescale at https://www.freescale.com/tower with the exception of the custom relay module.

5.7.1 TWR-K60N512: Kinetis K60 Module


TWRK60QSG: TWR-K60N512-KIT Quick Start Guide

TWRK60N512QSDLAB: TWR-K60N512 Quick Start Demo Software and Lab Guide

5.7.2 TWR-K40X256: Kinetis K40 Module


TWRK40QSG: TWR-K40X256-KIT Quick Start Guide

TWRK40X256QSDLAB: TWR-K40X256 Quick Start Demo Software and Lab Guide

5.7.3 TWR-SENSOR-PAK: Sensor Module

TWRSNSRPKUM_C: TWR-SENSOR-PAK User Manual Rev C

TWRSNSRPKQSG: Quick Start Guide for TWR-SENSOR-PAK

5.7.4 TWR-LCD: Graphical LCD Module

TWRLCDUM: TWR-LCD User Manual

TWRLCDLAB: TWR-LCD Lab tutorials

TWRLCDQSG: TWR-LCD Quick Start Guide
5.7.5  TWR-WIFI-RS2101: 802.11n Wi-Fi module featuring Redpine Signals

TWRWFIRS2101_QSG: Quick Start Guide for TWR-WIFI-RS2101
TWRWFIRS2101LAB: TWR-WIFI-RS2101 Lab Guide

5.7.6  TWR-SER: Serial Module

TWRSERUM: Tower System Serial Module - User Manual