



Z-Wave Software Structure: Learn about Command Classes and Reference Code

2020



Introduction

The Z-Wave protocol is an interoperable, wireless, RF-based communications technology designed specifically for control, monitoring and status reading applications in residential and light commercial environments.



- Interoperable
- Mesh Network
- International Standard
 - ITU-9959
- Low Power
 - Frequently Listening Slave (FLiRS)
- Sub 1 GHz
- Fully Backward Compatible
- Standardized Specifications
 - Command Classes

The Language of Z-Wave

- Command Class Types

- Application
- Management
- Encapsulation
- Network

- Types of Commands

- Set

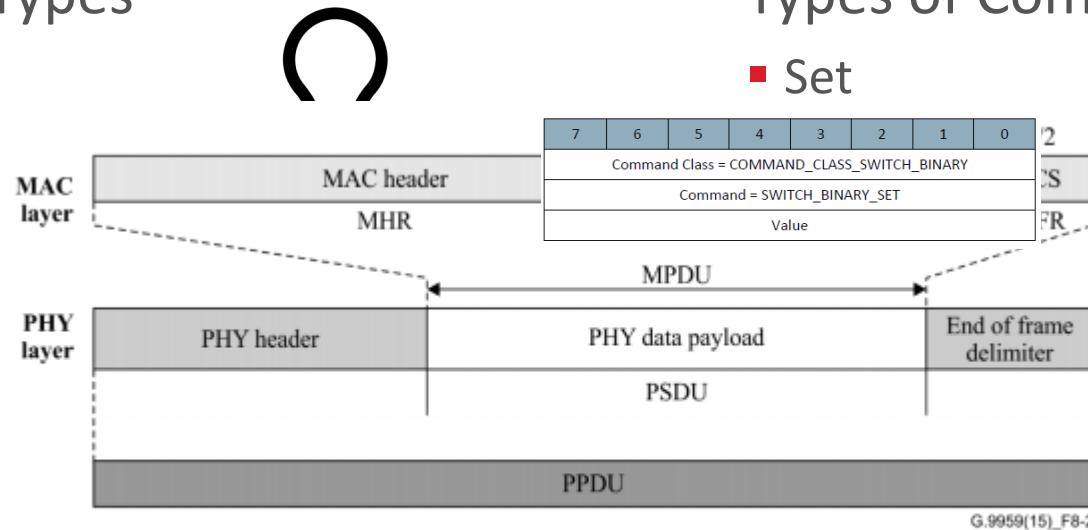
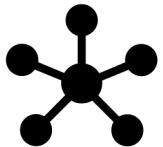


Figure 8-2 – General MPDU structure



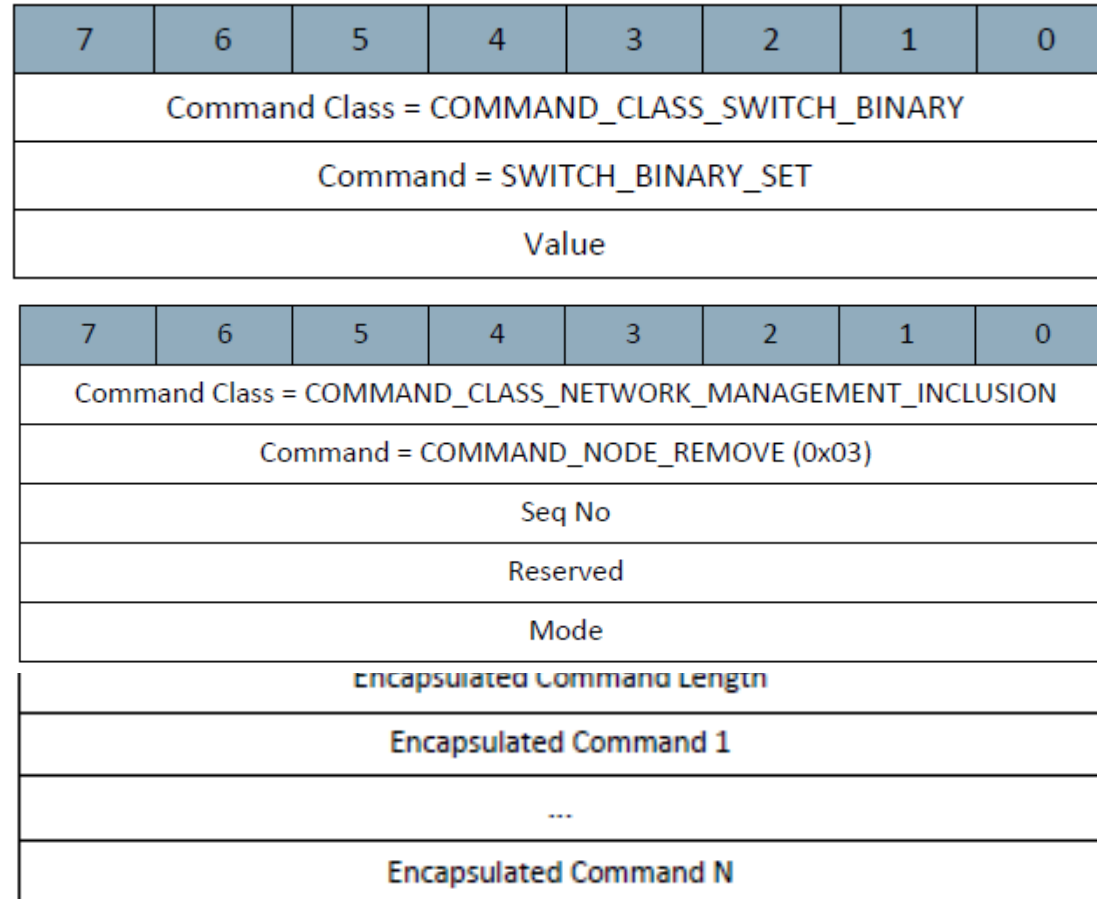
port



*<https://www.silabs.com/wireless/z-wave/specification>

Examples of the different command classes

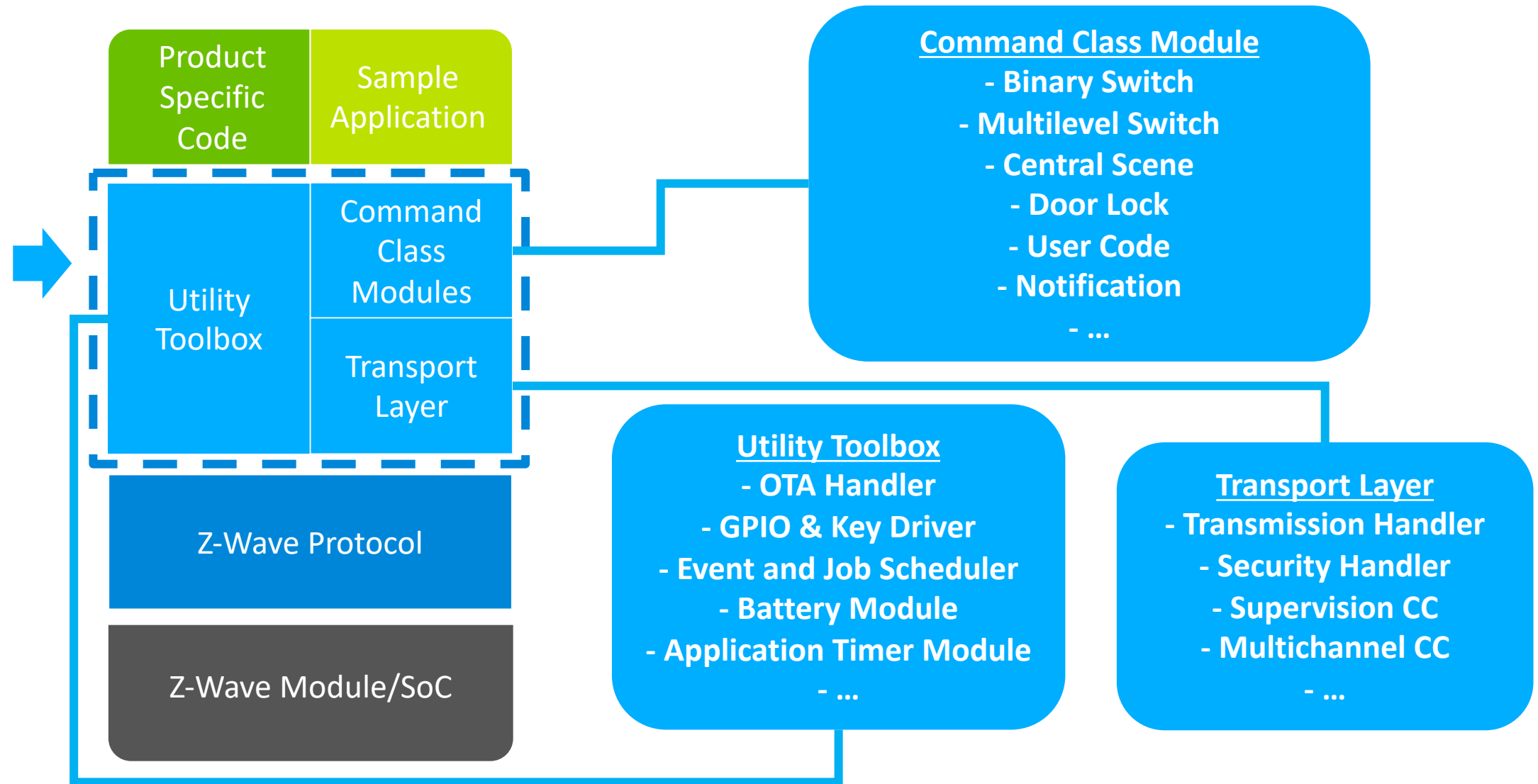
- Application
 - Binary Switch Set
 - Door Lock
 - Thermostat Setpoint
- Encapsulation
 - Supervision
 - Security S2
- Management
 - Version
 - Battery
- Network
 - Remove
 - Node Provisioning



*<https://www.silabs.com/wireless/z-wave/specification>

End Device Design and the Application Framework

Application Framework



The Sample Applications

Always On

Switch On/OFF 

 Power Strip 

 Wall Switch 

FLiRS

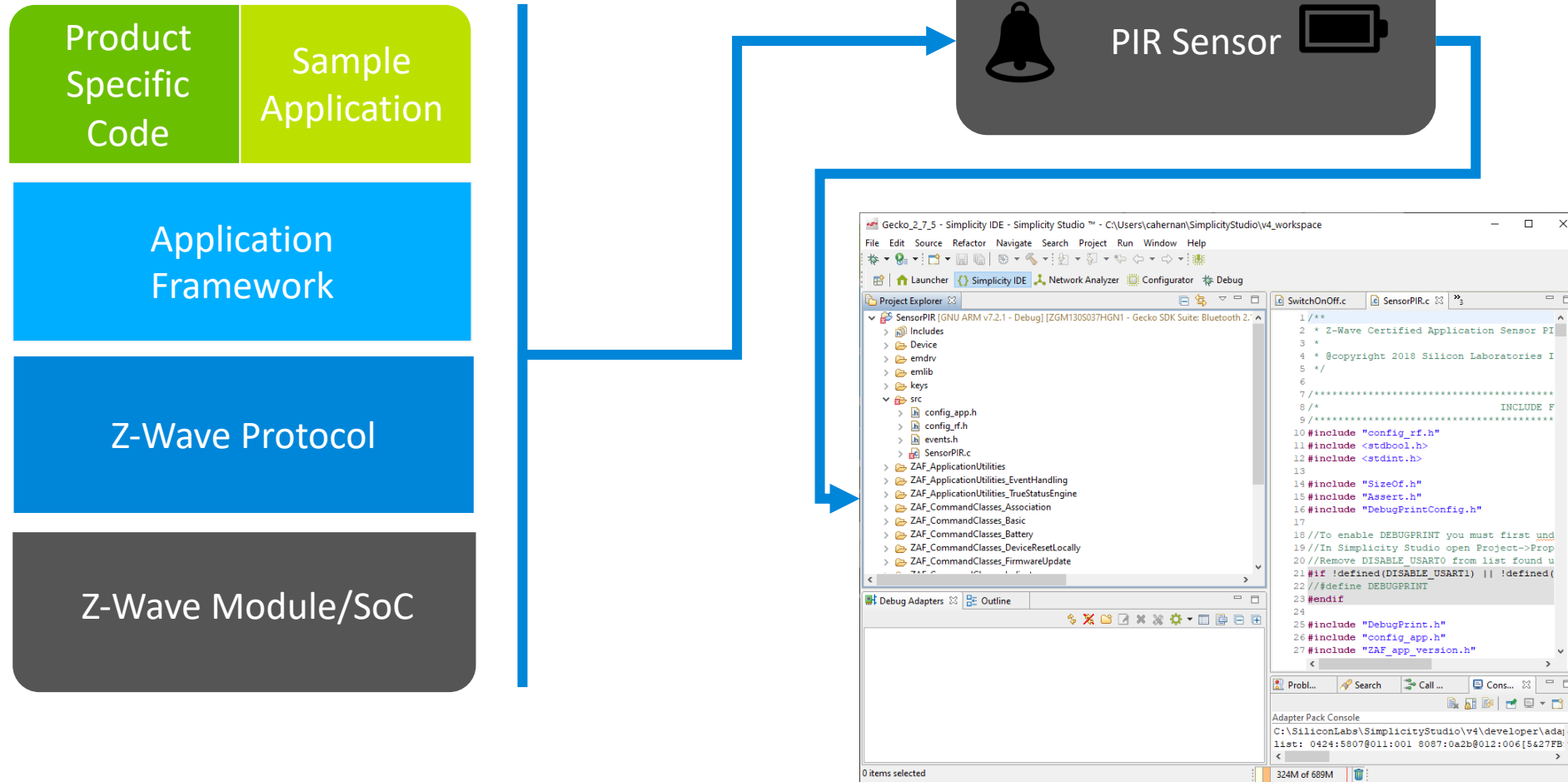
 Door Lock 

Sleepy

 PIR Sensor 

 D/W Sensor 

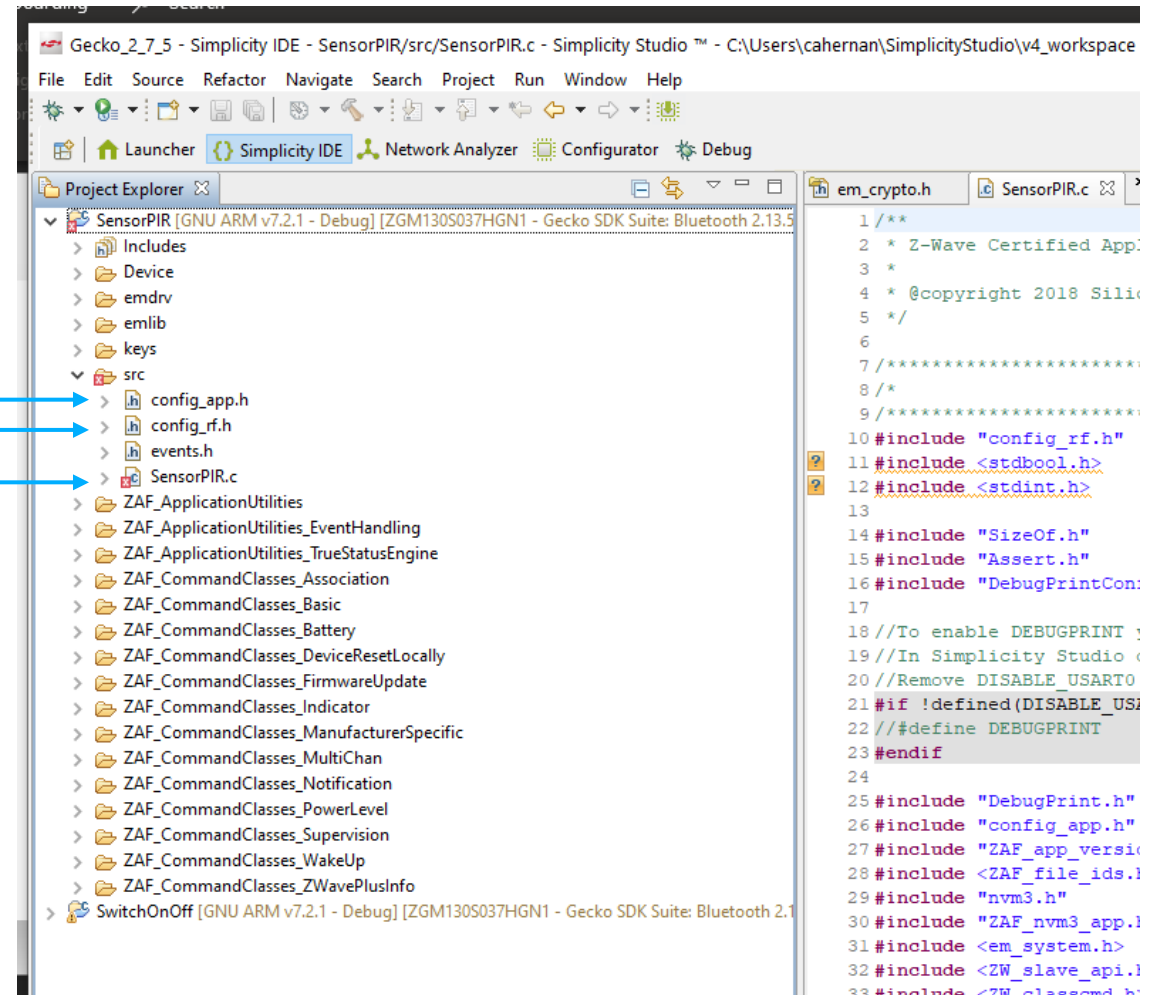
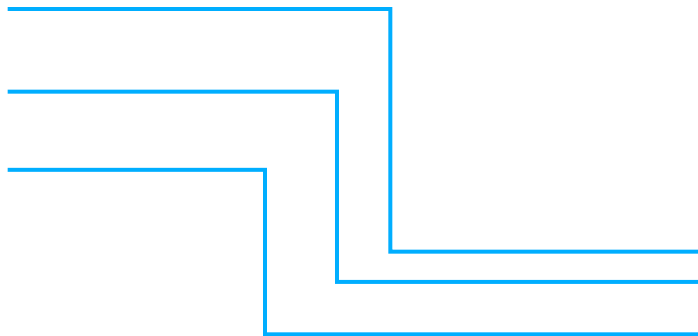
End Device Design from Sample Applications



Application Configuration

Configuration needed in 3 files

- config_app.h
- config_rf.h
- main source file



Config_app.h

- Device Role
- Device Type

```
//@ [DEVICE_OPTIONS_MASK_ID]
#define DEVICE_OPTIONS_MASK    APPLICATION_NODEINFO_NOT_LISTENING

//@ [APP_TYPE_ID]
#define APP_ROLE_TYPE ZWAVEPLUS_INFO_REPORT_ROLE_TYPE_SLAVE_SLEEPING_REPORTING
```

Device Class	Window Covering	Dimmer Switch
Generic	Multilevel Device	Multilevel Device
Specific	Motor Control	Specific Not Used

Config_app.h continued


```
)/*****  
 *  
 * Defines used to initialize the Manufacturer Specific Command Class.  
 *  
 *****/  
#define APP_MANUFACTURER_ID    MFG_ID_ZWAVE  
#define APP_PRODUCT_ID        PRODUCT_ID_SensorPIR
```

- Manufacturer Information
- Association Information

```
#define AGITABLE_LIFELINE_GROUP \  
{COMMAND_CLASS_BATTERY, BATTERY_REPORT}, \  
{COMMAND_CLASS_NOTIFICATION_V8, NOTIFICATION_REPORT_V8}, \  
{COMMAND_CLASS_DEVICE_RESET_LOCALLY, DEVICE_RESET_LOCALLY_NOTIFICATION}, \  
{COMMAND_CLASS_INDICATOR, INDICATOR_REPORT_V3}  
  
#define AGITABLE_ROOTDEVICE_GROUPS \  
{{ASSOCIATION_GROUP_INFO_REPORT_PROFILE_NOTIFICATION, NOTIFICATION_REPORT_HOME_SECURITY_V4}, {COMMAND_CLASS_BAS
```

Config_rf.h

- Config_rf.h
 - RF Power



```
// The maximum allowed Tx power in deci dBm
#define APP_MAX_TX_POWER      0

// The deci dBm output measured at a PA setting of 0dBm
#define APP_MEASURED_0DBM_TX_POWER 33
```

Main Source File

- RF Frequency
- Command Class List

```
▶/**
 * Unsecure node information list.
 * Be sure Command classes are not duplicated in both lists.
 * CHANGE THIS - Add all supported non-secure command classes here
 **/
static uint8_t cmdClassListNonSecureNotIncluded[] =
{
    COMMAND_CLASS_ZWAVEPLUS_INFO,
    COMMAND_CLASS_TRANSPORT_SERVICE_V2,
    COMMAND_CLASS_SECURITY,
    COMMAND_CLASS_SECURITY_2,
    COMMAND_CLASS_SUPERVISION,
    COMMAND_CLASS_FIRMWARE_UPDATE_MD_V5
};

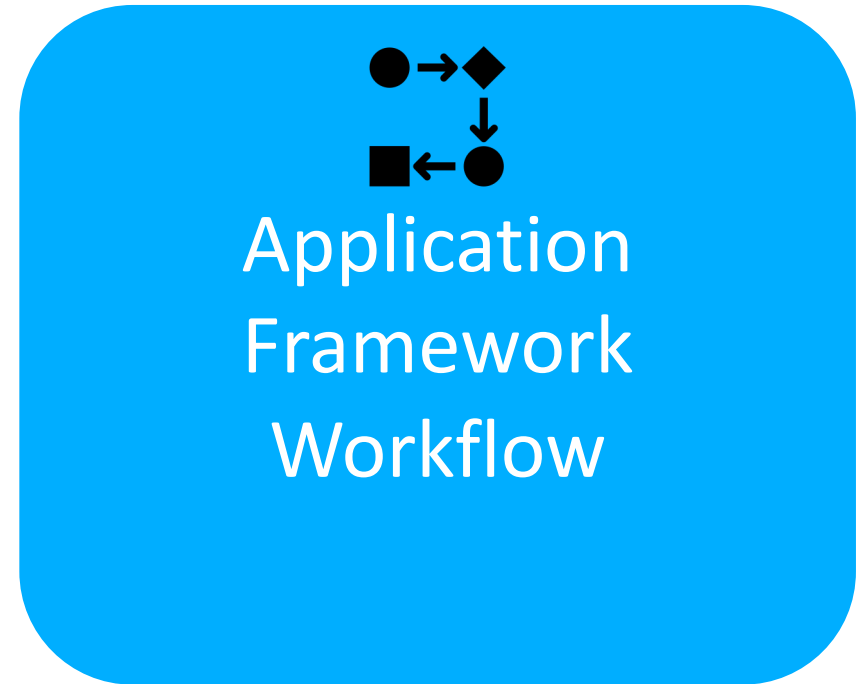
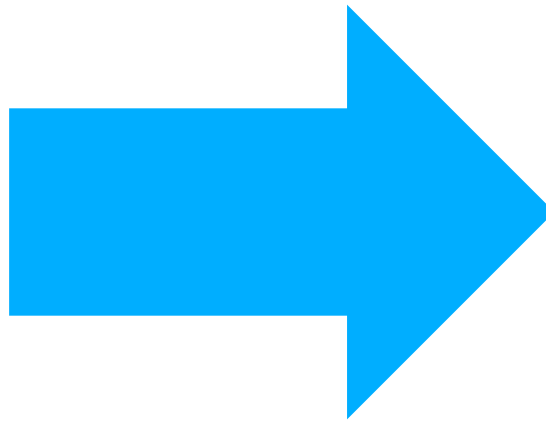
▶/**
 * Unsecure node information list Secure included.
 * Be sure Command classes are not duplicated in both lists.
 * CHANGE THIS - Add all supported non-secure command classes here
 **/
static uint8_t cmdClassListNonSecureIncludedSecure[] =
{
    COMMAND_CLASS_ZWAVEPLUS_INFO,
    COMMAND_CLASS_TRANSPORT_SERVICE_V2,
    COMMAND_CLASS_SECURITY,
    COMMAND_CLASS_SECURITY_2,
    COMMAND_CLASS_SUPERVISION
};
```

```
static const SRadioConfig_t RadioConfig =
{
    .iListenBeforeTalkThreshold = ELISTENBEFORETALKTRESHOLD_DEFAULT,
    .iTxPowerLevelMax = APP_MAX_TX_POWER,
    .iTxPowerLevelAdjust = APP_MEASURED_ODBM_TX_POWER,
    .eRegion = APP_FREQ
};

▶/**
 * Secure node information list.
 * Be sure Command classes are not duplicated in both lists.
 * CHANGE THIS - Add all supported secure command classes here
 **/
static uint8_t cmdClassListSecure[] =
{
    COMMAND_CLASS_VERSION,
    COMMAND_CLASS_MANUFACTURER_SPECIFIC,
    COMMAND_CLASS_DEVICE_RESET_LOCALLY,
    COMMAND_CLASS_INDICATOR,
    COMMAND_CLASS_POWERLEVEL,
    COMMAND_CLASS_BATTERY,
    COMMAND_CLASS_DOOR_LOCK,
    COMMAND_CLASS_USER_CODE,
    COMMAND_CLASS_ASSOCIATION_V2,
    COMMAND_CLASS_MULTI_CHANNEL_ASSOCIATION_V2,
    COMMAND_CLASS_ASSOCIATION_GRP_INFO,
    COMMAND_CLASS_FIRMWARE_UPDATE_MD_V5
};
```

Configuration Checklist

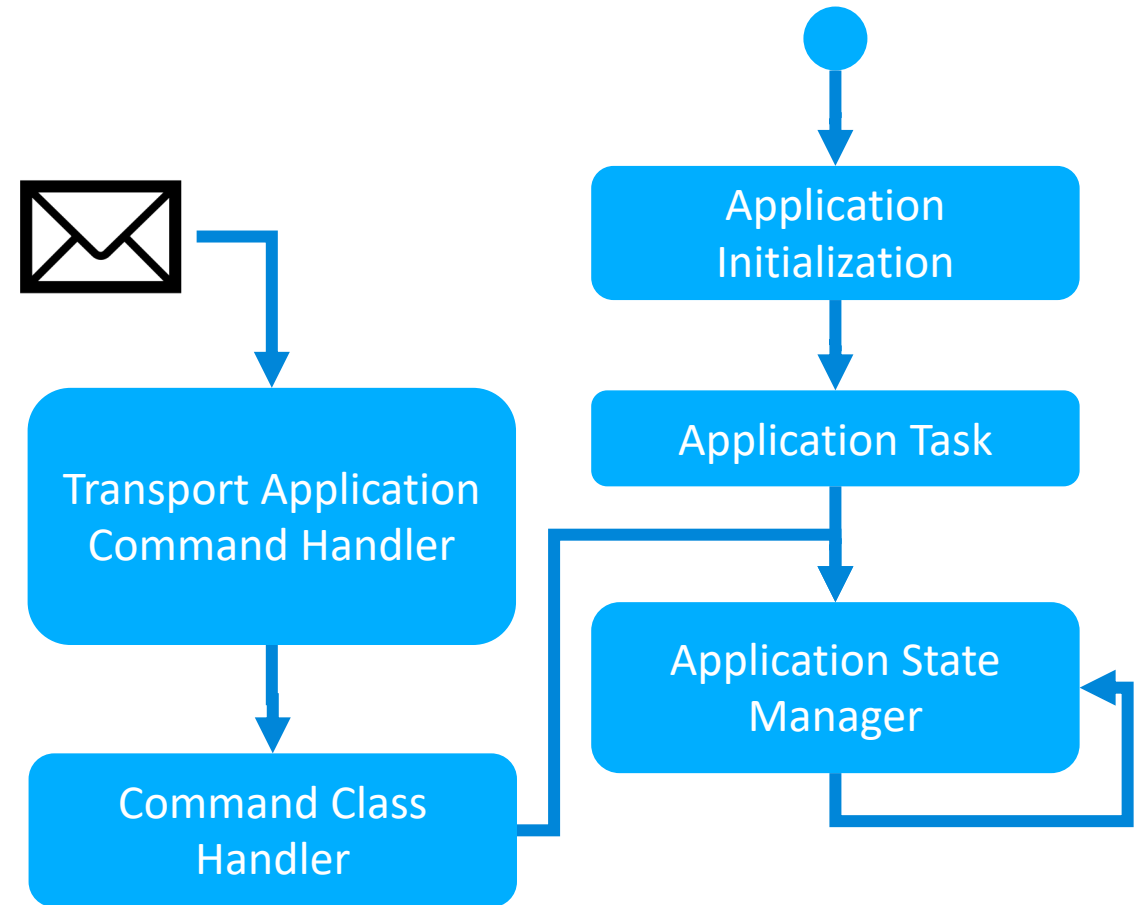
- ✓ ■ Role Type
- ✓ ■ Device Type
- ✓ ■ Manufacturer Information
- ✓ ■ Association Groups
- ✓ ■ TX Power
- ✓ ■ Command class list
- ✓ ■ RF region



Important Functions and Framework Flow

Important Main File Functions

- ***ApplicationInit***
 - Initialize the application.
- ***ApplicationTask***
 - Setup of events and IO.
- ***AppStateManager***
 - Core state machine of framework
- ***Transport_ApplicationCommandHandlerEx***
 - Incoming RF frames
- ***CommandClass functions, e.g. CC_Basic_Set_handler***
 - Application logic for various command classes



Events: Learn Mode State

```
case STATE_APP_LEARN_MODE:  
    if (EVENT_APP_FLUSHMEM_READY == event)  
    {  
        ~~~~~  
    }  
    if ((BTN_EVENT_SHORT_PRESS (APP_BUTTON_LEARN_RESET) == (BUTTON_EVENT)event) ||  
        (EVENT_SYSTEM_LEARNMODE_STOP == (EVENT_SYSTEM)event))  
    {  
        ~~~~~  
    }  
    if (EVENT_SYSTEM_LEARNMODE_FINISHED == (EVENT_SYSTEM)event)  
    {  
        ~~~~~  
    }  
    break;
```

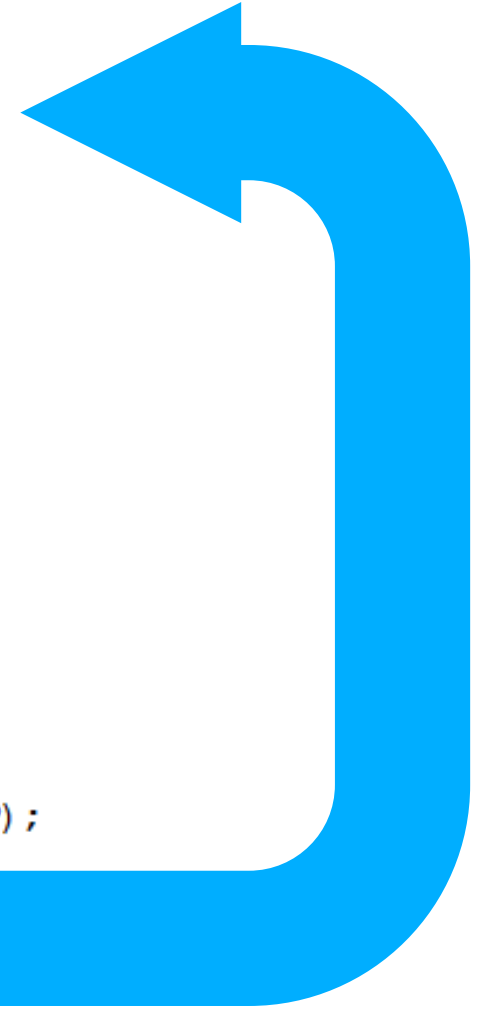
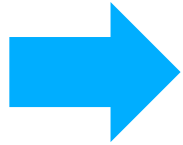

Events: Idle State

```
case STATE_APP_IDLE:
    /* The PIR_EVENT_BTN could be allocated to either a slider or a button.
     * - The slider will always send a DOWN event when moved to the ON position.
     * - A button will send SHORT_PRESS or HOLD event when pressed. Only the
     *   HOLD event will be followed by an UP event when the button is
     *   released. Since we need the UP event later to cancel the power
     *   lock, we ignore the SHORT_PRESS event here.
     */
    if ((BTN_EVENT_DOWN(PIR_EVENT_BTN) == (BUTTON_EVENT)event) ||
        (BTN_EVENT_HOLD(PIR_EVENT_BTN) == (BUTTON_EVENT)event))
        ZAF_PM_StayAwake(&m_RadioPowerLock, 0);
    DPRINT("\r\n*!* PIR_EVENT_BTN");
```

Switching States: Idle State

```
case STATE_APP_IDLE:
    if ((BTN_EVENT_DOWN(PIR_EVENT_BTN) == (BUTTON_EVENT)event) ||
        (BTN_EVENT_HOLD(PIR_EVENT_BTN) == (BUTTON_EVENT)event))
    {
        ZAF_PM_StayAwake(&m_RadioPowerLock, 0);
        DPRINT("\r\n*!*!* PIR_EVENT_BTN");
        ChangeState(STATE_APP_TRANSMIT_DATA);
        // ...

        /*Add event's on job-queue*/
        ZAF_JobHelperJobEnqueue(EVENT_APP_BASIC_START_JOB);
        ZAF_JobHelperJobEnqueue(EVENT_APP_NOTIFICATION_START_JOB);
        ZAF_JobHelperJobEnqueue(EVENT_APP_START_TIMER_EVENTJOB_STOP);
    }
}
```



Power Manager and Energy Modes

Power Manager

- Z-Wave Protocol controls the energy mode
- Will always go to sleep mode if possible

Power Locks

- lock the chip from entering sleep mode

Power Lock Type

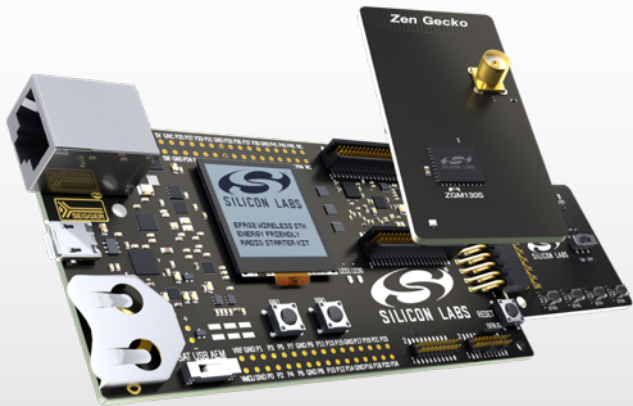
- PM_TYPE_RADIO
don't enter EM2/EM3/EM4
- PM_TYPE_PERIPHERAL
don't enter EM3/EM4

EM0 Active	Radio available
EM1 Sleep	MCU sleeping. Radio available
EM2 Deep Sleep	RAM Retention
EM3 Stop	RAM Retention
EM4 Shutoff	MCU shut down. No RAM retention. Wake up by Reset or Interrupt.

```
void SomeFunction(void)
{
    ZAF_PM_StayAwake(&m_RadioPowerLock, 0);
    :
    /* Do something where the radio module is required */
    :
    ZAF_PM_Cancel(&m_RadioPowerLock);
    :
}
```

Z-Wave 700 Components

WIRELESS STARTER KIT



One kit for both end device and gateway development

ZGM130S – SiP MODULE



End-devices & gateways
LGA64 9x9 mm SiP

EFR32ZG14 – MODEM SoC



Gateways only
QFN32 5x5 mm SoC

Development Kit

The hardware in the kit

- WSTK Main Development Board, 2 pcs
- BRD4200A (BRD4202A) Radio Board with ZGM130S
- BRD8029A EXP Board, 2 pcs
- UZB7 Controller USB Dongle
- Zniffer USB Dongle



Documentation Online

- Z-Wave Support Website:

- <https://www.silabs.com/wireless/z-wave>

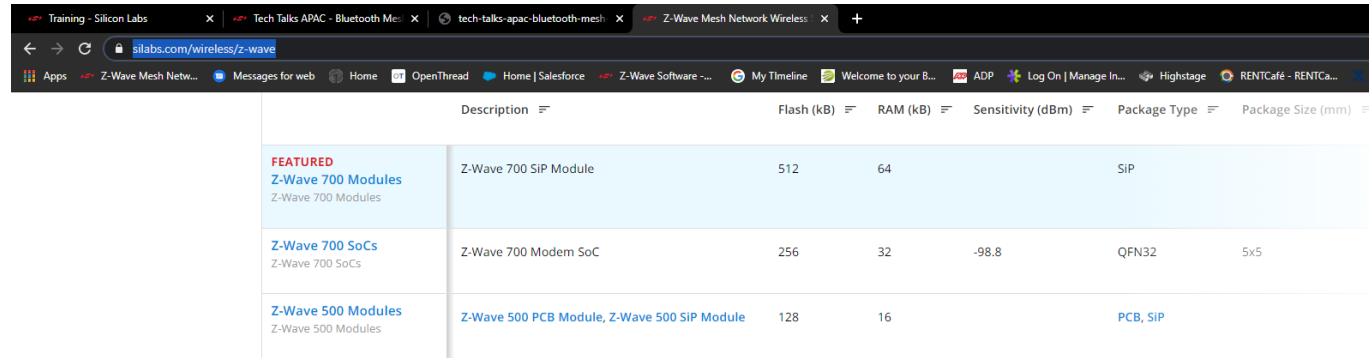
- Certification
- Specifications
- Learning Center

- Simplicity Studio:

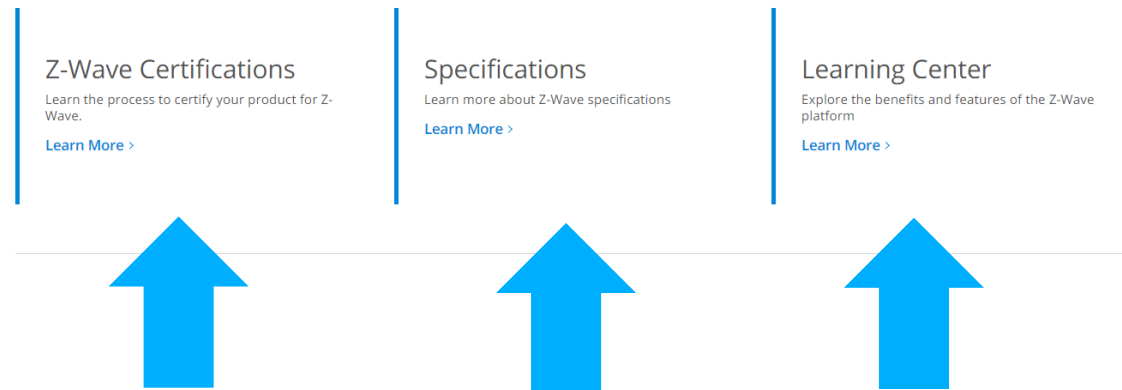
- <https://www.silabs.com/products/development-tools/software/simplicity-studio>

- ITU 9959:

- <file:///C:/Users/cahernan/Downloads/T-REC-G.9959-201202-S!!PDF-E.pdf>



	Description	Flash (kB)	RAM (kB)	Sensitivity (dBm)	Package Type	Package Size (mm)
FEATURED Z-Wave 700 Modules Z-Wave 700 Modules	Z-Wave 700 SIP Module	512	64		SIP	
Z-Wave 700 SoCs Z-Wave 700 SoCs	Z-Wave 700 Modem SoC	256	32	-98.8	QFN32	5x5
Z-Wave 500 Modules Z-Wave 500 Modules	Z-Wave 500 PCB Module, Z-Wave 500 SIP Module	128	16		PCB, SIP	



Documentation in IDE

- Demos
 - Preconfigured Sample Apps
- Sample Applications
- SDK Documentation

The screenshot shows the Silicon Labs IDE interface for the product ZGM130S037HGN1. The interface includes a top menu bar, a search bar, and a toolbar with icons for Launcher, Simplicity IDE, Network Analyzer, Configurator, and Debug. The main workspace is divided into three vertical panes: Demos, Software Examples, and SDK Documentation. The Demos pane lists various sample applications like Bootloader, Door Lock keypad, Power Strip, SensorPIR, SerialAPI, Switch On/Off, and Wall Controller. The Software Examples pane lists examples for Flex SDK 2.7.5.0, Gecko Bootloader 1.10.3, and Z-Wave SDK 7.13.6.0, including DoorLockKeyPad, PowerStrip, SensorPIR, SwitchOnOff, and WallController. The SDK Documentation pane lists documentation for Flex SDK 2.7.5.0, Gecko Bootloader 1.10.3, Micrium OS Kernel 5.8.2, and Z-Wave SDK 7.13.6.0, including API References, Application Notes, Fundamentals, Getting Started, Quick Start Guides, Release Notes, User's Guides, and Z-Wave Specifications. A red arrow points to the SDK Documentation pane, and blue arrows point to the Demos, Software Examples, and SDK Documentation tabs. A red dashed box highlights the SDK Documentation content.

Thank you!

silabs.com

