Rye, Esten A.

esten.rye@ryezone.com

location triangulation

Using RF Time of Flight Ranging

Contents

[Table of Figures 3](#_Toc514028147)

[Project Overview 4](#_Toc514028148)

[Processor Evaluation 5](#_Toc514028149)

[Task Definitions 7](#_Toc514028150)

[Tracked Object 7](#_Toc514028151)

[System Initialization 7](#_Toc514028152)

[Register Beacons 8](#_Toc514028153)

[Receive Time of Flight Distance Transmission from Beacon 8](#_Toc514028154)

[Triangulate Location Task 8](#_Toc514028155)

[Write Display Task 9](#_Toc514028156)

[Beacons 9](#_Toc514028157)

[System Initialization Task 10](#_Toc514028158)

[Start TOF Burst Measurement Task 10](#_Toc514028159)

[Calculate Distance Task 10](#_Toc514028160)

[Transmit TOF Data Task 11](#_Toc514028161)

[Task Schedule 11](#_Toc514028162)

[Tracked Object (coordinator.c) Schedule 11](#_Toc514028163)

[Beacon (enddevice.c) Schedule 11](#_Toc514028164)

[Processor Bandwidth Estimation 12](#_Toc514028165)

[Memory Estimate 12](#_Toc514028166)

[Tracked Object (coordinator.c) Memory Utilization Estimate 12](#_Toc514028167)

[Beacon (enddevice.c) Memory Utilization Estimate 12](#_Toc514028168)

[Real Time Operating System (RTOS) Justification 12](#_Toc514028169)

[Microcontroller Block Diagram 13](#_Toc514028170)

[Memory Map – ROM – Tracked Object (coordinator.c) 14](#_Toc514028171)

[Memory Map – RAM – Tracked Object (coordinator.c) 15](#_Toc514028172)

[Memory Map – ROM – Beacon (enddevice.c) 16](#_Toc514028173)

[Memory Map – RAM – Beacon (enddevice.c) 17](#_Toc514028174)

[Hardware I/O Map 18](#_Toc514028175)

[Hardware-Imposed Software Constraints 19](#_Toc514028176)

[Beacon Constraints 19](#_Toc514028177)

[Tracked Object Constraints 19](#_Toc514028178)

[Software-Imposed Hardware Constraints 19](#_Toc514028179)

[Beacon Constraints 19](#_Toc514028180)

[Tracked Object Constraints 19](#_Toc514028181)

[References 20](#_Toc514028182)

[Appendix A – Tracked Object (coordinator.c) 21](#_Toc514028183)

[Appendix B – Beacon (enddevice.c) 54](#_Toc514028184)

# Table of Figures

[Figure 1: High-level Implementation 4](#_Toc514028037)

[Figure 2: Processor Evaluation Results - Core Processor Features 5](#_Toc514028038)

[Figure 3: Processor Evaluation Results - Processor Performance Metrics 5](#_Toc514028039)

[Figure 4: Processor Evaluation Results - Memory Size and Processor Capacity 6](#_Toc514028040)

[Figure 5: Processor Evaluation Results - Networking Features and RTOS Support 6](#_Toc514028041)

[Figure 6: Processor Evaluation Results - Implementation Costs 6](#_Toc514028042)

[Figure 7: DR1047 Development Board Layout (NXP, 2010) 7](#_Toc514028043)

[Figure 8: Tracked Object - System Initialization 7](#_Toc514028044)

[Figure 9:Tracked Object - Pair Beacons Task Flow 8](#_Toc514028045)

[Figure 10: Tracked Object – Receive Distances Task Flow 8](#_Toc514028046)

[Figure 11: Tracked Object - Triangulate Location Task Flow 8](#_Toc514028047)

[Figure 12: Tracked Object - Write Display Task Flow 9](#_Toc514028048)

[Figure 13: DR1048 Sensor Board (NXP, 2010) 9](#_Toc514028049)

[Figure 14: Beacon - System Init Task Flow 10](#_Toc514028050)

[Figure 15: Beacon – Start TOF Task Flow 10](#_Toc514028051)

[Figure 16: Beacon - Calculate Average Distances 10](#_Toc514028052)

[Figure 17: Beacon - TOF Echo Task Flow 11](#_Toc514028053)

[Figure 18: Originally Proposed Tracked Object Schedule 11](#_Toc514028054)

[Figure 19: Originally Proposed Beacon Schedule 11](#_Toc514028055)

[Figure 20: JN5148-001 Block Diagram (NXP, 2010) 13](#_Toc514028056)

[Figure 21: Memory Map – Tracked Object (coordinator.c) – ROM 14](#_Toc514028057)

[Figure 22: Memory Map – Tracked Object (coordinator.c) – RAM 15](#_Toc514028058)

[Figure 23: Memory Map – Beacon (enddevice.c) - ROM 16](#_Toc514028059)

[Figure 24: Memory Map – Beacon (enddevice.c) - RAM 17](#_Toc514028060)

[Figure 25: JN5148 Hardware I/O Map (NXP, 2010) 18](#_Toc514028061)

# Project Overview

The purpose of this project is to design and implement a Real Time Embedded System that can locate the position of a tracked object on a two-dimensional plane based on the time of flight of RF signals transmitted to two separate, fixed beacons.

To calculate the distances a and b, the microcontroller in the tracked object will use time of flight data from each beacon and the speed of light (Tipler, 1999) to calculate the tracked object’s distance (Electronic, 2011) from each beacon.

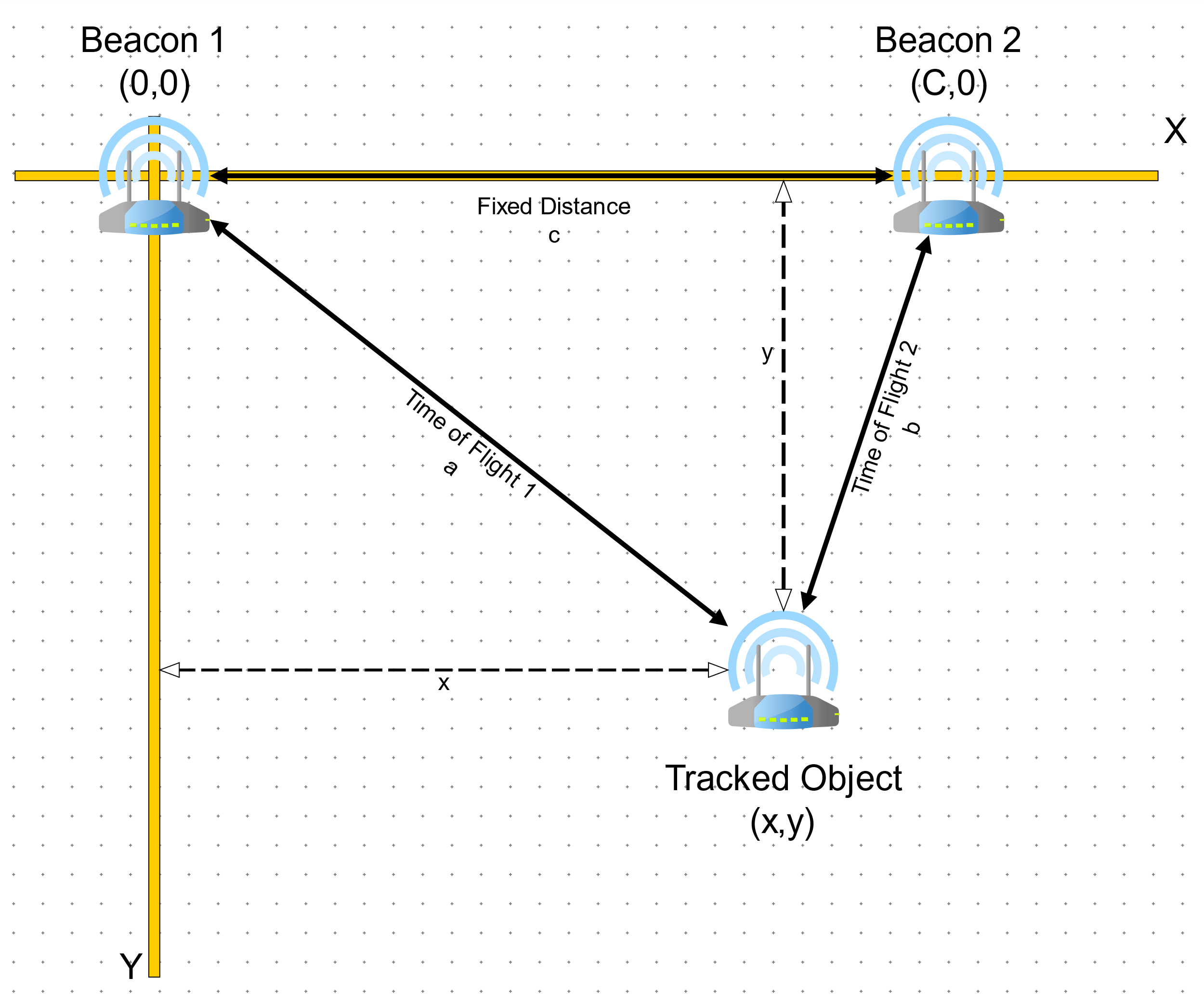


Figure 1: High-level Implementation

To calculate the y component of the tracked object’s position, the microcontroller in the tracked object will use Heron’s formula to calculate the height of a triangle given lengths of three sides. (wikiHow, 2018)

To calculate the x component, the microcontroller will use the Pythagorean theorem to calculate the length of the third side of a triangle given the length of the other two.

The inherent limitation of this project is that the method of triangulation being implemented only uses two beacons. The drawback of this approach is that the tracked object has no way of knowing when it has crossed the X or Y axis on the two-dimensional plane. For the purposes of this project, an assumption has been made that there are physical barriers in place to prevent the tracked object from reporting erroneous results by escaping across the X or Y axis.

# Processor Evaluation

The following criteria were used to select the microprocessor for this project:

* Does the vendor supply a Time-Of-Flight SDK for calculating round-trip time between the nodes?
* Does the vendor supply a reliable messaging SDK for communicating between the nodes?
* Does the microcontroller provide enough RAM to support growth of the beacon mesh network?
* Availability of development boards.
* Availability of local support.
* Cost.

Five microcontrollers were evaluated for selection in this project. Below is a set of tables that summarize the results of the evaluation. The processor selected was the NXP JN-5148 wireless microcontroller. The major factors that drove its selection were the vendor provided Time-Of-Flight and Networking SDKs, the large amount of available RAM for maintaining routing tables for the beacon mesh network and the availability of free evaluation hardware.



Figure 2: Processor Evaluation Results - Core Processor Features



Figure 3: Processor Evaluation Results - Processor Performance Metrics



Figure 4: Processor Evaluation Results - Memory Size and Processor Capacity



Figure 5: Processor Evaluation Results - Networking Features and RTOS Support



Figure 6: Processor Evaluation Results - Implementation Costs

# Task Definitions

The microcontrollers in this project have two roles. The first role is the Beacon role. In the Beacon role, the microcontroller acts as a repeater, parroting back time of flight requests from the Tracked Object. The second role is the Tracked Object role. In the Tracked Object role, the microcontroller is responsible for constructing the mesh network, sending time of flight requests, calculating its position relative to the beacons in the mesh network, and outputting that information to the screen.

## Tracked Object

The tracked object is being represented by a Jennic/NXP DR1047 development board with a Jennic/NXP JN5148 wireless microcontroller installed in the Jennic Module Connector. Below is an image of the development board (NXP, 2010):

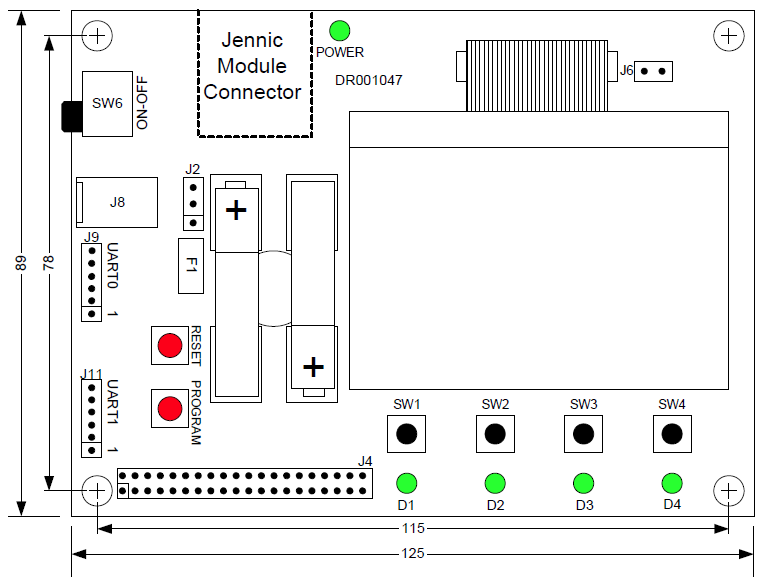


Figure 7: DR1047 Development Board Layout (NXP, 2010)

### System Initialization

This task is executed once on power up of the tracked object microcontroller. This job configures and initializes the mesh network used to facilitate communication between the tracked object and the beacons.

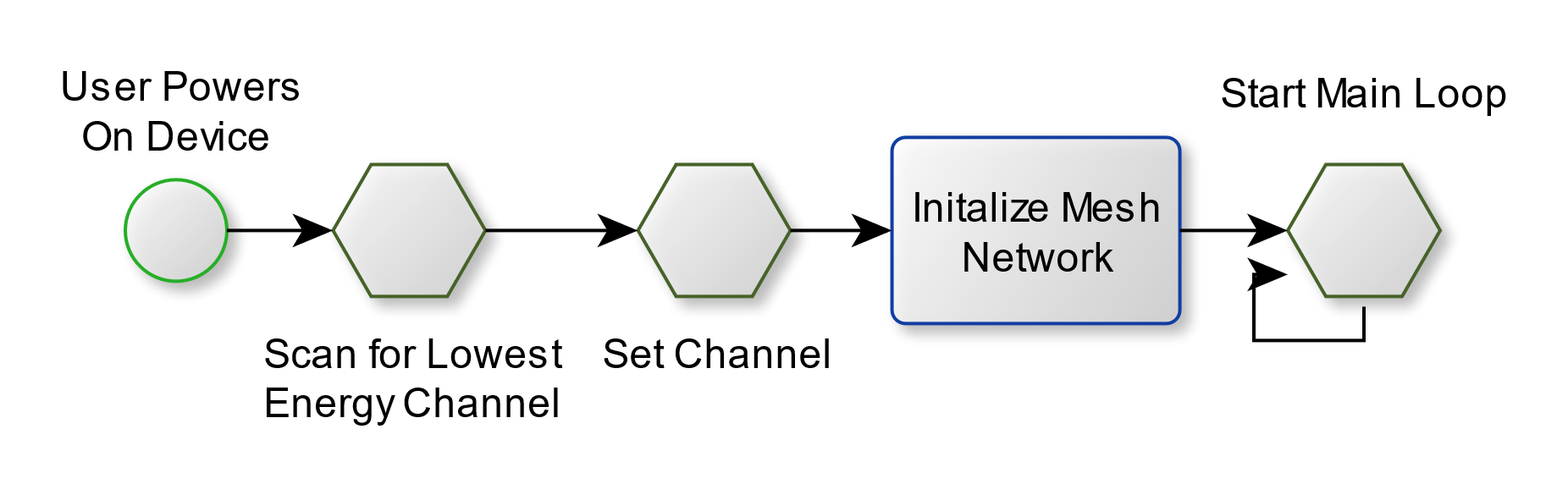


Figure 8: Tracked Object - System Initialization

### Register Beacons

This task is the first task executed by the tracked object microcontroller. This tasks job is to ensure two beacons are joined to the mesh network. This task is the highest priority task at boot. The task flow is depicted below.

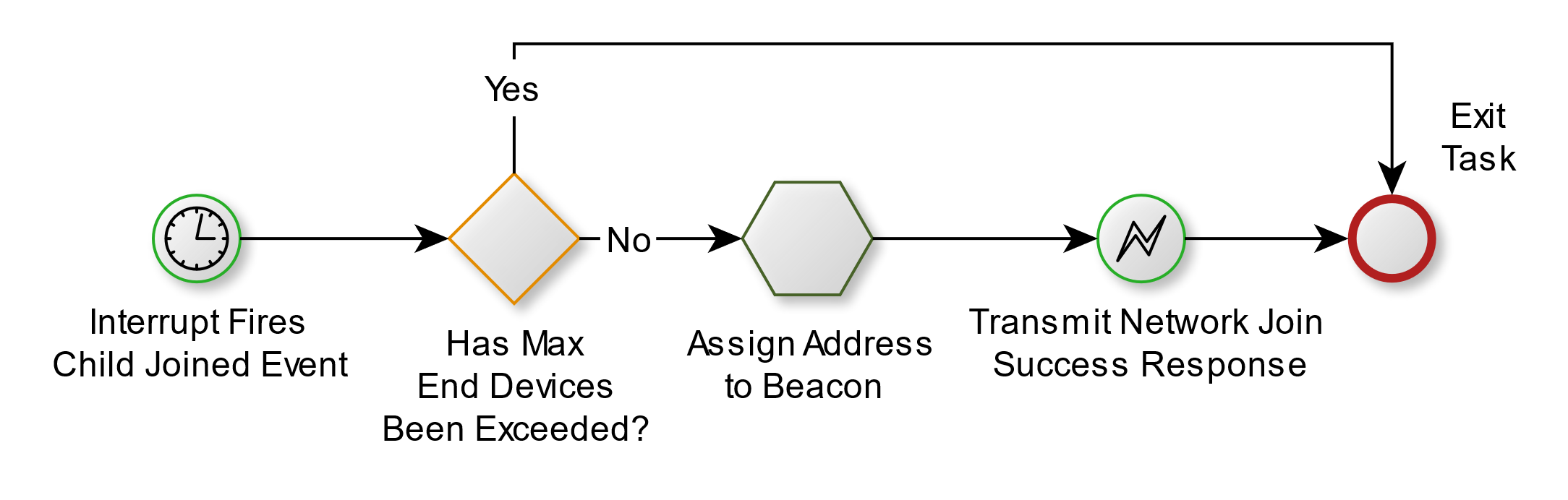


Figure 9:Tracked Object - Pair Beacons Task Flow

### Receive Time of Flight Distance Transmission from Beacon

This task interacts with the Time of Flight Engine to calculate distances *a* and *b* and writes them to a shared memory location.

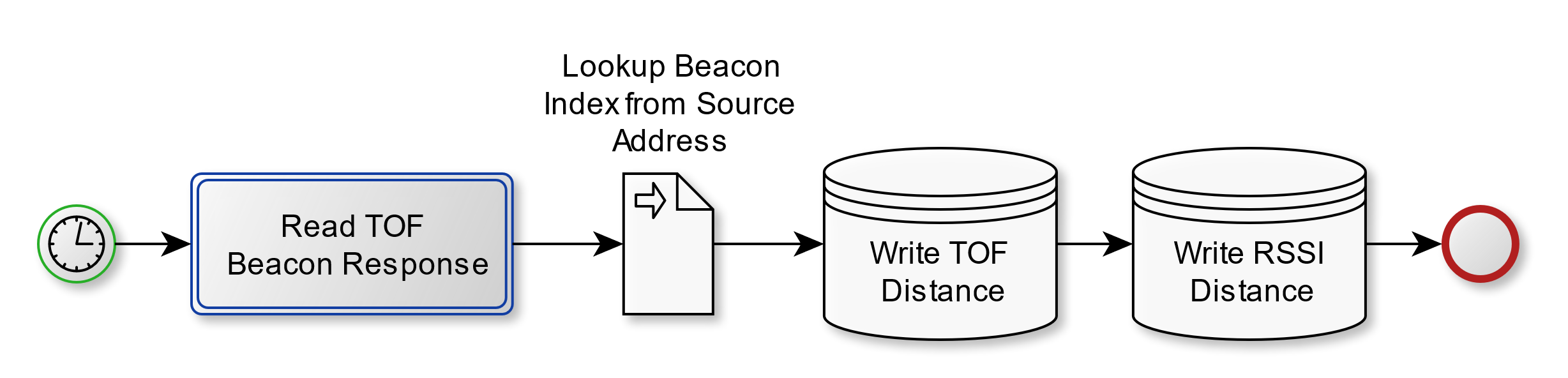


Figure 10: Tracked Object – Receive Distances Task Flow

### Triangulate Location Task

This task reads distances a and b from a shared memory location and calculates coordinates *x* and *y* and writes them to a shared memory location.

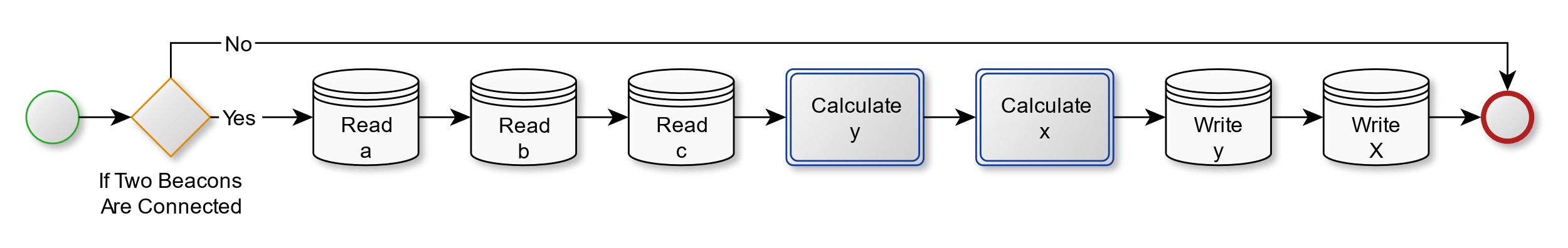


Figure 11: Tracked Object - Triangulate Location Task Flow

### Write Display Task

This task reads coordinates x and y from a shared memory location and displays them on the built-in display.

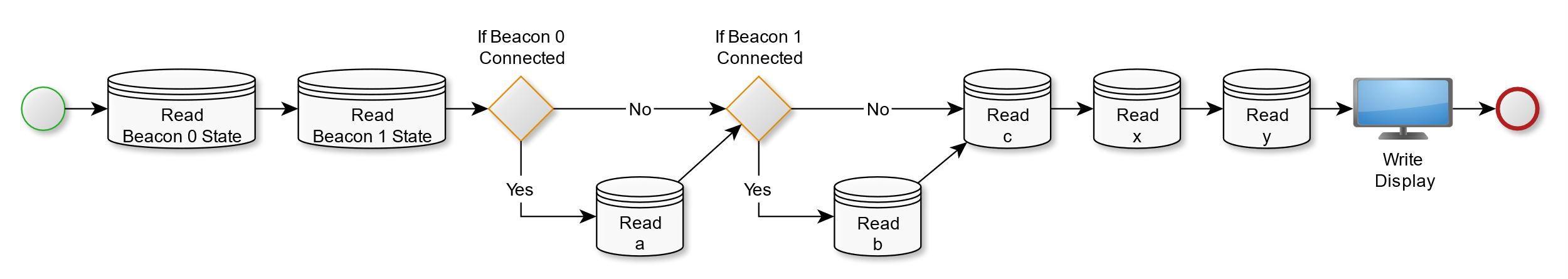


Figure 12: Tracked Object - Write Display Task Flow

## Beacons

There are two beacons in this project. These devices are being represented by a Jennic/NXP DR1048 sensor development board board with a Jennic/NXP JN5148 wireless microcontroller installed in the Jennic Module Connector. Below is a diagram representing the available connections on the board.

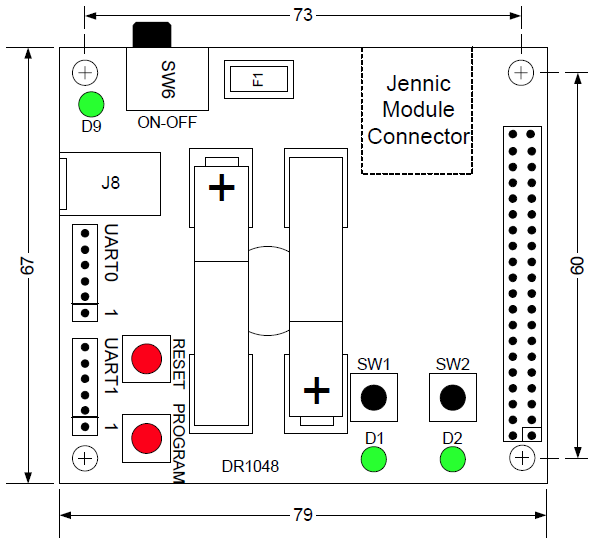


Figure 13: DR1048 Sensor Board (NXP, 2010)

### System Initialization Task

This task is executed once on power up of the tracked object microcontroller. This job locates and joins the beacon to the mesh network used to facilitate communication between the beacon and the tracked object.

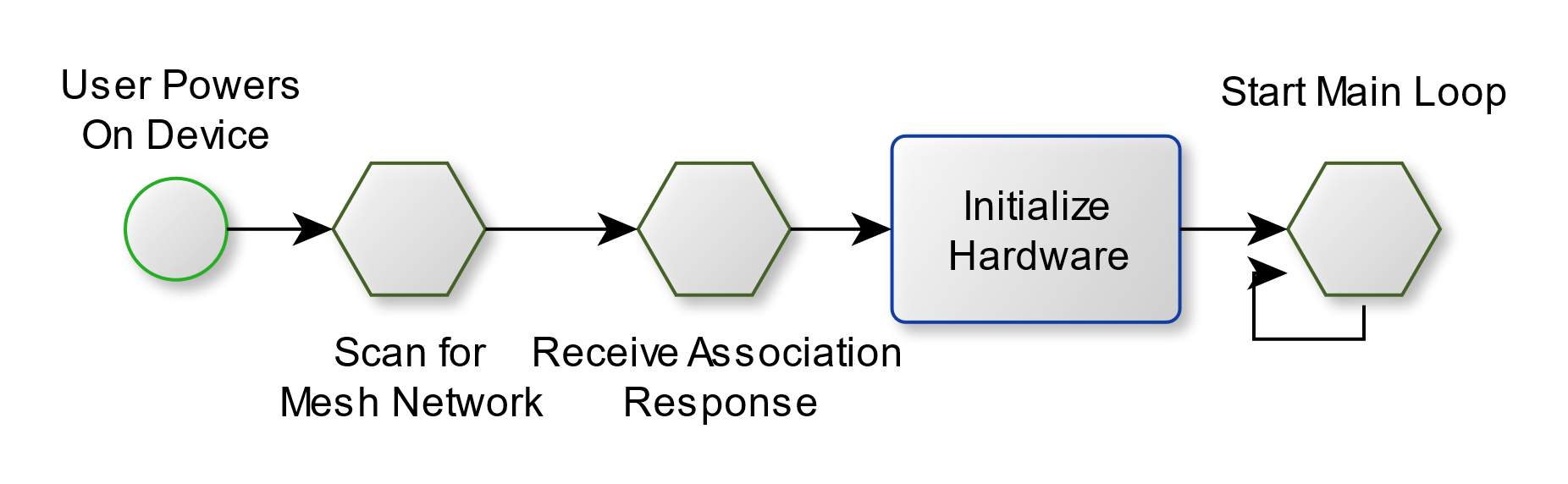


Figure 14: Beacon - System Init Task Flow

### Start TOF Burst Measurement Task

This task starts a forward TOF burst measurement against the Tracked Object.

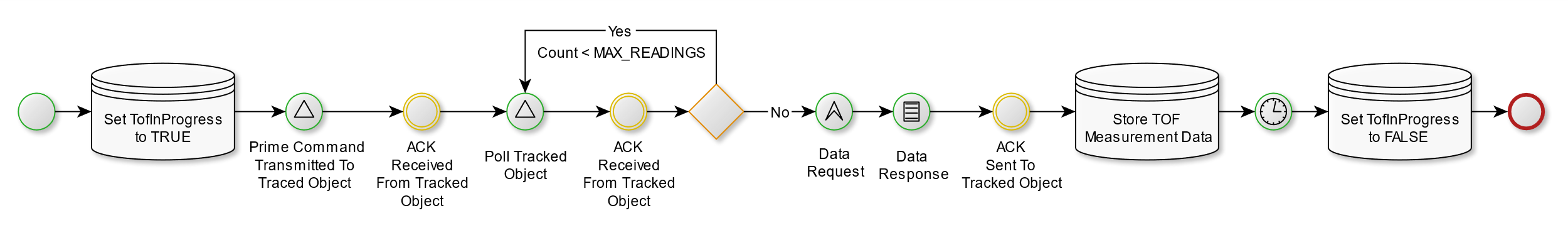


Figure 15: Beacon – Start TOF Task Flow

### Calculate Distance Task

This task calculates the average TOF and RSSI distances from a TOF burst measurement.

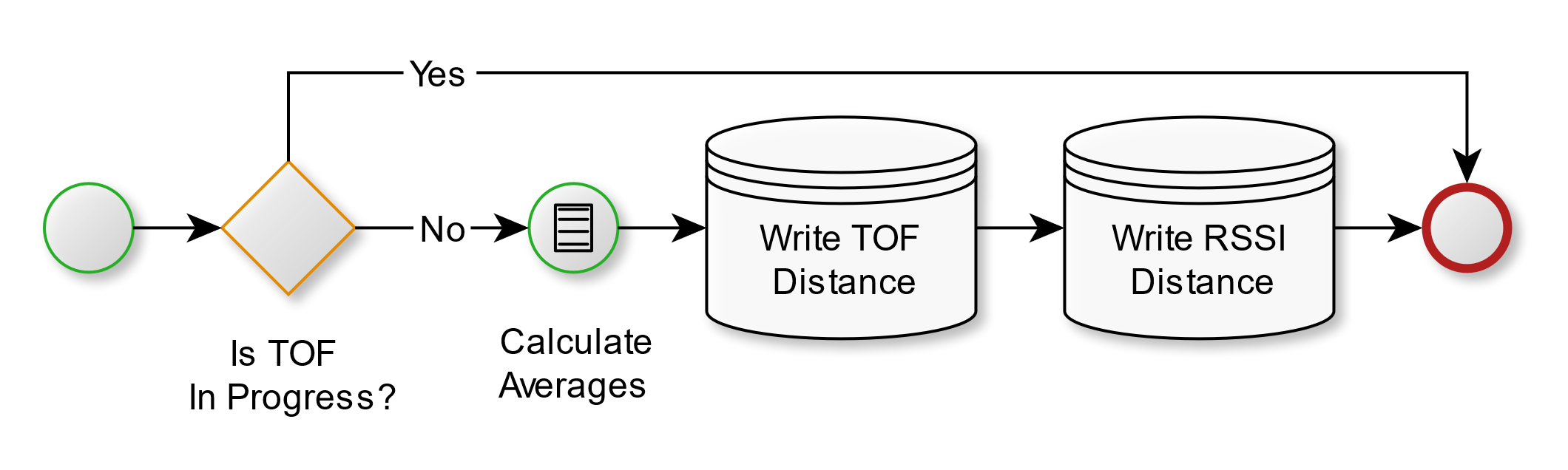


Figure 16: Beacon - Calculate Average Distances

### Transmit TOF Data Task

This task transmits the TOF distances to the Tracked Object.

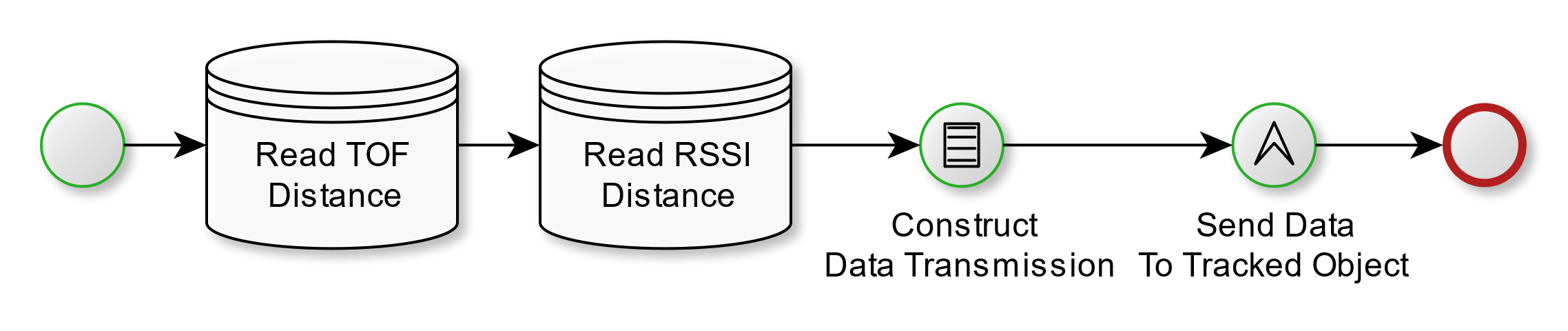


Figure 17: Beacon - TOF Echo Task Flow

# Task Schedule

## Tracked Object (coordinator.c) Schedule

This module is currently using a Round-Robin with Interrupts scheduling method. The main loop treats the following tasks in order:

* Write Display Task (lcd\_BuildStatusScreen)
* Receive Time of Flight Distance Transmission from Beacon (vProcessEventQueues)
* Triangulate Location Task (task\_CalculateXYPos)

The project had originally intended to use a fixed schedule that executed completely once per second. The schedule was to be divided into twenty 50 ms execution blocks. The originally proposed schedule is diagramed below:

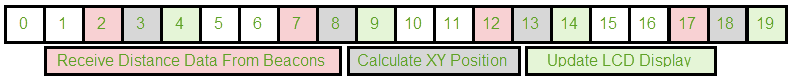


Figure 18: Originally Proposed Tracked Object Schedule

## Beacon (enddevice.c) Schedule

This module is currently using a Round-Robin with Interrupts scheduling method. The main loop treats the following tasks in order:

* Start TOF Burst Measurement Task (task\_StartTof)
* Calculate Distance Task (task\_CalculateDistance)
* Transmit TOF Data Task (tx\_Distance)

The project had originally intended to use a fixed schedule that executed completely once per second. The schedule was to be divided into twenty 50 ms execution blocks. The originally proposed schedule is diagramed below:

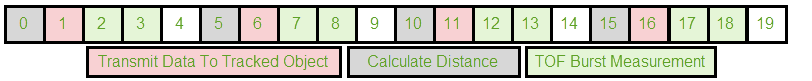


Figure 19: Originally Proposed Beacon Schedule

# Processor Bandwidth Estimation

The following estimations were calculated using the originally proposed schedules assuming each task used the entirety of its time slot.

* Tracked Object Processor Bandwidth Estimation: 60%
* Beacon Processor Bandwidth Estimation: 80%

# Memory Estimate

## Tracked Object (coordinator.c) Memory Utilization Estimate

* Non-Volatile: 504 bytes
* Volatile: 26 kilobytes

## Beacon (enddevice.c) Memory Utilization Estimate

* Non-Volatile: 486 bytes
* Volatile:

# Real Time Operating System (RTOS) Justification

The NXP JN-5148 wireless microcontroller is supplied with a complete SDK that includes the Eclipse Integrated Development Environment, a flash programmer application, a development shell with compiler, and an internally developed RTOS called JenOS. The MAC networking SDK for messaging over RF and Time-Of-Flight (TOF) ranging SDK are extensions to JenOS. Because these two critical SDKs are tied to JenOS, the project will use an RTOS. Use of the JenOS RTOS also provides the additional benefit of being able to define callback functions to handle processing of asynchronous SDK. An example of this would be the instruction to the TOF Ranging engine to perform a burst of 20 measurements. The processor offloads this work to the transceiver module and continues processing until an interrupt occurs with the set of results.

# Microcontroller Block Diagram

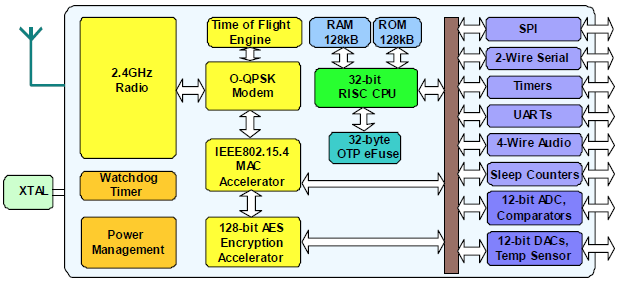


Figure 20: JN5148-001 Block Diagram (NXP, 2010)

# Memory Map – ROM – Tracked Object (coordinator.c)

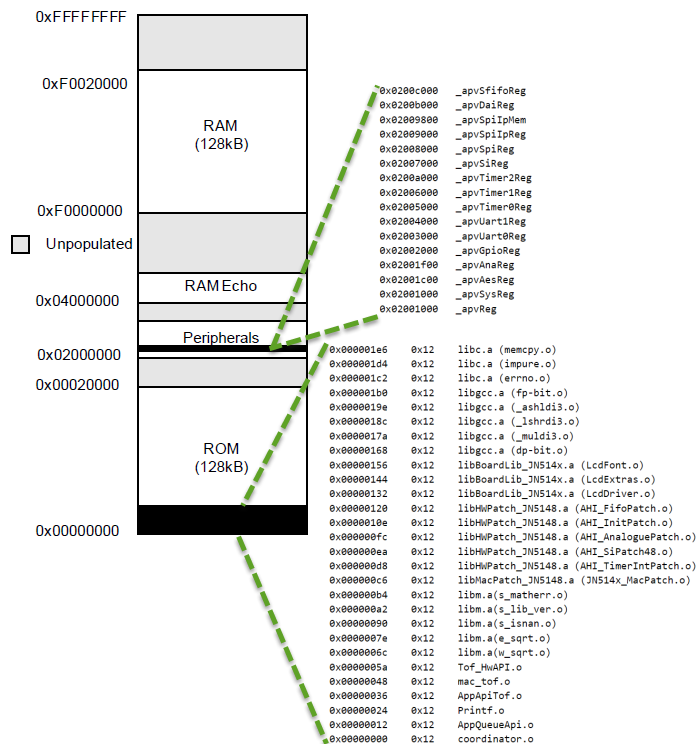


Figure 21: Memory Map – Tracked Object (coordinator.c) – ROM

# Memory Map – RAM – Tracked Object (coordinator.c)

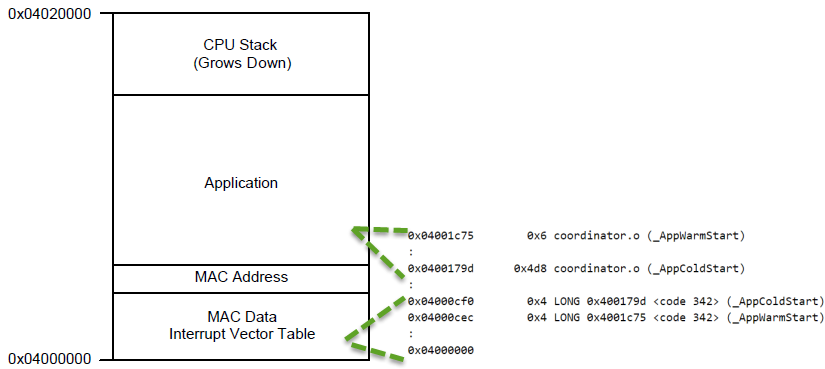


Figure 22: Memory Map – Tracked Object (coordinator.c) – RAM

# Memory Map – ROM – Beacon (enddevice.c)

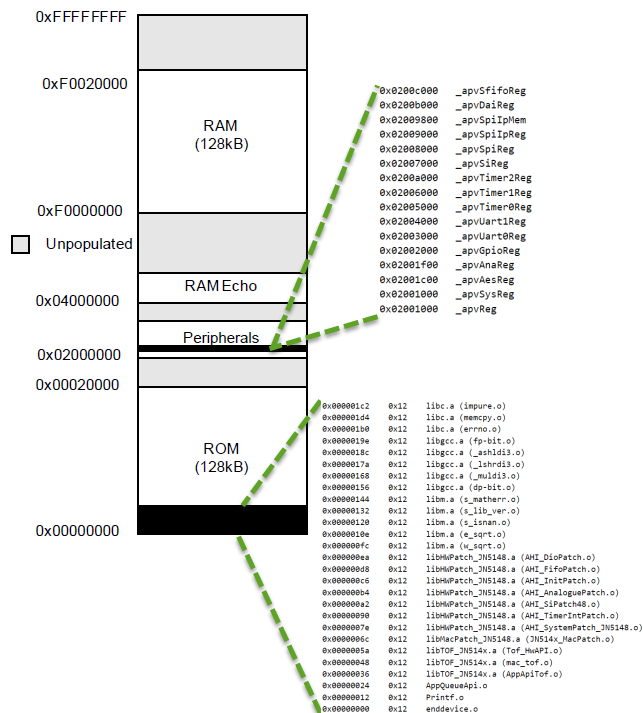


Figure 23: Memory Map – Beacon (enddevice.c) - ROM

# Memory Map – RAM – Beacon (enddevice.c)

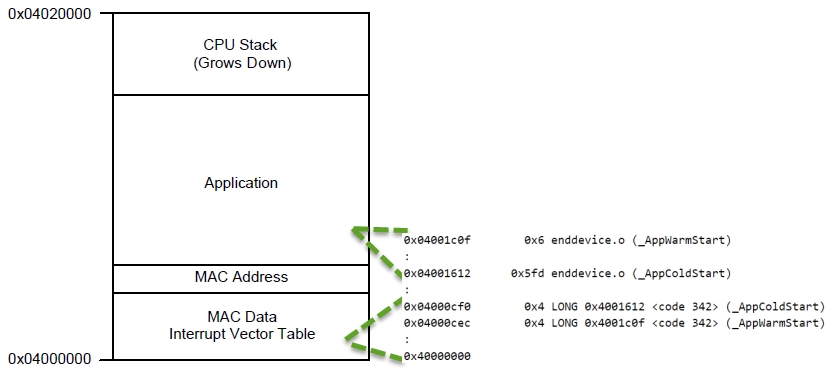


Figure 24: Memory Map – Beacon (enddevice.c) - RAM

# Hardware I/O Map

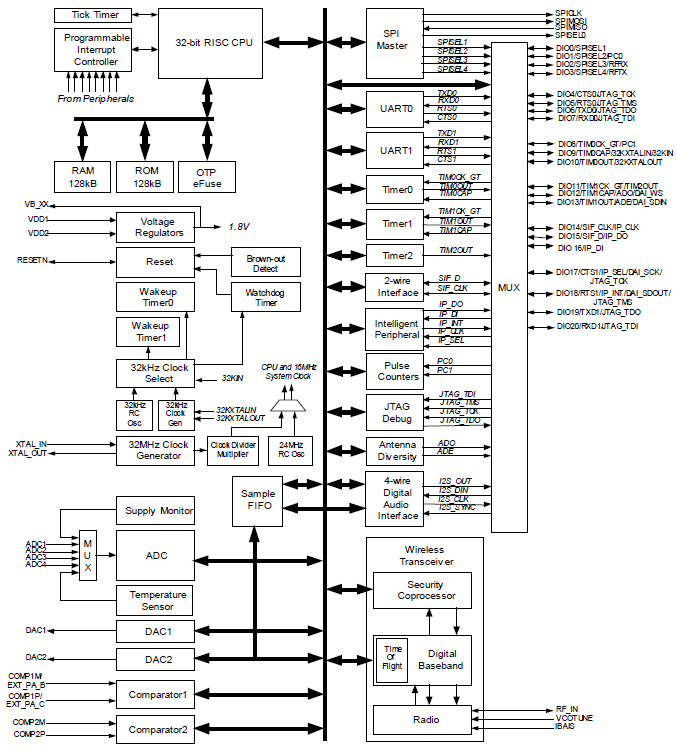


Figure 25: JN5148 Hardware I/O Map (NXP, 2010)

# Hardware-Imposed Software Constraints

## Beacon Constraints

* Maximum packet payload of 90 bytes from beacon to tracked object unencrypted. (NXP, 2010)
* Maximum packet payload of 82 bytes from beacon to beacon unencrypted. (NXP, 2010)

## Tracked Object Constraints

* Maximum packet payload of 89 bytes when broadcasting from tracked object to all beacons. (NXP, 2010)
* Maximum packet payload of 82 bytes from tracked object to beacon. (NXP, 2010)

# Software-Imposed Hardware Constraints

## Beacon Constraints

* UART port for serial debugging and programming.
* Two LEDs, minimum, for indication of beacon role assignment and flashing indication of continued execution.

## Tracked Object Constraints

* 8 row, 127 column LCD Display for displaying current state to the user.
* LCD Display must have sub-100ms response time for refreshes.
* Three buttons, minimum, for setting RF channel.
* Three LEDs, minimum for indicating network initialization, beacon 0 network registration and beacon 1 network registration.
* UART port for serial debugging and programming.

# References

Adafruit. (2018). *Adafruit Feather 32u4 RFM95 LoRa Radio- 868 or 915 MHz - RadioFruit.* Retrieved 05 09, 2018, from Adafruit.com: https://www.adafruit.com/product/3078

Adafruit. (2018). *Adafruit Feather M0 with RFM95 LoRa Radio - 900MHz - RadioFruit.* Retrieved 05 08, 2018, from Adafruit.com: https://www.adafruit.com/product/3178

ARM. (2018). *Cortex-M3*. Retrieved 05 09, 2018, from arm Developer: https://developer.arm.com/products/processors/cortex-m/cortex-m3

Beyond Semiconductor. (2014, 12). *BEYOND BA20 PIPELINEZERO PROCESSOR.* Retrieved 05 09, 2018, from Beyond Semiconductor: https://www.beyondsemi.com/107/snemi-dokument/beyond-ba20-datasheet/34/

Electronic, J. (2011, 03 20). *Distance Measurement with Radio Waves*. Retrieved 04 25, 2018, from Instructables.com: http://www.instructables.com/id/Distance-measurement-with-radio-waves/

Microchip. (2016, 04). *ATmega16U4/ATmega32U4 8-bit Microcontroller with 16/32K bytes of ISP Flash and USB Controller Datasheet.* Retrieved 05 08, 2018, from Microchip: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4\_Datasheet.pdf

Microchip. (2018, 04). *SAM D21 Family Low-Power, 32-bit Cortex-M0+ MCU with Advanced Analog.* Retrieved 05 08, 2018, from Microchip: http://ww1.microchip.com/downloads/en/DeviceDoc/SAM-D21-Family-Datasheet-DS40001882B.pdf

NXP. (2010, 04 02). Data Sheet: JN5148-001 IEEE802.15.4 Wireless Microcontroller. Retrieved 04 25, 2018

NXP. (2010, 09 28). JenNet Stack User Guide. Retrieved 05 09, 2018

NXP. (2010, 03 16). JN5148-EK010 ZigBee PRO Evaluation Kit. Retrieved 04 27, 2018

NXP. (2013, 12 06). *Data Sheet: JN516x IEEE802.15.4 Wireless Microcontroller.* Retrieved 05 09, 2018, from NXP.com: https://www.nxp.com/docs/en/data-sheet/JN516X.pdf

NXP. (2017, 11 7). *JN517X IEEE802.15.4 Wireless Microcontroller Product Data Sheet.* Retrieved 05 09, 2018, from NXP: https://www.nxp.com/docs/en/data-sheet/JN517X.pdf

Tipler, P. A. (1999). *Physics for Scientists and Engineers* (Fourth ed.). New York, NY, U.S.A.: W.H. Freeman and Company. Retrieved 04 25, 2018

wikiHow. (2018, 04 20). *How to Find the Height of a Triangle*. Retrieved 04 25, 2018, from wikiHow: https://www.wikihow.com/Find-the-Height-of-a-Triangle

# Appendix A – Tracked Object (coordinator.c)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Include files \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <jendefs.h>

#include <AppHardwareApi.h>

#include <AppQueueApi.h>

#include <mac\_sap.h>

#include <mac\_pib.h>

#include <AppApiTof.h>

#include <LedControl.h>

#include "LcdDriver.h"

#include "config.h"

#include "Printf.h"

#include <math.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Macro Definitions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define UART E\_AHI\_UART\_0

#define BYTE\_TO\_BINARY\_PATTERN "%c%c%c%c%c%c%c%c"

#define BYTE\_TO\_BINARY(byte) \

(byte & 0x80 ? '1' : '0'), \

(byte & 0x40 ? '1' : '0'), \

(byte & 0x20 ? '1' : '0'), \

(byte & 0x10 ? '1' : '0'), \

(byte & 0x08 ? '1' : '0'), \

(byte & 0x04 ? '1' : '0'), \

(byte & 0x02 ? '1' : '0'), \

(byte & 0x01 ? '1' : '0')

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Type Definitions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef enum

{

E\_STATE\_IDLE,

E\_STATE\_ENERGY\_SCANNING,

E\_STATE\_COORDINATOR\_STARTED,

}teState;

/\* Data type for storing data related to all end devices that have associated \*/

typedef struct

{

bool\_t bIsAssociated;

int32 i32TofDistance;

uint32 u32RssiDistance;

uint16 u16ShortAdr;

uint32 u32ExtAdrL;

uint32 u32ExtAdrH;

uint8 u8TxPacketSeqNb;

uint8 u8RxPacketSeqNb;

}tsEndDeviceData;

typedef struct

{

/\* Data related to associated end devices \*/

uint16 u16NbrEndDevices;

tsEndDeviceData sEndDeviceData[MAX\_END\_DEVICES];

teState eState;

uint8 u8Channel;

double x;

double y;

}tsCoordinatorData;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Local Function Prototypes \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vInitSystem(void);

PRIVATE void vStartEnergyScan(void);

PRIVATE void vStartCoordinator(void);

PRIVATE void vProcessEventQueues(void);

PRIVATE void vProcessIncomingMlme(MAC\_MlmeDcfmInd\_s \*psMlmeInd);

PRIVATE void vProcessIncomingMcps(MAC\_McpsDcfmInd\_s \*psMcpsInd);

PRIVATE void vProcessIncomingHwEvent(AppQApiHwInd\_s \*psAHI\_Ind);

PRIVATE void vHandleNodeAssociation(MAC\_MlmeDcfmInd\_s \*psMlmeInd);

PRIVATE void vHandleEnergyScanResponse(MAC\_MlmeDcfmInd\_s \*psMlmeInd);

PRIVATE void vHandleMcpsDataInd(MAC\_McpsDcfmInd\_s \*psMcpsInd);

PRIVATE void vHandleMcpsDataDcfm(MAC\_McpsDcfmInd\_s \*psMcpsInd);

PRIVATE void vProcessReceivedDataPacket(uint8 \*pu8Data, uint8 u8Len, uint16 u16Address);

PRIVATE void vPutChar(unsigned char c);

PRIVATE void reverse(char \*str, int len);

PRIVATE int intToStr(uint32 x, char str[], int d);

PRIVATE uint32 GetDistance(uint16 iEndDevice);

PRIVATE void lcd\_BuildStatusScreen(void);

PRIVATE void lcd\_UpdateStatusScreen(void);

PRIVATE void interrupt\_handleDistanceTransmissionReceived(uint8 \*pu8Data, uint8 u8Len, uint16 u16Address);

PRIVATE void task\_CalculateXYPos(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Local Variables \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Handles from the MAC \*/

PRIVATE void \*s\_pvMac;

PRIVATE MAC\_Pib\_s \*s\_psMacPib;

PRIVATE tsCoordinatorData sCoordinatorData;

PRIVATE bool\_t bLedState;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Exported Functions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: AppColdStart

\*

\* DESCRIPTION:

\* Entry point for application from boot loader. Initialises system and runs

\* main loop.

\*

\* RETURNS:

\* Never returns.

\*

\* NOTES: Demo Application Boiler Plate.

\* Modified to call custom code.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PUBLIC void AppColdStart(void)

{

volatile int n=0;

#ifdef WATCHDOG\_ENABLED

vAHI\_WatchdogStop();

#endif

vAHI\_UartEnable(UART);

vAHI\_UartReset(UART, TRUE, TRUE);

vAHI\_UartSetClockDivisor(UART, E\_AHI\_UART\_RATE\_38400);

vAHI\_UartReset(UART, FALSE, FALSE);

vInitSystem();

vInitPrintf((void \*)vPutChar);

vLcdResetDefault();

lcd\_BuildStatusScreen();

//Enable TOF ranging.

vAppApiTofInit(TRUE);

vStartEnergyScan();

vLedInitRfd();

while (1)

{

for(n=0;n<1000000;n++);

bLedState = !bLedState;

vLedControl(0, bLedState);

lcd\_BuildStatusScreen();

vProcessEventQueues();

task\_CalculateXYPos();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: AppWarmStart

\*

\* DESCRIPTION:

\* Entry point for application from boot loader. Simply jumps to AppColdStart

\* as, in this instance, application will never warm start.

\*

\* RETURNS:

\* Never returns.

\*

\* NOTES: Demo Application Boiler Plate.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PUBLIC void AppWarmStart(void)

{

AppColdStart();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Local Functions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vInitSystem

\*

\* DESCRIPTION:

\* Initializes hardware and initial data values.

\*

\* RETURNS:

\* void

\*

\* NOTES: Demo Application Boiler Plate for system initialization.

\* Modified to initialize new data added to the application.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vInitSystem(void)

{

/\* Setup interface to MAC \*/

(void)u32AppQApiInit(NULL, NULL, NULL);

(void)u32AHI\_Init();

/\* Initialise coordinator state \*/

sCoordinatorData.eState = E\_STATE\_IDLE;

sCoordinatorData.u16NbrEndDevices = 0;

int i;

for (i=0; i<MAX\_END\_DEVICES; i++)

{

sCoordinatorData.sEndDeviceData[i].bIsAssociated = FALSE;

sCoordinatorData.sEndDeviceData[i].i32TofDistance = 0;

sCoordinatorData.sEndDeviceData[i].u32RssiDistance = 0;

sCoordinatorData.sEndDeviceData[i].u8RxPacketSeqNb = 0;

sCoordinatorData.sEndDeviceData[i].u8TxPacketSeqNb = 0;

}

/\* Set up the MAC handles. Must be called AFTER u32AppQApiInit() \*/

s\_pvMac = pvAppApiGetMacHandle();

s\_psMacPib = MAC\_psPibGetHandle(s\_pvMac);

/\* Set Pan ID and short address in PIB (also sets match registers in hardware) \*/

MAC\_vPibSetPanId(s\_pvMac, PAN\_ID);

MAC\_vPibSetShortAddr(s\_pvMac, COORDINATOR\_ADR);

/\* Enable receiver to be on when idle \*/

MAC\_vPibSetRxOnWhenIdle(s\_pvMac, TRUE, FALSE);

/\* Allow nodes to associate \*/

s\_psMacPib->bAssociationPermit = 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessEventQueues

\*

\* DESCRIPTION:

\* Check each of the three event queues and process and items found.

\*

\* PARAMETERS: Name RW Usage

\* void

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling event queues.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessEventQueues(void)

{

MAC\_MlmeDcfmInd\_s \*psMlmeInd;

MAC\_McpsDcfmInd\_s \*psMcpsInd;

AppQApiHwInd\_s \*psAHI\_Ind;

/\* Check for anything on the MCPS upward queue \*/

do

{

psMcpsInd = psAppQApiReadMcpsInd();

if (psMcpsInd != NULL)

{

vProcessIncomingMcps(psMcpsInd);

vAppQApiReturnMcpsIndBuffer(psMcpsInd);

}

} while (psMcpsInd != NULL);

/\* Check for anything on the MLME upward queue \*/

do

{

psMlmeInd = psAppQApiReadMlmeInd();

if (psMlmeInd != NULL)

{

vProcessIncomingMlme(psMlmeInd);

vAppQApiReturnMlmeIndBuffer(psMlmeInd);

}

} while (psMlmeInd != NULL);

/\* Check for anything on the AHI upward queue \*/

do

{

psAHI\_Ind = psAppQApiReadHwInd();

if (psAHI\_Ind != NULL)

{

vProcessIncomingHwEvent(psAHI\_Ind);

vAppQApiReturnHwIndBuffer(psAHI\_Ind);

}

} while (psAHI\_Ind != NULL);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessIncomingMlme

\*

\* DESCRIPTION:

\* Process any incoming managment events from the stack.

\*

\* PARAMETERS: Name RW Usage

\* psMlmeInd

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling incoming 802.15.4

\* managment events.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessIncomingMlme(MAC\_MlmeDcfmInd\_s \*psMlmeInd)

{

switch (psMlmeInd->u8Type)

{

case MAC\_MLME\_IND\_ASSOCIATE: /\* Incoming association request \*/

if (sCoordinatorData.eState == E\_STATE\_COORDINATOR\_STARTED)

{

vHandleNodeAssociation(psMlmeInd);

}

break;

case MAC\_MLME\_DCFM\_SCAN: /\* Incoming scan results \*/

if (psMlmeInd->uParam.sDcfmScan.u8ScanType == MAC\_MLME\_SCAN\_TYPE\_ENERGY\_DETECT)

{

if (sCoordinatorData.eState == E\_STATE\_ENERGY\_SCANNING)

{

/\* Process energy scan results and start device as coordinator \*/

vHandleEnergyScanResponse(psMlmeInd);

}

}

break;

default:

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessIncomingData

\*

\* DESCRIPTION:

\* Process incoming data events from the stack.

\*

\* PARAMETERS: Name RW Usage

\* psMcpsInd

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling incoming 802.15.4 data.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessIncomingMcps(MAC\_McpsDcfmInd\_s \*psMcpsInd)

{

/\* Only handle incoming data events one device has been started as a

coordinator \*/

if (sCoordinatorData.eState >= E\_STATE\_COORDINATOR\_STARTED)

{

switch(psMcpsInd->u8Type)

{

case MAC\_MCPS\_IND\_DATA: /\* Incoming data frame \*/

vHandleMcpsDataInd(psMcpsInd);

break;

case MAC\_MCPS\_DCFM\_DATA: /\* Incoming acknowledgement or ack timeout \*/

vHandleMcpsDataDcfm(psMcpsInd);

break;

default:

break;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleMcpsDataDcfm

\*

\* DESCRIPTION:

\*

\* PARAMETERS: Name RW Usage

\* psMcpsInd

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling 802.15.4 Data

\* Acknowledgements.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleMcpsDataDcfm(MAC\_McpsDcfmInd\_s \*psMcpsInd)

{

if (psMcpsInd->uParam.sDcfmData.u8Status == MAC\_ENUM\_SUCCESS)

{

/\* Data frame transmission successful \*/

}

else

{

/\* Data transmission falied after 3 retries at MAC layer. \*/

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleMcpsDataInd

\*

\* DESCRIPTION:

\*

\* PARAMETERS: Name RW Usage

\* psMcpsInd

\* RETURNS:

\*

\* NOTES: Demo Application Boiler Plate for handling 802.15.4 Received Data

\* Packet Events.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleMcpsDataInd(MAC\_McpsDcfmInd\_s \*psMcpsInd)

{

MAC\_RxFrameData\_s \*psFrame;

psFrame = &psMcpsInd->uParam.sIndData.sFrame;

/\* Check application layer sequence number of frame and reject if it is

the same as the last frame, i.e. same frame has been received more

than once. \*/

uint16 u16EndDeviceIndex = psFrame->sSrcAddr.uAddr.u16Short - 1;

if (psFrame->au8Sdu[0] >= sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u8RxPacketSeqNb)

{

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u8RxPacketSeqNb++;

vProcessReceivedDataPacket(&psFrame->au8Sdu[1],

(psFrame->u8SduLength) - 1,

psFrame->sSrcAddr.uAddr.u16Short);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessReceivedDataPacket

\*

\* DESCRIPTION:

\*

\* PARAMETERS: Name RW Usage

\* pu8Data Packet Data Received

\* u8Len Size of Data Array.

\* u16Address Source Address

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling Received data packets.

\* Modified to call custom data handler.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessReceivedDataPacket(uint8 \*pu8Data, uint8 u8Len, uint16 u16Address)

{

vPrintf("\nReceived data Packet %i long\n", u8Len);

if (u8Len >= 1)

{

uint8 firstByte = pu8Data[0];

switch(firstByte)

{

case 0xd1:

interrupt\_handleDistanceTransmissionReceived(&pu8Data[1], u8Len-1, u16Address);

break;

default:

vPrintf("Unexpected data packet.\n");

break;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: interrupt\_handleDistanceTransmissionReceived

\*

\* DESCRIPTION:

\* Data handler for Distance Transmission Received Event.

\*

\* PARAMETERS: Name RW Usage

\* pu8Data Packet Data Received

\* u8Len Size of Data Array.

\* u16Address Source Address

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void interrupt\_handleDistanceTransmissionReceived(uint8 \*pu8Data, uint8 u8Len, uint16 u16Address)

{

#ifdef DEBUG\_DISTANCE\_TRANSMISSION

vPrintf("TOF Byte0: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Data[0]));

vPrintf("TOF Byte1: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Data[1]));

vPrintf("TOF Byte2: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Data[2]));

vPrintf("TOF Byte3: "BYTE\_TO\_BINARY\_PATTERN"\n\n", BYTE\_TO\_BINARY(pu8Data[3]));

vPrintf("RSSI Byte0: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Data[4]));

vPrintf("RSSI Byte1: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Data[5]));

vPrintf("RSSI Byte2: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Data[6]));

vPrintf("RSSI Byte3: "BYTE\_TO\_BINARY\_PATTERN"\n\n", BYTE\_TO\_BINARY(pu8Data[7]));

#endif

uint32 highByte = ((uint32)pu8Data[0]) << 24;

uint32 midHighByte = ((uint32)pu8Data[1]) << 16;

uint32 midLowByte = ((uint32)pu8Data[2]) << 8;

uint32 lowByte = ((uint32)pu8Data[3]);

uint16 u16EndDeviceIndex = u16Address - 1;

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].i32TofDistance = ((int32)highByte) | midHighByte | midLowByte | lowByte;

highByte = ((uint32)pu8Data[4]) << 24;

midHighByte = ((uint32)pu8Data[5]) << 16;

midLowByte = ((uint32)pu8Data[6]) << 8;

lowByte = ((uint32)pu8Data[7]);

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u32RssiDistance = highByte | midHighByte | midLowByte | lowByte;

vPrintf("\nDistance Transmission Received From Beacon %i.\nTOF Distance: %i cm\nRSSI Distance: %i cm\n", u16Address, sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].i32TofDistance, sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u32RssiDistance);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessIncomingHwEvent

\*

\* DESCRIPTION:

\* Process any hardware events.

\*

\* PARAMETERS: Name RW Usage

\* psAHI\_Ind

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessIncomingHwEvent(AppQApiHwInd\_s \*psAHI\_Ind)

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleNodeAssociation

\*

\* DESCRIPTION:

\* Handle request by node to join the network.

\*

\* PARAMETERS: Name RW Usage

\* psMlmeInd

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling Node Association.

\* Modified to set additional value for display purposes.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleNodeAssociation(MAC\_MlmeDcfmInd\_s \*psMlmeInd)

{

uint16 u16ShortAdr = 0xffff;

uint16 u16EndDeviceIndex;

MAC\_MlmeReqRsp\_s sMlmeReqRsp;

MAC\_MlmeSyncCfm\_s sMlmeSyncCfm;

if (sCoordinatorData.u16NbrEndDevices < MAX\_END\_DEVICES)

{

/\* Store end device address data \*/

u16EndDeviceIndex = sCoordinatorData.u16NbrEndDevices;

u16ShortAdr = END\_DEVICE\_START\_ADR + sCoordinatorData.u16NbrEndDevices;

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u16ShortAdr = u16ShortAdr;

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u32ExtAdrL =

psMlmeInd->uParam.sIndAssociate.sDeviceAddr.u32L;

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].u32ExtAdrH =

psMlmeInd->uParam.sIndAssociate.sDeviceAddr.u32H;

sCoordinatorData.sEndDeviceData[u16EndDeviceIndex].bIsAssociated = TRUE;

vPrintf("Beacon %i Associated: %i\n", u16EndDeviceIndex, u16ShortAdr);

sCoordinatorData.u16NbrEndDevices++;

sMlmeReqRsp.uParam.sRspAssociate.u8Status = 0; /\* Access granted \*/

}

else

{

sMlmeReqRsp.uParam.sRspAssociate.u8Status = 2; /\* Denied \*/

}

/\* Create association response \*/

sMlmeReqRsp.u8Type = MAC\_MLME\_RSP\_ASSOCIATE;

sMlmeReqRsp.u8ParamLength = sizeof(MAC\_MlmeRspAssociate\_s);

sMlmeReqRsp.uParam.sRspAssociate.sDeviceAddr.u32H = psMlmeInd->uParam.sIndAssociate.sDeviceAddr.u32H;

sMlmeReqRsp.uParam.sRspAssociate.sDeviceAddr.u32L = psMlmeInd->uParam.sIndAssociate.sDeviceAddr.u32L;

sMlmeReqRsp.uParam.sRspAssociate.u16AssocShortAddr = u16ShortAdr;

sMlmeReqRsp.uParam.sRspAssociate.u8SecurityEnable = FALSE;

/\* Send association response. There is no confirmation for an association

response, hence no need to check \*/

vAppApiMlmeRequest(&sMlmeReqRsp, &sMlmeSyncCfm);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vStartEnergyScan

\*

\* DESCRIPTION:

\* Starts an enery scan on the channels specified.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling 802.15.4 Energy Scans.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vStartEnergyScan(void)

{

/\* Structures used to hold data for MLME request and response \*/

MAC\_MlmeReqRsp\_s sMlmeReqRsp;

MAC\_MlmeSyncCfm\_s sMlmeSyncCfm;

sCoordinatorData.eState = E\_STATE\_ENERGY\_SCANNING;

/\* Start energy detect scan \*/

sMlmeReqRsp.u8Type = MAC\_MLME\_REQ\_SCAN;

sMlmeReqRsp.u8ParamLength = sizeof(MAC\_MlmeReqStart\_s);

sMlmeReqRsp.uParam.sReqScan.u8ScanType = MAC\_MLME\_SCAN\_TYPE\_ENERGY\_DETECT;

sMlmeReqRsp.uParam.sReqScan.u32ScanChannels = SCAN\_CHANNELS;

sMlmeReqRsp.uParam.sReqScan.u8ScanDuration = ENERGY\_SCAN\_DURATION;

vAppApiMlmeRequest(&sMlmeReqRsp, &sMlmeSyncCfm);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleEnergyScanResponse

\*

\* DESCRIPTION:

\* Selects a channel with low enery content for use by the wireless UART.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for handling 802.15.4 Energy Scan

\* Response.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleEnergyScanResponse(MAC\_MlmeDcfmInd\_s \*psMlmeInd)

{

uint8 u8MinEnergy;

uint8 i;

u8MinEnergy = (psMlmeInd->uParam.sDcfmScan.uList.au8EnergyDetect[0]) ;

sCoordinatorData.u8Channel = CHANNEL\_MIN;

/\* Search list to find quietest channel \*/

while (i < psMlmeInd->uParam.sDcfmScan.u8ResultListSize)

{

if ((psMlmeInd->uParam.sDcfmScan.uList.au8EnergyDetect[i]) < u8MinEnergy)

{

u8MinEnergy = (psMlmeInd->uParam.sDcfmScan.uList.au8EnergyDetect[i]);

sCoordinatorData.u8Channel = i + CHANNEL\_MIN;

}

i++;

}

vStartCoordinator();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vStartCoordinator

\*

\* DESCRIPTION:

\* Starts the network by configuring the controller board to act as the PAN

\* coordinator.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS:

\* TRUE if network was started successfully otherwise FALSE

\*

\* NOTES: Demo Application Boiler Plate for starting 802.15.4 Network.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vStartCoordinator(void)

{

/\* Structures used to hold data for MLME request and response \*/

MAC\_MlmeReqRsp\_s sMlmeReqRsp;

MAC\_MlmeSyncCfm\_s sMlmeSyncCfm;

sCoordinatorData.eState = E\_STATE\_COORDINATOR\_STARTED;

/\* Start Pan \*/

sMlmeReqRsp.u8Type = MAC\_MLME\_REQ\_START;

sMlmeReqRsp.u8ParamLength = sizeof(MAC\_MlmeReqStart\_s);

sMlmeReqRsp.uParam.sReqStart.u16PanId = PAN\_ID;

sMlmeReqRsp.uParam.sReqStart.u8Channel = sCoordinatorData.u8Channel;

sMlmeReqRsp.uParam.sReqStart.u8BeaconOrder = 0x0F;

sMlmeReqRsp.uParam.sReqStart.u8SuperframeOrder = 0x0F;

sMlmeReqRsp.uParam.sReqStart.u8PanCoordinator = TRUE;

sMlmeReqRsp.uParam.sReqStart.u8BatteryLifeExt = FALSE;

sMlmeReqRsp.uParam.sReqStart.u8Realignment = FALSE;

sMlmeReqRsp.uParam.sReqStart.u8SecurityEnable = FALSE;

vAppApiMlmeRequest(&sMlmeReqRsp, &sMlmeSyncCfm);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: lcd\_BuildStatusScreen

\*

\* DESCRIPTION:

\* Builds the LCD output presented to the user.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void lcd\_BuildStatusScreen(void)

{

#ifdef DEBUG\_LCD

vPrintf("lcd\_BuildStatusScreen\n");

#endif

vLcdClear();

vLcdWriteText("Esten Rye", 0, 0);

vLcdWriteTextRightJustified("SEIS 740", 0, 127);

vLcdWriteText("TOF Triangulation", 1, 0);

vLcdWriteText("Node 0:", 2, 0);

vLcdWriteTextRightJustified("Off", 2, 60);

vLcdWriteText("Node 1:", 2, 64);

vLcdWriteTextRightJustified("Off", 2, 123);

vLcdWriteText("A:", 3, 0);

vLcdWriteText("B:", 4, 0);

vLcdWriteText("C:", 5, 0);

vLcdWriteText("X:", 6, 0);

vLcdWriteText("Y:", 7, 0);

lcd\_UpdateStatusScreen();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: GetDistance

\*

\* DESCRIPTION:

\* Retrieves a distance measurement for a specified end device.

\*

\* PARAMETERS: Name RW Usage

\* iEndDevice index of the end device in the routing table to use.

\*

\* RETURNS: uint32 distance result.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE uint32 GetDistance(uint16 iEndDevice)

{

uint32 distance;

if (sCoordinatorData.sEndDeviceData[iEndDevice].i32TofDistance < 50)

{

distance = sCoordinatorData.sEndDeviceData[iEndDevice].u32RssiDistance;

}

else

{

distance = sCoordinatorData.sEndDeviceData[iEndDevice].i32TofDistance;

}

return distance;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: lcd\_UpdateStatusScreen

\*

\* DESCRIPTION:

\* Updates the LCD output presented to the user.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void lcd\_UpdateStatusScreen(void)

{

#ifdef DEBUG\_LCD

vPrintf("lcd\_UpdateStatusScreen\n");

#endif

bool\_t beacon0Assigned = sCoordinatorData.sEndDeviceData[0].bIsAssociated;

bool\_t beacon1Assigned = sCoordinatorData.sEndDeviceData[1].bIsAssociated;

#ifdef DEBUG\_LCD

vPrintf("Beacon 0 Associated: %i\n", beacon0Assigned);

vPrintf("Beacon 1 Associated: %i\n", beacon1Assigned);

#endif

if (beacon0Assigned)

{

vLcdWriteTextRightJustified(" On", 2, 60);

}

else

{

vLcdWriteTextRightJustified("Off", 2, 60);

}

if (beacon1Assigned)

{

vLcdWriteTextRightJustified(" On", 2, 123);

}

else

{

vLcdWriteTextRightJustified("Off", 2, 123);

}

char output[20];

intToStr(GetDistance(0), output, 0);

vLcdWriteTextRightJustified(output, 3, 127);

intToStr(GetDistance(1), output, 0);

vLcdWriteTextRightJustified(output, 4, 127);

vLcdWriteTextRightJustified("120", 5, 127);

intToStr((int)sCoordinatorData.x, output, 0);

vLcdWriteTextRightJustified(output, 6, 127);

intToStr((int)sCoordinatorData.y, output, 0);

vLcdWriteTextRightJustified(output, 7, 127);

vLcdRefreshAll();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vPutChar

\*

\* DESCRIPTION:

\* Updates the UART output presented to the user.

\*

\* PARAMETERS: Name RW Usage

\* c

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for supporting vPrintf functionality.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vPutChar(unsigned char c) {

while ((u8AHI\_UartReadLineStatus(UART) & E\_AHI\_UART\_LS\_THRE) == 0);

vAHI\_UartWriteData(UART, c);

while ((u8AHI\_UartReadLineStatus(UART) & (E\_AHI\_UART\_LS\_THRE | E\_AHI\_UART\_LS\_TEMT)) != (E\_AHI\_UART\_LS\_THRE | E\_AHI\_UART\_LS\_TEMT));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vPutChar

\*

\* DESCRIPTION:

\* reverses a string 'str' of length 'len'

\*

\* PARAMETERS: Name RW Usage

\* str Y string to reverse

\* len N length of the string.

\*

\* RETURNS: void

\*

\* NOTES: retrieved from https://www.geeksforgeeks.org/convert-floating-point-number-string/

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void reverse(char \*str, int len)

{

int i=0, j=len-1, temp;

while (i<j)

{

temp = str[i];

str[i] = str[j];

str[j] = temp;

i++; j--;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vPutChar

\*

\* DESCRIPTION:

\* Converts a given positive integer x to string str[]. d is the number

\* of digits required in output. If d is more than the number

\* of digits in x, then 0s are added at the beginning.

\*

\* PARAMETERS: Name RW Usage

\* uint32 N integer to convert.

\* str Y string to write result to

\* len N number of leading zeros.

\*

\* RETURNS: void

\*

\* NOTES: retrieved from https://www.geeksforgeeks.org/convert-floating-point-number-string/

\* modified to support uint32 numbers.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int intToStr(uint32 x, char str[], int d)

{

int i = 0;

while (x)

{

str[i++] = (x%10) + '0';

x = x/10;

}

// If number of digits required is more, then

// add 0s at the beginning

while (i < d)

{

str[i++] = '0';

}

reverse(str, i);

str[i] = '\0';

return i;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vPutChar

\*

\* DESCRIPTION:

\* Calculates the XY position of the coordinator based on time of flight data

\* received from beacon nodes.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void task\_CalculateXYPos(void)

{

int32 a = (int32)GetDistance(0);

int32 b = (int32)GetDistance(1);

int32 c = (int32)120;

vPrintf("\nCalculate XY Position\nA: %i\nB: %i\nC: %i\n", a, b, c);

if (a > 0 && b > 0)

{

int32 s = (a + b + c) / 2;

int32 n = s \* (s-a) \* (s-b) \* (s-c);

double y = 2 \* sqrt(n) / c;

double x = sqrt(pow(a, 2) - pow(y, 2));

vPrintf("N: %i\nS: %i\nX: %i\nY: %i\n", n, s, (int)x, (int)y);

sCoordinatorData.y = y;

sCoordinatorData.x = x;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* END OF FILE \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# Appendix B – Beacon (enddevice.c)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Include files \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <jendefs.h>

#include <AppHardwareApi.h>

#include <AppQueueApi.h>

#include <mac\_sap.h>

#include <mac\_pib.h>

#include <AppApiTof.h>

#include "Printf.h"

#include <Math.h>

#include <LedControl.h>

#include "config.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Macro Definitions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MAX\_READINGS 20

#define UART E\_AHI\_UART\_0

#define BYTE\_TO\_BINARY\_PATTERN "%c%c%c%c%c%c%c%c"

#define BYTE\_TO\_BINARY(byte) \

(byte & 0x80 ? '1' : '0'), \

(byte & 0x40 ? '1' : '0'), \

(byte & 0x20 ? '1' : '0'), \

(byte & 0x10 ? '1' : '0'), \

(byte & 0x08 ? '1' : '0'), \

(byte & 0x04 ? '1' : '0'), \

(byte & 0x02 ? '1' : '0'), \

(byte & 0x01 ? '1' : '0')

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Type Definitions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef enum

{

E\_STATE\_IDLE,

E\_STATE\_ACTIVE\_SCANNING,

E\_STATE\_ASSOCIATING,

E\_STATE\_ASSOCIATED

} teState;

typedef struct

{

teState eState;

uint8 u8Channel;

uint8 u8TxPacketSeqNb;

uint8 u8RxPacketSeqNb;

uint16 u16Address;

int32 i32TofDistance;

uint32 u32RssiDistance;

} tsEndDeviceData;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Local Function Prototypes \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vInitSystem(void);

PRIVATE void vProcessEventQueues(void);

PRIVATE void vProcessIncomingMlme(MAC\_MlmeDcfmInd\_s \*psMlmeInd);

PRIVATE void vProcessIncomingMcps(MAC\_McpsDcfmInd\_s \*psMcpsInd);

PRIVATE void vProcessIncomingHwEvent(AppQApiHwInd\_s \*psAHI\_Ind);

PRIVATE void vStartActiveScan(uint32 u32ChannelstoScan);

PRIVATE void vHandleActiveScanResponse(MAC\_MlmeDcfmInd\_s \*psMlmeInd);

PRIVATE void vStartAssociate(void);

PRIVATE void vHandleAssociateResponse(MAC\_MlmeDcfmInd\_s \*psMlmeInd);

PRIVATE void vHandleMcpsDataInd(MAC\_McpsDcfmInd\_s \*psMcpsInd);

PRIVATE void vHandleMcpsDataDcfm(MAC\_McpsDcfmInd\_s \*psMcpsInd);

PRIVATE void vProcessReceivedDataPacket(uint8 \*pu8Data, uint8 u8Len);

PRIVATE void vPutChar(unsigned char c);

PRIVATE void task\_StartTof(void);

PRIVATE void task\_CalculateDistance(void);

PRIVATE void tx\_Distance(int32 i32TofDistance, uint32 u32RssiDistance);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Local Variables \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Handles from the MAC \*/

PRIVATE void \*s\_pvMac;

PRIVATE MAC\_Pib\_s \*s\_psMacPib;

PRIVATE tsEndDeviceData sEndDeviceData;

PRIVATE bool\_t bLedState;

PRIVATE uint32 u32Tick = 0;

PRIVATE uint8 u8CurrentTxHandle = 0x00;

eTofReturn eTofStatus = -1;

volatile bool\_t bTofInProgress = FALSE;

tsAppApiTof\_Data asTofData[MAX\_READINGS];

/\* RSSI to Distance (cm) lookup table. Generated from formula in JN-UG-3063 \*/

uint32 au32RSSIdistance[] = { 502377, 447744, 399052, 355656, 316979, 282508,

251785, 224404, 200000, 178250, 158866, 141589, 126191, 112468, 100237,

89337, 79621, 70963, 63246, 56368, 50238, 44774, 39905, 35566, 31698,

28251, 25179, 22440, 20000, 17825, 15887, 14159, 12619, 11247, 10024,

8934, 7962, 7096, 6325, 5637, 5024, 4477, 3991, 3557, 3170, 2825, 2518,

2244, 2000, 1783, 1589, 1416, 1262, 1125, 1002, 893, 796, 710, 632,

564, 502, 448, 399, 356, 317, 283, 252, 224, 200, 178, 159, 142, 126,

112, 100, 89, 80, 71, 63, 56, 50, 45, 40, 36, 32, 28, 25, 22, 20, 18,

16, 14, 13, 11, 10, 9, 8, 7, 6, 6, 5, 4, 4, 4, 3, 3, 3, 2, 2 };

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Exported Functions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vTofCallback

\*

\* DESCRIPTION:

\* This function is passed to bAppApiGetTof. Function is called when the tof

\* readings have been completed and stored in asTofData.

\*

\* PASSED:

\* eTofReturn eStatus,

\*

\* RETURNS:

\* None

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void vTofCallback(eTofReturn eStatus)

{

eTofStatus = eStatus;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: AppColdStart

\*

\* DESCRIPTION:

\* Entry point for application from boot loader. Initialises system and runs

\* main loop.

\*

\* RETURNS:

\* Never returns.

\*

\* NOTES: Demo Application Boiler Plate

\* Modified to execute custom code.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PUBLIC void AppColdStart(void)

{

int n;

#ifdef WATCHDOG\_ENABLED

vAHI\_WatchdogStop();

#endif

vAHI\_UartEnable(UART);

vAHI\_UartReset(UART, TRUE, TRUE);

vAHI\_UartSetClockDivisor(UART, E\_AHI\_UART\_RATE\_115200);

vAHI\_UartReset(UART, FALSE, FALSE);

vInitPrintf((void \*)vPutChar);

/\* Clear screen and tabs \*/

vPrintf("\x1B[2J\x1B[H\x1B[3g");

vPrintf("Time of Flight Triangulation Demo\n");

for(n = 0; n < MAX\_READINGS; n++)

{

asTofData[n].s32Tof = 0;

asTofData[n].s8LocalRSSI = 0;

asTofData[n].u8LocalSQI = 0;

asTofData[n].s8RemoteRSSI = 0;

asTofData[n].u8RemoteSQI = 0;

asTofData[n].u32Timestamp = 0;

asTofData[n].u8Status = 0;

}

vInitSystem();

/\* Enable TOF ranging. \*/

vAppApiTofInit(TRUE);

vPrintf("Starting Scan\n");

vStartActiveScan(SCAN\_CHANNELS);

vLedInitRfd();

while (1)

{

if (u32Tick++ > (bTofInProgress ? 10000 : 100000))

{

bLedState = !bLedState;

vLedControl(0, bLedState);

u32Tick = 0;

}

if (sEndDeviceData.eState >= E\_STATE\_ASSOCIATED)

{

task\_StartTof();

/\* Check for return code to have been set in callback \*/

if (eTofStatus != -1)

{

if (eTofStatus == TOF\_SUCCESS)

{

task\_CalculateDistance();

tx\_Distance(sEndDeviceData.i32TofDistance, sEndDeviceData.u32RssiDistance);

}

else

{

vPrintf("\nToF failed with error %d", eTofStatus);

}

/\* Reset flags for next ToF burst \*/

eTofStatus = -1;

bTofInProgress = FALSE;

}

}

vProcessEventQueues();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: AppWarmStart

\*

\* DESCRIPTION:

\* Entry point for application from boot loader. Simply jumps to AppColdStart

\* as, in this instance, application will never warm start.

\*

\* RETURNS:

\* Never returns.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PUBLIC void AppWarmStart(void)

{

AppColdStart();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* Local Functions \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vInitSystem

\*

\* DESCRIPTION:

\* Initializes hardware and device state data.

\*

\* RETURNS:

\* void

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vInitSystem(void)

{

/\* Setup interface to MAC \*/

(void)u32AppQApiInit(NULL, NULL, NULL);

(void)u32AHI\_Init();

/\* Enable high power modules \*/

vAHI\_HighPowerModuleEnable(TRUE, TRUE);

/\* Initialise end device state \*/

sEndDeviceData.eState = E\_STATE\_IDLE;

sEndDeviceData.u8TxPacketSeqNb = 0;

sEndDeviceData.u8RxPacketSeqNb = 0;

/\* Set up the MAC handles. Must be called AFTER u32AppQApiInit() \*/

s\_pvMac = pvAppApiGetMacHandle();

s\_psMacPib = MAC\_psPibGetHandle(s\_pvMac);

/\* Set Pan ID in PIB (also sets match register in hardware) \*/

MAC\_vPibSetPanId(s\_pvMac, PAN\_ID);

/\* Enable receiver to be on when idle \*/

MAC\_vPibSetRxOnWhenIdle(s\_pvMac, TRUE, FALSE);

vPrintf("Done Init\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: task\_StartTof

\*

\* DESCRIPTION:

\* Starts a TOF Forward Burst measurement that takes MAX\_READINGS number of

\* measurements.

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void task\_StartTof(void)

{

// Loop an obsurdly long number of times to slow down TOF captures.

// ran out of time to do a proper scheduler

uint64 wait;

for (wait = 0; wait < 100000000000000000000000uLL; wait++);

/\* Create address for coordinator \*/

MAC\_Addr\_s sAddr;

sAddr.u8AddrMode = 2;

sAddr.u16PanId = PAN\_ID;

sAddr.uAddr.u16Short = COORDINATOR\_ADR;

if(bTofInProgress==FALSE)

{

if (bAppApiGetTof( asTofData, &sAddr, MAX\_READINGS, API\_TOF\_FORWARDS, vTofCallback))

{

vPrintf("\nForward burst started");

bTofInProgress = TRUE;

} else {

vPrintf("\nFailed to start ToF");

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: task\_CalculateDistance

\*

\* DESCRIPTION:

\* Calculates the average i32TofDistance and the average u32RssiDistance

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void task\_CalculateDistance(void)

{

int32 n, s32Mean, s32StanDev;

double dStd, dMean;

uint8 u8NumErrors;

double dAcc = 0.0;

sEndDeviceData.u32RssiDistance = 0;

u8NumErrors = 0;

vPrintf("\n\n| # \x1BH| ToF (ps) \x1BH| Lcl RSSI \x1BH| Lcl SQI \x1BH| Rmt RSSI \x1BH| Rmt SQI \x1BH| Timestamp \x1BH| Status \x1BH|");

vPrintf("\n--------------------------------------------------------------------------------");

for(n = 0; n < MAX\_READINGS; n++)

{

vPrintf("\n|%d",n);

/\* Only include successful readings \*/

if (asTofData[n].u8Status == MAC\_TOF\_STATUS\_SUCCESS)

{

dAcc += asTofData[n].s32Tof;

sEndDeviceData.u32RssiDistance += au32RSSIdistance[asTofData[n].s8LocalRSSI];

sEndDeviceData.u32RssiDistance += au32RSSIdistance[asTofData[n].s8RemoteRSSI];

vPrintf("\t|%i\t|%d\t|%d\t|%d\t|%d\t|%d\t|%d\t|",

asTofData[n].s32Tof,

asTofData[n].s8LocalRSSI,

asTofData[n].u8LocalSQI,

asTofData[n].s8RemoteRSSI,

asTofData[n].u8RemoteSQI,

asTofData[n].u32Timestamp,

asTofData[n].u8Status);

}

else

{

u8NumErrors++;

vPrintf("\t|-\t|-\t|-\t|-\t|-\t|-\t|%d\t|",

asTofData[n].u8Status);

}

}

/\* Calculate statistics \*/

if(u8NumErrors != MAX\_READINGS)

{

dMean = dAcc / (MAX\_READINGS - u8NumErrors);

/\* Calculate standard deviation = sqrt((1/N)\*(sigma(xi-xmean)2) \*/

dStd = 0.0;

/\* Accumulate sum of squared deviances \*/

for(n = 0; n < MAX\_READINGS; n++)

{

if(asTofData[n].u8Status == MAC\_TOF\_STATUS\_SUCCESS)

{

dStd += ((double)asTofData[n].s32Tof - dMean) \* ((double)asTofData[n].s32Tof - dMean);

}

}

/\* std = sqrt(mean of sum of squared deviances) \*/

dStd /= (MAX\_READINGS - u8NumErrors);

dStd = sqrt(dStd);

s32StanDev = (int32)dStd;

s32Mean = (int32)dMean;

/\* Calculate distances \*/

sEndDeviceData.i32TofDistance = dMean \* 0.03;

sEndDeviceData.u32RssiDistance /= (MAX\_READINGS - u8NumErrors) \* 2;

}

else

{

s32StanDev = 0;

s32Mean = 0;

sEndDeviceData.i32TofDistance = 0;

sEndDeviceData.u32RssiDistance = 0;

}

vPrintf("\n\nStandDev (ToF): %ips, Mean (ToF): %ips, Errors: %d",

s32StanDev,

s32Mean,

u8NumErrors);

vPrintf("\nDistance (ToF): %icm, Distance (RSSI): %dcm",

sEndDeviceData.i32TofDistance,

sEndDeviceData.u32RssiDistance);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessEventQueues

\*

\* DESCRIPTION:

\* Check each of the three event queues and process and items found.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessEventQueues(void)

{

MAC\_MlmeDcfmInd\_s \*psMlmeInd;

MAC\_McpsDcfmInd\_s \*psMcpsInd;

AppQApiHwInd\_s \*psAHI\_Ind;

/\* Check for anything on the MCPS upward queue \*/

do

{

psMcpsInd = psAppQApiReadMcpsInd();

if (psMcpsInd != NULL)

{

vProcessIncomingMcps(psMcpsInd);

vAppQApiReturnMcpsIndBuffer(psMcpsInd);

}

} while (psMcpsInd != NULL);

/\* Check for anything on the MLME upward queue \*/

do

{

psMlmeInd = psAppQApiReadMlmeInd();

if (psMlmeInd != NULL)

{

vProcessIncomingMlme(psMlmeInd);

vAppQApiReturnMlmeIndBuffer(psMlmeInd);

}

} while (psMlmeInd != NULL);

/\* Check for anything on the AHI upward queue \*/

do

{

psAHI\_Ind = psAppQApiReadHwInd();

if (psAHI\_Ind != NULL)

{

vProcessIncomingHwEvent(psAHI\_Ind);

vAppQApiReturnHwIndBuffer(psAHI\_Ind);

}

} while (psAHI\_Ind != NULL);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessIncomingHwEvent

\*

\* DESCRIPTION:

\* Process any hardware events.

\*

\* PARAMETERS: Name RW Usage

\* psAHI\_Ind

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessIncomingHwEvent(AppQApiHwInd\_s \*psAHI\_Ind)

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessIncomingMlme

\*

\* DESCRIPTION:

\* Process any incoming managment events from the stack.

\*

\* PARAMETERS: Name RW Usage

\* psMlmeInd

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessIncomingMlme(MAC\_MlmeDcfmInd\_s \*psMlmeInd)

{

/\* We respond to several MLME indications and confirmations, depending

on mode \*/

switch (psMlmeInd->u8Type)

{

/\* Deferred confirmation that the scan is complete \*/

case MAC\_MLME\_DCFM\_SCAN:

if (sEndDeviceData.eState == E\_STATE\_ACTIVE\_SCANNING)

{

vHandleActiveScanResponse(psMlmeInd);

}

break;

/\* Deferred confirmation that the association process is complete \*/

case MAC\_MLME\_DCFM\_ASSOCIATE:

/\* Only respond to this if associating \*/

if (sEndDeviceData.eState == E\_STATE\_ASSOCIATING)

{

vHandleAssociateResponse(psMlmeInd);

}

break;

default:

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessIncomingData

\*

\* DESCRIPTION:

\* Process incoming data events from the stack.

\*

\* PARAMETERS: Name RW Usage

\* psMcpsInd

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessIncomingMcps(MAC\_McpsDcfmInd\_s \*psMcpsInd)

{

/\* Only handle incoming data events one device has been started as a

coordinator \*/

if (sEndDeviceData.eState >= E\_STATE\_ASSOCIATED)

{

switch(psMcpsInd->u8Type)

{

case MAC\_MCPS\_IND\_DATA: /\* Incoming data frame \*/

vHandleMcpsDataInd(psMcpsInd);

break;

case MAC\_MCPS\_DCFM\_DATA: /\* Incoming acknowledgement or ack timeout \*/

vHandleMcpsDataDcfm(psMcpsInd);

break;

default:

break;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleMcpsDataDcfm

\*

\* DESCRIPTION:

\*

\* PARAMETERS: Name RW Usage

\*

\* RETURNS:

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleMcpsDataDcfm(MAC\_McpsDcfmInd\_s \*psMcpsInd)

{

if (psMcpsInd->uParam.sDcfmData.u8Status == MAC\_ENUM\_SUCCESS)

{

/\* Data frame transmission successful \*/

}

else

{

/\* Data transmission failed after 3 retries at MAC layer. \*/

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleMcpsDataInd

\*

\* DESCRIPTION:

\*

\* PARAMETERS: Name RW Usage

\*

\* RETURNS:

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleMcpsDataInd(MAC\_McpsDcfmInd\_s \*psMcpsInd)

{

MAC\_RxFrameData\_s \*psFrame;

psFrame = &psMcpsInd->uParam.sIndData.sFrame;

if (psFrame->sSrcAddr.uAddr.u16Short == COORDINATOR\_ADR)

{

if (psFrame->au8Sdu[0] >= sEndDeviceData.u8RxPacketSeqNb)

{

sEndDeviceData.u8RxPacketSeqNb++;

vProcessReceivedDataPacket(&psFrame->au8Sdu[1],

(psFrame->u8SduLength) - 1);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vProcessReceivedDataPacket

\*

\* DESCRIPTION:

\*

\* PARAMETERS: Name RW Usage

\*

\* RETURNS:

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vProcessReceivedDataPacket(uint8 \*pu8Data, uint8 u8Len)

{

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vStartAssociate

\*

\* DESCRIPTION:

\* Start the association process with the network coordinator.

\*

\* PARAMETERS: Name RW Usage

\* None.

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\* Assumes that a network has been found during the network scan.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vStartAssociate(void)

{

MAC\_MlmeReqRsp\_s sMlmeReqRsp;

MAC\_MlmeSyncCfm\_s sMlmeSyncCfm;

sEndDeviceData.eState = E\_STATE\_ASSOCIATING;

/\* Create associate request. We know short address and PAN ID of

coordinator as this is preset and we have checked that received

beacon matched this \*/

sMlmeReqRsp.u8Type = MAC\_MLME\_REQ\_ASSOCIATE;

sMlmeReqRsp.u8ParamLength = sizeof(MAC\_MlmeReqAssociate\_s);

sMlmeReqRsp.uParam.sReqAssociate.u8LogicalChan = sEndDeviceData.u8Channel;

sMlmeReqRsp.uParam.sReqAssociate.u8Capability = 0x80; /\* We want short address, other features off \*/

sMlmeReqRsp.uParam.sReqAssociate.u8SecurityEnable = FALSE;

sMlmeReqRsp.uParam.sReqAssociate.sCoord.u8AddrMode = 2;

sMlmeReqRsp.uParam.sReqAssociate.sCoord.u16PanId = PAN\_ID;

sMlmeReqRsp.uParam.sReqAssociate.sCoord.uAddr.u16Short = COORDINATOR\_ADR;

/\* Put in associate request and check immediate confirm. Should be

deferred, in which case response is handled by event handler \*/

vAppApiMlmeRequest(&sMlmeReqRsp, &sMlmeSyncCfm);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleAssociateResponse

\*

\* DESCRIPTION:

\* Handle the response generated by the stack as a result of the associate

\* start request.

\*

\* PARAMETERS: Name RW Usage

\* psMlmeInd

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleAssociateResponse(MAC\_MlmeDcfmInd\_s \*psMlmeInd)

{

/\* If successfully associated with network coordinator \*/

if (psMlmeInd->uParam.sDcfmAssociate.u8Status == MAC\_ENUM\_SUCCESS)

{

vPrintf("Associated");

sEndDeviceData.u16Address = psMlmeInd->uParam.sDcfmAssociate.u16AssocShortAddr;

sEndDeviceData.eState = E\_STATE\_ASSOCIATED;

}

else

{

vStartActiveScan(SCAN\_CHANNELS);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vStartActiveScan

\*

\* DESCRIPTION:

\* Start a scan to search for a network to join.

\*

\* PARAMETERS: Name RW Usage

\* u32ChannelstoScan

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vStartActiveScan(uint32 u32ChannelstoScan)

{

MAC\_MlmeReqRsp\_s sMlmeReqRsp;

MAC\_MlmeSyncCfm\_s sMlmeSyncCfm;

sEndDeviceData.eState = E\_STATE\_ACTIVE\_SCANNING;

/\* Request scan \*/

sMlmeReqRsp.u8Type = MAC\_MLME\_REQ\_SCAN;

sMlmeReqRsp.u8ParamLength = sizeof(MAC\_MlmeReqScan\_s);

sMlmeReqRsp.uParam.sReqScan.u8ScanType = MAC\_MLME\_SCAN\_TYPE\_ACTIVE;

sMlmeReqRsp.uParam.sReqScan.u32ScanChannels = u32ChannelstoScan;

sMlmeReqRsp.uParam.sReqScan.u8ScanDuration = 3;

vAppApiMlmeRequest(&sMlmeReqRsp, &sMlmeSyncCfm);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vHandleActiveScanResponse

\*

\* DESCRIPTION:

\* Handle the reponse generated by the stack as a result of the network scan.

\*

\* PARAMETERS: Name RW Usage

\* psMlmeInd

\*

\* RETURNS:

\* None.

\*

\* NOTES: Demo Application Boiler Plate

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vHandleActiveScanResponse(MAC\_MlmeDcfmInd\_s \*psMlmeInd)

{

MAC\_PanDescr\_s \*psPanDesc;

uint8 i;

if (psMlmeInd->uParam.sDcfmScan.u8ScanType == MAC\_MLME\_SCAN\_TYPE\_ACTIVE)

{

if (psMlmeInd->uParam.sDcfmScan.u8Status == MAC\_ENUM\_SUCCESS)

{

i = 0;

while (i < psMlmeInd->uParam.sDcfmScan.u8ResultListSize)

{

psPanDesc = &psMlmeInd->uParam.sDcfmScan.uList.asPanDescr[i];

if ((psPanDesc->sCoord.u16PanId == PAN\_ID)

&& (psPanDesc->sCoord.u8AddrMode == 2)

&& (psPanDesc->sCoord.uAddr.u16Short == COORDINATOR\_ADR)

/\* Check it is accepting association requests \*/

&& (psPanDesc->u16SuperframeSpec & 0x8000))

{

sEndDeviceData.u8Channel = psPanDesc->u8LogicalChan;

vStartAssociate();

return;

}

i++;

}

}

}

//sbarf: updated section 24\_7\_08

// if there are remaining unscanned channels left to scan, scan them

if(psMlmeInd->uParam.sDcfmScan.u32UnscannedChannels != 0)

vStartActiveScan(psMlmeInd->uParam.sDcfmScan.u32UnscannedChannels);

else

/\* Failed to find coordinator: keep trying \*/

vStartActiveScan(SCAN\_CHANNELS);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: vPutChar

\*

\* DESCRIPTION:

\* Updates the UART output presented to the user.

\*

\* PARAMETERS: Name RW Usage

\* c

\*

\* RETURNS: void

\*

\* NOTES: Demo Application Boiler Plate for supporting vPrintf functionality.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void vPutChar(unsigned char c) {

while ((u8AHI\_UartReadLineStatus(UART) & E\_AHI\_UART\_LS\_THRE) == 0);

vAHI\_UartWriteData(UART, c);

while ((u8AHI\_UartReadLineStatus(UART) & (E\_AHI\_UART\_LS\_THRE | E\_AHI\_UART\_LS\_TEMT)) != (E\_AHI\_UART\_LS\_THRE | E\_AHI\_UART\_LS\_TEMT));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* NAME: tx\_Distance

\*

\* DESCRIPTION:

\* Transmits the i32TofDistance and u32RssiDistance to the coordinator.

\*

\* RETURNS: void

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PRIVATE void tx\_Distance(int32 i32TofDistance, uint32 u32RssiDistance)

{

/\* Structures used to hold data for MLME request and response \*/

MAC\_McpsReqRsp\_s sMcpsReqRsp;

MAC\_McpsSyncCfm\_s sMcpsSyncCfm;

uint8 \*pu8Payload;

/\* Create frame transmission request \*/

sMcpsReqRsp.u8Type = MAC\_MCPS\_REQ\_DATA;

sMcpsReqRsp.u8ParamLength = sizeof(MAC\_McpsReqData\_s);

/\* Set handle so we can match confirmation to request \*/

sMcpsReqRsp.uParam.sReqData.u8Handle = u8CurrentTxHandle;

u8CurrentTxHandle +=1;

/\* Use short address for source \*/

sMcpsReqRsp.uParam.sReqData.sFrame.sSrcAddr.u8AddrMode = 2;

sMcpsReqRsp.uParam.sReqData.sFrame.sSrcAddr.u16PanId = PAN\_ID;

sMcpsReqRsp.uParam.sReqData.sFrame.sSrcAddr.uAddr.u16Short = sEndDeviceData.u16Address;

/\* Use short address for destination \*/

sMcpsReqRsp.uParam.sReqData.sFrame.sDstAddr.u8AddrMode = 2;

sMcpsReqRsp.uParam.sReqData.sFrame.sDstAddr.u16PanId = PAN\_ID;

sMcpsReqRsp.uParam.sReqData.sFrame.sDstAddr.uAddr.u16Short = COORDINATOR\_ADR;

/\* Frame requires ack but not security, indirect transmit or GTS \*/

sMcpsReqRsp.uParam.sReqData.sFrame.u8TxOptions = MAC\_TX\_OPTION\_ACK;

/\* Set payload, only use first 8 bytes \*/

sMcpsReqRsp.uParam.sReqData.sFrame.u8SduLength = 10;

pu8Payload = sMcpsReqRsp.uParam.sReqData.sFrame.au8Sdu;

vPrintf("\nTransmitting Distance to Coordinator\n");

pu8Payload[0] = sEndDeviceData.u8TxPacketSeqNb++;

pu8Payload[1] = (uint8)(0xd1);

pu8Payload[2] = (uint8)(((uint32)i32TofDistance & 0xff000000uL) >> 24);

pu8Payload[3] = (uint8)(((uint32)i32TofDistance & 0x00ff0000uL) >> 16);

pu8Payload[4] = (uint8)(((uint32)i32TofDistance & 0x0000ff00uL) >> 8);

pu8Payload[5] = (uint8)((uint32)i32TofDistance & 0x000000ffuL);

pu8Payload[6] = (uint8)((u32RssiDistance & 0xff000000uL) >> 24);

pu8Payload[7] = (uint8)((u32RssiDistance & 0x00ff0000uL) >> 16);

pu8Payload[8] = (uint8)((u32RssiDistance & 0x0000ff00uL) >> 8);

pu8Payload[9] = (uint8)(u32RssiDistance & 0x000000ffuL);

#ifdef DEBUG\_DISTANCE\_TRANSMISSION

vPrintf("TOF Byte0: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Payload[2]));

vPrintf("TOF Byte1: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Payload[3]));

vPrintf("TOF Byte2: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Payload[4]));

vPrintf("TOF Byte3: "BYTE\_TO\_BINARY\_PATTERN"\n\n", BYTE\_TO\_BINARY(pu8Payload[5]));

vPrintf("RSSI Byte0: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Payload[6]));

vPrintf("RSSI Byte1: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Payload[7]));

vPrintf("RSSI Byte2: "BYTE\_TO\_BINARY\_PATTERN"\n", BYTE\_TO\_BINARY(pu8Payload[8]));

vPrintf("RSSI Byte3: "BYTE\_TO\_BINARY\_PATTERN"\n\n", BYTE\_TO\_BINARY(pu8Payload[9]));

#endif

vAppApiMcpsRequest(&sMcpsReqRsp, &sMcpsSyncCfm);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\* END OF FILE \*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/