Smart Appliance Design

Reference Manual

Document Number: DRM129
Rev. 0, 04/2013

Contents

Section number **Title** **Page**

Chapter 1
Introduction

Chapter 2
System Overview

Chapter 3
User Interface (UI)

Chapter 4
Led Driver Board

Chapter 5
Control Board

Chapter 6
Modbus Protocol

Chapter 7
Home Automation Gateway

Chapter 8
Home Automation User's Guide

Chapter 1
Introduction

1.1 Introduction

The refrigerator is a necessary appliance in urban households, its marketing story is accompanied by continuous improvement and innovation in design and features that adapt to the trends of modern life, and today one of the most recent is developed in Freescale.

Freescale continues with the development of advanced technologies that allow greater productivity, optimization, and higher-performance converged products that improve customers lives.

This design reference manual describes the development of a Smart Appliance System using Freescale Microcontroller Solutions.

1.2 Smart appliance system advantages

- A system that uses Freescale technology to control and increment the features of a common house appliance as a refrigerator.
- Energy and power cost saving due to the use of advanced motor control algorithms, compressor motor type, energy saving functions related to open and close times (vacation mode), and communication to a home automation hub that decides the best times of the day to turn the refrigerator on at full power. This is based on a smart energy system that provides information from an energy provider.
- Provides security for the user to know the status of the refrigerator through internet.

1.3 Freescale solutions

The main concept behind the smart appliance system is to highlight the following features:

- Energy efficient appliance—Refrigerator with overall lower power consumption

- User interface (UI)—Friendly operation for the user
- Home automation—Demonstrates capabilities to set energy usage profiles, this depends on grid information from a smart hub or from manual user input.
- Smart grid—Allows for an appliance to receive information about energy costs and peak power consumption to distribute loads in electrical service.
- Smart home gateway—Serves as an information path from systems at home to the external network.
- Advanced motor control in compressor—Compressor uses a low-voltage BLDC motor, instead of a common-use universal motor. The controller is able to regulate compressor speed for faster freezing times or use slower speeds to simply maintain a constant temperature. BLDC motors are also less acoustically noisy than universal (brushed) motors. Controlled motors are also known to consume less energy than uncontrolled motors because torque is optimized.
- Internal light—Control color and light intensity of the internal refrigerator giving users a sense of aesthetic customization.
- Harness reduction—Wire reductions in the system using the Modbus communication protocol.

1.4 How a refrigerator works

A refrigerator is an appliance that uses a compression system. It uses fluid gas that changes from gaseous to liquid state and vice versa. When it evaporates, it cools the interior. This enables the preservation of food for a longer period of time. The main components for a refrigerator are

- Evaporator—The refrigerant liquid flows through the pipes inside the freezer and turns into steam.
- Condenser—The gas condenses and as it changes form, it releases heat into the environment.
- Compressor—It circulates the refrigerant liquid. A belt connects it to the motor.
- Electric Motor—This drives the compressor

1.5 Smart appliance sections

The following block diagram illustrates each section of the system:

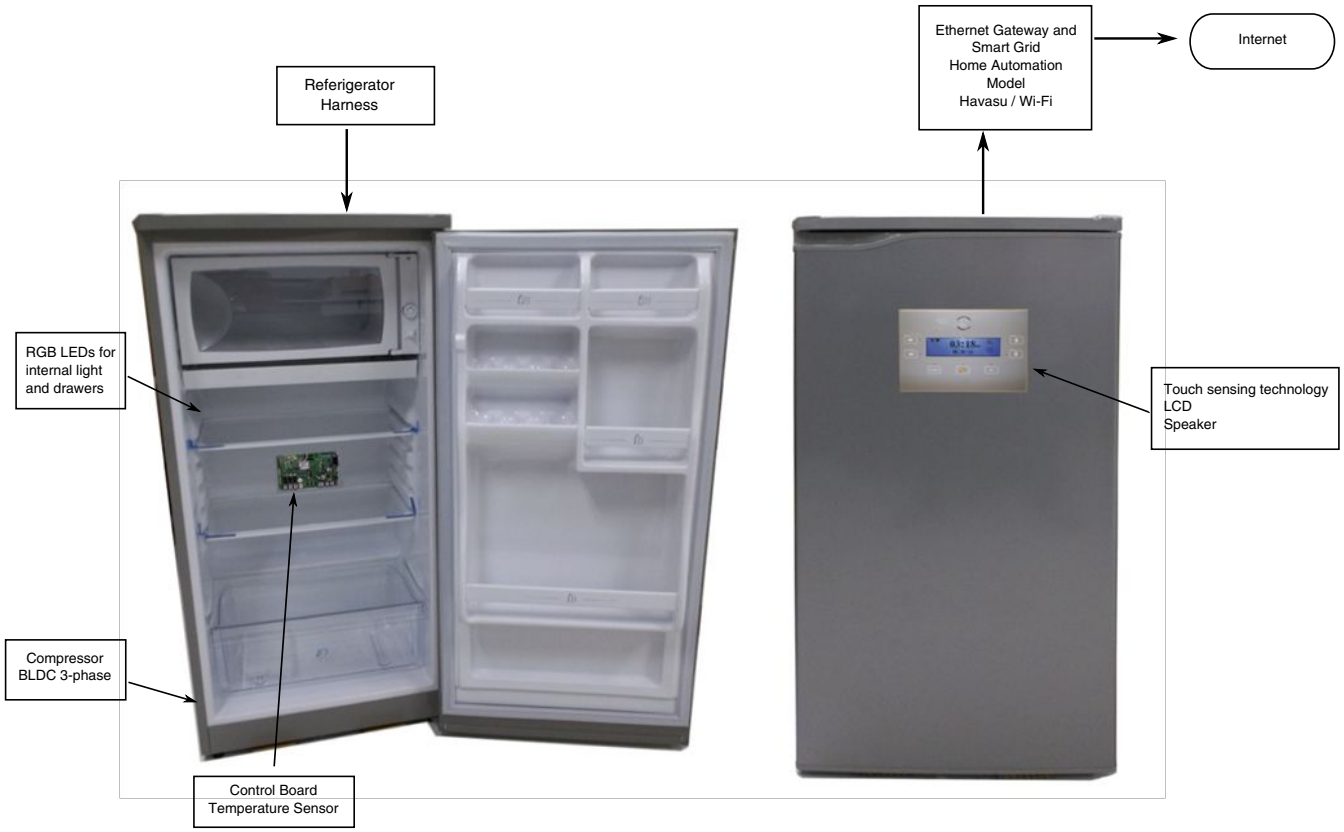

Figure 1-1. Smart appliance sections block diagram

Chapter 2
System Overview

2.1 Description

A smart appliance project can receive information from the environment looking for the most adequate solution to improve comfort for the user and efficiency in refrigerator operation.

The smart appliance system is connected to hardware components with different qualities that make up a single system. This design reference manual is dedicated to explain the solution firmware of each block and the hardware designed to be used at home, taking into consideration environmental and electromechanical specifications among others.

The system is described in several stages, this includes functions such as;

- Human-machine interface (HMI) with an LCD screen (graphical user interface) and touch sensing buttons for operation
- Compressor control
- Temperature control
- Modbus to communicate internal interfaces (control board, user interface, and LED control board)
- Wireless ZigBee communications to communicate with smart energy or home automation devices. Both user interface and control board are designed to support a Freescale ZigBee radio. The user interface is currently the subsystem programmed to manage wireless comms.

2.2 Smart appliance interfaces

2.2.1 Low-end user interface

The low-end user interface implements basic user interface features to configure appliance, check status, and turn on and off features.

Main characteristics for this block are:

- Touch-sensing implementation for buttons
- Custom segment LCD for the user interface
- Custom communications (based on Modbus) to the control board and drive LEDs
- Optimized for low power
- Uses MC9S08LH64 for Touch-sensing, LCD user interface, Modbus communication, and ZegBee.

Figure 2-1. User interface low-end

Figure 2-1 shows a low-end interface example.

2.2.2 Mid-end user interface

The mid-end user interface implements more features to configure an appliance, check status, and turn on and off features using simple-to-use touch sensing technology in a graphic menu. Figure 2-2 shows a mid-end example.

Main characteristics for this block are:

- Touch-sensing implementation for buttons
- Opaque capacitive film for touch sensing technology capability using TSS
- Dot-Matrix LCD screen for the user interface (Displaytech, S64240C – 240 x 64 dots, monochromatic)
- Wireless communications to the control board (harness replacement)
- Record/play voice messages
- Uses the MC9S08LH64 for touch-sensing, LCD, SPI, SD Card, wireless communication, and Modbus communication
- ZigBee transceiver to home automation gateway

Figure 2-2. User interface mid-end

2.3 Smart appliance system block diagram

The following block diagram illustrates a Freescale solution:

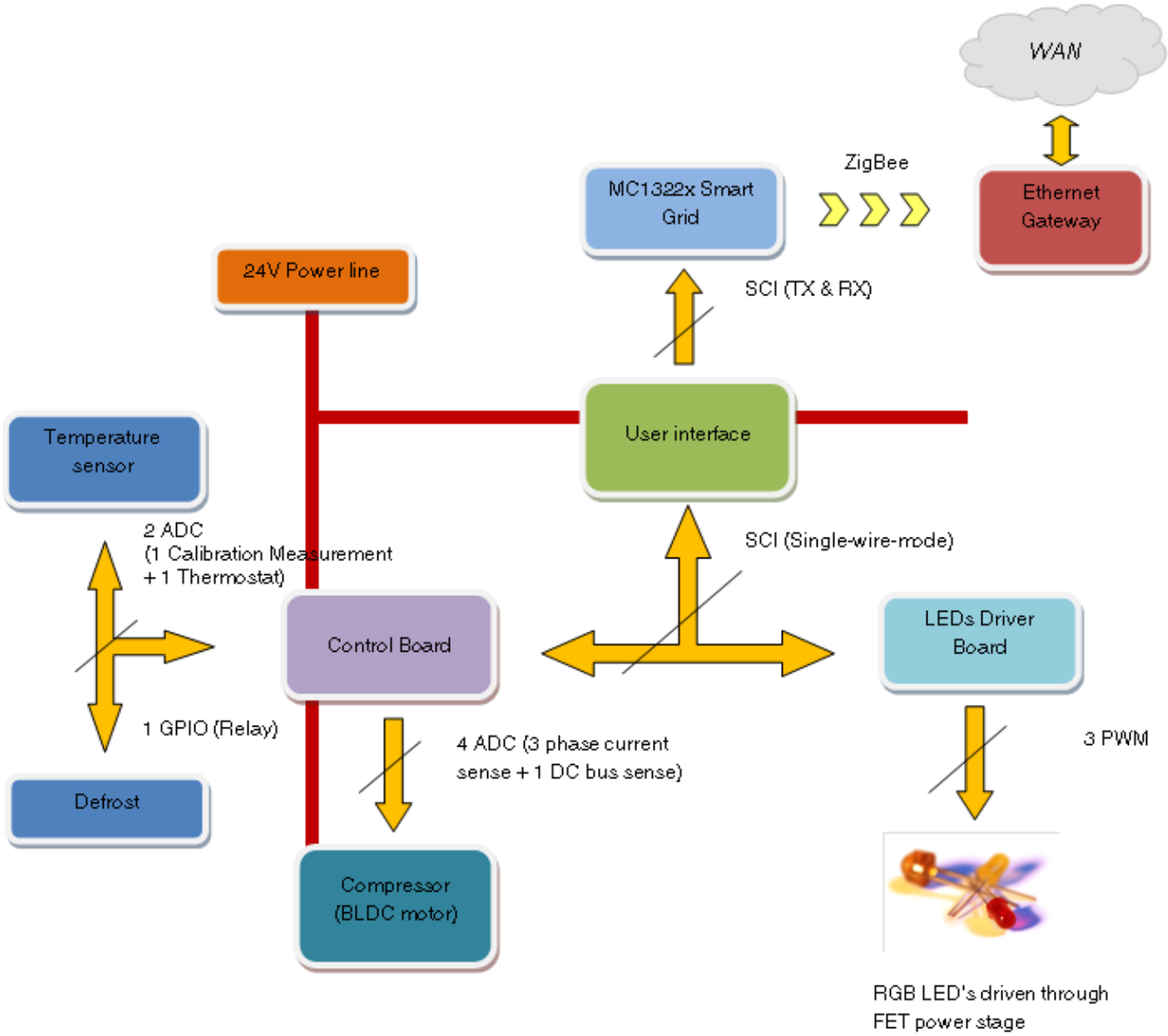

Figure 2-3. Smart appliance system block diagram

Chapter 3
User Interface (UI)

3.1 Features

The user interface (UI) is everything designed into a device where a human can interact. This includes: display, keyboard or keypad, auditive feedback (buzzer or speakers), lighting, levers, cameras, help messages, and so on.

The smart appliance system presents the UI as the core system for the refrigerator with features such as:

- Core main processor MC9S08LH64 compatible peripheral module
  - External Oscillator 32.768 Khz
  - Bus frequency 20 Mhz
  - Integrated LCD driver
- 802.15.4 Communications to the mid-end user interface that receives configuration commands and serves as a bridge to home gateway.
- Modbus One-Wire-SCI to interconnect the system
- Audio driver
- Application software including the TSS (Touch-Sensing Software library)
- LCD (Liquid Crystal Display)
- BDM connector for programming and debugging
- Capacitive film for electrode key connectors
- ITO glass connector
- Power supply + 24 V
- Board optimized to low power

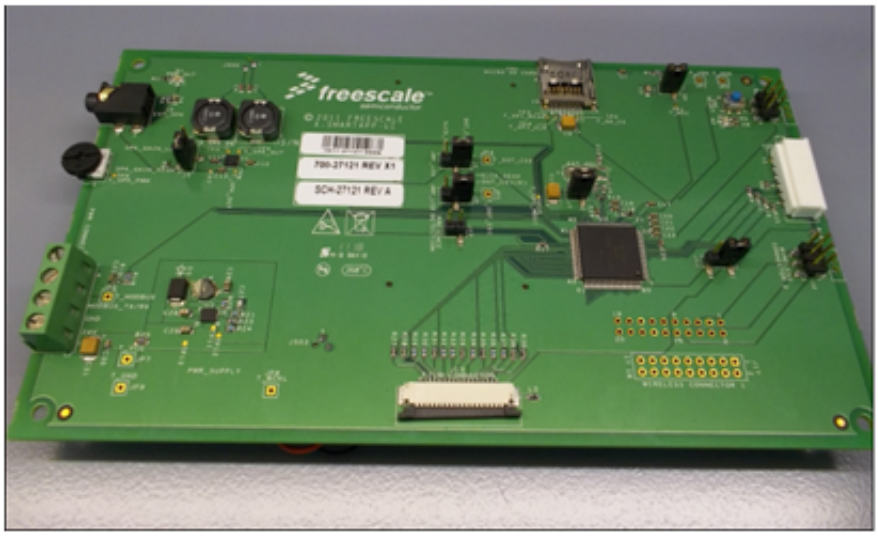

Figure 3-1. User interface board (bottom)

3.2 Block diagram

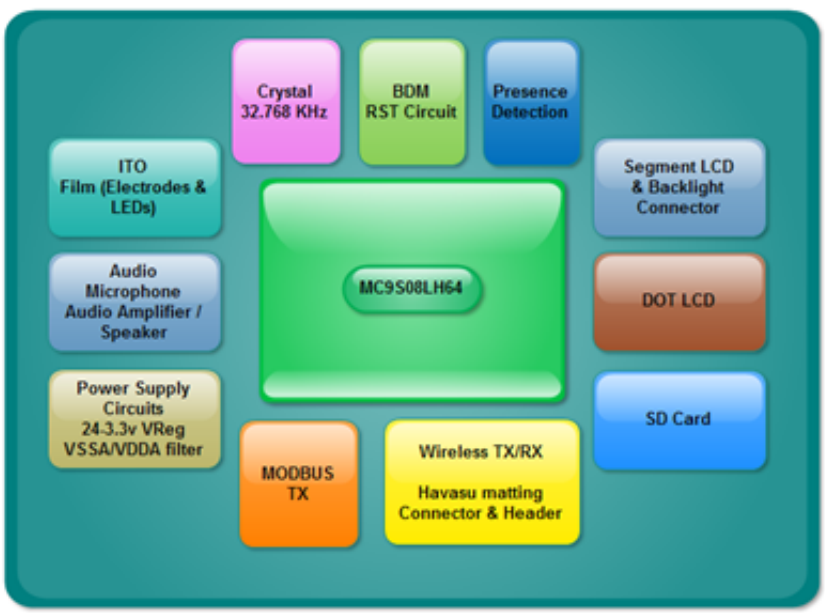

Figure 3-2. User interface block diagram

3.3 Placement (UI)

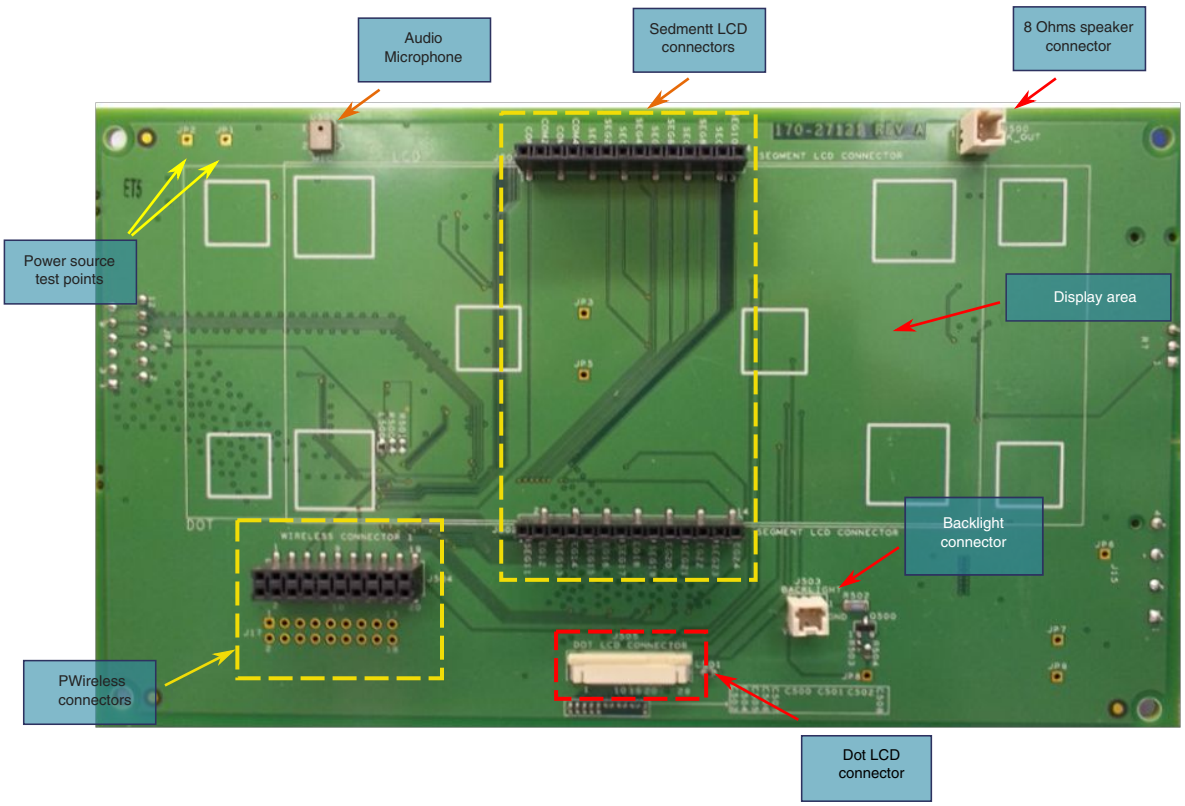

Figure 3-3. User interface placement top side

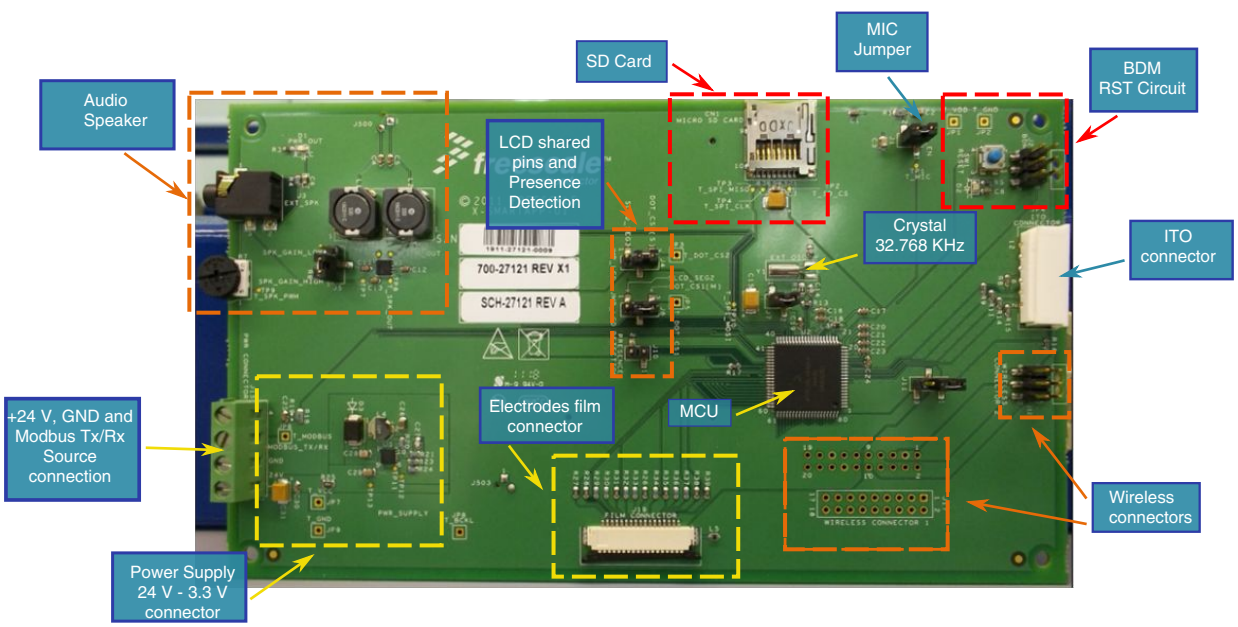

Figure 3-4. User interface placement bottom side

3.4 Hardware description

Both user interfaces are implemented on the same board. This type of design allows a cost reduction opportunity by being able to buy more of the same components for different product versions.

3.4.1 Clock source

An external crystal of 32.768 KHz on EXTAL/XTAL is necessary to generate a BUSCLK frequency to 20 MHz. The figure below demonstrates the connections.

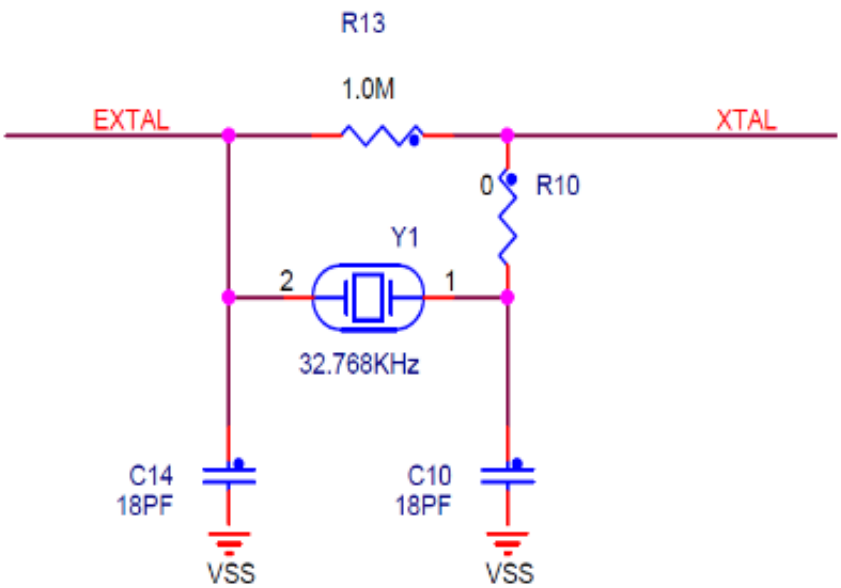

Figure 3-5. Crystal connection

3.4.2 Displays

The user interface board uses a segment LCD and a Dot LCD in low-end and mid-end versions that are configurable through jumpers.

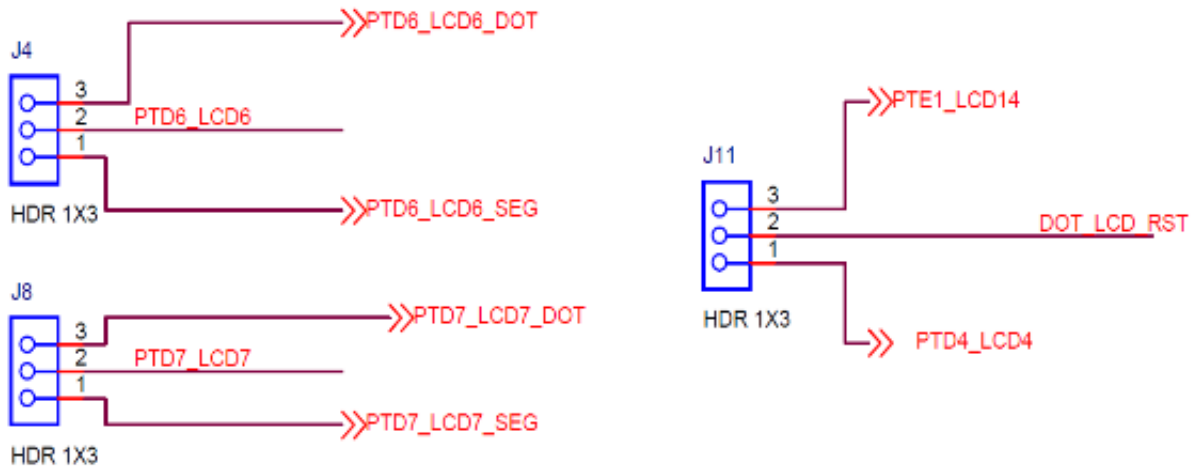

Figure 3-6. Jumpers shared for low-end and mid-end versions

3.4.2.1 Segment LCD

The custom display is used in the low-end version that show parameters to modify on the refrigerator such as clock, temperature level, turbo and vacation modes, alarm, wireless, and control LEDs using the liquid crystal display driver of the MC9S08LH64. (See datasheet in Appendix B).

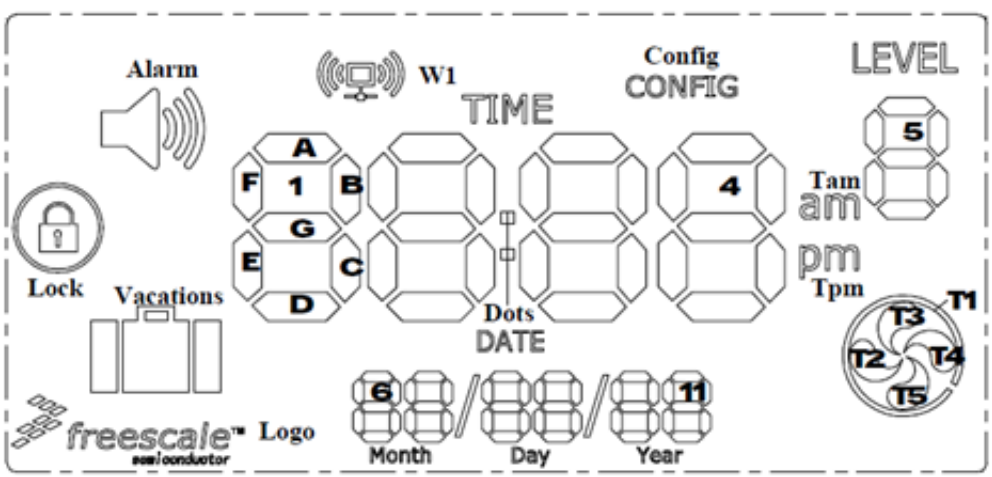

Figure 3-7. Custom LCD (low-end versions)

3.4.2.2 Dot LCD

The Dot LCD controls the same parameters as the LCD low-end version does. It has improvements such as recording voice, display menus and images, and making the function more comfortable for the user. The Dot LCD is controlled by SPI communication. The pin assignments in the Dot LCD are shared with the SD card module, Segments LCD, ITO connector, and film connector.

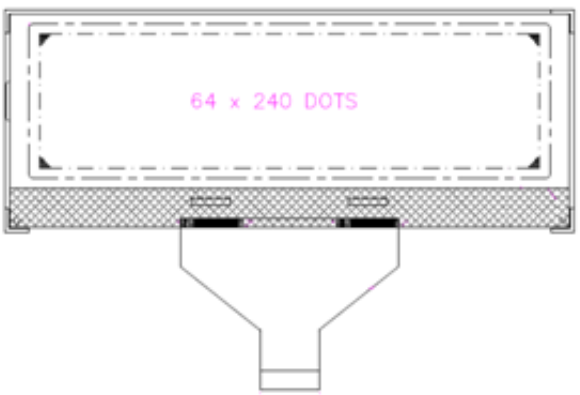

Figure 3-8. Dot LCD S64240C (mid-end version)

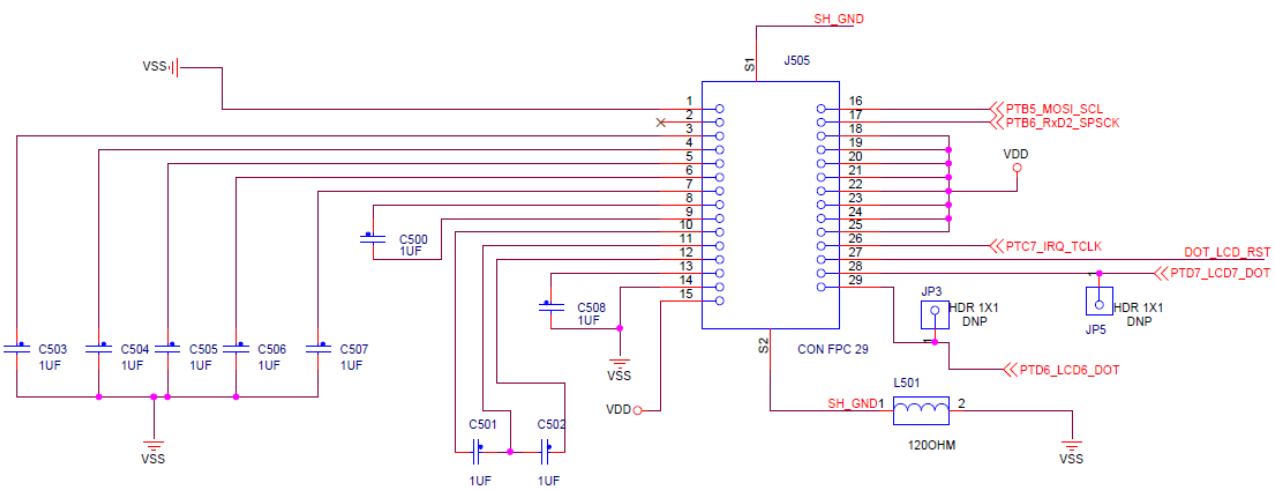

Figure 3-9. Dot LCD connection diagram

3.4.3 Micro SD card

The Micro SD card is module designed to save information routed by the MCU for mid-end versions only. When the user is in the record voice menu, the SD memory is in operation and avoids the use of the MCU memory. The way this module communicates is through SPI. The connections diagram is shown below.

NOTE

A maximum of 2 GB memory can be used for an SD card.

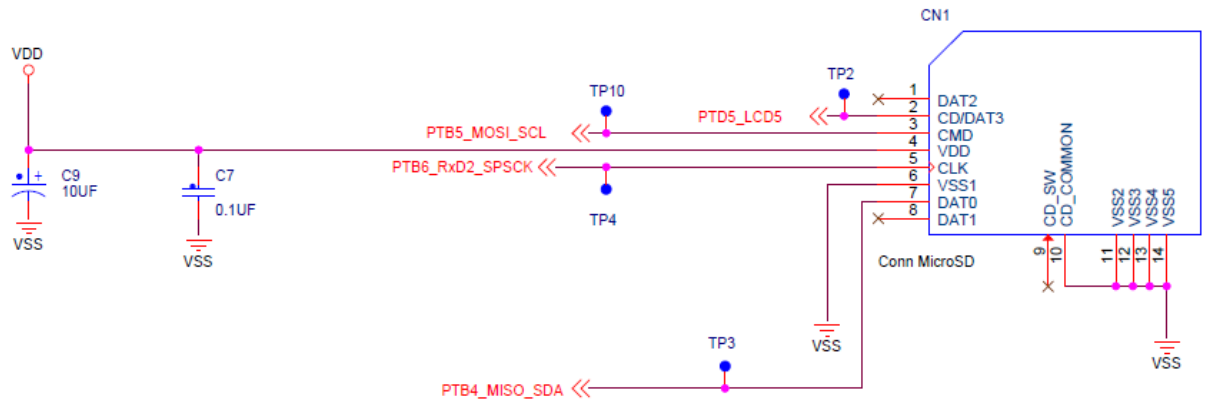

Figure 3-10. Micro SD connection

3.4.4 Audio

The smart appliance user interface has an audio block that saves and reproduces voice messages and also the alarm sound.

3.4.4.1 Microphone (MIC)

The following hardware has a microphone with enhanced RF protection set to a gain of 20 db. The circuit can be enabled or disabled by the jumper J1.

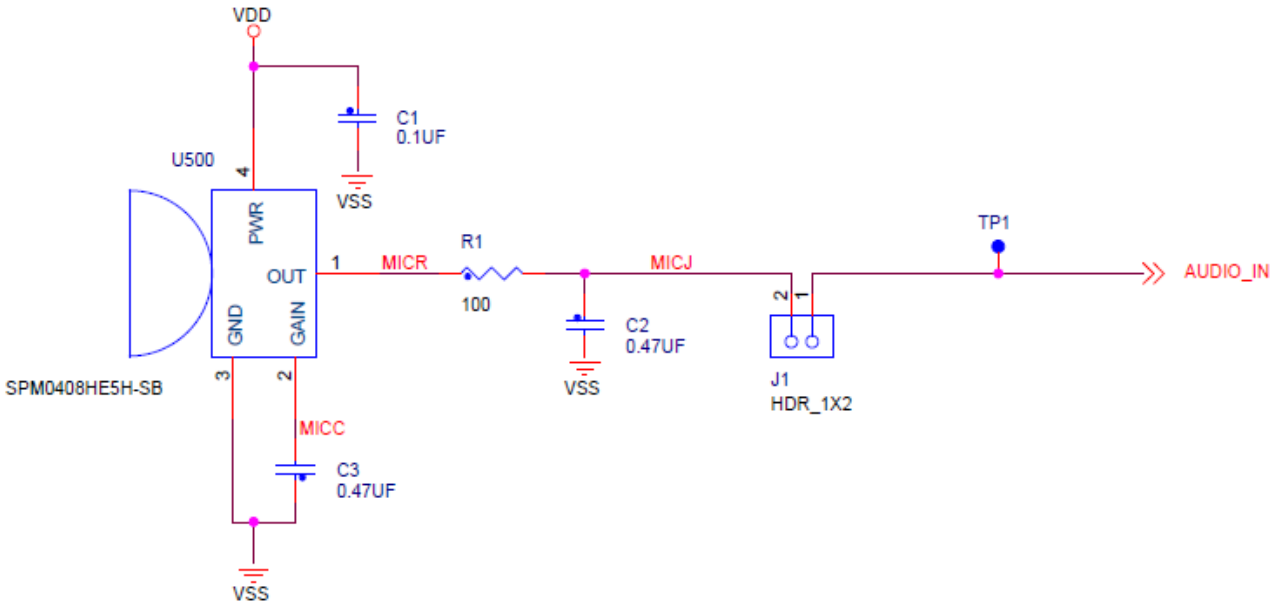

Figure 3-11. MIC Connection

3.4.4.2 Speaker

The following hardware configuration implements an amplification output system with fixed gain class D, modulated by a PWM, and combines a speaker with a jack connection for common headphones. This way, if the headphone jack is active, then the speaker is disabled. On the other hand, if the headphone jack is not activated, then the speaker output is active.

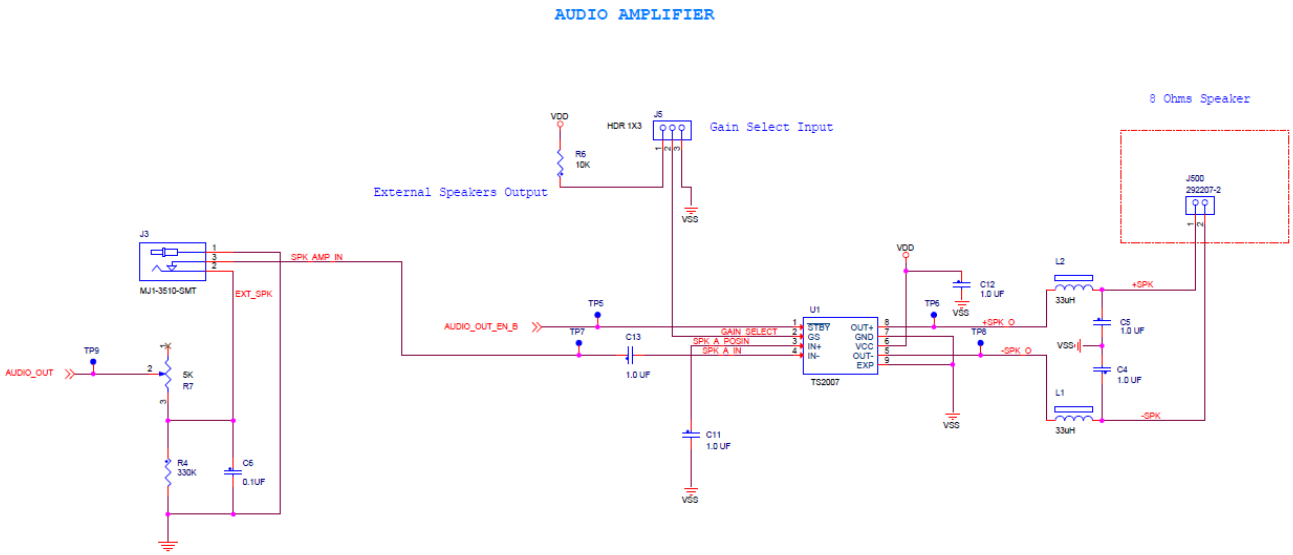

Figure 3-12. Speaker connection

3.4.5 Electrode keys

The interface to the user is controlled by electrodes through a capacitive film and an ITO, see shared pins in Figure 3-6. To enhance the functions of the electrodes the distances among the MCU, the capacitive membrane, and ITO are minimal . Each electrode has a pull-up resistor of 1 M Ω..

NOTE

Electrode keys hardware does not support both functions.

The capacitive membrane includes:

- Seven electrode keys
- Seven LEDs

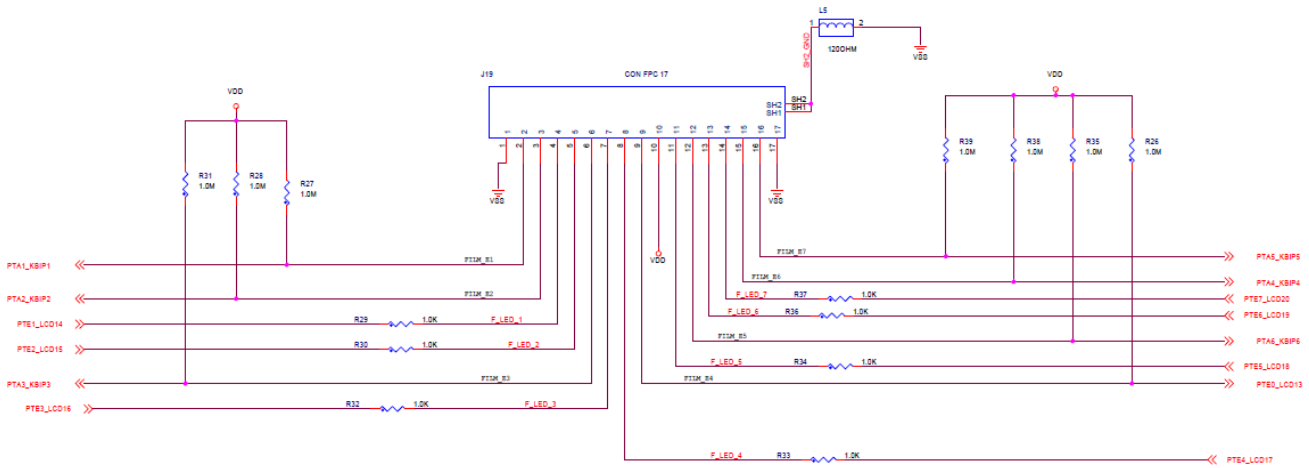

Figure 3-13. Used pin MC9S08LH64 to film connector

The ITO includes 12 electrodes and six of them are shared with segment LCD.

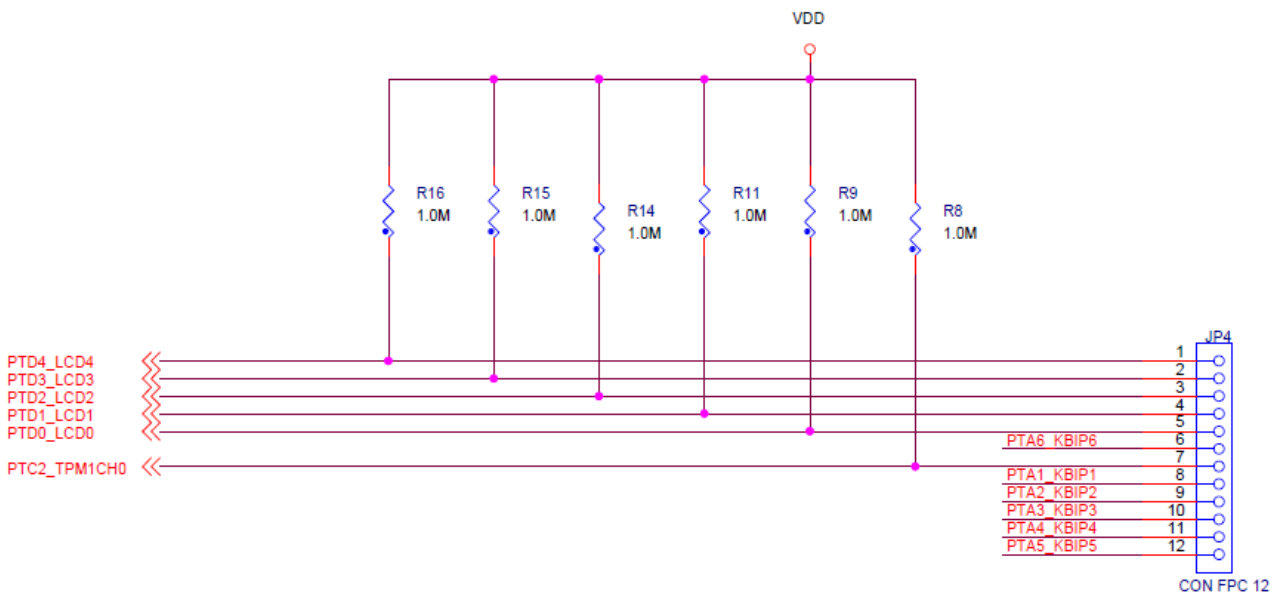

Figure 3-14. Used pin MC9S08LH64 to ITO connector

3.4.6 Presence detection

The purpose of this feature is to detect when the user is near a smart appliance. The code is not included in this reference design application; however, the hardware is ready to be used in future applications or customer request.

The main idea for this circuit is to use an electrode with a big size due to a higher capacitance. This is controlled with the TSS library as presence detection.

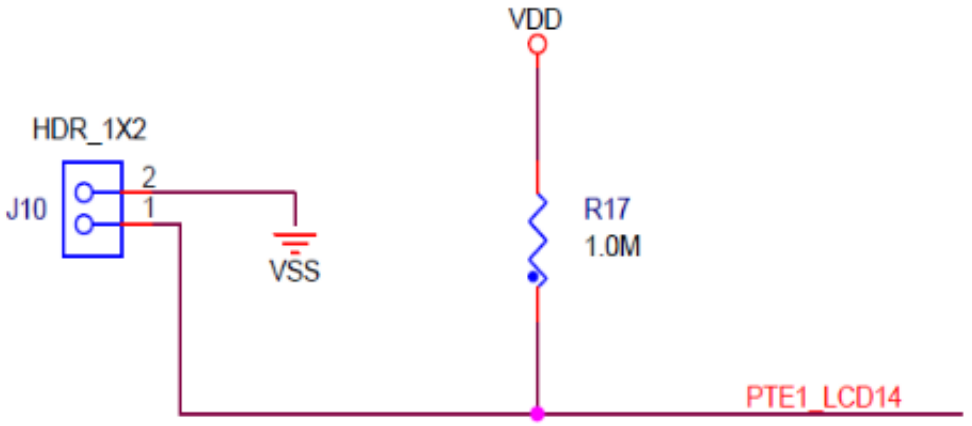

Figure 3-15. Presence detection circuit

3.4.7 Power source

The UI receives energy from of a power source that provides 24 V. The components on the board are powered with a +3.3 V regulator.

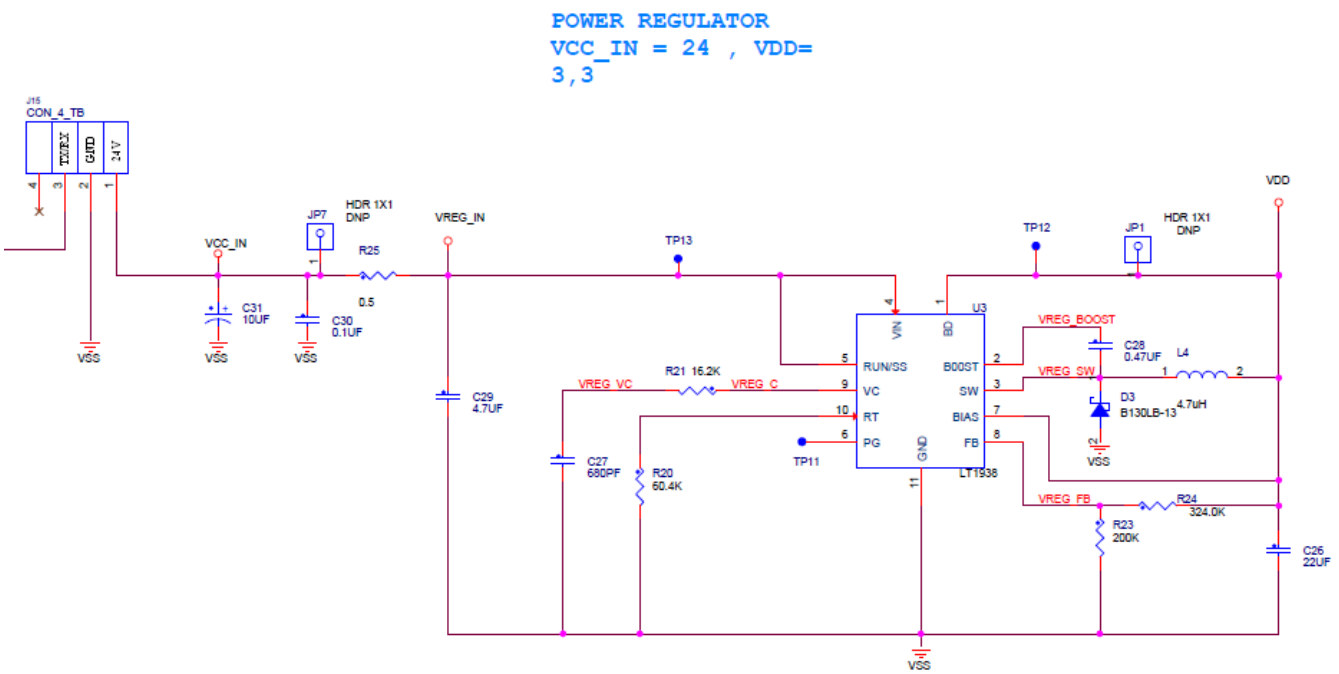

Figure 3-16. User interface power source

3.4.8 Modbus

The user interface design uses UART which provides a method of communication among several devices.

A single-wire-mode for Modbus communication protocol is used to interconnect the system.

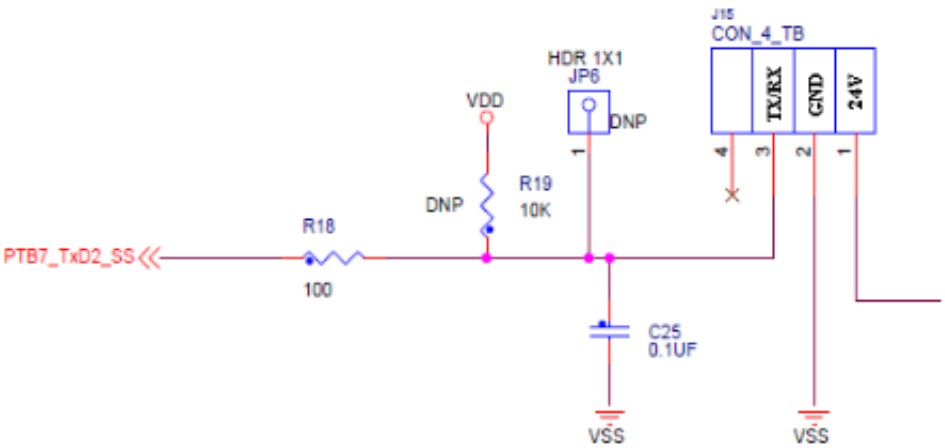

Figure 3-17. Modbus circuit diagram

The HA gateway end device communication protocol is based on a master or slave using SCI TX and RX for the user interface.

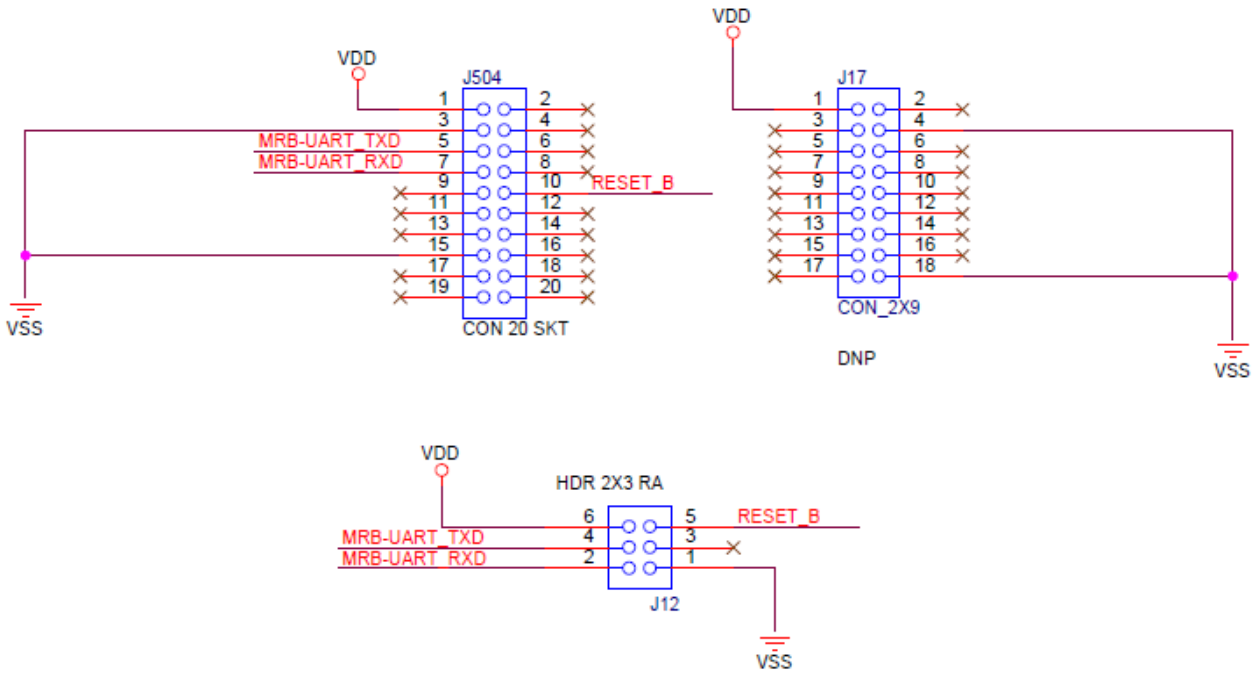

Figure 3-18. 1323x-MRB wireless connectors (user interface)

3.4.9 Debugging interface

The BDM connection is used on this board for easier debugging. Table 3-1 shows the pin assignment and shows the connection diagram.

Table 3-1. Used pin MC9S08LH64 to the BDM connector

Pin # S08LH64	Signal Name
34	PTB2/RESET
45	PTC6/ACMPO/BKGD/MS

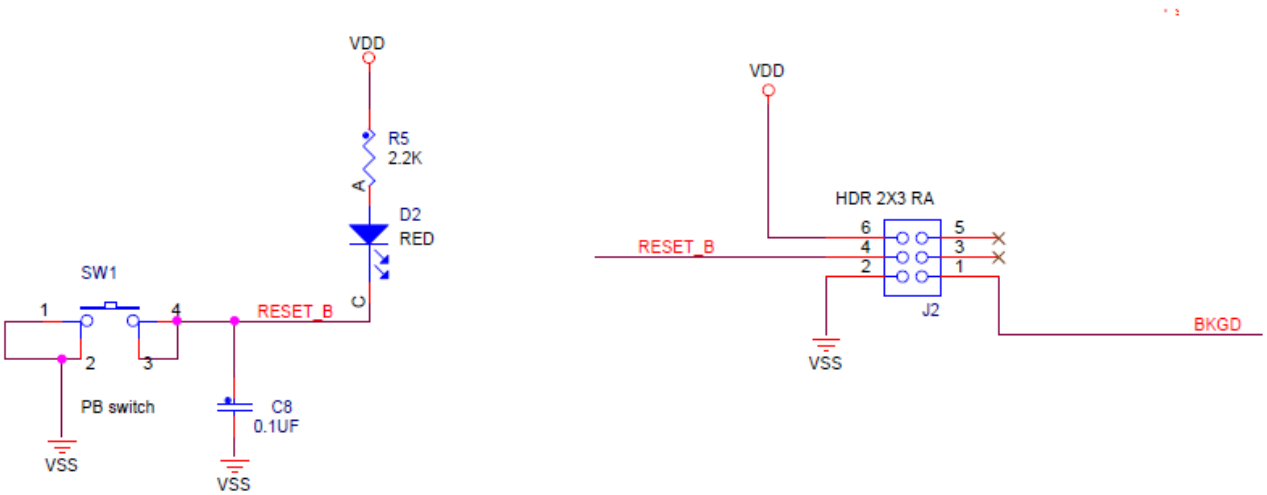

Figure 3-19. BDM and RST connection

3.5 Firmware User Interface

3.5.1 Overview

The user interface software is designed to provide a bridge between machine and people to allow control of all system functions.

The software modules that interact with the MCU peripherals, Hardware Abstraction Layer (HAL) are independent to the modules that process the information Hardware Independent Layer (HIL).

The use of this architecture reduces the dependency among all the blocks and helps improve portability of the software drivers to different applications.

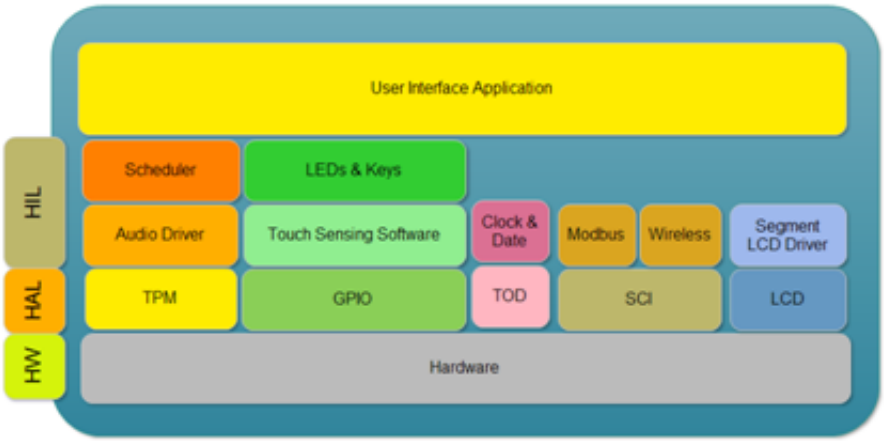

Figure 3-20. UI software architecture low-end

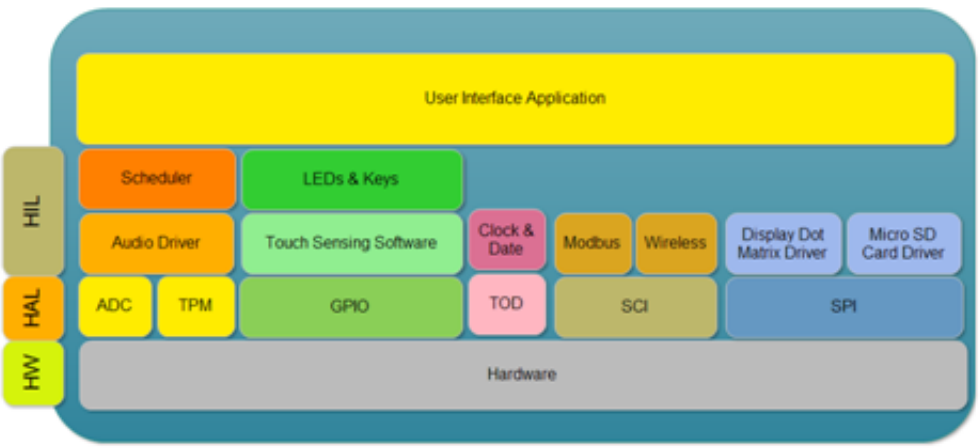

Figure 3-21. UI software architecture mid-end

3.5.2 Flowchart

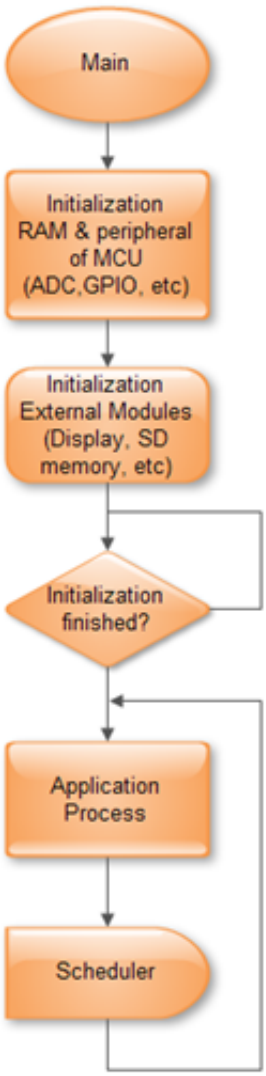

Figure 3-22. High-level UI process flow diagram

Figure 3-22 is a general flowchart for initialization of peripherals, registers, and the variables used in the code. They control the necessary hardware to comply with the specifications of the user interface process.

The software is designed with a scheduler that selects a process whenever the system or the user requests a new task at a given time, which optimizes the processing to obtain better control of the system.

3.5.3 Scheduler

The objective of the scheduler is to minimize both the wait time and response time. Scheduling is essential to achieve multi-programming. A multi-programmed system has many processes that require a resource from the MCU at a certain time. This happens when the processes are in a Ready state. If there is a processor available, then a process should be selected to be executed later. This process is selected by the code and is called a scheduler.

To implement this algorithm, the process is maintained as a queue FIFO and is controlled by interruptions or via software to generate a base time such as:

- TOD
- TPM

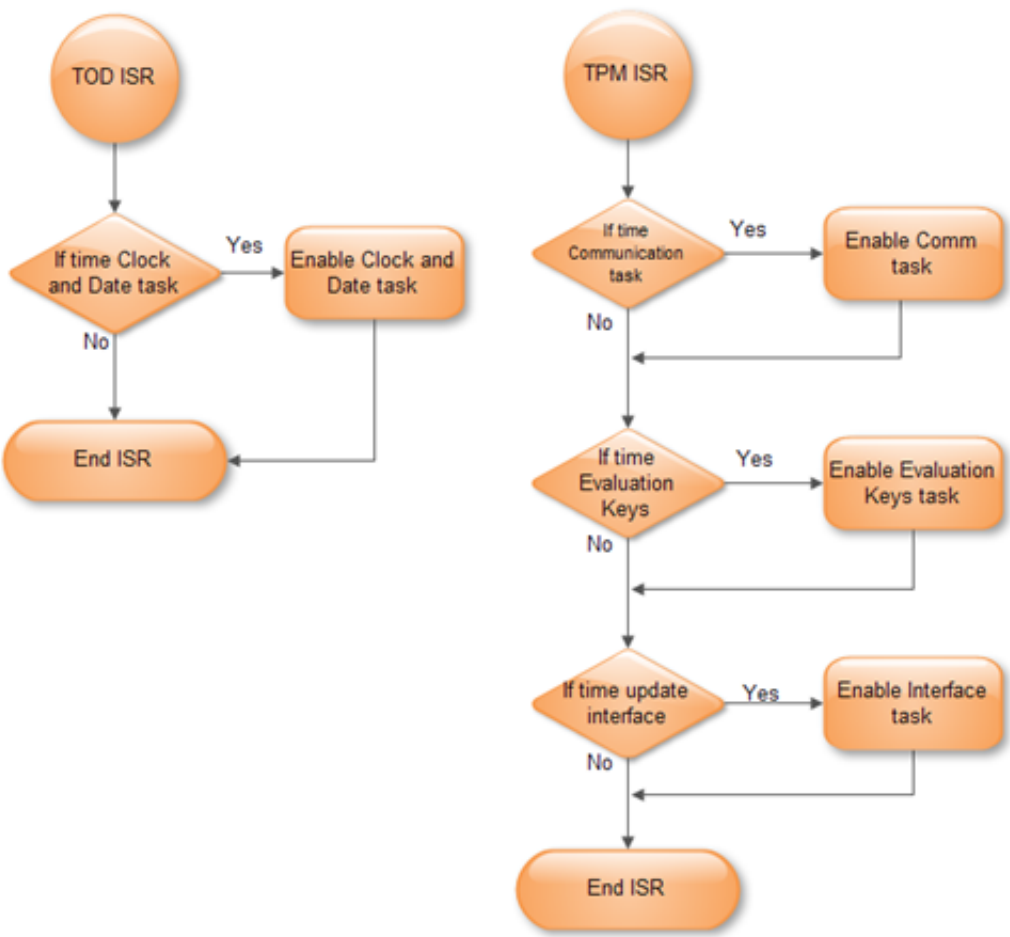

Figure 3-23. Scheduler flowchart

3.5.4 System management

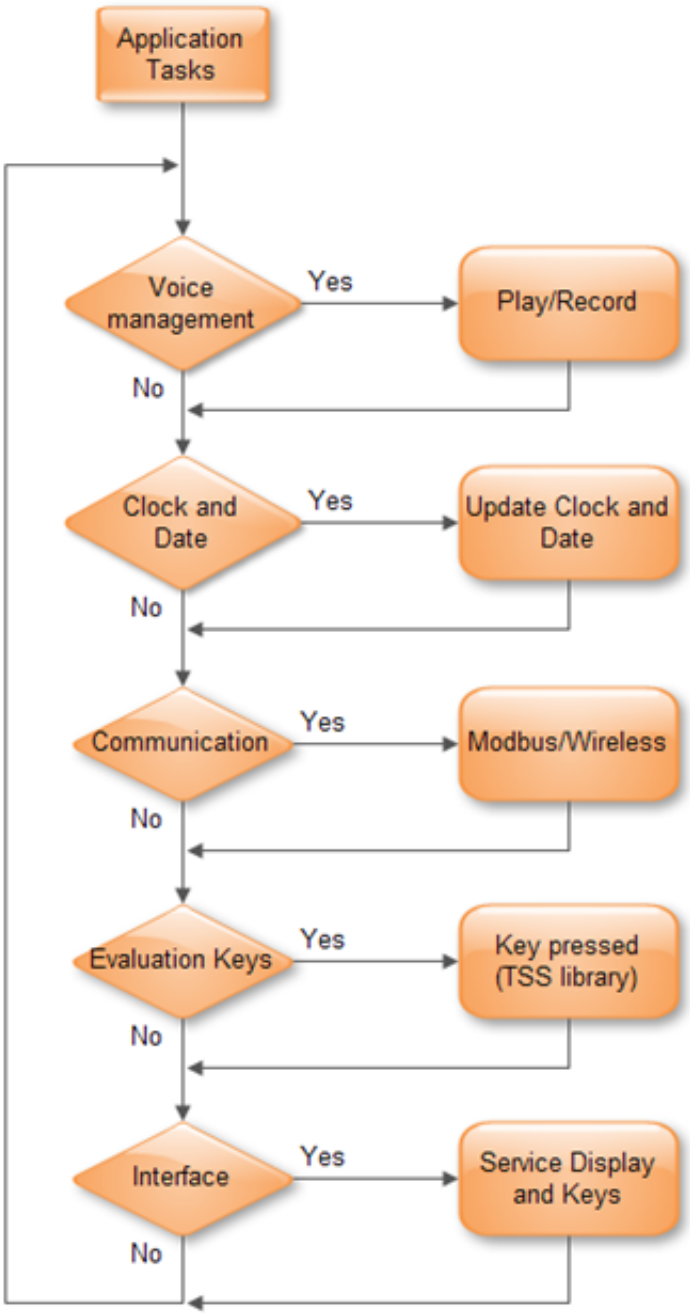

Figure 3-24. Application process flowchart

3.5.5 Sound management

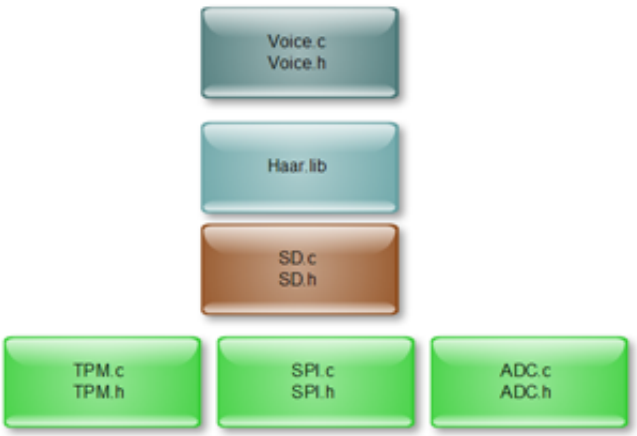

Figure 3-25. Voice messages SW modules

The Smart Appliance UI aims this function to record voice messages, store them in an external memory, and play them at anytime.

The driver sound is optimized with an audio compression system and a memory unit. The following bullets describe the solution for the sound manager.

Voice block diagram

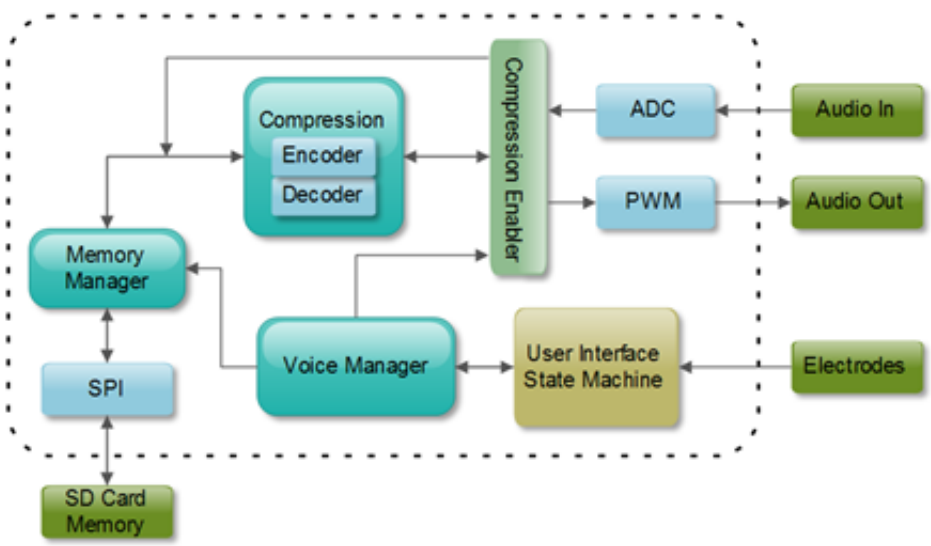

Figure 3-26. Voice block diagram

Sound Input/Output mechanism

The Sound Input/Output is controlled by a single TPM of MCU which is enabled or disabled if the user is in voice messages menu.

An ADC is triggered by the TPM to a 16 KHz sample rate to save a sound input signal which is generated by a microphone circuit. A PWM signal is used to modulate an audio output signal and is configured to an 80 KHz frequency due to the filter placed in the speaker circuit.

Sound compression

The selected compression algorithm is based on Haar Wavelets similar to Fourier analysis in that it allows a target function over an interval to be represented in terms of an orthonormal function basis.

The Haar transform provides a transform domain where energy is concentrated in specific regions that can be better visualized on the next image.

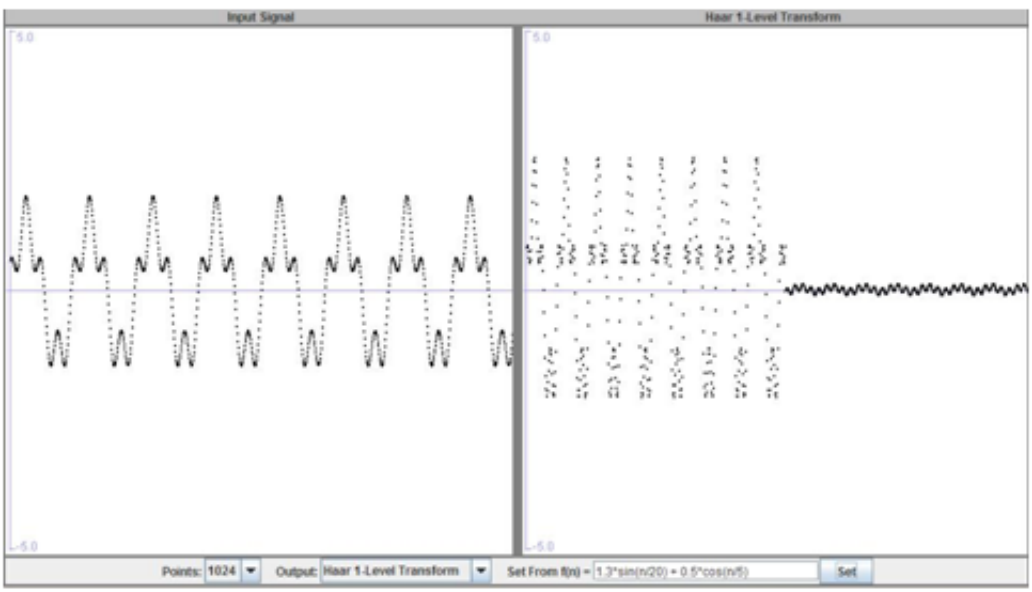

Figure 3-27. Haar Transformation

Sound recorder/player flowchart

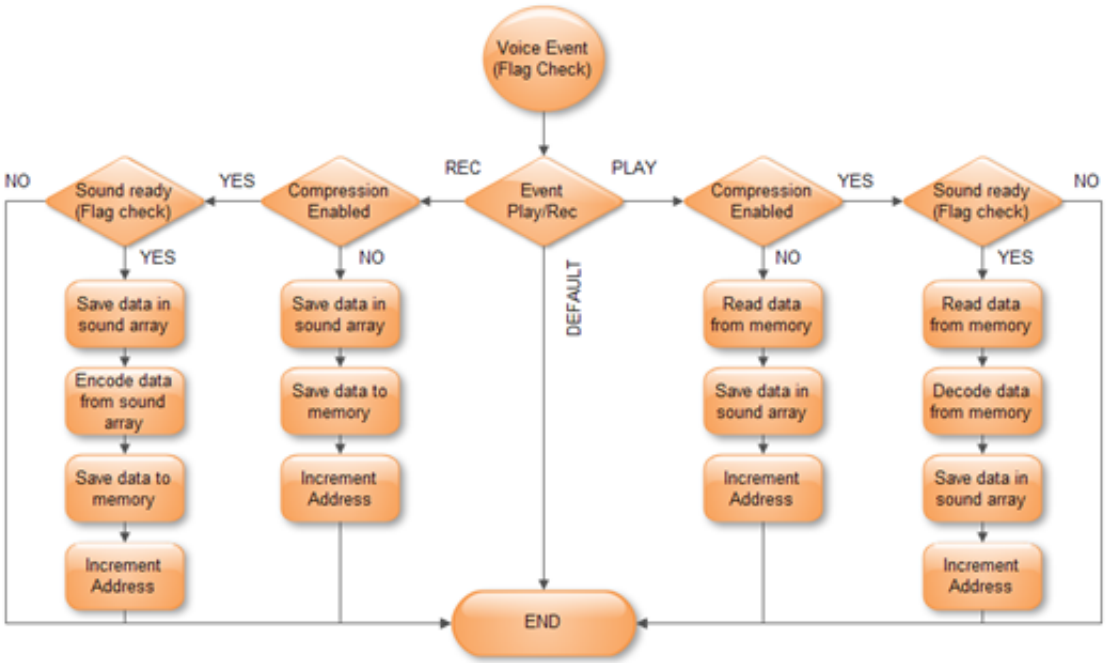

Figure 3-28. Sound recorder/player flowchart

3.5.6 Clock and date

The TOD module is used to create the calendar functionality. With this module, the user interface creates the time and date. It displays the day-of-the-week and time in 12 hrs or a 24 hr format.

Time also synchronizes with the time of record or play and the time on backlight for display functions.

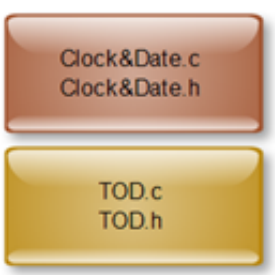

Figure 3-29. Clock and Date SW modules

3.5.7 Keys

The user interface is designed with Freescale the Touch Sensing Software (TSS 2.5) library that controls seven electrode buttons that are used for function keys. These keys are programmed to allow the system command interpreter to execute functions.

NOTE

For further information on the Freescale Touch Sensing Software library visit www.freescale.com/touchsensing

A state machine is designed for control keys and it executes the most important functions of the user interface system, such as:

- Navigate in the different menus and go out them
- Enable or disable functions of refrigeration control
- Modify parameters of operation control
- Set the clock and date
- Lock the system

Menus in both low-end and mid-end interfaces use the same type of operation for the state machine, and differ in functions such as Sound and Display only. The following table shows the different menus and functions that can be modified or accessed with keys in both interfaces.

Table 3-2. Low-end vs. mid-end

Functions	Low-End	Mid-End
Turbo mode	YES	YES
Vacations mode	YES	YES
Alarm	YES	YES
Lock system	YES	YES
Temperature level	YES	YES
Clock settings menu	YES	YES
Date settings menu	YES	YES
Voice messages reproduce record and erase	NO	YES
Configuration menu	NO	YES
Speed communication	NO	YES
Speaker volume	NO	YES
Internal Light	YES	YES
Light test mode	YES	YES
Backlight control		YES

The configuration electrodes measurement method is CTS in low-end and mid-end interfaces to avoid the use of extra TPM. It can be found in the files shown in figure below.

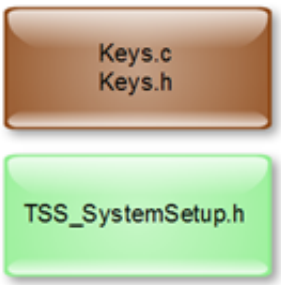

Figure 3-30. Keys SW modules

3.5.8 Display

The HAL driver for the LCD module provides the module initialization functions and routines to display information. For the low-end LCD, the HAL driver gives the configuration options in blink mode. However the high-end LCD driver gives the options of configuration image-based.

The HIL routines display relevant information for the user interface application. They can write the date and time as well as the refrigerator functions among other features.

The updates in the display will depend of the state machine of Service Key.

Figure 3-31. Low-end display SW modules

Figure 3-32. High-end display SW modules

Chapter 4
Led Driver Board

4.1 Features

The LED driver board is used to control the internal light, its intensity or LED RGB color is based on an array of transistors as triggers. It also checks if the door is open or closed and determines if the internal light is enabled or disabled. Other features on this board are explained as follows:

- MC9S08SE8 compatible peripheral module
  - Bus frequency 8 MHz
  - Timer Pulse-Width Modulator
- BDM connector for programming and debugging
- On board + 3.3 V regulator as the power supply to components
- 1 SCI TX to Single Wire Module used in Modbus protocol
- Darlington transistor arrays controlled by three channels of PWM
- Connection diagram of the status door
- Power supply + 24 V, One-Wire-SCI, and GND on the same connector

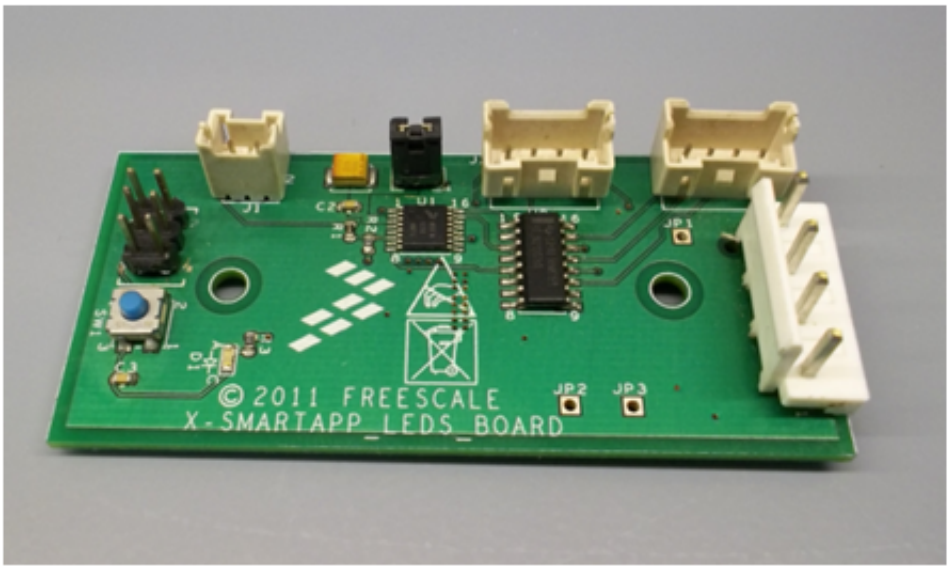

Figure 4-1. LED driver board (top)

4.2 Block diagram

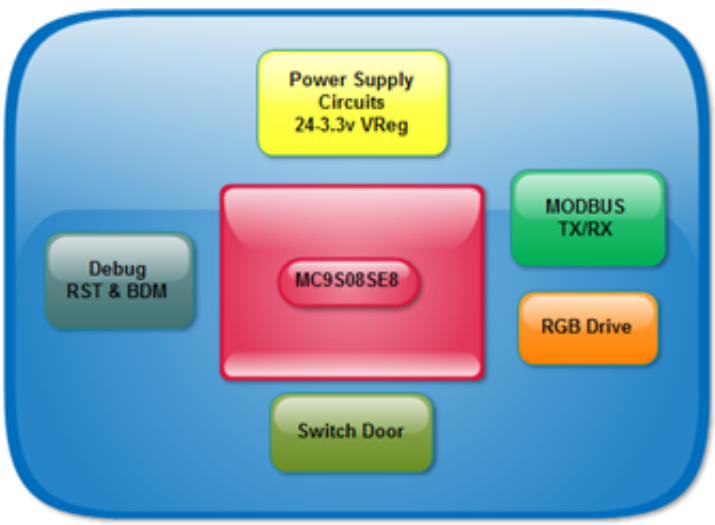

Figure 4-2. LED driver board block diagram

4.3 Placement — LED Drive Board

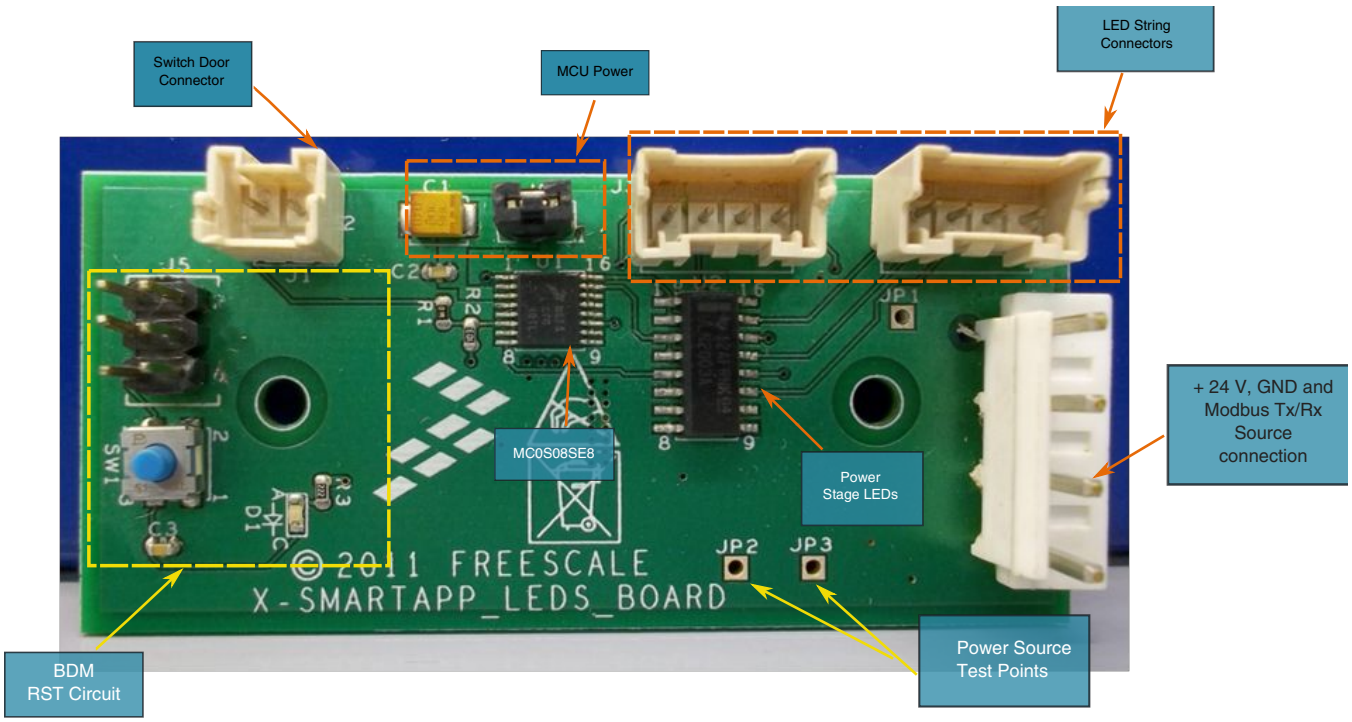

Figure 4-3. LEDs board top side

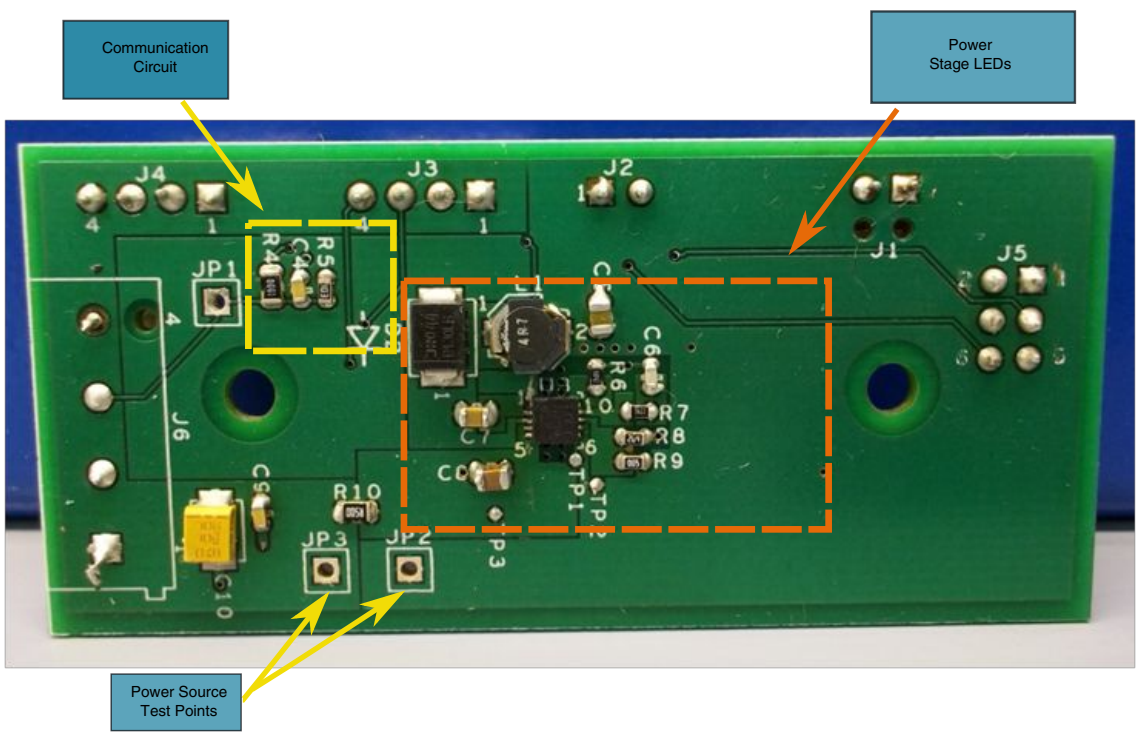

Figure 4-4. LEDs board bottom side

4.4 Hardware description

4.4.1 Door switch

This circuit is designed to know the status door and for SW to determine whether it is necessary for the light is to be on/off or to enable the alarm.

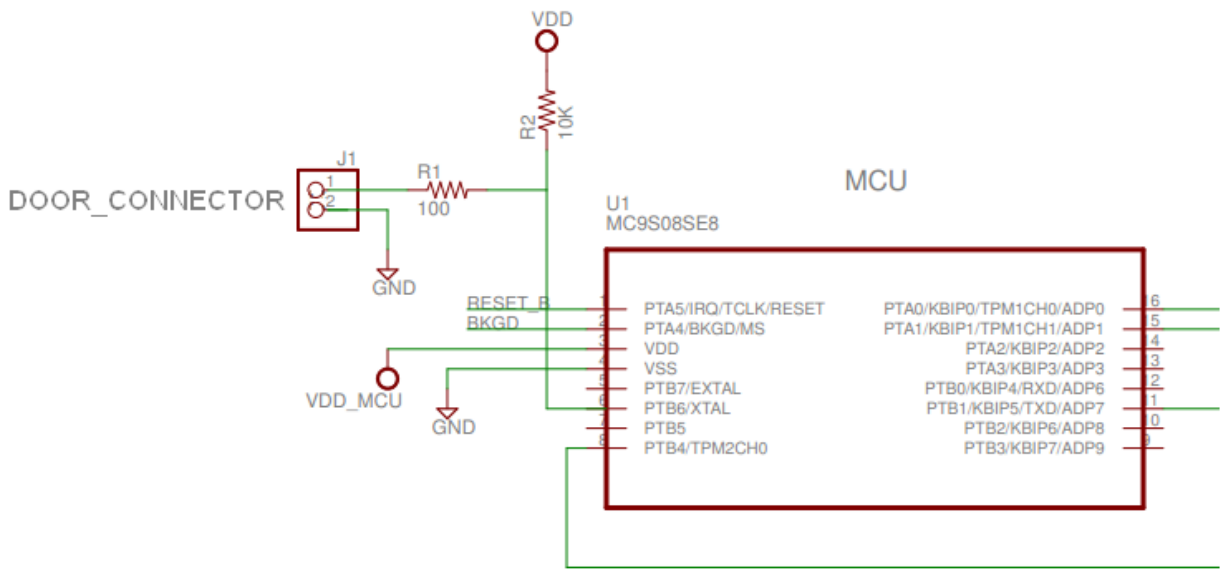

Figure 4-5. Circuit status door

4.4.2 LED circuit driver

The LED driver board is designed to provide the system a general illumination and high-brightness to the cooling chamber. The following block shows a Darlington transistor array (ULN2003AN) for the control LEDs using PWM signals as trigger for the different colors: red, green, and blue.

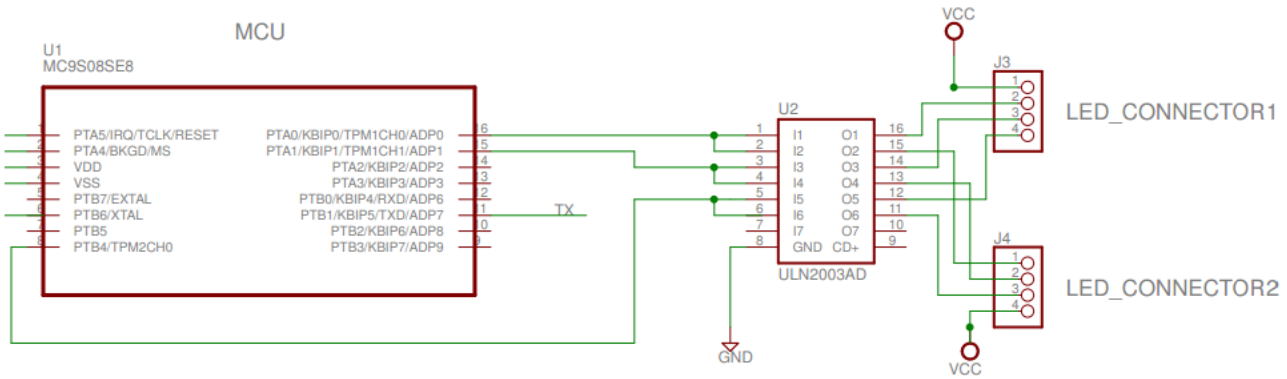

Figure 4-6. Control LED RGB

The two string LEDs used in this circuit are powered to 24 V. The use of these LEDs can change according to customer needs.

4.4.3 Communication hardware

The following circuit is designed to interconnect the system used in control register and status.

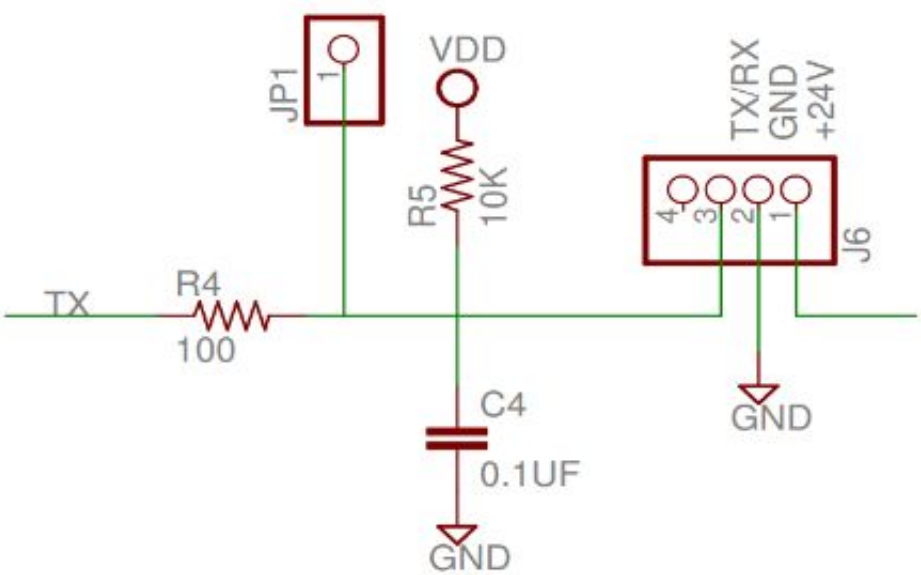

Figure 4-7. Circuit SCI One-Wire

4.4.4 Debug and Reset Connections

A serial background command such as WRITE_BYTE can be used to allow an external debug host to force a target system reset. (See User Interface BDM and RST connection).

4.4.5 Power source

The LEDs driver board uses the same power source circuit design as the user interface board (refer to User Interface power source).

4.5 Firmware LED driver board

4.5.1 Overview

The LEDs firmware was extended to support communication between the UI and the RGB panel extension. A new command is added to setup the color of each LED and the state of refrigerator door. Communication protocol is explained in chapter 6 Modbus protocol.

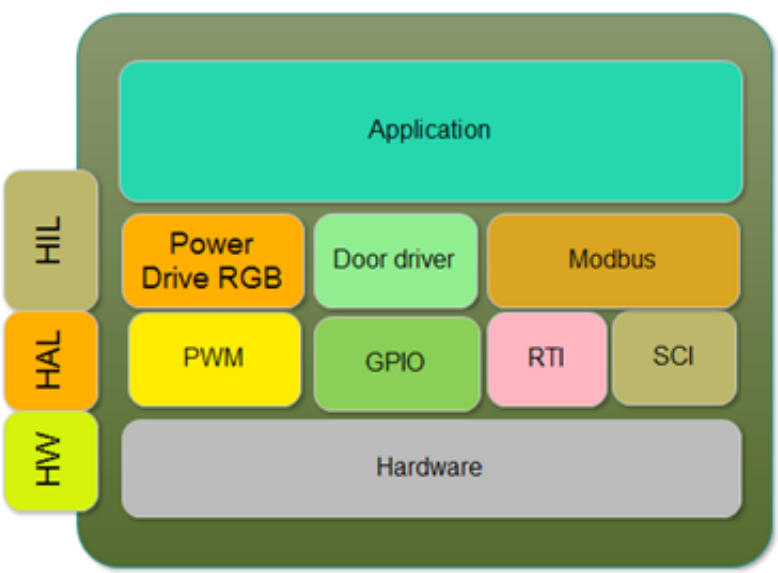

Figure 4-8. LED driver board SW architecture

4.5.2 Flowchart

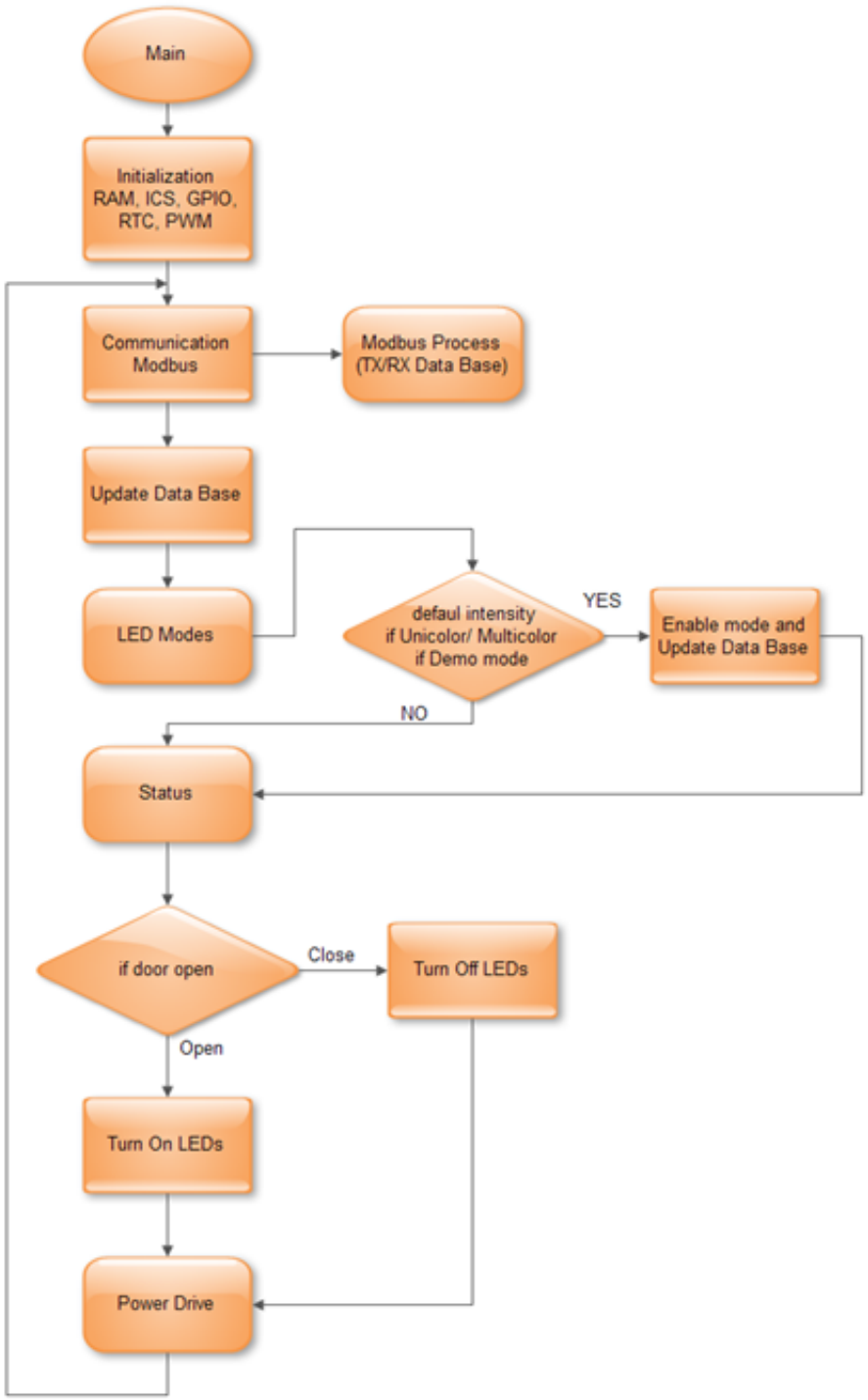

Figure 4-9. High-Level LED drive flowchart

4.5.3 RGB power drive

The LEDs can be turned on or off in the cooling chamber, this depends on the status door. If the door is open and the LEDs are turned on, the RGB control can be performed. If the door is closed, then a logic signal is detected and the LEDS are turned off (via software).

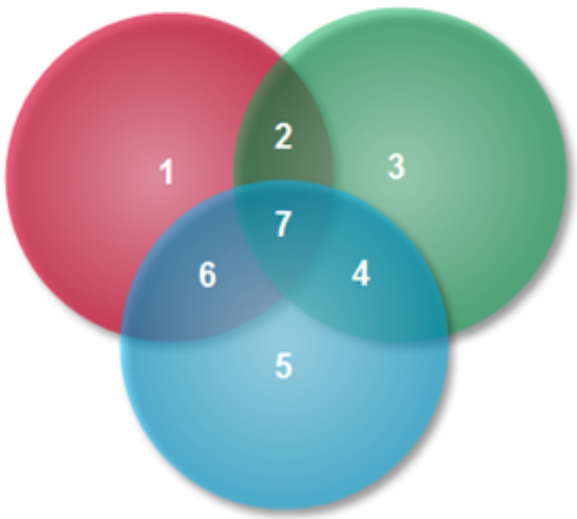

Figure 4-10. RGB Combination Color

In Figure 4-10, the circles represent all the colors and the numbers represent the sequence of individual colors that will be modulated to a maximum of 255 units, with a duty cycle of a PWM signal and using a function with a sequence data on each line. The colors fade from the current values to the new values.

The sequence data on each line has the following fields:

- Red PWM value 0 = 0% (LED off) through to 255 = 100% (LED fully on)
- Green PWM value 0 = 0% (LED off) through to 255 = 100% (LED fully on)
- Blue PWM value 0 = 0% (LED off) through to 255 = 100% (LED fully on)

Changes in LED brightness are more noticeable between 0 and 128 than from 128 to 255.

In demo mode, the end of all available sequence data, both the fade rate and hold time fields must be set to 255.

Chapter 5
Control Board

5.1 Features

The smart appliance low voltage control board is the control and instrumentation board for the Smart Appliance system. The control board implements features such as:

- Solid State Motor Control for a BLDC Compressor to regulate refrigerator temperature.
- Temperature sensing using a thermostat.
- Control the temperature distribution fans.
- Communication with low-end and mid-end blocks in the system via a one-wire Modbus type protocol reducing harness size.
- On/Off relay defrost resistor control for Freezer no-frost functions.
- Obtain and communicate temperature measurements.
- 802.15.4 Communications with User Interface and home gateway to receive configuration commands.
- 56F8006 DSC used as main processor.

Figure 5-1. Control board (top)

5.2 Control Board Block diagram

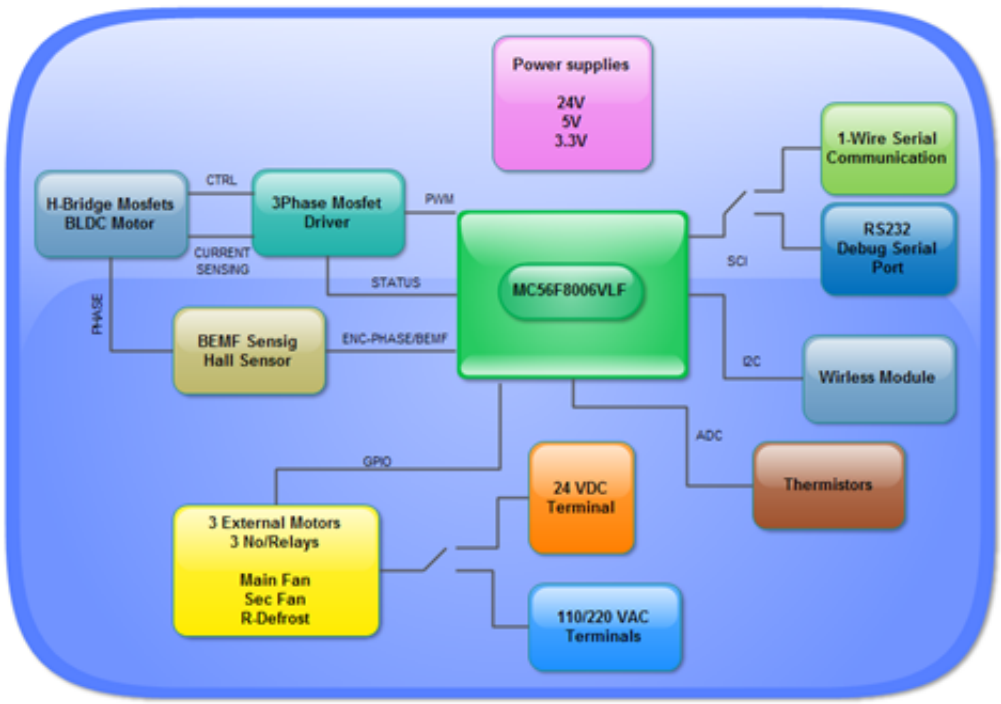

Figure 5-2. Control board block diagram

5.3 Placement (Control Board)

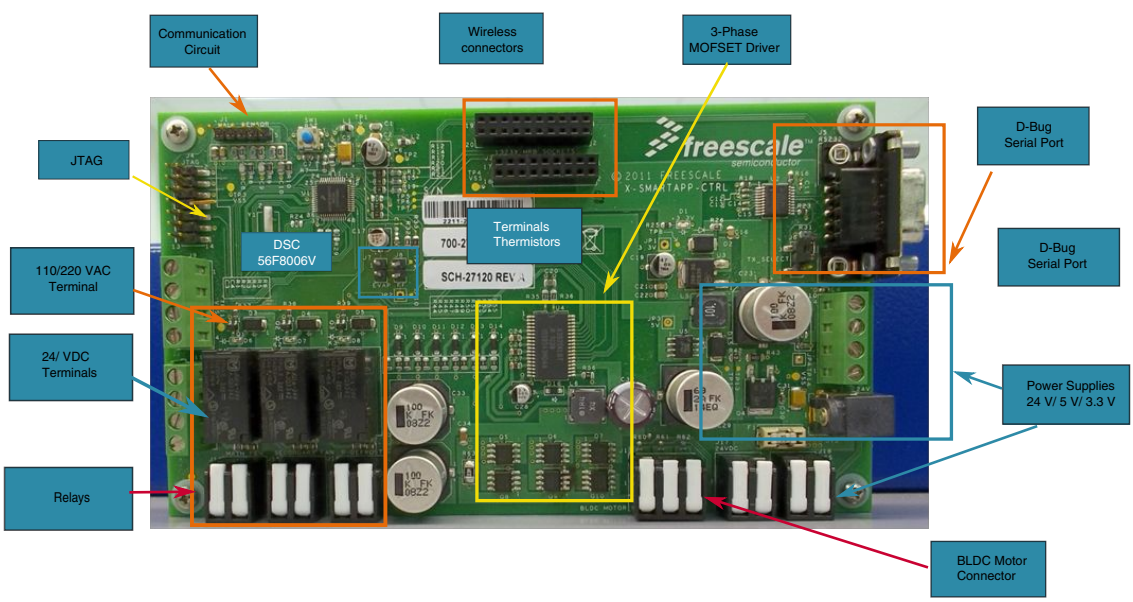

Figure 5-3. Control Board placement

5.4 Hardware description

5.4.1 Power

The smart appliance low voltage control board is powered by a 24 volt, 5 Amp external power source. It has a 24 VDC, 5V DC, and 3.3 VDC regulators.

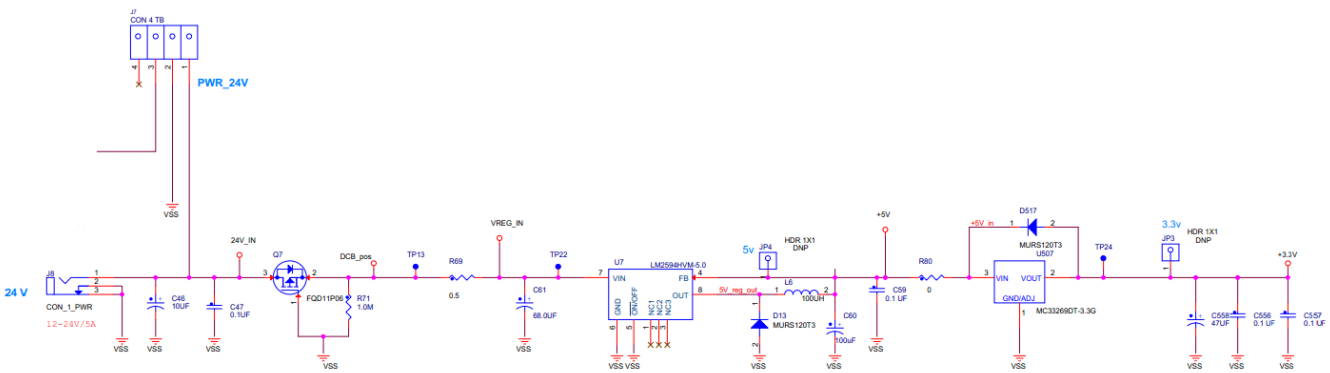

Figure 5-4. Control board power source

5.4.2 Clocking

The 56F8006 DSC does not use external clocking, but the board is prepared to use a quartz crystal for future updates.

5.4.3 1323X Modular reference board connector

The control board optionally interfaces to a MC1322x modular reference board (MRB) board. For specifics see Figure 5-5.

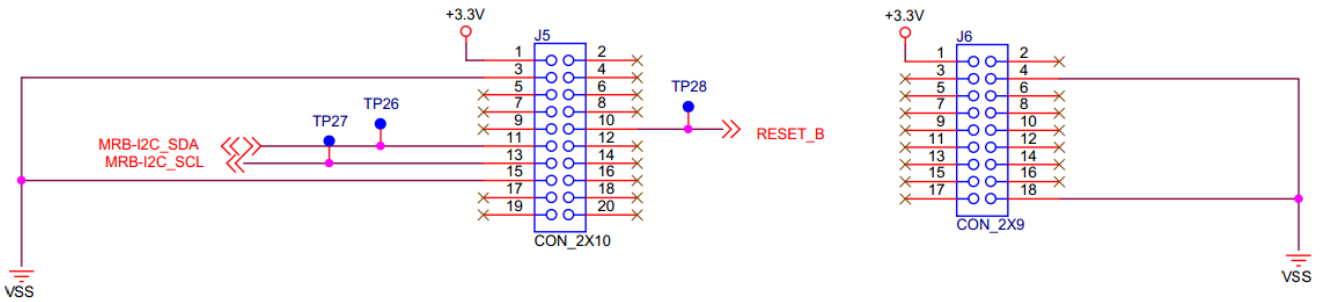

Figure 5-5. 1323X - MRB sockets

5.4.4 Serial Interface

The smart appliance low voltage control board includes a single wire, half-duplex serial interface. Output for the SCI0 TXD pin goes directly to the 3 pin terminal block which also includes a 24 volt power and ground (see LED driver board Circuit SCI One-Wire).

5.4.5 Aux RS232 interface

The SCI0 (RXD/TXD) pin can also be connected to an additional RS-232 transceiver and DB9 connector to be used as an additional debug port. It has a jumper available on the board to switch the TXD to the single pin interface or to the auxiliary RS232 interface.

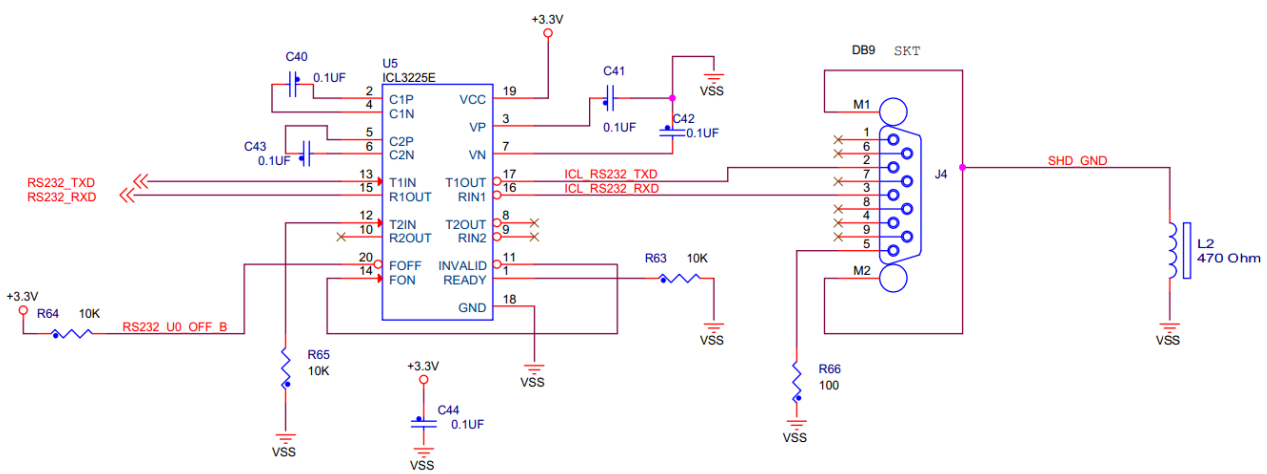

Figure 5-6. RS232 Debug serial port

5.4.6 3-Phase motor control inverter

The smart appliance low voltage control board has a 3-phase low-voltage motor control inverter that includes the following characteristics:

- 6-FET 3-phase inverter capable of handling 24 volts and 5 Amps
- FET pre-driver
- DC bus shunt resistor.
- Signal conditioning for ADC feedback for each of the three phase voltages and for the 4 DC bus shunt resistor

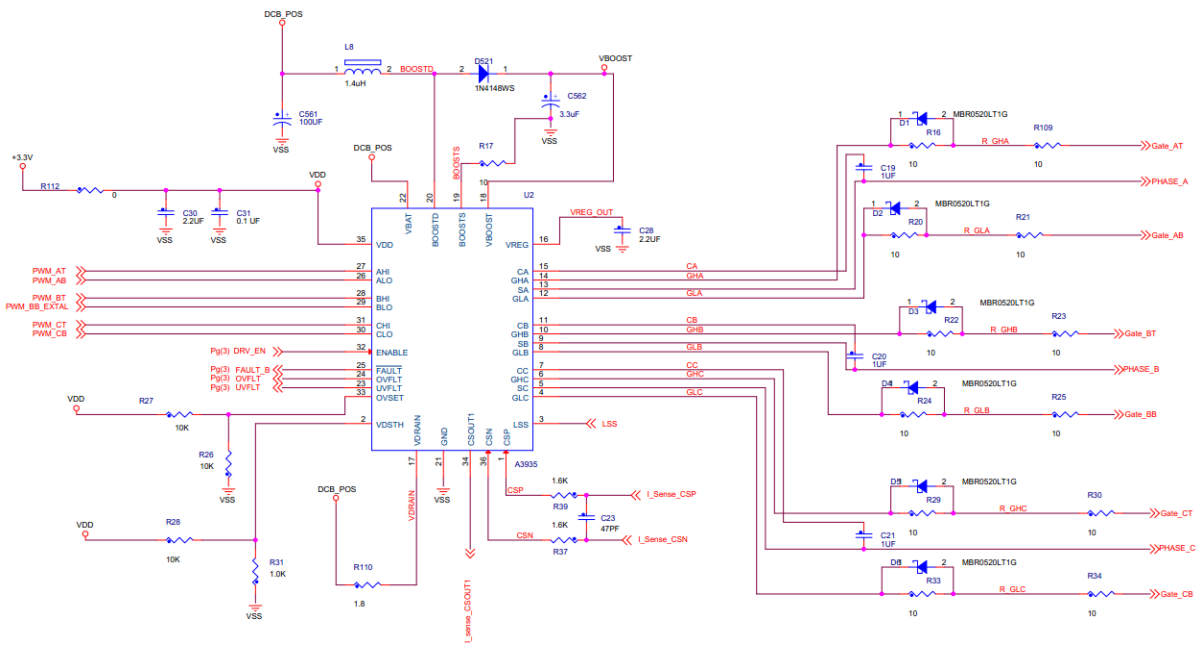

Figure 5-7. FER pre-driver DC Bus current sensing shunt amplifier

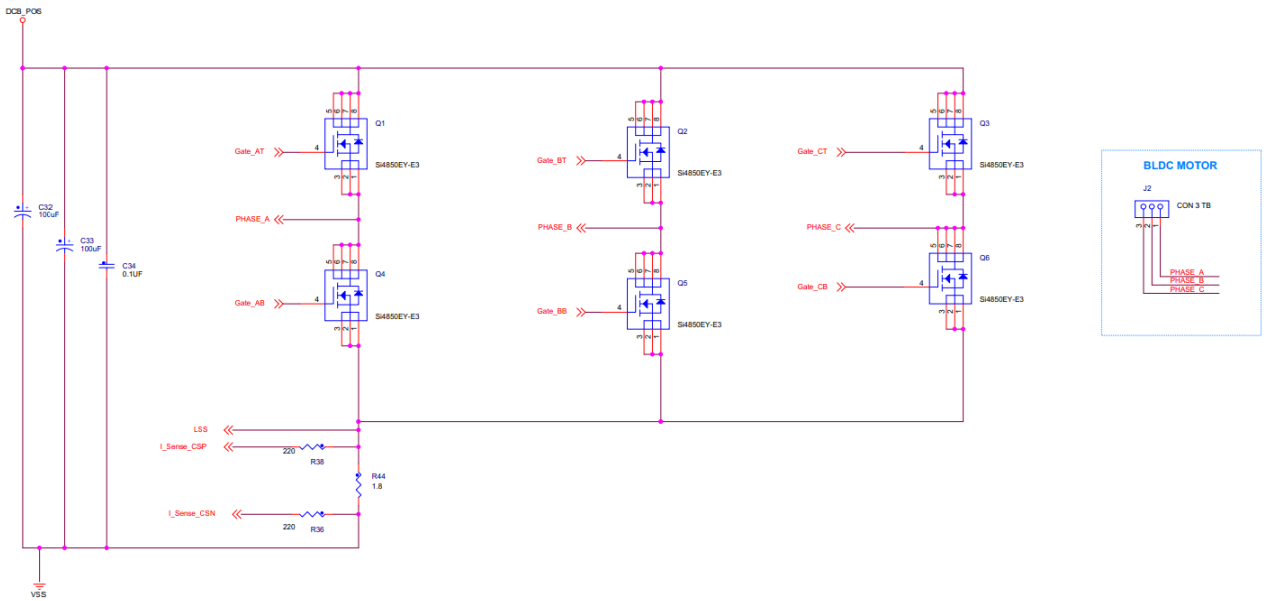

Figure 5-8. 3-phase FET inverter for BLDC motor

5.4.7 Control connectors

The control board has several power connectors designed to switch to either 24 VDC or 110/220 VAC with internal relays for refrigerator power devices (fans, resistor).

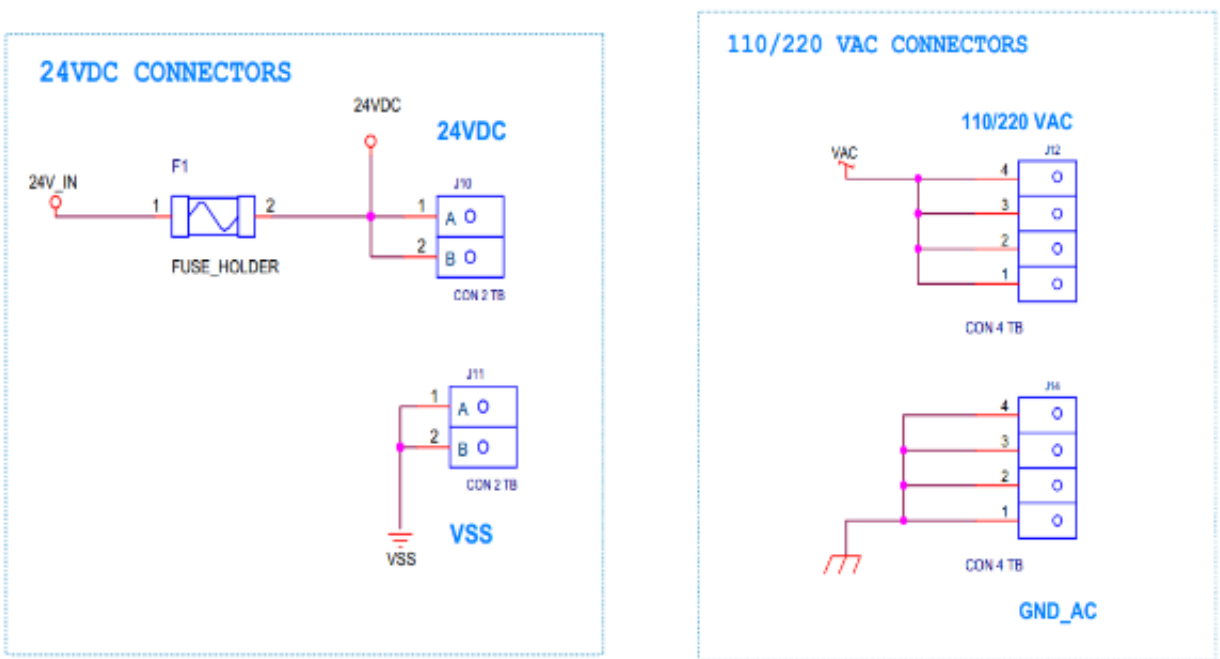

Figure 5-9. Control connectors

5.4.8 Relays

The control board has a main fan, secondary fan, and defrost resistor relays with configuration normally open. Relays are controlled with 12 or 24 VDC from a board via a FET transistor with a current return diode for coil current peaks. The diagram connection of the three relays is shown in the following figure.

NOTE

There is no hardware control design for fans and the defrost resistor.

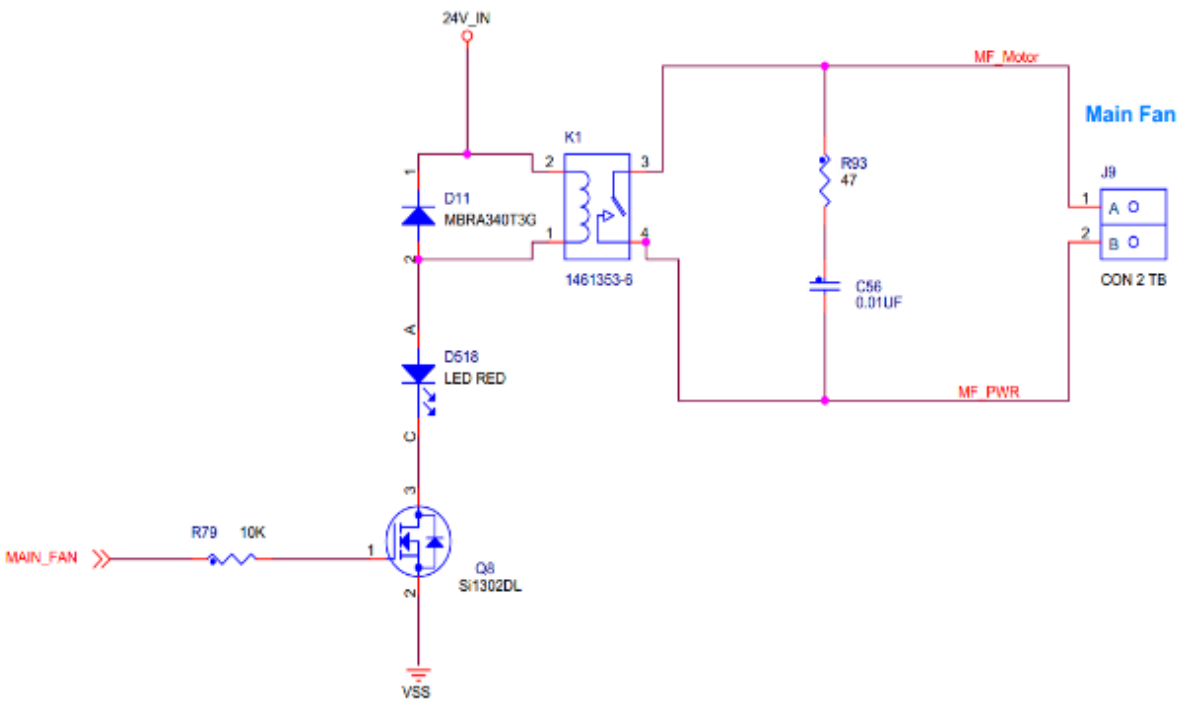

Figure 5-10. Relay diagram connection

5.4.9 Hall sensor

The control board has an additional Hall sensor circuit used for proximity switching, positioning, and speed detection in a motor control. The smart appliance reference design does not use this function but the circuit is implemented for any future customer request.

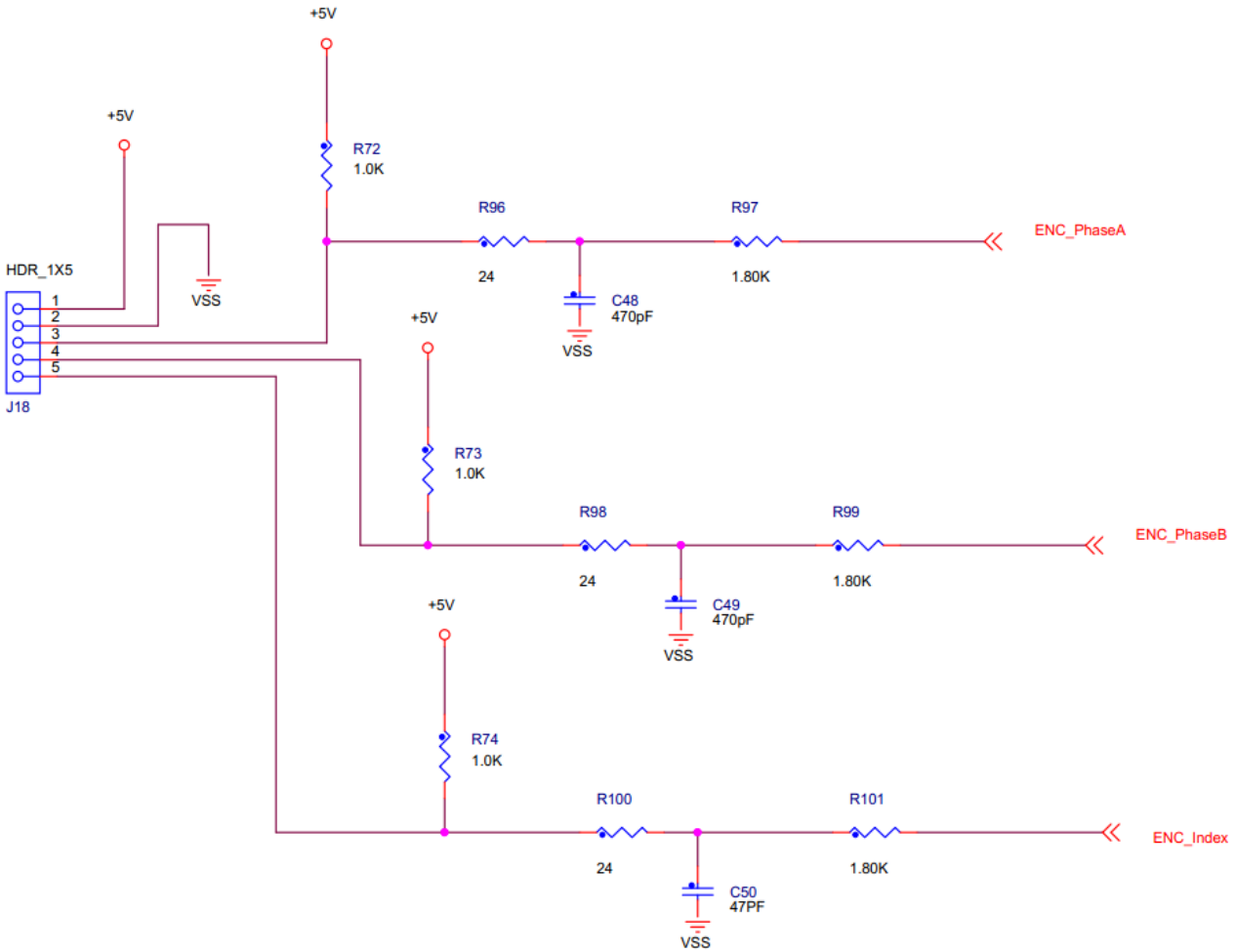

Figure 5-11. Hall sensor circuit diagram

5.4.10 Thermostat

A thermostat is used to regulate the temperature of a Smart Appliance System keeping the temperature at a desired level. Thermistors are devices that change its impedance depending o the temperature, the board uses a combination of capacitors and resistances to calculate impedance based on the internal charge time. Circuit is shown below.

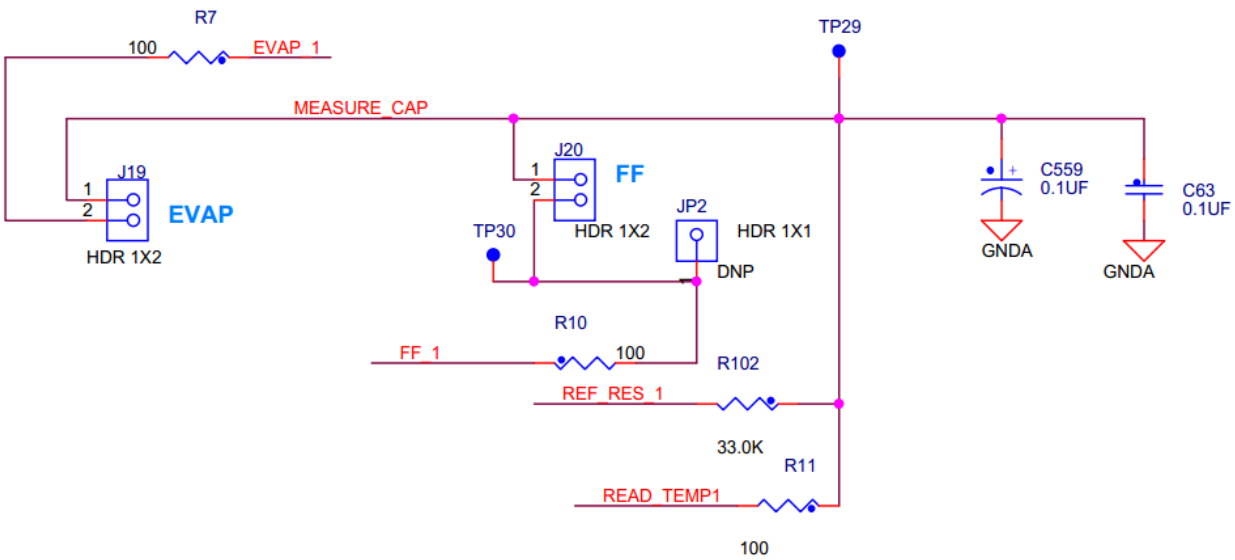

Figure 5-12. Temperature control

5.5 Firmware

5.5.1 Overview

Most of the processing system is carried out on the control board. The refrigerator is a closed circuit that relies on a refrigerant gas and consists of two processes, a compression and a decompression, which are made using a compressor and generates cold inside. Due to these changes, the energy in the system is very variable in addition to temperature control, making the system more complex and requiring better processing.

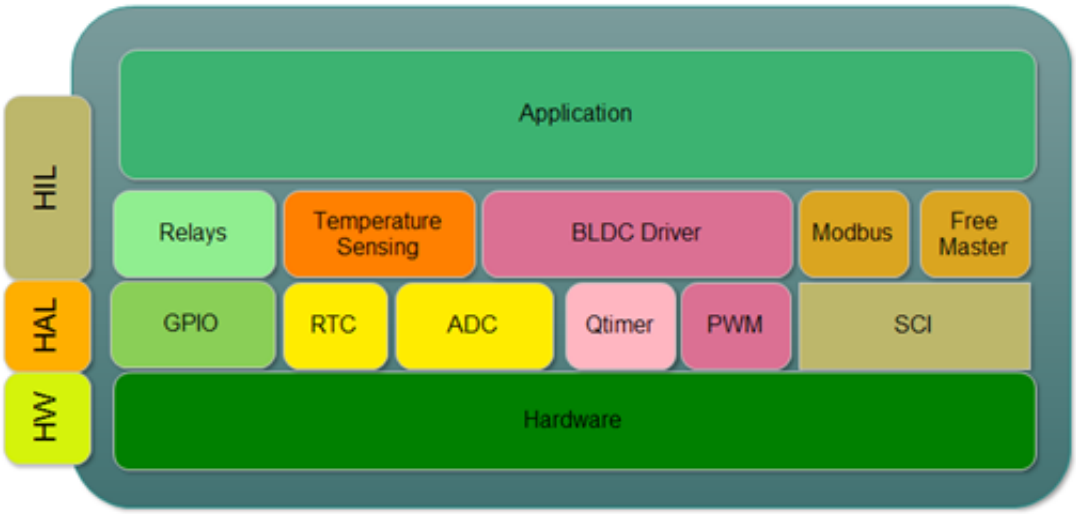

Figure 5-13. Control board SW architecture

5.5.2 Flowchart

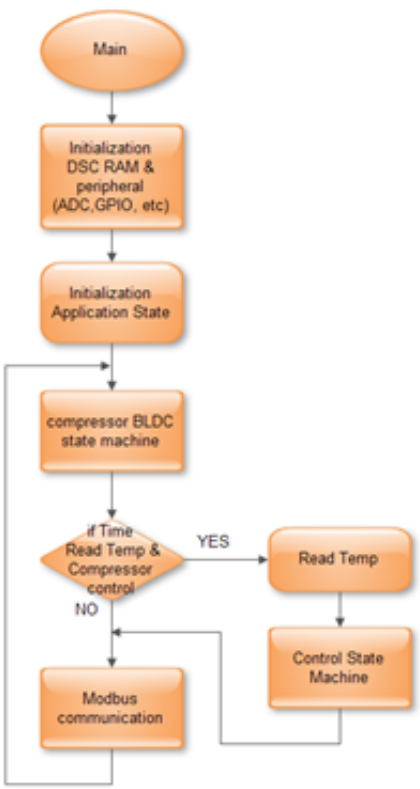

Figure 5-14. High level control board flowchart

5.5.3 Control state machine

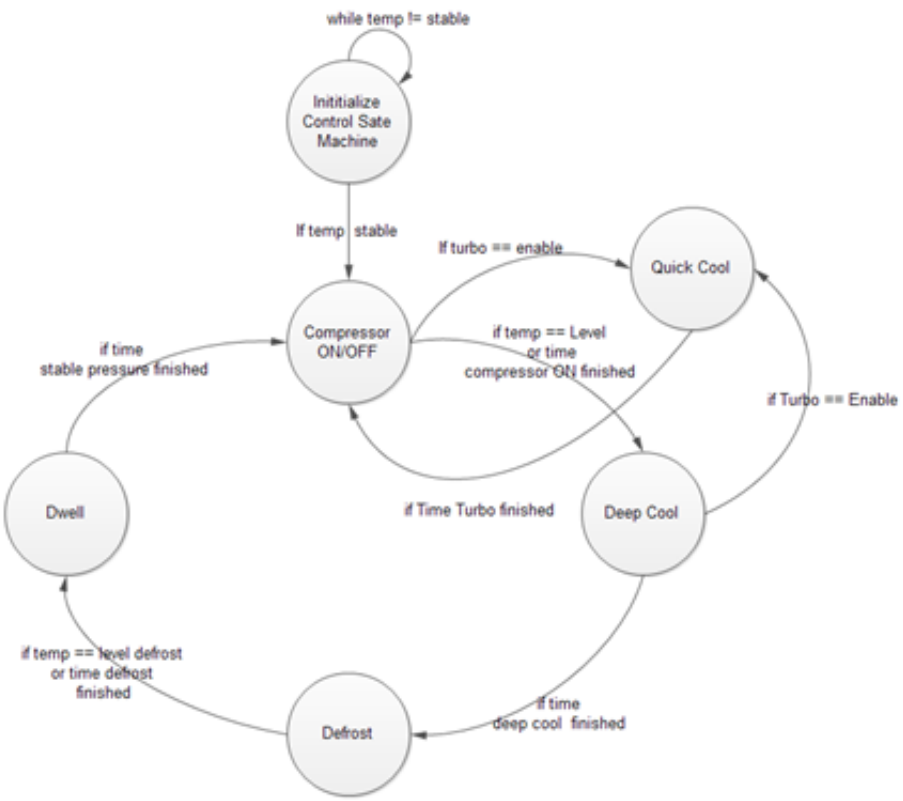

Figure 5-15. Refrigerator control state diagram

5.5.4 Temperature control

The temperature reading is made with a thermistor and due to the percentage error that a conventional refrigerator has as a rule, the temperature reading is done through a reference circuit RC and with the help of an ADC and an RTC calculates the load times of a capacitor (see the hardware description for the thermistor).

The obtained data is processed by using the capacitor time-charge formula T = RC, where T is the load time and C is a known capacitance value. As a result, the value of the thermistor resistance can be obtained. The resistance value is compared in a table of resistances to convert the resistance value to temperature.

The files below show the process of temperature control.

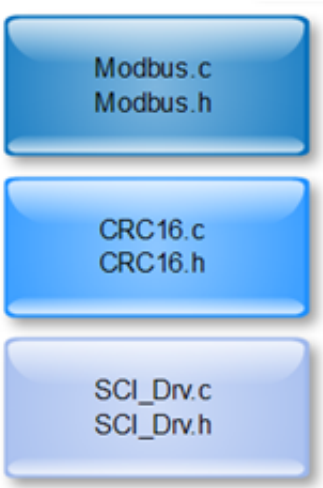

Figure 5-16. Temperature control modules

5.5.5 Defrost and fans

The control board has the capability to enable or disable the fans and defrost. However, this design reference has no control logic for these features.

Chapter 6
Modbus Protocol

6.1 Introduction

Smart appliance communications are based on Modbus protocol. This is a multi-point protocol based on master/slaves architecture. Functions and overview are explained below.

Modbus is focused to modular applications that use general application data base(s). This characteristic makes it an optimal protocol for Smart Appliance application because of the module system required for easy reference designs changes. General data base application is explained below.

6.2 Modbus General Operation

The Modbus works with a set of instructions and commands that drive the communication system. Although there are many communication commands over this protocol, smart appliance application use read and write commands only; this is due to the small information quantity required to control the system.

Read command reads information from the slave application data base and put it over the master data base. Read command basic data structure is given by the slave ID, number of command, direction/size of registers, data, and check data bytes.

Write command writes information from the master application data base and put it over the slave data base. Write command basic data structure is given by the slave ID, number of command, direction/size of registers, and check data bytes.

6.3 Data base general operation

The data base is the bridge between application and communications to have a modular system that can be easily fixed, maintained, and expanded. All these advantages make easy applications for many appliance topics.

The data base is required to match the application with communication by matching important control global variables with data base. The communication system refreshes the global system variables required for appliance control.

6.4 Communication system main operation

Refrigerator system needs to be a robust system; therefore, the communication system has failure detects, and checks constantly the status of the entire system. System will be in wait state until an event occurs, and then communication cycle will begin with most priority information and then check lower priority.

Communication system could have two main operation modes:

Refresh data base cycle all operating time — over this mode MCU will check the entire status of the data base giving important data lower refreshing time.

Refresh data base cycle only when changes take place: MCU only will refresh database when slaves have an update in its data base.

6.4.1 Read command operation

The Read command executes the following steps:

1. Master waits for the timer Modbus task to be zero and slaves are waiting for the master to “talk”.
2. When the service timer for the Modbus expires the master sends a chain to the slave by a defined function vfnMaster_Send_Read(x,y,z) where you introduce slave ID (x), first data register direction (y) where the data starts to be read, and registers (z) to be read. You must make sure that the register and slave do exist and that they are in range.
3. The master waits for the answer from the slave or waits for the fault timer to expire. If the fault timer expires, the master MCU increases an error counter and then checks it again or tries in the next loop.

4. If master receives a correct message, it saves the data in its data base via function vfnMaster_Put_on_DB(x), where you introduce the first data register direction (x) to start saving the data.

6.4.2 Write command operation

The write command executes the following steps in its operation:

1. Master waits for the service timer Modbus task to be zero and slaves are waiting for master to talk.
2. When service timer for Modbus expires, the master sends a chain to the slave by a defined function vfnMaster_Send_Write(x,y,z,*p) where you introduce slave ID (x), first data register direction (y) where the data is going to start to be written, number of registers (z) to be written, and a data pointer (*p) which is the direction of the information the master is going to transmit to the slave. You must be sure that register and slave do exist and are in range.
3. Master waits for the answer from the slave or waits for the fault timer to expire. If the fault timer expires, the master MCU can save the error then check it again or try it in next loop.
4. If master receives a correct message, it will be continuous with the next communication instruction.

NOTE

Only continuous blocks of data can be written.

6.4.3 Synchronization between messages

Synchronize timers for communications is the most important setting for correct system operation and development. Main timers are fault and reception. These timers provide the bus synchronization, and each one must be set according to data size and communication speed.

6.4.4 Communication errors

An error in the communication line is detected by a faults counter that probes the robustness of the communication system in any environment.

6.5 Driver application use

The Modbus driver application works to refresh the data base in a cycle mode. For this operation mode, communications have a cycle to check data information. All time communications are made between master and slaves to share information without an update request.

Modbus communication protocol is explained in detail in the next application. This code is implemented for the first stage of the smart appliance project.

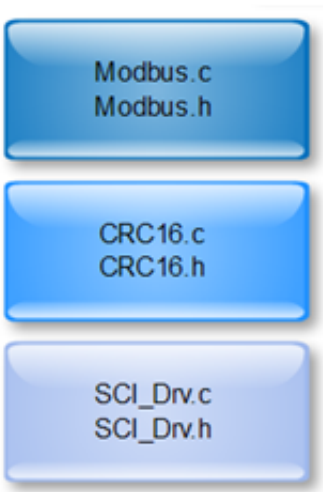

Figure 6-1. Modbus files modules

6.6 Communication driver current implementation

Basic communications start configuring baud rate and timer for Modbus on all devices on the internet. The speed configured on this project is 4800 bauds. Timers for Modbus are configured with 1 ms overflow interruption which cause the timers used for the driver to decrease at a constant time.

Chapter 7
Home Automation Gateway

7.1 Introduction

This document describes the design of a Home Automation Gateway using Freescale technologies. The Home Automation Gateway is a device that communicates with smart appliances connected to a Zigbee network under the Home Automation Profile and allows the user to control them via an embedded web server. Also, communication with a different Zigbee network created under the Smart Energy Profile is possible. Information about energy consumption and pricing information can be obtained this way.

7.2 Application features

The Home Automation Gateway (HAGateway) works as a communications bridge between several networks using different interfaces and protocols.

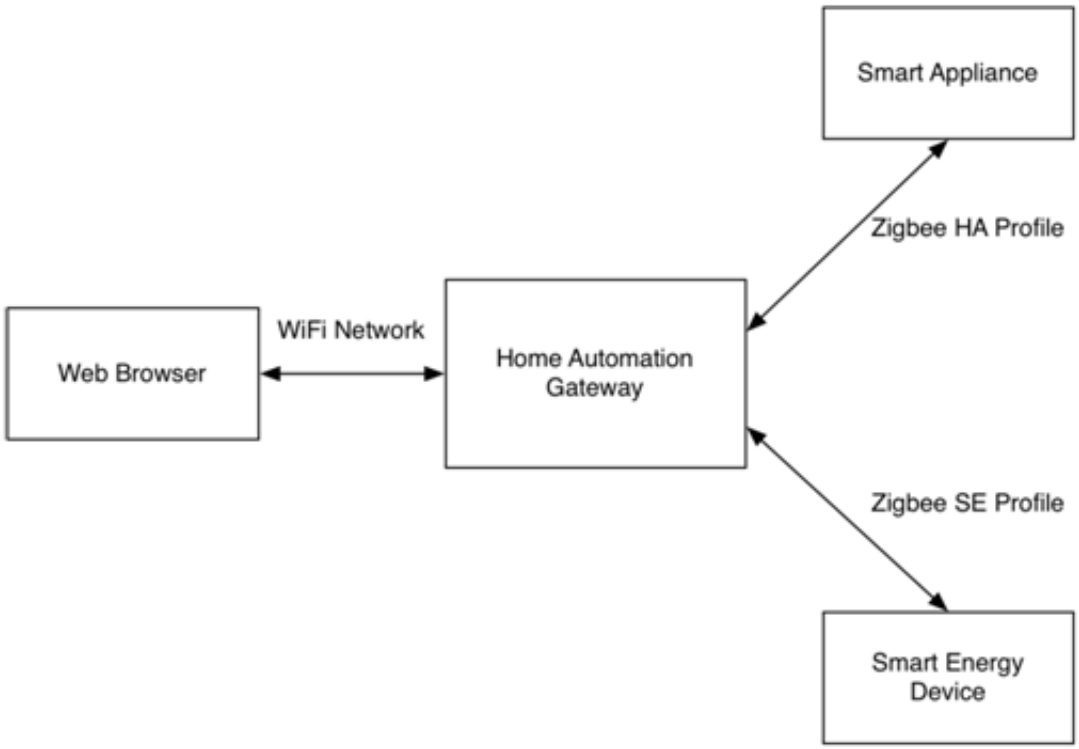

Figure 7-1. Home Automation Gateway

The user can interact with Zigbee-enabled smart appliances by visiting a website hosted on an embedded server in the HAGateway. This website is built dynamically, according to the status of the Zigbee Home Automation network and displays the status of every connected device.

The configuration and control website allows user interaction by implementing CGI-like functionality. A new device details page is created for each device that has joined the Zigbee network. Each page contains detailed information and allows control of certain properties, depending on the kind of appliance it is intended to control. For the purpose of this reference design, only one smart appliance, a refrigerator, is implemented.

7.3 Home automation gateway components

The HAGateway is based on the Kinetis K60 MCU. This platform implements functionality that allows user interaction by creating a software model of the Zigbee network. Internet Protocol stack is implemented in this microcontroller using an external module, G1011MI, as the physical and MAC layers.

A secondary processor is used to communicate with the Zigbee network. Both platforms are wired and implement a proprietary serial protocol. This protocol is detailed later in this document in section 3.4 HA to Zigbee Serial Communication Protocol.

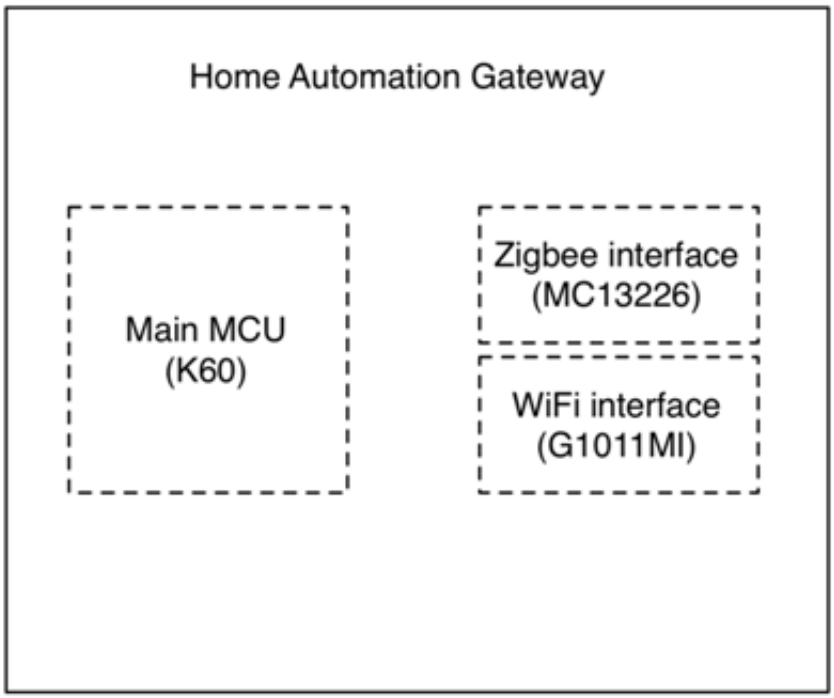

Figure 7-2. Home Automation Gateway wireless interfaces

7.4 Zigbee wireless technology

Wireless technologies allow a wide range of devices to interconnect and work together to give users great levels of control and automated behaviors. Zigbee is a standards-based wireless technology designed to create low-cost, low-power wireless devices. Using the 2.4 GHz band, Zigbee defines a set of standards intended to provide solutions to several consumer, business, government, and industrial needs.

The Home Automation Gateway is capable of acting as a Zigbee 2007 coordinator. This allows the device to create a Zigbee network and communicate with other Zigbee enabled devices.

ZigBee networks are composed of several device types: ZigBee Coordinator, ZigBee Routers, and ZigBee End Devices. Coordinators control the formation and security of networks, routers extend the range of networks and end devices perform specific sensing or control functions. The HAGateway is a Zigbee coordinator. It is responsible of creating the Home Automation network, which is a Personal Area Network (PAN), and allows other devices to join it.

A Public Application Profile runs on ZigBee devices and contains specific details about what information a device can communicate and how this device should interact with other devices on the ZigBee network. The HAGateway is capable of communicating with two profiles: Home Automation and Smart Energy.

7.5 Zigbee home automation profile

ZigBee Home Automation offers a global standard for interoperable products. It enables smart homes that can control appliances, lighting, environment, energy management and security, as well as the expandability to connect with other ZigBee networks. This profile supports devices for controlling lighting, closures, HVAC systems, intruder alarm systems, and other generic devices.

For this reference design, the end device has been implemented using a set of the functions defined for generic devices. A detailed description of this implementation is available in section Zigbee end device.

7.6 Zigbee smart energy profile

ZigBee Smart Energy is the world's leading standard for interoperable products that monitor, control, inform, and automate the delivery and use of energy and water.

This standard supports the diverse needs of a global ecosystem of utilities, product manufacturers, and government groups as they plan to meet future energy and water needs.

The HAGateway is capable of communicating with Smart Energy devices using inter-PAN communication, which allows anonymous transfer of data without joining a particular network.

7.7 Freescale Software Tools

Freescale provides a set of useful software tools that adapt to the developers needs, allowing solution of complex problems in very short development cycles.

This reference design is based on some of these resources, thus making the result a modular embedded software application with very valuable qualitative properties that would allow extensibility and flexibility to meet the needs of a final product.

Libraries that implement complex communication protocol stacks are a very valuable resource. The HAGateway is based on wireless communication, using RTCS, which is Freescale's MQX IP implementation, to allow connectivity with standard networking and Beestack to communicate with remote Zigbee devices.

7.7.1 MQX Real-Time Operating System (RTOS)

The Freescale MQX Real-Time Operating System (RTOS) provides real-time performance within a small, configurable footprint. This RTOS is designed to allow developers to configure and balance code size with performance requirements.

It is important to mention it is not hard to migrate legacy application code to a Freescale MQX-based platform. The Freescale MQX RTOS is designed to have a modern, component-based microkernel architecture allowing for customization by feature, size, and speed by selecting the components engineers wish to include while meeting the tight memory constraints of embedded systems.

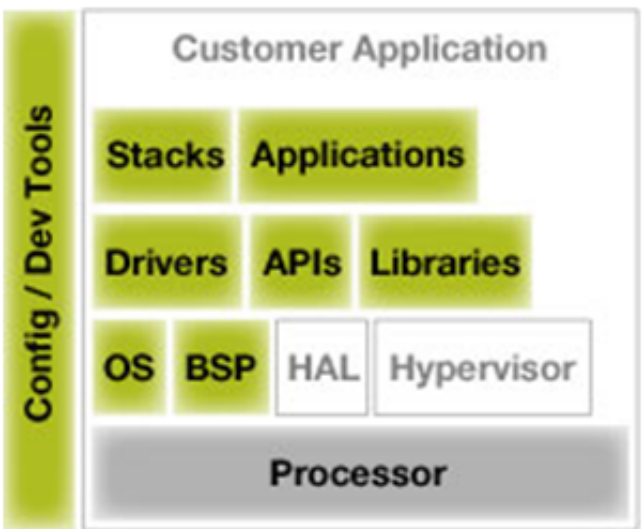

Figure 7-3. MQX Architecture

7.7.2 MQX Key Benefits

As stated by Freescale in its documentation resources, these are some of the key benefits of using MQX as a development platform for embedded products:

- Small code density — The Freescale MQX RTOS was designed for speed and size efficiency in embedded systems. The RTOS delivers true real-time performance, with context switch and low-level interrupt routines hand-optimized in assembly. It

can also be configured to take as little as 12 KB of ROM and 2.5 K RAM on ColdFire v2 (CFV2), including kernel, two task applications,one LW Semaphore, interrupt stack, queues, and a memory manager.

- Component-based architecture — Provides a fully functional RTOS core with additional, optional services. Freescale MQX RTOS includes 25 components eight are core components, and 17 are optional. Components are linked only if needed, preventing unused functions from bloating the memory footprint.
- Full and lightweight components — Key components are included in both full and lightweight versions for further control of size, RAM/ROM utilization and performance options. These components include: lightweight semaphores, events, timers, logs, and a memory component.
- Real-time, Priority-based preemptive, multithreading — In an RTOS, threads execute in order of their priority. If a high-priority thread becomes ready to run, it can, within a small and bounded time interval, take over the CPU from any lower-priority thread that may be executing. Moreover, the high-priority thread can run uninterrupted until it has finished what it needs to do. This approach, known as priority-based preemptive scheduling, allows high-priority threads to meet their deadlines consistently, no matter how many other threads are competing for CPU time.
- Optimized for Freescale architecture — Optimized assembly code to accelerate key real-time portions of the RTOS such as context switching.
- Scheduling — Freescale MQX RTOS provides the developer faster development time by relieving engineers from create or maintain an efficient scheduling system and interrupt handling. It is also significantly useful if one requires the use of multiple communication protocols like USB or TCP/IP.
- Code Reuse — Freescale MQX RTOS provides a framework with a simple API to build and organize the features across Freescales broad portfolio of embedded processors.
- Intuitive API — Writing code for Freescale MQX RTOS is straight forward with a complete API and available reference documentation.
- Fast boot sequence — A fast boot sequence ensures the application is running quickly after the hardware has been reset.
- Simple Message Passing — Messages can be passed either from a system pool or a private pool and can be sent with either an urgent status, or a user defined priority, and can be broadcast or task specific. For maximum flexibility, a receiving task can be operating on either the same CPU as the sending task or on a different CPU within the same system.

7.7.3 RTCS

Real–Time TCP/IP Communication Suite (RTCS) provides IP networking for the Freescale MQX Software Solutions. Freescale MQX RTCS provides a rich assortment of TCP/IP networking application protocols and uses the Freescale MQX RTOS drivers for Ethernet and serial connectivity.

RTCS is implemented in ANSI C, and full source code is provided. It is completely re-entrant and is responsive to the demands of real-time systems. The RTCS supports any number of hardware interfaces and any number of IP addresses on each hardware interface.

This is a summary of featured characteristics:

- Supports TCP, IP, UDP, ARP, ICMP, CIDR, IGMP, and PPP
- Supports application-layer protocols such as DNS resolver only, RPC/XDR, BootP, DHCP, HTTP, FTP, TFTP, Telnet, SNMPv1, and SNMPv2c
- Supports lower-layer protocols such as Ethernet (IEEE 802.3) and PPP (includes CHAP, LCP, PAP, CCP, and IPCP)
- Is compatible with RFC 1122 (Requirements for IPv4 Hosts)
- Is compatible with RFC 1812 (Requirements for IPv4 Routers)
- Provides a Berkeley Socket (BSD) API and supports stream and datagram sockets
- Supports high-performance, re-entrant operation
- Supports multihoming and multiple devices
- Includes configurable network parameters
- Supports dynamically configurable gateways
- Provides network status and diagnostics
- Is written in 100% ANSI C

7.8 Beestack Zigbee protocol stacks

The BeeStack architecture builds on the ZigBee protocol stack. Based on the OSI Seven-Layer model, the ZigBee stack ensures interoperability among networked devices. The physical (PHY), media access control (MAC), and network (NWK) layers create the foundation for the application (APL) layers.

The BeeStack defines additional services to improve the communication between layers of the protocol stack. At the Application Layer, the application support layer (ASL) facilitates information exchange between the Application Support Sub-Layer (APS) and application objects. Finally, ZigBee Device Objects (ZDO), in addition to other manufacturer-designed applications, allow for a wide range of useful tasks applicable to home and industrial automation.

BeeStack uses an IEEE® 802.15.4-compliant MAC/PHY layer that is not part of ZigBee itself. The PHY layer encompasses features specified by IEEE 802.15.4 for packet-based, wireless transport. The MAC sub-layer supports features specific to low-power radio frequency networks.

The NWK layer defines routing, network creation and configuration, and device synchronization. The application framework (AF) supports a rich array of services that define ZigBee functionality. ZigBee Device Objects (ZDO) implement application-level services in all nodes using profiles. A security service provider (SSP) is available to the layers that use encryption (NWK and APS).

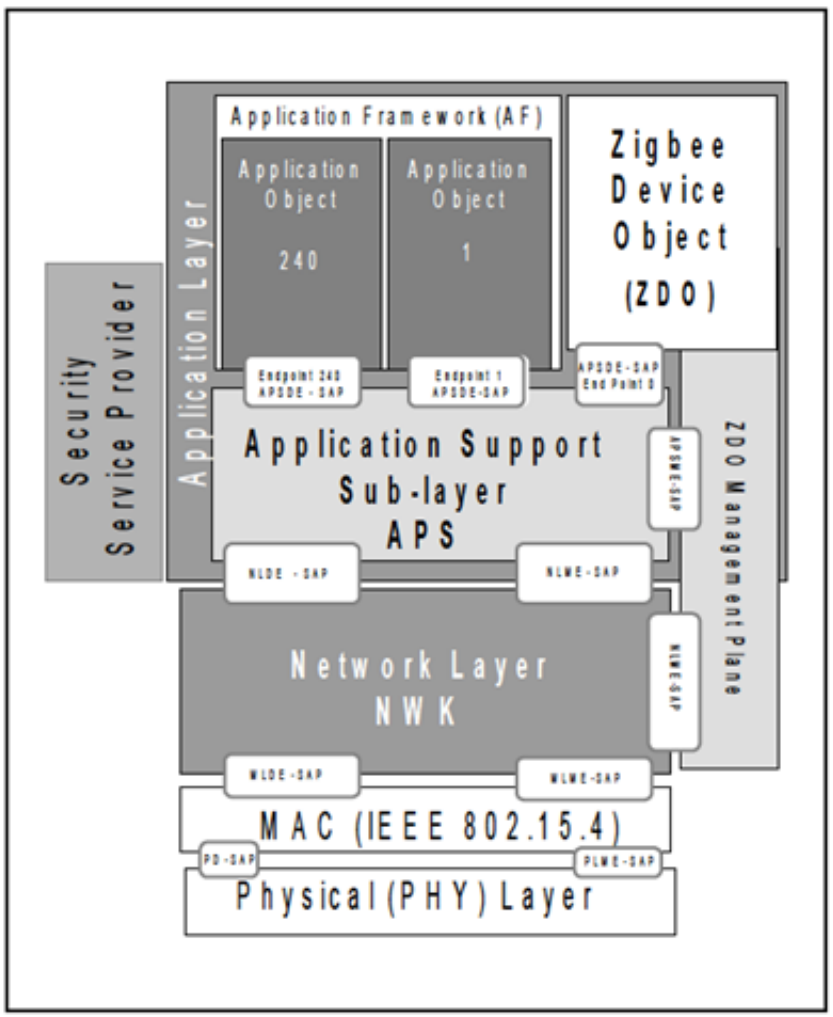

Figure 7-4. Zigbee Layers

7.9 External references

For more information about Freescale MQX RTOS visit:

http://www.freescale.com/mqx

Written documentation about Beestack is available at:
http://www.freescale.com/beekit

Additional details and documentation about Zigbee standards can be found at the official Zigbee Alliance website:
http://www.zigbee.org

7.10 Hardware Description

7.10.1 Hardware implementation

This application has been designed to run on standard Freescale hardware development tools, including a Tower system and various 1322x development kits. The following boards are needed to run this application:

- One Tower system including the following modules to host main application:
  - TWR-K60
  - TWR-WIFI-G1011MI
- Two MC13226-SRB. One is used as Zigbee coordinator and is part of the Home Automation Gateway, the other is used as a Zigbee end device and is part of the smart appliance.
- One MC13226-LPB. This board is used to create another network under the Smart Energy Profile.

7.10.2 Featured products

Two main platforms are used to build the Home Automation Gateway. The main application runs on a K60, a member of the new 32-bit, ARM Cortex-M4 portfolio. Software running in this microcontroller is based on MQX and demanding tasks, like standard IP communications, are performed by this core.

All Zigbee-based communications are accomplished using a secondary platform, the MC13226. This is a Platform-In-Package that includes an IEEE 802.15.4 transceiver and a 32-bit, ARM7 TDMI-based microcontroller. Stack libraries are included in a ROM image to allow applications of Zigbee 2007 Profile 2 (PRO) maximizing the amount of available RAM for the application.

7.10.3 Kinetis Family

Kinetis MCUs represent the most scalable portfolio of ARM Cortex-M4 in the industry. Kinetis features the latest low-power innovations, high performance, and high precision mixed-signal capability. Kinetis MCUs are supported by a market-leading enablement bundle from Freescale and ARM third party ecosystem partners.

All Kinetis families include a powerful array of analog, communication, timing, and control peripherals with the level of feature integration increasing with flash memory size and the number of inputs/outputs. Features common to all Kinetis families include:

- Core:
  - ARM Cortex-M4 Core delivering 1.25 DMIPS/MHz with DSP instructions (floating-point unit available on certain Kinetis families)
  - Up to 32-channel DMA for peripheral and memory servicing with minimal CPU intervention
  - Broad range of performance levels rated at maximum CPU frequencies of 50 MHz, 72 MHz, 100 MHz, 120
  - MHz, and 150 MHz
- Ultra-low power:
  - 10 low power operating modes for optimizing peripheral activity and wake-up times for extended battery life.
  - Low–leakage wake-up unit, low power timer, and low power RTC for additional low power flexibility
  - Industry-leading fast wake-up times
- Memory:
  - Scalable memory footprints from 32 KB flash / 8 KB RAM to 1 MB flash / 128 KB RAM. Independent flash banks enable concurrent code execution and firmware updates
  - Optional 16 KB cache memory for optimizing bus bandwidth and flash execution performance. Offered on K10, K20, and K60 family devices with CPU performance of up to 150 MHz.
  - FlexMemory with up to 512 KB FlexNVM and up to 16 KB FlexRAM. FlexNVM can be partitioned to support additional program flash memory (example, the bootloader), data flash (example storage for large tables), or EEPROM backup. FlexRAM supports EEPROM byte-write/byte-erase operations and dictates the maximum EEPROM size. EEPROM endurance capable of exceeding ten million cycles
  - EEPROM erase/write times an order of magnitude faster than traditional EEPROM

  - Multi-function external bus interface capable of interfacing to external memories, gate-array logic
  - DDR memory controller
  - NAND flash controller

- Mixed-signal analog:
  - Fast, high precision 16-bit ADCs, 12-bit DACs, programmable gain amplifiers, high speed comparators
  - Internal voltage reference. Powerful signal conditioning, conversion and analysis capability with reduced system cost

- Human Machine Interface (HMI):
  - Capacitive Touch Sensing Interface with full low-power support and a minimal current adder when enabled

- Connectivity and communications:
  - UARTs with ISO7816 and IrDA support, I2S, CAN, I2C, and DSPI
  - USB OTG controller

- Reliability, safety, and security:
  - Hardware cyclic redundancy check engine for validating memory contents/ communication data and increased system reliability
  - Independent-clocked COP for protection against code runaway in fail-safe applications
  - External watchdog monitor
  - Secure storage and tamper detect

- Timing and Control:
  - Powerful FlexTimers which support general purpose, PWM, and motor control functions
  - Carrier Modulator Transmitter for IR waveform generation
  - Programmable Interrupt Timer for RTOS task scheduler time base or trigger source for ADC conversion and
  - programmable delay block

- System:
  - 5 V tolerant GPIO with pin interrupt functionality
  - Wide operating voltage range from 1.71 V to 3.6 V with flash programmable down to 1.71 V with fully
  - Functional flash and analog peripherals
  - Ambient operating temperature ranges from -40 °C to 105 °C

The K60 MCU family includes IEEE 1588 Ethernet, full and high-speed USB 2.0 On-The-Go with device charger detect capability, hardware encryption and tamper detection capabilities.

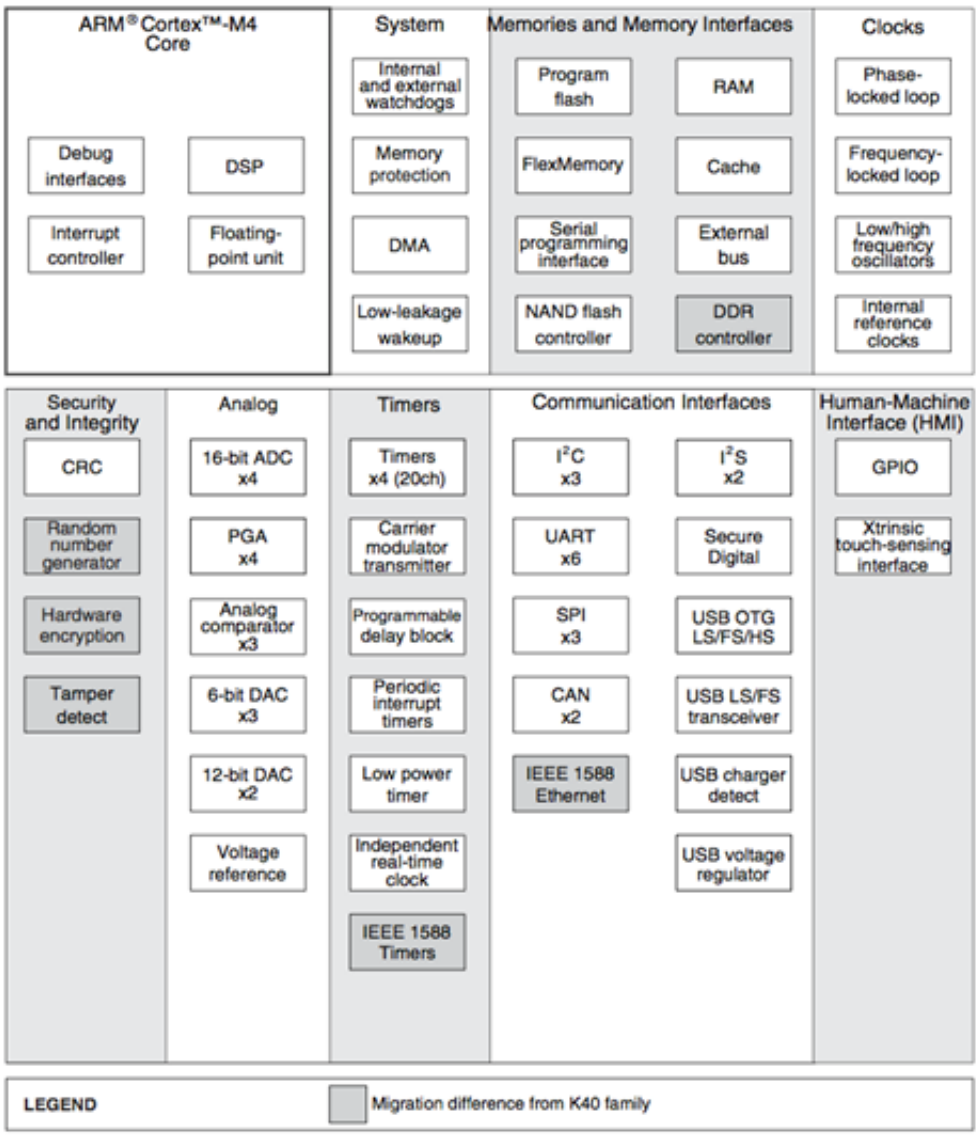

Figure 7-5. K60 Block Diagram

7.10.4 MC1322x

The MC1322x family is Freescale's third-generation ZigBee platform which incorporates a complete, low power, 2.4 GHz radio frequency transceiver, 32-bit ARM7 core based MCU, hardware acceleration for both the IEEE 802.15.4 MAC and AES security, and a full set of MCU peripherals into a 99-pin LGA Platform-in-Package (PiP).

The MC1322x MCU resources offer superior processing power for ZigBee applications. A full 32-bit ARM7TDMI-S core operates up to 26 MHz. A 128 Kbyte FLASH memory is mirrored into a 96 Kbyte RAM for upper stack and applications software. In addition, an 80 Kbyte ROM is available for boot software, standardized IEEE 802.15.4 MAC and communications stack software. A full set of peripherals and Direct Memory Access (DMA) capability for transceiver packet data complement the processor core.

In addition to the best-in-class MCU performance and power, the MC1322x also provides best-in-class power savings. Typical transmit current is 29 mA and typical receive current is 22 mA with the CPU at 2 MHz operation and even lower with the bus stealing enabled. On board power supply regulation is provided for source voltages from 2.0 Vdc to 3.6 Vdc. Numerous low current modes are available to maximize battery life including sleep or restricted performance operation.

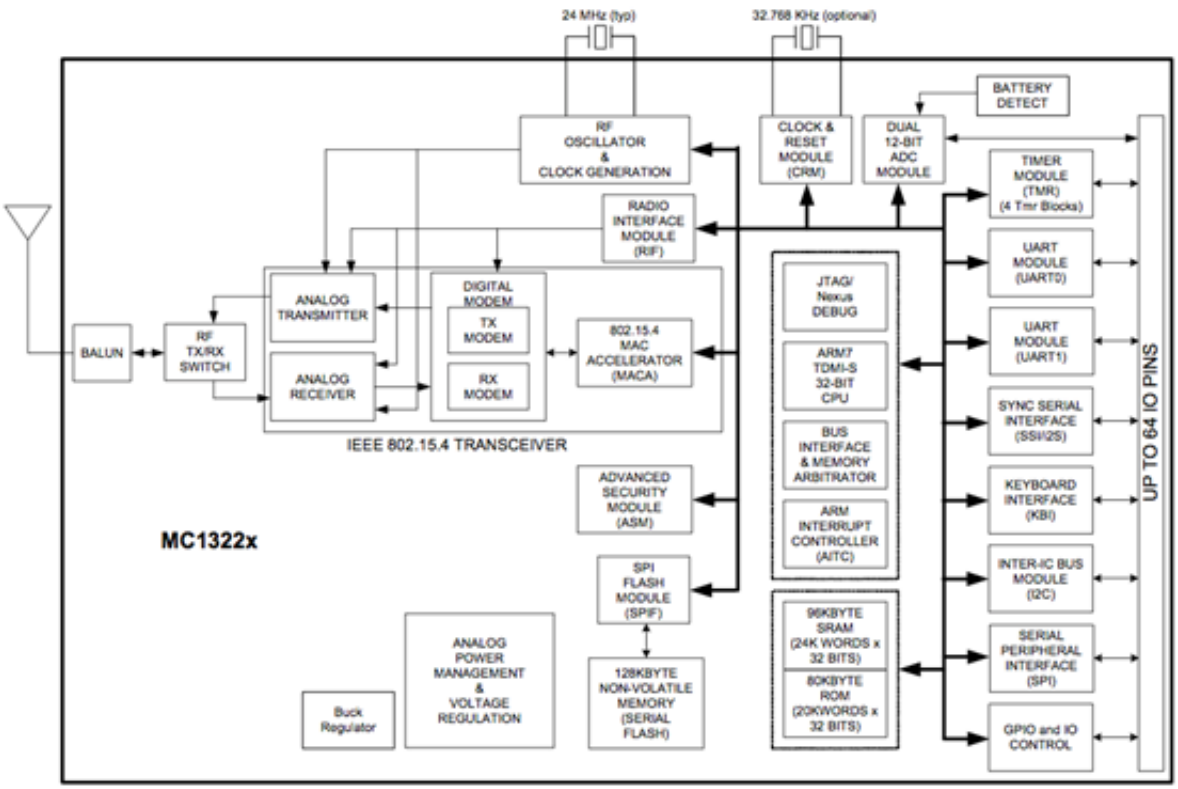

Figure 7-6. MC1322x Block Diagram

7.10.5 Additional hardware features

The Tower system is a set of Freescale modular development boards that allows immediate software development before custom hardware is finished. Interchangeable and reusable modules make it easy to customize a design.

7.11 Firmware Description

7.11.1 System overview

The Home Automation Gateway is an application composed by multiple embedded platforms, including a Kinetis MCU (K60) running the main application and allowing external systems and users to interact via a web-based interface, and an MC13226, which is a Zigbee-capable System-in-Package capable of monitoring and controlling remote smart devices inside a home automation network as well as communicating with an external network to publish and update energy consumption and pricing information.

Taking advantage of this hardware architecture, application concerns have been distributed between the two separate platforms, allowing a clear separation of responsibilities and favoring system extensibility.

7.11.2 Home automation gateway application

The main application running in the K60 MCU is responsible of modeling the local network and publishing remote devices attributes on a web based interface.

Taking advantage of MQX RTOS and its libraries, users are able to interact with remote devices using conventional browsers. This embedded website displays static and dynamic content and is compatible with the most common internet-browsers.

7.11.3 Home automation gateway functionality

The main application is responsible for the following functions:

1. Web-based interface — A website that allows user interaction to configure and control devices connected to the local automation network. All sections follow a dynamic architecture displaying the actual status of the network. A section for each device is created after connecting to the Zigbee network. From this section, device attributes can be configured and controlled individually. Static content is available using the TFS library.
2. CGI-like functionality — All dynamic content is provided to the web interface using a collection of generic functions that adapt to each kind of endpoint and attribute supported by the system.

3. HA Gateway — On the application level, a model of the local Zigbee network is created. Network creation and permission to join; among other operations, are also managed by this component. All information provided to the interface layer is available through this collection of structures and its interfaces.
4. Networking — The lower layer of this embedded application performs communication with external systems. A proprietary serial protocol is used to communicate with the Zigbee platform and the RTCS MQX library processes IP-based requests.

7.11.4 Main application architecture

The software architecture of this embedded application is component oriented. This design philosophy is useful when systems need to be scalable and its functionality extended over time

From an information viewpoint, it is easy to identify the flow of data from the system to the user, and how it is transformed by the in-between function.

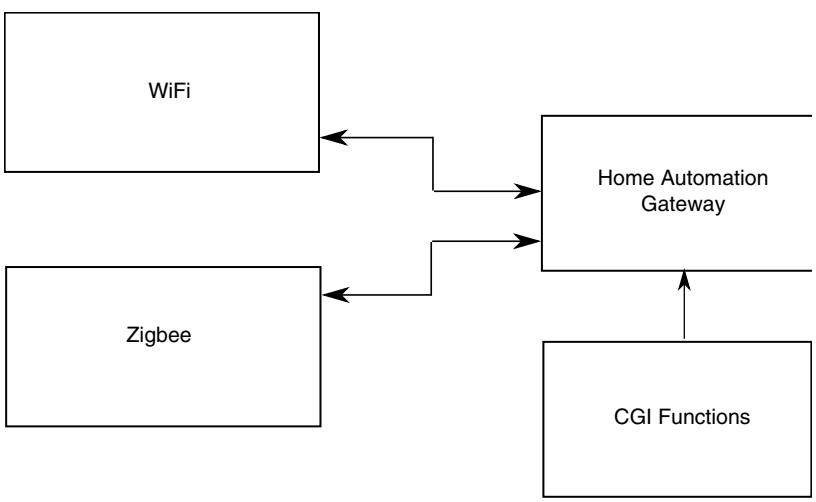

Figure 7-7. Main application components

By classifying all functional blocks into three main layers, where the higher layers represent more abstract formats of data (user friendly formats), it is also possible to identify levels of responsibility and exposed interfaces on both extremes, high level and low level interfaces.

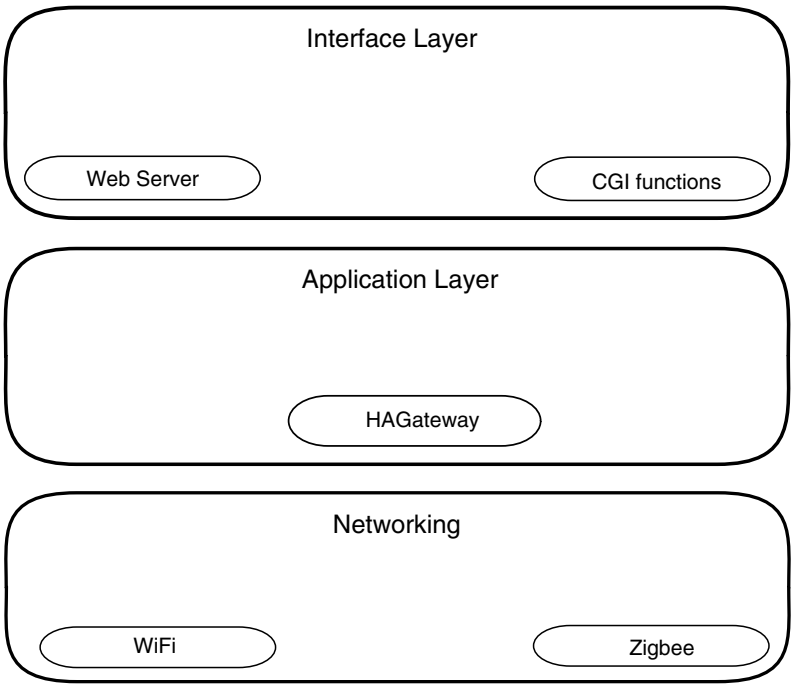

Figure 7-8. Main application layers

7.11.5 Home automation network software model

The core of this application is a software model of the local automation network. This software model is a collection of software structures named nodes that represent each device connected to the network. In this model, each node contains its network information and other attributes needed to send information and commands.

When a remote device connects to the Zigbee network, a new node instance is initialized with its information, including Zigbee-level capabilities and other descriptors.

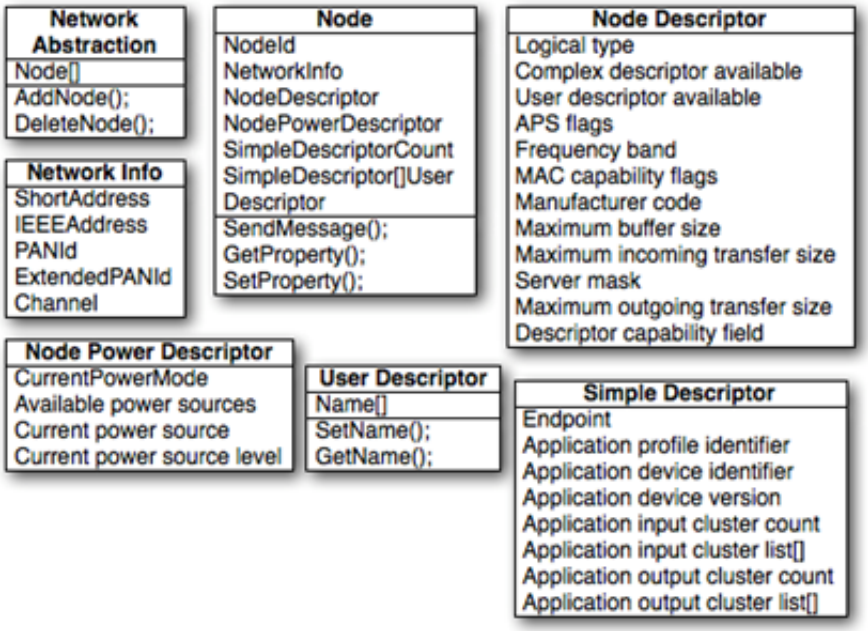

Figure 7-9. Network abstraction structures

7.11.6 Web interface

As mentioned before, a web-based interface is used to interact with devices connected to the automation network. Content in this website is mostly dynamically generated, however, three main sections can be identified:

Home — General information about the home automation network and the devices in it is available in this section.

Energy — When a separate network with smart energy profile is near, this page displays a graph that displaces current prices on every time of the day. Also, a table with energy consumption information for every device in the local network is displayed.

Device[n] — A section for each device is available by clicking the tab with its previously defined User Descriptor. Detailed information for a determined endpoint and its attributes is accessible from this page.

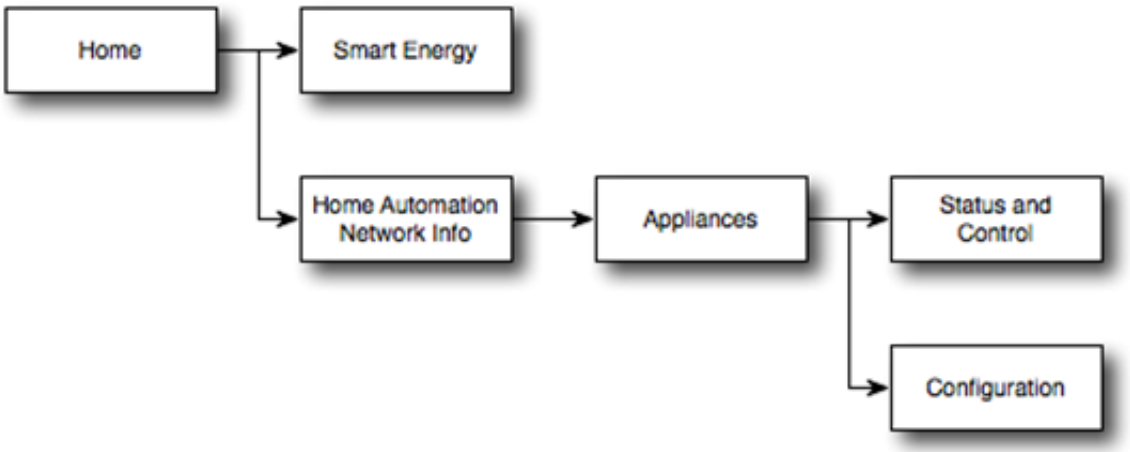

Figure 7-10. Embedded website structure

7.11.7 Dynamic web content and functionality

Dynamic functionality is implemented using CGI-like functions. The CGI Link table is defined as follows:

```
const HTTPD_CGI_LINK_STRUCT HAGateway_cgi_lnk_tbl[] =
{
  {"nwkStatus",      cgi_nwkStatus},
  {"deviceList",     cgi_deviceList},
  {"epList",         cgi_epList},
  {"devConsList",    cgi_deviceConsList},
  {"formNetwork",    cgi_formNetwork},
  {"toggleJoin",     cgi_toggleJoin},
  {"upCmd17",        cgi_upCmdEp17},
  {"upCmd18",        cgi_upCmdEp18},
  {"downCmd17",      cgi_downCmdEp17},
  {"downCmd18",      cgi_downCmdEp18},
  {"newOpMode",      cgi_newOpModeEp20},
  {"setControlEp21",cgi_setControlEp21},
  {"changeUsrDesc", cgi_usrDescConfig},
  {"selectDevice",   cgi_selectDevice},
  {"energy",         cgi_energyTable},
  { 0, 0 }    // DO NOT REMOVE - last item - end of table
};
```

A conceptual description for each function is provided below:

nwkStatus — Returns the information regarding IP_ADDRESS, Network Formation, and Permit Join status.

deviceList — Basic information for each device that has joined the network is available through this function.

epList — Returns the state of the endpoints the node contains in the Device tab.

devConsList — Returns the table of devices with their current power consumption in the Energy tab.

formNetwork — Network creation takes place after calling this function.

toggleJoin — Sends a command to the Zigbee Coordinator to toggle the Permit Join flag.

upCmd17 — Sends a levelUp command to EndPoint 17 of the device.

upCmd18 — Sends a levelUp command to EndPoint 18 of the selected device.

downCmd17 — Sends a levelDown command to EndPoint 17 of the selected device.

downCmd18 — Sends a levelDown command to EndPoint 18 of the selected device.

newOpMode — Sends a presentValue command to EndPoint 20 of the selected device.

setControlEp21 — Sends a presentValue command to EndPoint 21 of the selected device.

changeUserDesc — The user descriptor property in a remote device is updated.

selectDevic — When selected, the Device tab verifies the nodes information to establish what device is selected.

energy — Returns values stored in the pricing table.

7.12 Zigbee Applications

7.12.1 Zigbee home automation projects

The Home Automation Gateway project includes three different Zigbee projects, each project for each kind of device:

- Zigbee Coordinator — This project is actually part of the home automation gateway device and is responsible for creating and managing the Zigbee home automation network. This device also performs inter-PAN communication.
- Zigbee End Device — This project is actually the Zigbee component of a smart appliance, for this design, a refrigerator and its most common functionality is implemented.
- Zigbee Smart Energy — This project represents a remote device in a separate network, normally a device which is part of the smart energy network created by the electric energy service provider.

7.12.2 Zigbee coordinator

The Zigbee Coordinator is implemented as an ESP Device and has two profiles implemented, so inter-PAN communication can be used. Each cluster that is part of a remote device must be implemented in this device, at least as a client cluster:

Home automation profile
- Output clusters:
  - Level
  - Temperature measurement
  - Multi–state value

Smart Energy Profile
- Output clusters:
  - Metering
  - Interpan communication
- Input clusters
  - Price
  - Interpan communication

7.12.3 Zigbee end device

The Zigbee end device is implemented as a smart appliance device and has two profiles implemented:

Home automation profile.
- Input clusters
  - Level
  - Temperature measurement
  - Multi–state value
- Smart energy profile
  - Ouput clusters
    - Price
  - Input clusters
    - Metering

7.12.4 Zigbee smart energy device

The Zigbee smart energy device should be part of the energy provider's network. Using the inter-PAN communication cluster, the home automation gateway asks for the scheduled prices.

The price cluster receives a Get Scheduled Prices Command and generates a Publish Price Command. The information used by the HAGateway contained in the Publish Price Command is:

- Start Time
- Duration
- Price

For more information in the complete command's payload refer to document ZigBee Smart Energy Profile Specification 075356r15 sub-clause D.4.2.3.2.

7.12.5 Summary of clusters and endpoints

As mentioned before, to communicate with a remote device using Zigbee, the same cluster must be implemented on both sides. In the diagram below, an "S" represents a server cluster, while a "C" represents a client cluster.

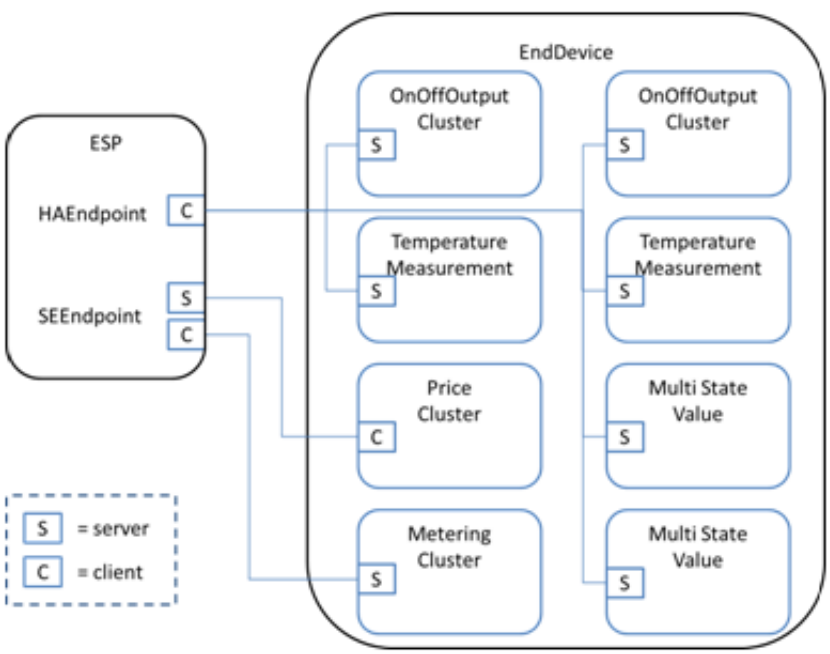

Figure 7-11. Client and server clusters

Register	Type	Description	Cluster	Attribute(hex)
Freezer Temp	Uint16	Temperature measurement in °C multiplied by 100.	Temperature	Temperature
Refrigerator Temp	Uint16	Temperature measurement in °C multiplied by 100.	Temperature	Temperature
Control	5-bit bitmap	• Operation mode (0) — 0 by time (no frost sensor), 1 by control (frost sensor). • Sensor Control (1) — 0 sensor disable, 1 sensor enable. • Resistance Control (2) — 0 turn off resistance, 1 turn on resistance. • Compressor Control (3) — 0 turn off compressor, 1 turn on compressor. • Alarm Status (4) — 0 Alarm off, 1 Alarm on.	Multistate Value	presentValue
status	4-bit bitmap	• Sensor Status (0) — 0 Frost not detected, 1 Frost detected • Defrost Status (1) — 0 Defrost incomplete, 1 Defrost completed. • Resistance Status (2) — 0 Resistance off, 1 Resistance on. • Compressor Status (3) — 0 Compressor off, 1 Compressor on.	Multistate Value	presentValue
Temp Level	Uint8[2]	Low part of the register is used to save refrigerator temperature level from 0-9. Initial level is 8. High part of the register is used to save freezer level temperature from 0-9. Initial level is 8.	Level Level	level Value (0-A) levelValue (0-A)

Table continues on the next page...

Register	Type	Description	Cluster	Attribute(hex)
mode	Uint8	• 000 — Normal operation mode. • 001 — Turbo operation mode. • 010 — Vacation operation mode. • 011 — Test operation mode. • 100 — Remote operation mode. • 101 — Lock mode.	Multistate Value	presentValue (0-5)
General Status	2-bit bitmap	• Alarm enabler (0) — 0 Disable, 1 Enable. • Open door Status (1) — 0 close, 1 open.		
power	Uint16	Power consumption measurement in Kwh in format where Power is multiplied by 100. (16-bit)	SimpleMeter	CurrValue

All EndPoints implemented in the Zigbee end device are linked to the registers from the control board as follows:

Endpoint 15 — Refrigerator temperature
Endpoint 16 — Freezer temperature
Endpoint 17 — Refrigerator level
Endpoint 17 — Freezer level
Endpoint 19 — Power consumption
Endpoint 20 — Operation Mode
Endpoint 21 — Control
Endpoint 22 — Status (includes Status and General Status registers)

More information about this implementation is outside of the scope of this document. Separate documentation is available.

7.13 HAGateway to Zigbee serial communication protocol

To enable communication between the main application and the Zigbee platform, a proprietary protocol has been designed and implemented. This protocol describes the transactions between the Network controller and the Home Automation Gateway.

The protocol sends and receives configuration and controls messages to the Home Automation Network and reports the status of the Network to the application level components.

Table 7-2. HAGateway-to-Zigbee packet structure

Header					Element
Packet Type	Command	Packet Id	Response Id	Size	Payload

Where every header field is 1 byte long:

Packet Type — To differentiate between ZDO_Management or APP_Management messages.

Command — Values in HAGateway.h to manage the different behaviors for the HAGateway.

Packet Id — A serialized identifier to send a response if requested by the command.

Response Id — The field with the Packet Id that requested the response message.

Size — The size of the message payload, (value 0 – 64)

The Message field has a size from 0 bytes to 64 bytes.

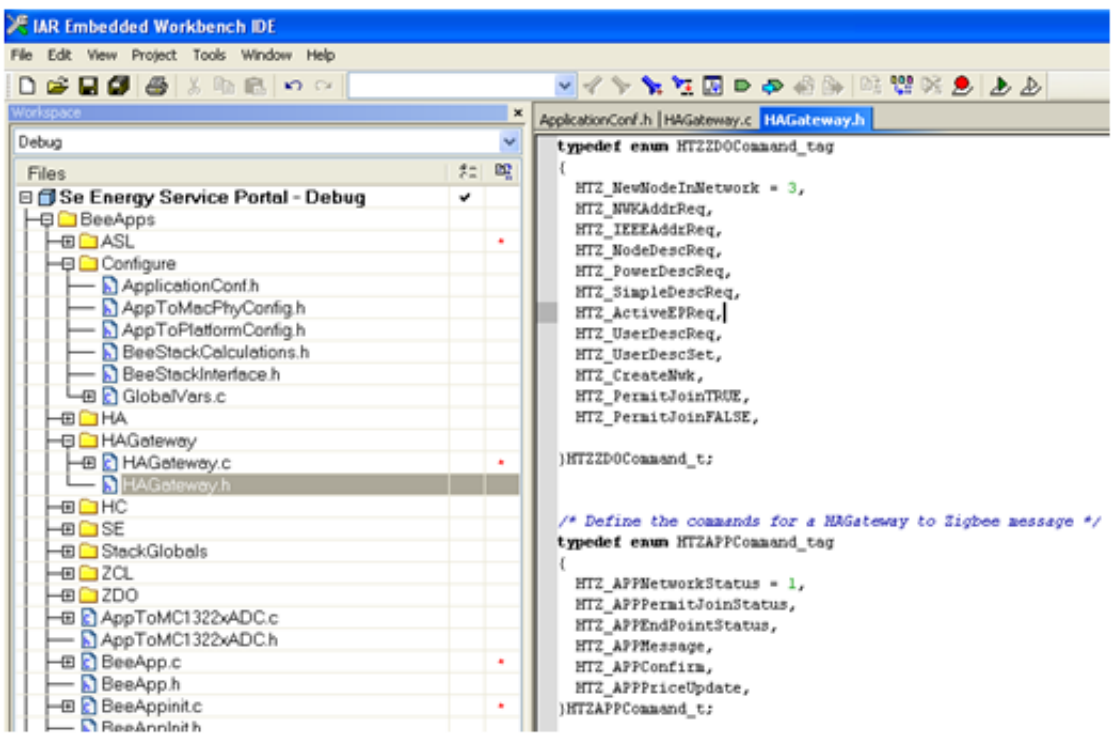

Figure 7-12. HAGateway packet structure definition

Chapter 8
Home Automation User's Guide

8.1 System setup

Some configuration needs to be performed before using this application. This section will guide you through this process, from configuring software to programming and connecting hardware boards.

8.1.1 Configuring the software

1. To recompile the MQX core, the library projects have to be imported (opened) into a CodeWarrior 10.1 workspace.

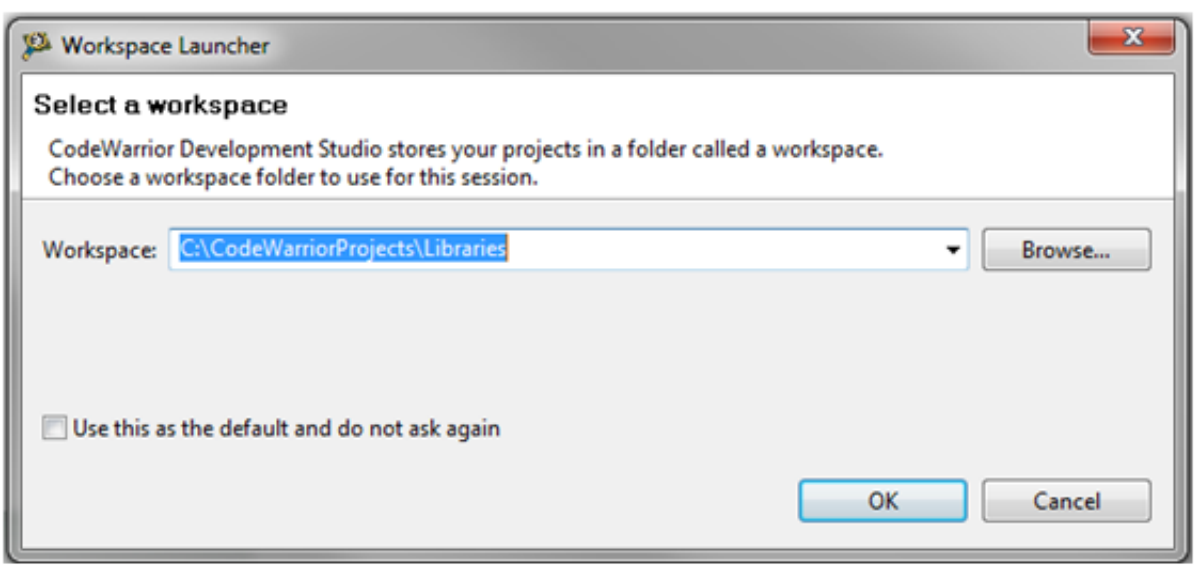

Figure 8-1. Opening MQX libraries

2. If you do not have a previous workspace with the library projects, select the menu File/Import/General/Existing Projects into a workspace and navigate to the MQX 3.7 installation directory. Select and import the following projects:
   - BSP project — <mqx_path>\mqx\build\cw10\bsp_twrk60n512_library

- PSP project — <mqx_path>\mqx\build\cw10\psp_twrk60n512_library
- RTCS project — <mqx_path>\rtcs\build\cw10\rtcs_twrk60n512_library

CAUTION

The Copy Projects into workspace check box in the file importer must be unchecked. Accidentally leaving this option enabled corrupts the project and the libraries will not be able to build.

3. Configure the library with the parameters below:
   - Open user_config.h:
     1. Make sure BSPCFG_ENABLE_GS which enables GainSpan Wi-Fi Driver Support is 1.
     2. Make sure the value of BSPCFG_ENABLE_HAGateway, which enables the HAGateway application is 1
     3. Change:

```
#define BSPCFG_ENABLE_TTYD        1
#define BSPCFG_ENABLE_ITTYD       0
   To
#if BSPCFG_ENABLE_HAGateway
  /* HAGateway UART interface */
    #define BSPCFG_ENABLE_TTYD      0
    #define BSPCFG_ENABLE_ITTYD     1
#else
    #define BSPCFG_ENABLE_TTYD      1
    #define BSPCFG_ENABLE_ITTYD     0
#endif
```

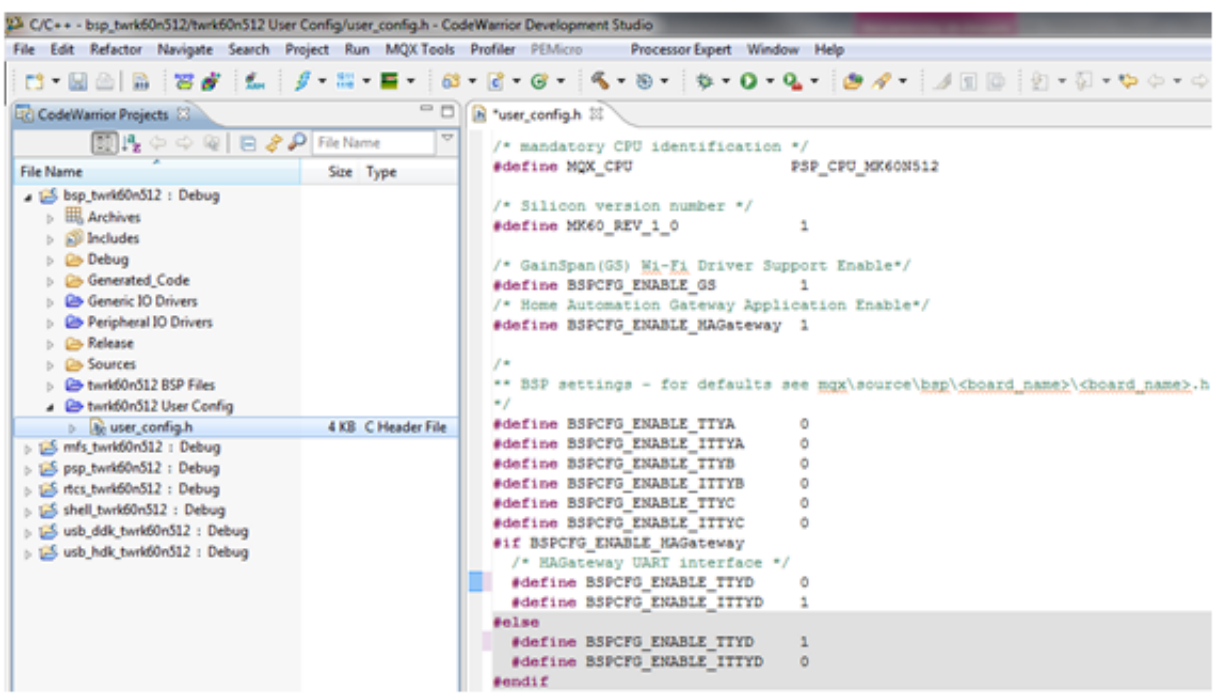

Figure 8-2. Configuring macro definitions

4. Save and build all three libraries

5. Select menu File/Switch Workspace/other.
6. Open the HAGateway Workspace in CW10.1.
7. Configure the Wi-Fi parameters
   - Open Config.h
     1. HTTP Server is setup as Static IP
        a. Modify ENET_IPADDR so IP address is the same sub-net as AP
     2. DEMOCFG_USE_WIFI defines the use of Wi-Fi interface.
     3. Modify #define DEMOCFG_SSID to specify AP.
     4. Modify #define DEMOCFG_PASSPHRASE to specify password.
8. Configure HAGateway app
   - Open HAGateway.h.
   - Modify #define HAGatewayNumberOfNodes to indicate the number of ZED's to support in tables.
9. Save the files and Build Projects
10. To configure the Zigbee Projects open two windows of the IAR Embedded Workbench IDE.
11. Select menu File/Open/Workspace

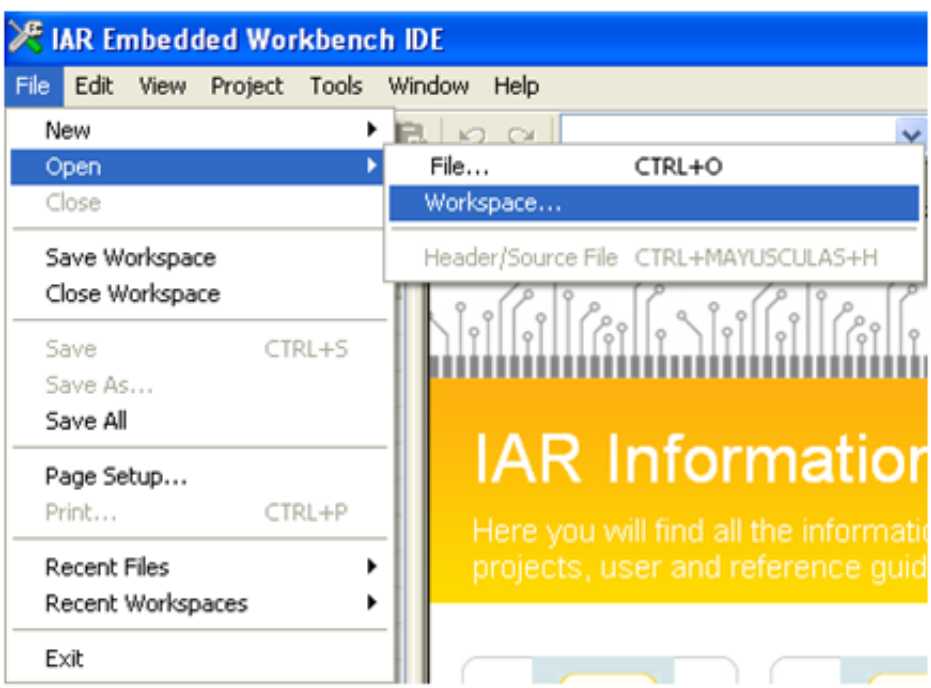

Figure 8-3. Open an IAR Workspace

12. Browse to the location of the project and select the .eww file and click open.
13. Repeat steps 11 and 12 in a new IAR Embedded Workbench IDE window to open the other project.
14. After the projects are open, go to BeeApps/Configure/ApplicationConf.h.

15. Modify #define mDefaultValueOfChannel_c to specify the desired Home Automation Network channel. It must be the same in the Zigbee Coordinator and Zigbee End Device projects.

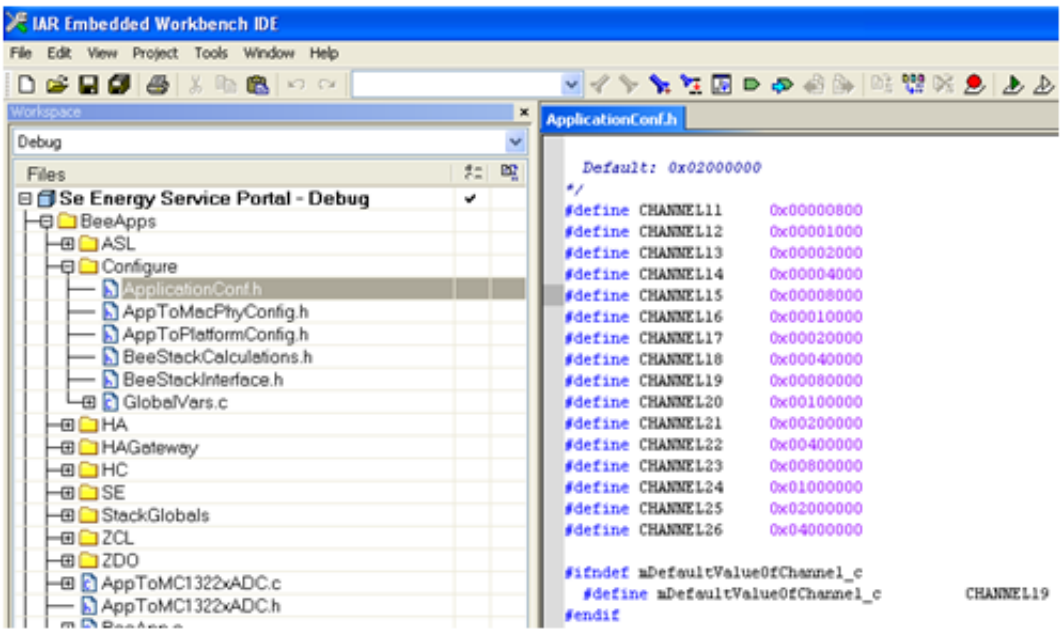

Figure 8-4. Zigbee channel configuration

16. Modify #define mDefaultValueOfPanId_c to specify the network's PAN Id. It must be the same in the Zigbee Coordinator and Zigbee End Device projects.
17. Modify #define mDefaultValueOfExtendedAddress_c to specify the device's IEEE Address. It must be different in the Zigbee Coordinator and Zigbee End Device projects.

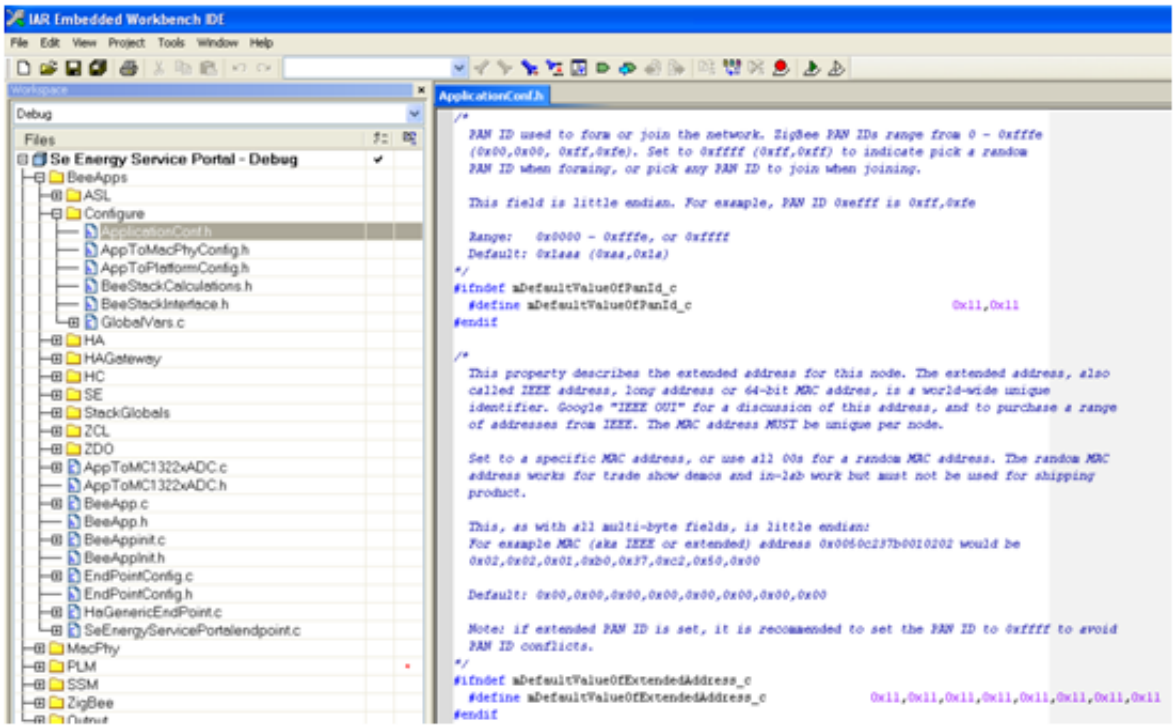

Figure 8-5. Configure IEEE Address

- For the Zigbee Coordinator project skip to step 21

18. Open BeeApps/StackGlobals/beeStack_Globals.c.
19. Modify the variable declaration gUserDesc.
20. The string must be a 16 byte length string, and all the unused bytes must be set to ‘ ‘ or 0x20 (space character). The new descriptor string size should only count the used bytes.

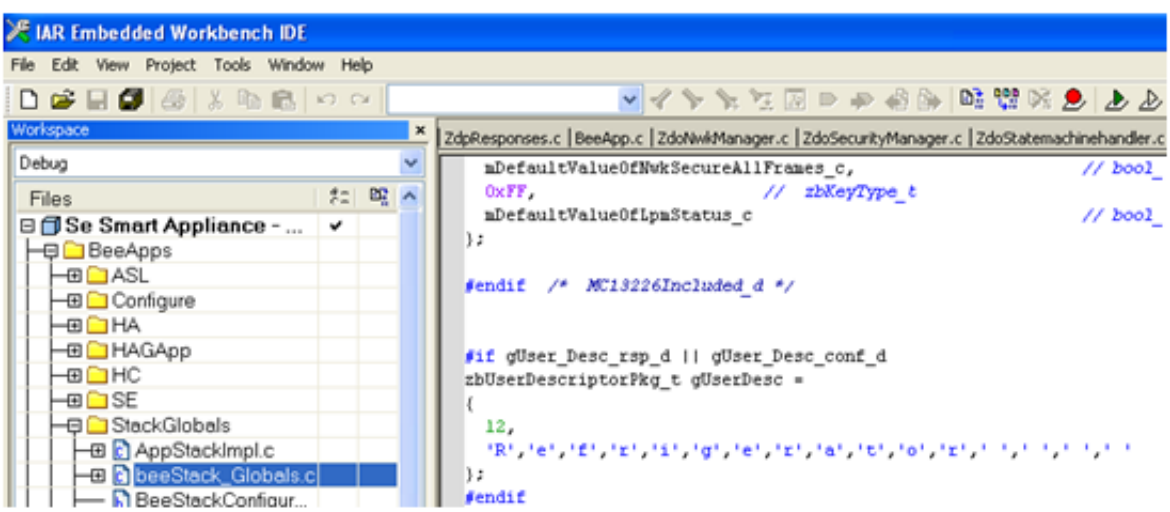

Figure 8-6. Configuring User Descriptor

21. Clean the project’s object code and rebuild.

8.1.2 Programming the Hardware

1. Connect one end of the USB cable to the PC and the other end to the Power/OSJTAG mini-B connector on the TWRK60N512 module.
2. Make sure the OSBDM driver is installed
3. Click on the HAGateway project, then click the Debug button, and enter Debug Configurations

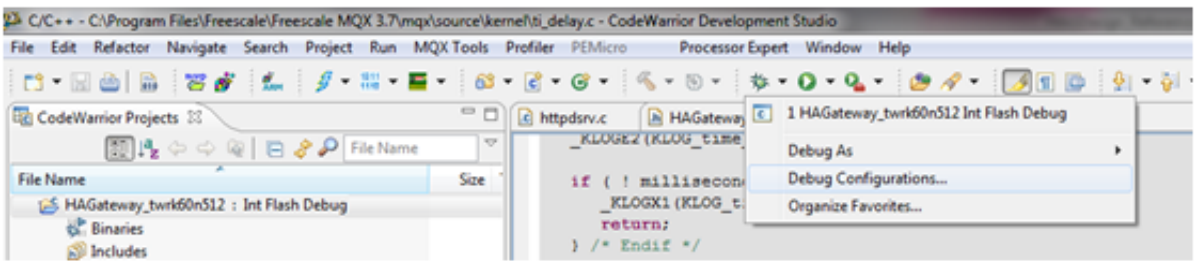

Figure 8-7. Configuring Codewarrior debugger

4. In Remote System select TWRK60N512 OSBDM/OSJTAG

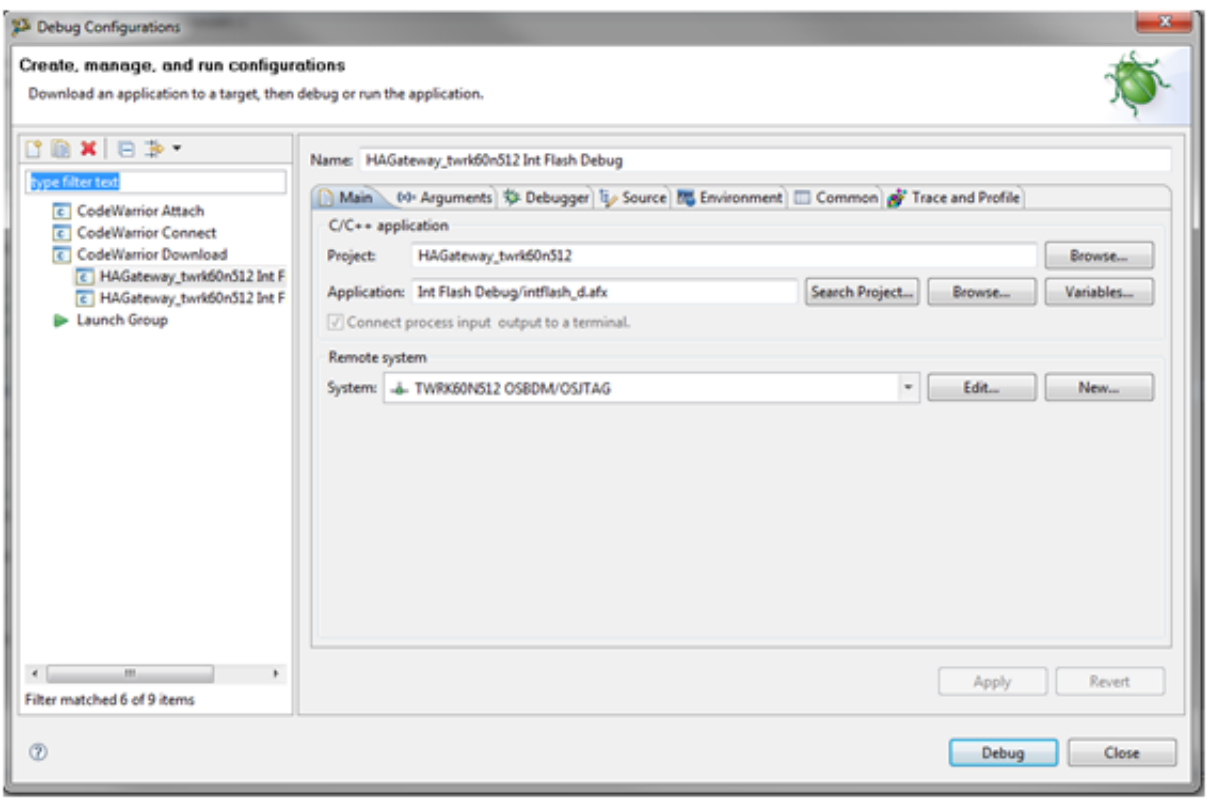

Figure 8-8. Selecting OSBDM/OSJTAG

5. Click Apply, then click Debug. The Program will now begin to download to the board.

6. In IAR Embedded Workbench IDE right-click in the project name and click on options.

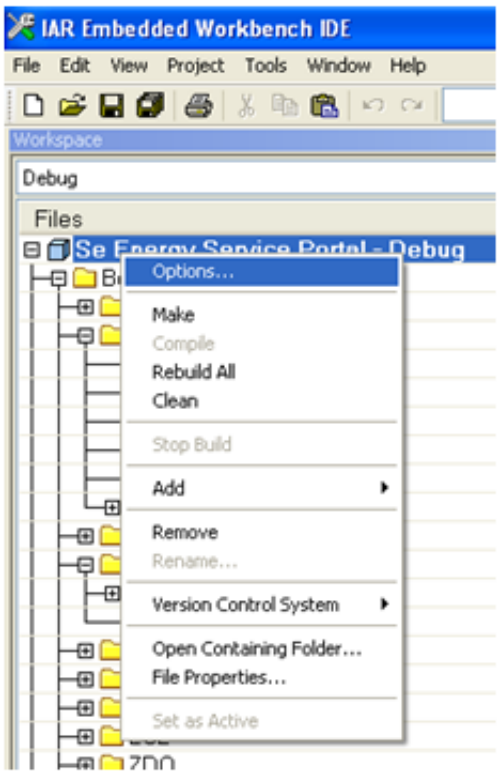

Figure 8-9. Configuring IAR hardware connections

7. Click on category Debugger, select J-Link/J-Trace in the Driver field and click OK.

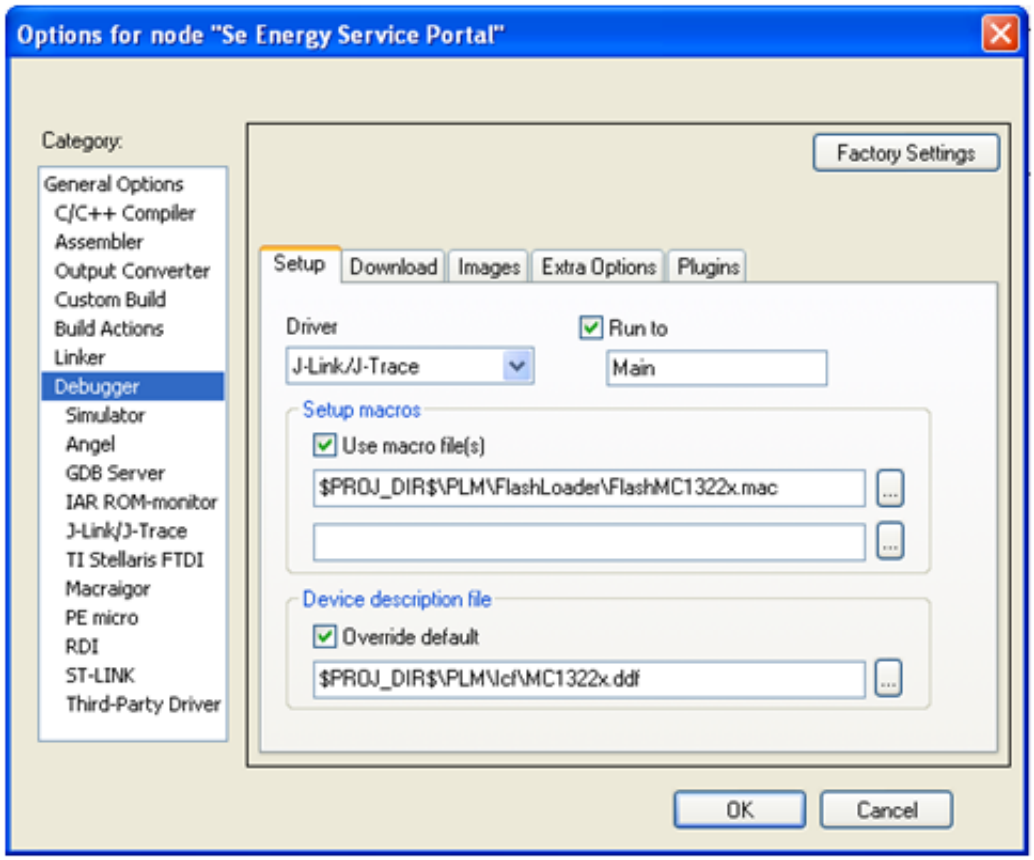

Figure 8-10. Selecting J-Link/J-Trace

8. Then click on the Download and Debug button.

8.1.3 Hardware connections

1. Insert the TWR-K60N512 cpu board, the primary side into the Functional Elevator connector
2. TWR-WIFIG1011MI board, the primary side into the Functional Elevator connector
3. There is no need to install a TWR-ELEV DUMMY, however it is strongly recommended to do so, it provides mechanical support.
4. On the TWR-WIFIG1011MI board configure the jumpers/switches as follows: J3:2-3 and J6:1-2, SW2 to the DOWN position (toward RUN), and SW1 in the DOWN position (toward TWR PWR).

Figure 8-11. Gainspan WiFi tower board

5. In the Functional Elevator 700-26006 — A SIDE EXPANSION PORT J8 connect Pin 65 (Ground) to the MC13226-SRB board (used as coordinator) I/O connector Pin 4.
6. In the Functional Elevator 700-26006 — A SIDE EXPANSION PORT J8 connect Pin 43 (UART3_RX) to the MC13226-SRB board (used as coordinator) I/O Connector Pin 17.
7. In the Functional Elevator 700-26006 — A SIDE EXPANSION PORT J8 connect Pin 44 (UART#_TX) to the MC13226-SRB board (used as coordinator) I/O Connector Pin 18.

8.2 Starting the system

1. Power the MC13226-SRB board (used as the coordinator). Turn the Off/On switch to the ON position (right).
2. At this point the LED PWR should be lit and the LED called LED1 must be lit.
3. Connect power via the USB A to mini B to Functional Elevator 700-26006: J5 slide its power switch SW1 to the ON position (UP).
4. At this point the following LED's should be lit TWR-WIFIG1011MI:D1 and D4, and TWR-K60N512: D21, D14, D15, D16, and D17.

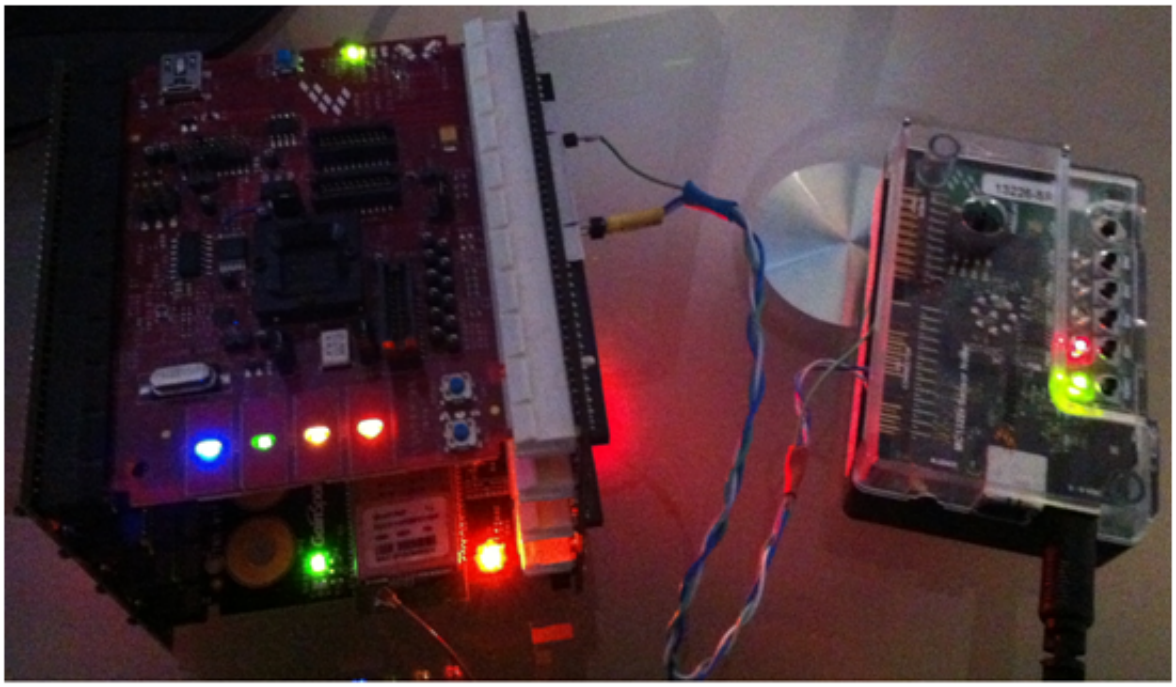

Figure 8-12. Home automation gateway hardware connections

8.3 Using the system

1. Connect the host PC to the router (wireless)
2. Using a web browser, access http://192.168.1.3 .
3. Click on the Form Network button to start the Home Automation Network.
4. All LEDs in the MC13226-SRB board will follow a sequence and then stop.

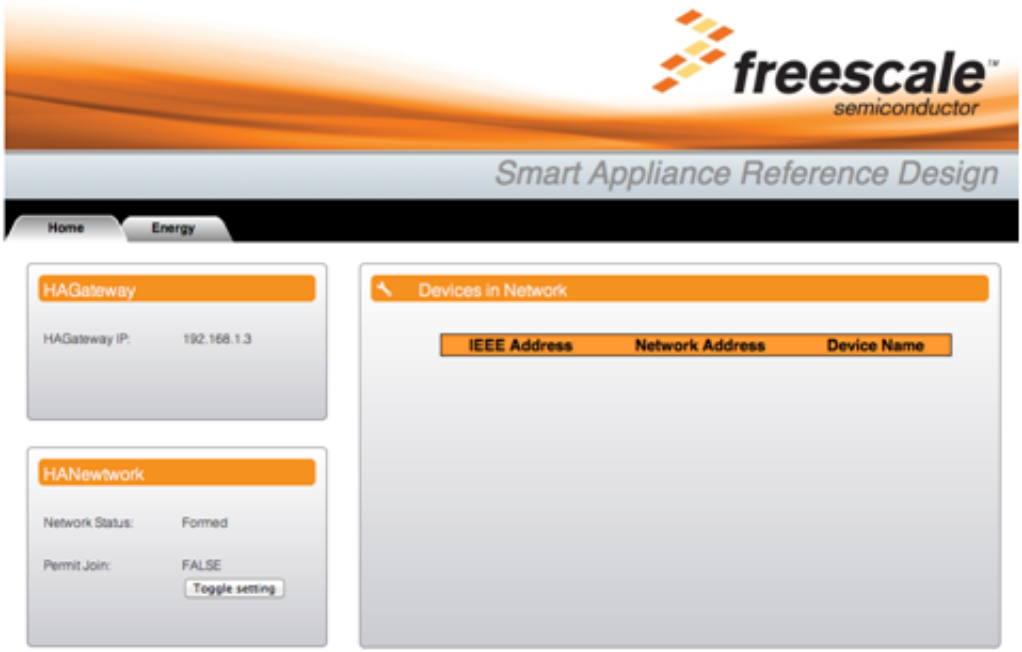

Figure 8-13. HAGateway Home section

5. When no device is connected to the network, only two tabs Home and Energy are visible and you will see an empty device table.
6. You can switch to the Energy tab

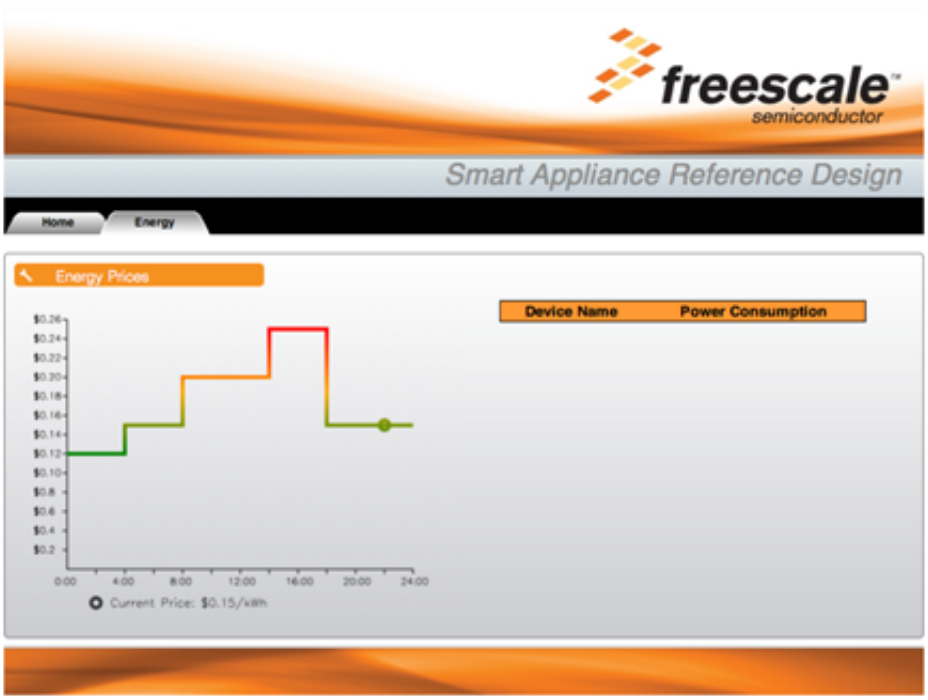

Figure 8-14. Energy tab

7. In case there are Energy prices information a 24 hour graph is visible.

8. When a device is joined to the network, the basic information in the Home tab in the device table is visible. If the Permit Join field is FALSE click the Toggle setting button

Figure 8-15. Home screen with one connected device (NWK ADDRESS 0x0001)

9. A tab is then created with the name of the new device.

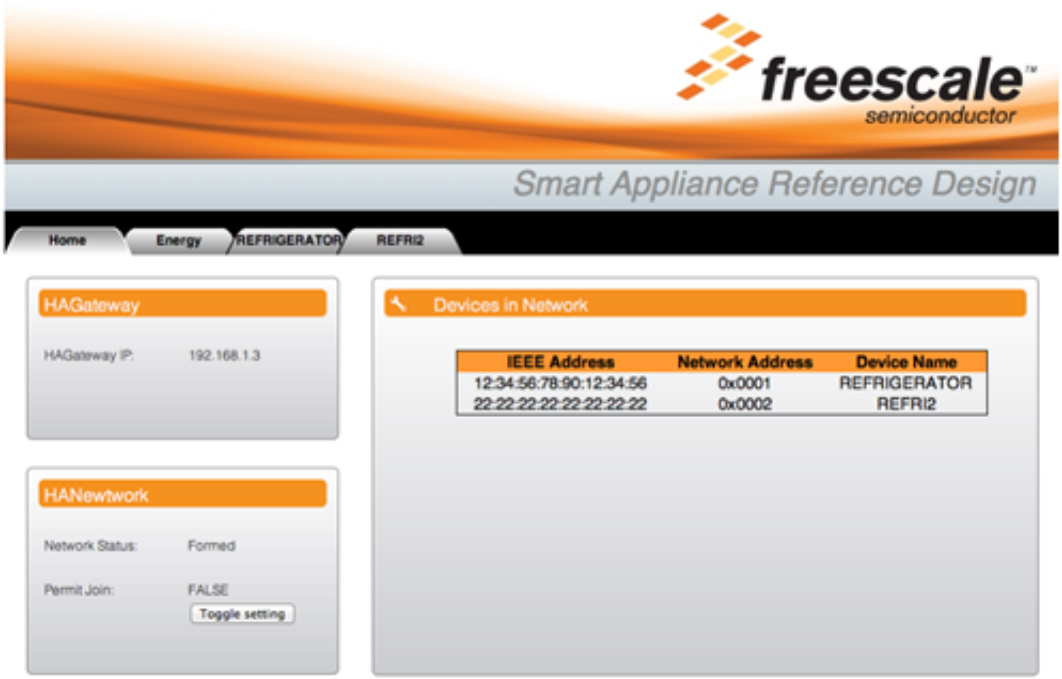

Figure 8-16. Home Screen with two connected devices (0x0001 and 0x0002)

10. In the Energy tab now we have a device in the Power Consumption table

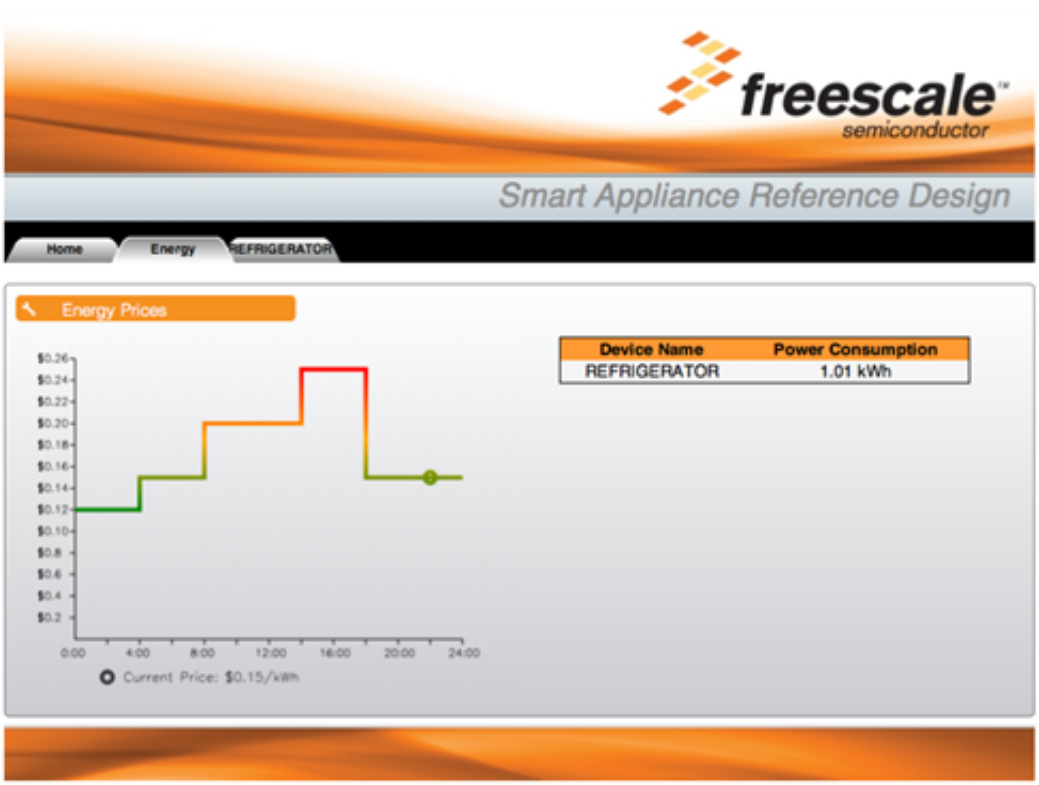

Figure 8-17. Energy section with one connected device

11. In the device tab the specific information is displayed.

Figure 8-18. Device section

12. The user controls are the Temperature Level field, Mode field, and the five checkboxes on the right, these checkboxes do not save their last state.

Appendix A
Database General Definition

Table A-1. General Control Functions Register

Functions	Register number	Modifiable bits high	Modifiable bits high	Description
*Freezer compartment temperature read	0	0000 0000	0000 0000	16 bits register to store temperature measurement in °C in format where temperature is multiplied by 100.
*Refrigerator compartment temperature read	1	0000 0000	0000 0000	16 bits register to store temperature measurement in °C in format where temperature is multiplied by 100.
*Refrigerator control	2	xxxx xxxx	xxx0 0000	Operation control register for refrigerator. Control • Operation mode (bit 0): 0 by time (without frost sensor), 1 by control (with frost sensor). • Sensor Control (bit 1): 0 sensor disable, 1 sensor enable. • Resistance Control (bit 2): 0 turn off resistance, 1 turn on resistance. • Compressor Control (bit 3): 0 turn off compressor, 1 turn on compressor. • Alarm Status (bit 4): 0 Alarm off, 1 Alarm on.

Table continues on the next page...

Table A-1. General Control Functions Register (continued)

Functions	Register number	Modifiable bits high	Modifiable bits high	Description
*Refrigerator status	3	xxxx xxxx	xxxx 0010	Operation mode status register for refrigerator. Status • Sensor Status (bit 0): 0 Frost not detected, 1 Frost detected. • Defrost Status (bit 1): 0 Defrost incomplete, 1 Defrost completed. • Resistance Status (bit 2): 0 Resistance off, 1 Resistance on. • Compressor Status (bit 3):0 Compressor off, 1 Compressor on.
*Temperature level	4	xxxx 1000	xxxx 1000	Low part of the register is used to save refrigerator temperature level from 0-9. Initial level is 8. High part of the register is used to save freezer level temperature from 0-9. Initial level is 8.
*General operation mode	5	xxxx xxxx	xxxx x000	Operation modes register for the entire system. • 000: Normal operation mode. • 001: Turbo operation mode. • 010: Vacation operation mode. • 011: Test operation mode. • 100: Remote operation mode. • 101:Lock mode.
*General open door and alarm status	6	xxxx xxxx	xxxx xx00	Open door alarm enable and status register. • Operation enabler (bit 0): 0 Alarm disable, 1 Alarm enable. • Open door Status (bit 1): 0 close, 1 open.

Table continues on the next page...

Table A-1. General Control Functions Register (continued)

Functions	Register number	Modifiable bits high	Modifiable bits high	Description
Power consumption	7	0000 0000	0000 0000	16 bits register to store Power consumption measurement in Kwh in format where Power is multiplied by 100.
Date and time (seconds)	8	0011 0000	0011 0000	System time register that contains 2 ASCII characters for seconds tens and units. High register part are tens and low part units. Initial value 00.
Date and time (minutes)	9	0011 0000	0011 0000	System time register that contains 2 ASCII characters for minutes tens and units. High register part are tens and low part units. Initial value 00.
Date and time (hours)	10	0011 0001	0011 0010	System time register that contains 2 ASCII characters for hours tens and units. High register part are tens and low part units. Initial value 12.
Date and time (day)	11	0011 0000	0011 0001	System date register that contains 2 ASCII characters for day on tens and units. High register part are tens and low part units. Initial value 01.
Date and time (month)	12	0011 0000	0011 0001	System date register that contains 2 ASCII characters for month on tens and units. High register part are tens and low part units. Initial value 01.
Date and time (year)	13	0011 0001	0011 0000	System date register that contains 2 ASCII characters for year on tens and units. High register part are tens and low part units. Initial value 10.
Time status	14	xxxx xxxx	xxxx xxx0	General time status register: • Time status(bit 0): 0 am, 1 pm.

Reference table

x	Bit has no importance (unused)
0	Bit has an initial condition of logic zero (used bit)
1	Bit has an initial condition of logic one(used bit)
–	Free memory space to future functions implementation
*	Fundamental for control implementation

Appendix B
Data Sheet Segment LCD

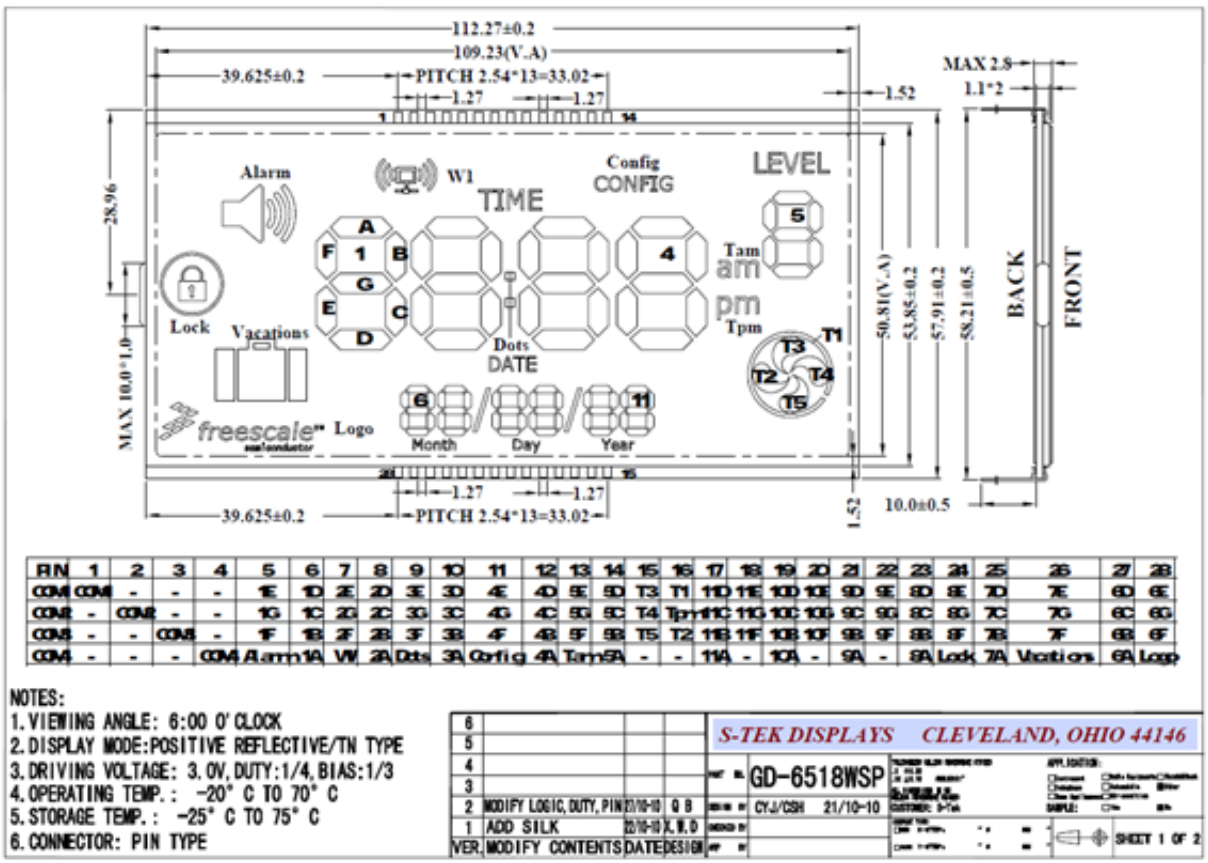

Figure B-1. Data sheet segment LCD

Appendix C
Reference Documents

The documents listed below should be referenced for more information regarding this design or the products featured within.

- MC9S08LH64 Reference Manual HCS08 Microcontrollers.
- MC9S08SE8 Reference Manual HCS08 Microcontrollers.
- 56F8006 Data Sheet, MC56F8006, Freescale Semiconductor, 2008.
- CodeWarrior™ Development Studio for Freescale™ 56800/E Digital Signal Controllers.
- Application note— Adding a Voice User Interface to M68HC05 (document number AN1292)
- TSS—Touch Sensing Software 2.5v.
- Secure Digital Card product manual, SanDisk 2003
- Free Master Software Users Manual, Freescale Semiconductor, 2004
- Masterflux TH BLDC Motor Controller Product Specification 030F0121
- http://www.Modbus-IDA.org

For a current list of documentation, go to www.freescale.com. This link Includes documentation, hardware, software and project management documents.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc.

Document Number: DRM129
Rev. 0, 04/2013