

KAGA FEI EVBnRF528101.2 Multi Sensor Board Beacon Firmware User Guide

Contents

- [1 KAGA FEI EVBnRF528101.2 Multi Sensor Board Beacon Firmware](#)
- [2 Introduction](#)
- [3 Hardware Overview](#)
- [4 Drivers](#)
- [5 Adding New Sensors](#)
- [6 Event Handler](#)
- [7 Add to Project](#)
- [8 Adding to Application](#)
- [9 Documents / Resources](#)
 - [9.1 References](#)
- [10 Related Posts](#)



KAGA FEI EVBnRF528101.2 Multi Sensor Board Beacon Firmware

Introduction

- This document provides an overview of the development resources for the KFEI Sensor Kit Beacon firmware application.
- It outlines the general architecture and use of the firmware.

Requirements

To send sensor data using BLE using the Multi-Sensor Board Beacon application, you need the following.

- Nordic SDK
- Segger Embedded Studio and NRF board support package (For installation and setup procedures, register the product according to the sheet provided when purchasing the evaluation board/evaluation KIT, and download the “SES
- NRF52 Quickstart Guide x.xx EN.pdf” for Segger Embedded Studio’s NRF quick start guide.).
- IF pluggable interface board(EY1SENSOR-SKT)
- Sensors for the desired operation
- J-link Lite
- Supported following module

Board	Chip
EJ2840AA2-EVB	nRF52840
ED2833AA2-EVB	nRF52833
EJ2833AA2-EVB	nRF52833
EC2832AA2-EVB	nRF52832
ES2832AA2-EVB	nRF52832
EJ2832AA2-EVB	nRF52832
EC2820AA2-EVB	nRF52820
ES2820AA2-EVB	nRF52820
EC2811AA2-EVB	nRF52811
ES2811AA2-EVB	nRF52811
EC2810AA2-EVB	nRF52810
ES2810AA2-EVB	nRF52810

Table 1 : supported Board

Resource Description

The software of the Multi Sensor Board Beacon application is configured as shown in the block diagram below. For BLE communication, Soft Device provided by Nordic is used, and it is set to be written together with the application when writing using SES.

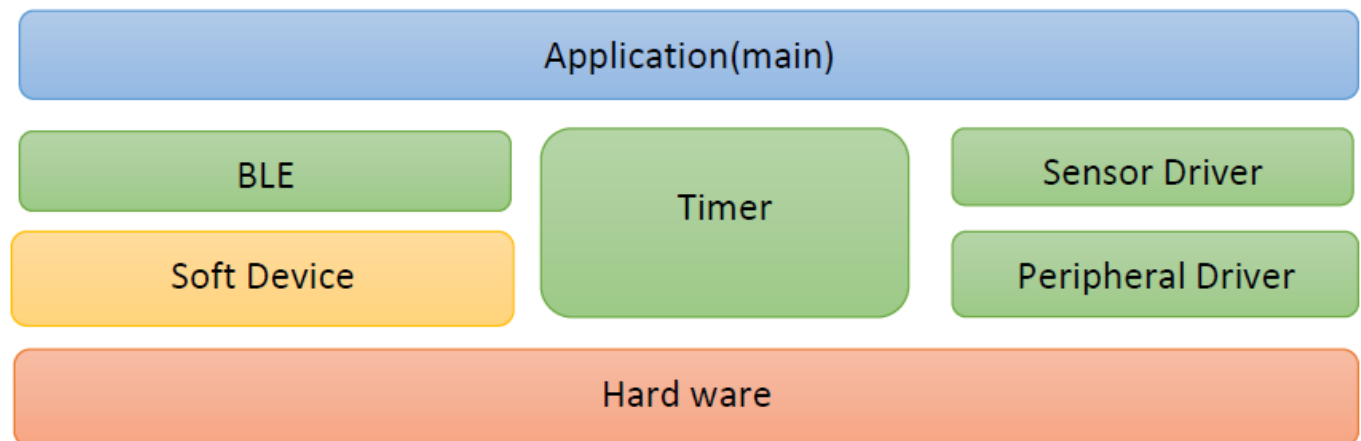
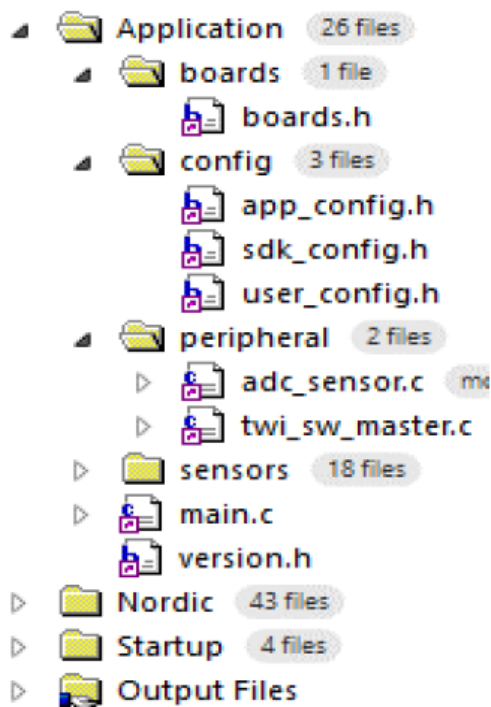


Figure 1: Software block diagram

- The following folders and files are included in the project for the Multi Sensor Board Beacon application. The function of each file is as follows.



Application resources

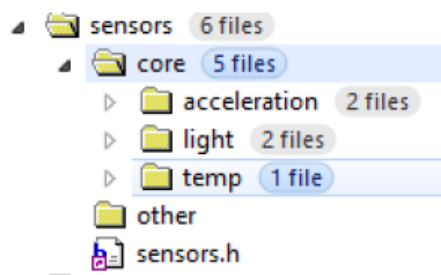
- Board information
 - IF board pin assignments
- Configuration files
 - Application configuration (BLE, timer)
 - SDK configuration
 - User configuration (sensor pins)
- Peripheral drivers
 - ADC
 - Software TWI
- Sensor drivers
- Main application
- Version information

Nordic SDK

Startup Files

Generated output files

Figure 2: Application configuration



Sensor drivers

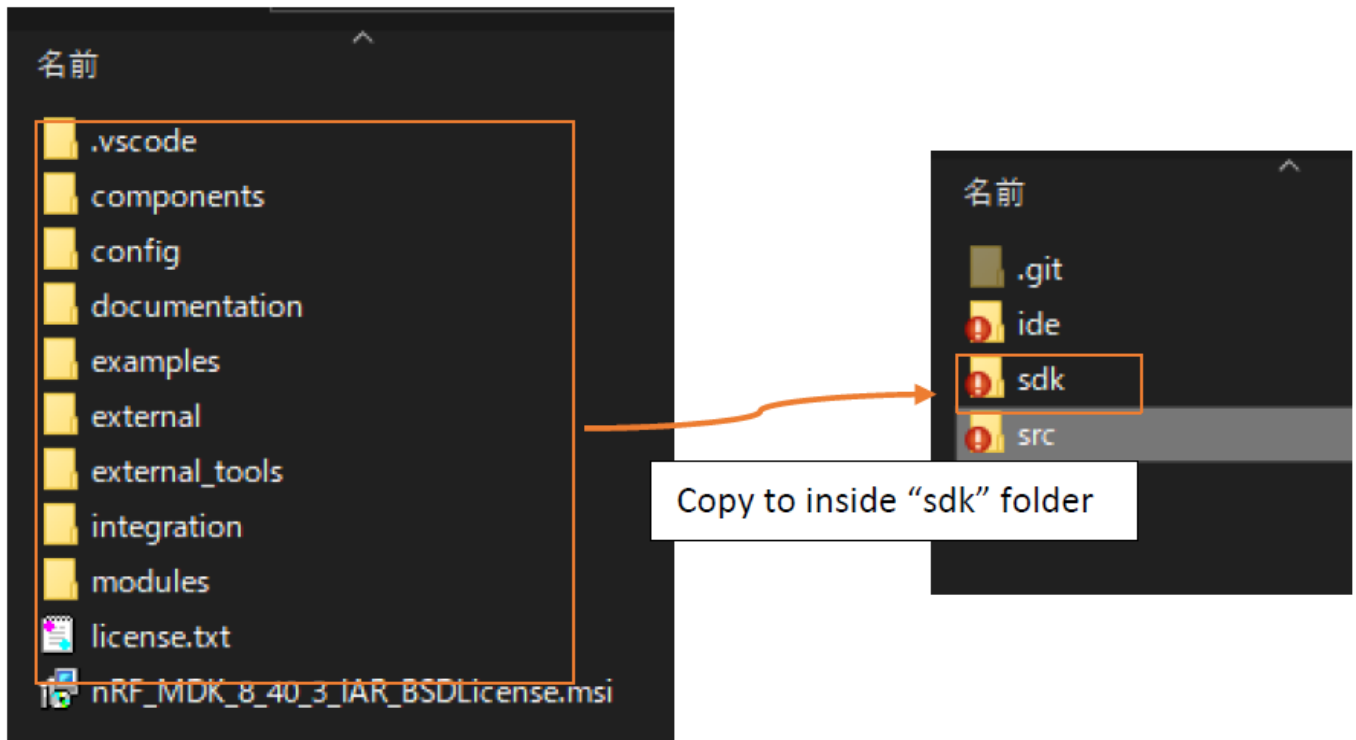
- Accelerometer sensors
- Light sensors
- Temperature sensors
- Other sensor

Common sensor header

Figure 3 Sensor driver configuration

Nordic SDK Settings

- The project for the Multi Sensor Board Beacon application uses NordicSDK. Copy the folder where you downloaded and decompressed the following version of Nordic SDK into the “sdk” folder.
- The following SDK version is used in the project of the Multi Sensor Board Beacon application, but please consider the latest version when using it for development.
- [SDK info]
- SDK Version 17.1.0 nRF5 SDK
- Download URL <https://www.nordicsemi.com/Software-and-tools/Software/nRF5-SDK/Download#infotabs>
- Nordic source code copyright belongs to Nordic



Application Overview

During initialization, the application prepares any connected sensors as well as the Softdevice and BLE stack. Once complete, a background timer is started that triggers at a preset interval and BLE advertising is started. The event triggered by the timer updates the BLE advertising data with new sensor data.

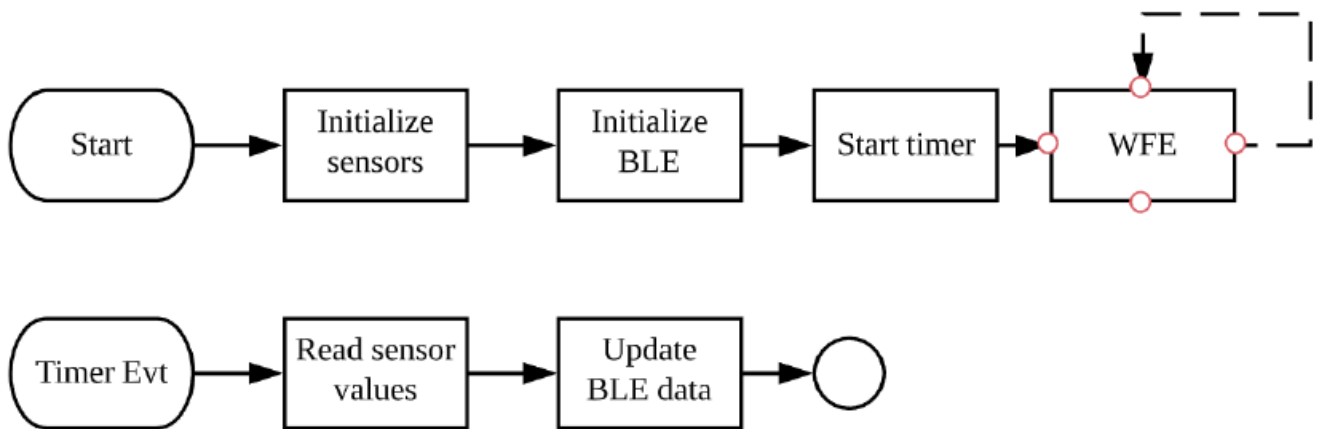


Figure 4: Application initialization and timer events

Application Timer

The timer responsible for triggering update events can be configured to faster or slower intervals. Care should be taken to ensure that the timer interval is sufficiently long that all sensor reads can be completed before the next interval. The value for configuring this timer can be found in the configuration header src/config/app_config.h.

```

9 /** Application control */
10 #define UPDATE_TIMER_RATE_MS      1000    /**< Interval for sensor updates, should be low
11 #define ADV_INTERVAL_MS          100      /**< Minimum 100ms, maximum 10.24s */
  
```

Figure 5: Application configuration options

BLE Advertising

- The device is configured to operate as a non-connectable beacon advertising at regular intervals with no timeout. The device is configured to transmit a primary advertising packet and a scan response packet using the format described in Figure 6.
- The rate at which advertising data is transmitted from the BLE radio can be set in the configuration header. The field ADV_INTERVAL_MS controls the rate at which packets are transmitted.

Advertising Formats

The primary advertising packet contains a series of sensor data fields aggregated under a Manufacturer Specific Data fragment. The secondary scan response packet contains the device name under a Complete Device Name fragment. The field contents are shown in Figure 6.

Byte	Field
1	Type ID (0x81)
2	Sequence (0 → 255)
3	Name
4	
5	
6	
7	Channel 1
8	
9	
10	Channel 2
11	
12	
13	Channel 3
14	
15	
16	Channel 4
17	
18	
19	Channel 5
20	
21	
22	Channel 6
23	
24	

Figure 6: Primary advertising packet – KFEI Sensor Kit data format - and scan response packet

- The firmware contains a global structure that matches the format of the packet contents.

```

28 typedef struct
29 {
30     uint8_t typenum;
31     uint8_t sequence_num;
32     char name[NAME_MAX_LEN];
33     sensor_ch_t accx_data;
34     sensor_ch_t accy_data;
35     sensor_ch_t accz_data;
36     sensor_ch_t light_data;
37     sensor_ch_t temp_data;
38
39 }ble_custom_adv_data_t;

```

Figure 7: Primary Advertising Packet-Structure

```

60 ble_custom_adv_data_t custom_data;

```

Figure 8: Primary Advertising Packet-Variable Definition

- This struct is used for holding sensor data read during the timer update events, and is transmitted over BLE radio at advertising intervals.
- For the final product, it is necessary to obtain a new company ID and set it.

Hardware Overview

Each supported board has its own configuration in the Segger Embedded Studio project. You can select the target board for compiling the application using the configuration drop-down box in SES. These configurations are set to target the relevant chip type and soft device required for the application.

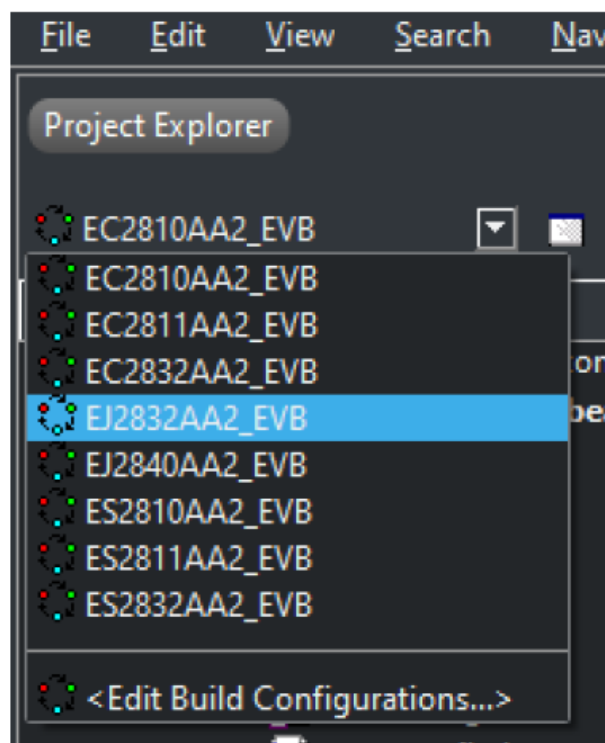


Figure 9 :drop-down list

Pin Assignment

- Due to the arrangement of pins on each board variant, the IF board will connect to a different set of pins.
- Pin assignments are predefined to maintain compatibility with the various layouts.
- The default pin assignments for each of the supported boards can be found in Application/boards/boards.h.

```
6  #if defined(EC2832AA2_EVB)
7      // 52832 large
8      #define PIN_CN3_3 (15)
9      #define PIN_CN3_4 (12)
0      #define PIN_CN4_3 (PIN_NC)
1      #define PIN_CN4_4 (31)
2
3      #define CN4_4_ADC (NRF_SAADC_INPUT_AIN7)
```

Figure 10: Pin assignments and valid slot assignments for the EC2832AAx-EVB board

- The table below shows the various pin connections for each board.
- AIN represents an analog-capable pin for use with ADC-based sensors.
- NFC represents pins that can be used for Near Field Communication, however this is disabled by default.

	CN3		CN4	
	Pin no	Port	Pin no	Port
EJ2840AA2-EVB	3	P0.28/AIN4	3	P0.29/AIN5
	4	P0.04/AIN2	4	P0.03/AIN1
ED2833AA2-EVB	3	P0.15	3	P0.04/AIN2
	4	-	4	P0.02/AIN0
EJ2833AA2-EVB	3	P0.28/AIN4	3	P0.29/AIN5
	4	P0.04/AIN2	4	P0.03/AIN1
EC2832AA2-EVB	3	P0.15	3	-
	4	P0.12	4	P0.31/AIN7
EJ2832AA2-EVB	3	P0.10/NFC	3	P0.02/AIN0
	4	P0.28/AIN4	4	P0.18
ES2832AA2-EVB	3	P0.03/AIN1	3	P0.04/AIN2
	4	P0.02/AIN0	4	P0.09/NFC
EC2820AA2-EVB	3	P0.02	3	P0.29
	4	-	4	P0.08
ES2820AA2-EVB	3	P0.03	3	P0.02
	4	-	4	P0.015
EC2811AA2-EVB	3	P0.15	3	-
	4	P0.12	4	P0.31/AIN7
ES2811AA2-EVB	3	P0.03/AIN1	3	P0.11
	4	P0.04/AIN2	4	P0.17
EC2810AA2-EVB	3	P0.15	3	-
	4	P0.12	4	P0.31/AIN7
ES2810AA2-EVB	3	P0.03/AIN1	3	P0.11
	4	P0.04/AIN2	4	P0.17

Table 2: GPIO Pin

Note: ADC inputs vary from board to board (i.e. AIN0/AIN1). While the boards.h header contains predefined assignments for simple configuration, care should be taken to ensure the correct ADC is selected to prevent hardware damage.

The attached sensor uses the pins described in the table below. Must be connected to the GPIO Pin of the board used.

Sensor	Driver	Pin
Light sensor	ADC	Pin4
Temperature sensors	ADC	Pin4
Accelerometer sensors	TWI	Pin3,Pin4

Table 3:Correspondence table of sensor and driver

NFC and Reset Pin Configuration

Some pins are configurable for use with NFC or GPIO. Some devices use the shared pins to connect with the IF board, so by default, this application configures the pins for GPIO use. The reset button located on the board is also configured to reset the device when pushed.

This functionality is controlled using the CONFIG_GPIO_AS_PINRESET and CONFIG_NFCT_PINS_AS_GPIOS definitions located in the Project Options → Common [configuration] → Preprocessor → Preprocessor Definitions.

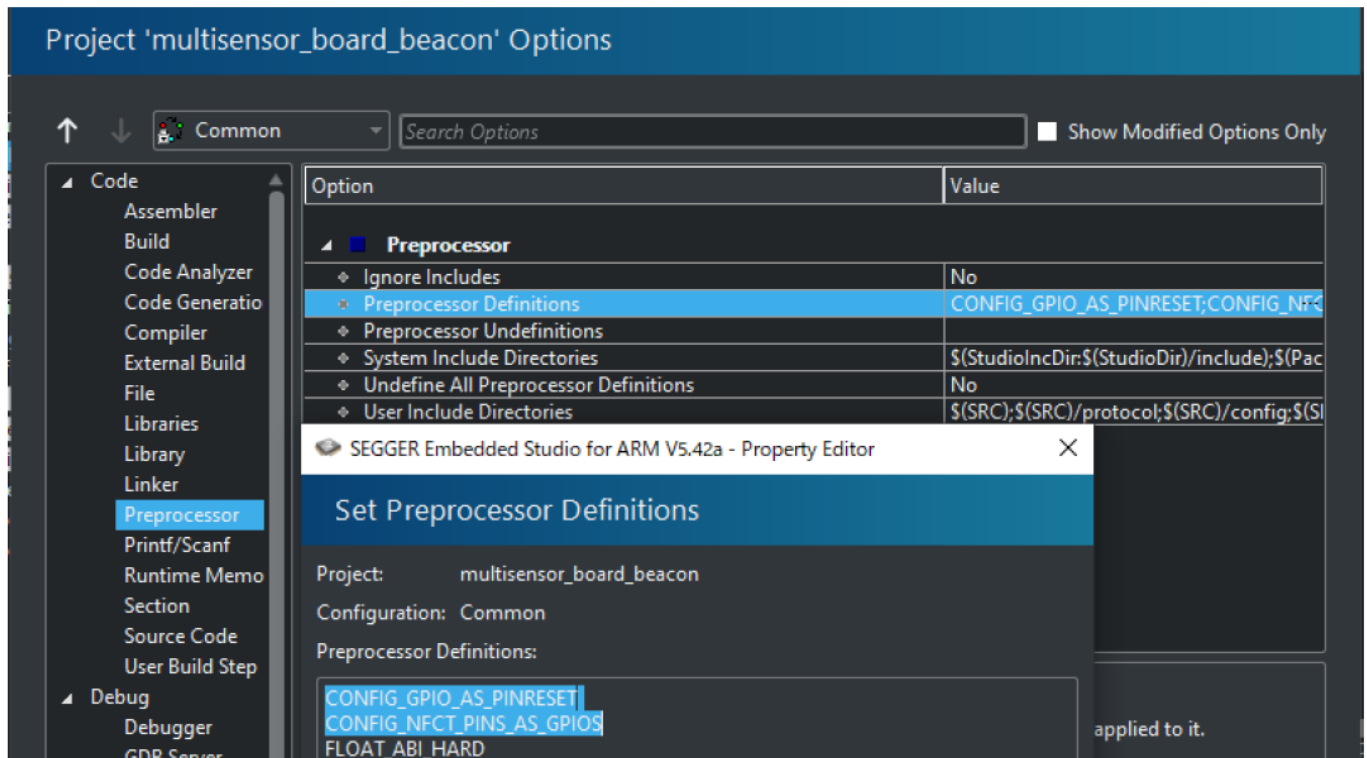


Figure 11: NFC / PinReset functionality

The relevant source code is located in the Startup/system_nrf52.c. The desired configuration options are written to the UICR registers as part of the board startup process.

Note: once written, UICR registers cannot be cleared by software. To change the functionality, a chip erase must first be performed.

```

181  /* Configure NFCT pins as GPIOs if NFCT is not to be used in your code. If CONFIG_NFCT_PINS_AS_GPIOS is not defined,
182  two GPIOs (see Product Specification to see which ones) will be reserved for NFC and will not be available as
183  normal GPIOs. */
184  #if defined (CONFIG_NFCT_PINS_AS_GPIOS)
185      if ((NRF_UICR->NFCPINS & UICR_NFCPINS_PROTECT_Msk) == (UICR_NFCPINS_PROTECT_NFC << UICR_NFCPINS_PROTECT_Pos)){
186          NRF_NVMC->CONFIG = NVMC_CONFIG_WEN_Wen << NVMC_CONFIG_WEN_Pos;
187          while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
188          NRF_UICR->NFCPINS &= ~UICR_NFCPINS_PROTECT_Msk;
189          while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
190          NRF_NVMC->CONFIG = NVMC_CONFIG_WEN_Ren << NVMC_CONFIG_WEN_Pos;
191          while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
192          NVIC_SystemReset();
193      }
194  #endif
195
196  /* Configure GPIO pads as pPin Reset pin if Pin Reset capabilities desired. If CONFIG_GPIO_AS_PINRESET is not
197  defined, pin reset will not be available. One GPIO (see Product Specification to see which one) will then be
198  reserved for PinReset and not available as normal GPIO. */
199  #if defined (CONFIG_GPIO_AS_PINRESET)
200      if (((NRF_UICR->PSELRESET[0] & UICR_PSELRESET_CONNECT_Msk) != (UICR_PSELRESET_CONNECT_Connected << UICR_PSELRESET_
201      ((NRF_UICR->PSELRESET[1] & UICR_PSELRESET_CONNECT_Msk) != (UICR_PSELRESET_CONNECT_Connected << UICR_PSELRESET_
202      NRF_NVMC->CONFIG = NVMC_CONFIG_WEN_Wen << NVMC_CONFIG_WEN_Pos;
203      while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
204      NRF_UICR->PSELRESET[0] = 21;
205      while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
206      NRF_UICR->PSELRESET[1] = 21;
207      while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
208      NRF_NVMC->CONFIG = NVMC_CONFIG_WEN_Ren << NVMC_CONFIG_WEN_Pos;
209      while (NRF_NVMC->READY == NVMC_READY_READY_Busy){}
210      NVIC_SystemReset();
211  }
212  #endif

```

Figure 12: NFC / PinReset startup configuration

Sensor configuration

- The interface board (IF) has two connectors for attaching various sensors. To use the sensors, a user must add the specific pin configuration in this file src/config/user_config.h
- Each board has different pin configurations and sensor connectors, in which you can refer to from the board connector document to be able to understand on how to configure the sensors.
- For example, the ADC pin of board EC2832AAx-EVB is in connector CN4 Pin 4 (CN4_4), while board EJ2832AAx-EVB is in connector CN3 Pin 4 (CN3_4).
- Users can add/remove the sensor pin assignments to suit their specific application requirements. A set of example pins is already configured in the user_config.h header.

```

10  #if defined(EC2832AA2_EVB)
11      // 52832 large
12
13      /* Example when using IMU sensor */
14      #define BMA400_SCL  PIN_CN3_4
15      #define BMA400_SDA  PIN_CN3_3
16
17      /* Example when using adc light sensor*/
18      #define LIGHT_ADC_PIN  CN4_4_ADC
19
20      /* Example when using adc temperature sensor*/
21      #define TEMP_ADC_PIN  CN4_4_ADC
22

```

- The Application/main.c/sensors_init function contains the initialization functions for the sensors that can be connected to the IF board.
- As an example, all of the available functions are listed (commented by default). Only the sensors that are connected are included in the application.

```

void sensors_init(void){
    // imu
    //  bma400_TWI_Init(BMA400_SCL, BMA400_SDA);

    // light
    //  light_adc_init(LIGHT_ADC_PIN);

    // temperature/humidity
    //  temp_adc_init(TEMP_ADC_PIN);
    return;
}

```

Figure 14: Sensor initialization functions

- Users can uncomment/comment the functions as required, or remove unused code.

- Likewise, the Application/main.c/on_timer_evt function contains the sensor reading functions. Sensors connected to the board are read during this function and the advertising data is updated.

```

376 static void on_timer_evt(void *context)
377 {
378     NRF_LOG_INFO("Event Reached, Updating data");
379
380     // Sequence number
381     static uint16_t _sequ_num = 0;
382
383     _sequ_num += 1;
384
385     custom_data.sequence_num = _sequ_num;
386     custom_data.typenum = TYPE_IDENTIFER;
387
388     /** Insert relevant sensor update calls here */
389     #if !(defined(EC2820AAx_EVB) || defined(ES2820AAx_EVB) )
390     // on_temp_adc_update();
391     on_light_update();
392     #endif
393     on_accelerometer_update();
394     // on_txpower_update();
395
396     update_adv_data();
397 }

```

Figure 15: Sensor value update functions

- As with the initialization function, the user must comment or remove unused features.

Sensor	Init function	Data read function
Light sensor	light_adc_init	on_light_update
Temperature sensors	temp_adc_init	on_temp_adc_update
Accelerometer sensors	bma400_TWI_Init	on_accelerometer_update

Table 4: Sensor function correspondence table

Drivers

- This section describes the existing peripheral and sensor drivers included in the Sensorkit application.

Peripheral Drivers

The Nordic SDK includes peripheral drivers suitable for most applications (e.g. TWI, GPIO). These drivers are the ones typically used by the sensor drivers. A small number of custom peripheral drivers have been included alongside the SDK. They are located in the Application/peripheral folder.

ADC

The ADC driver provides a single adc_read function to sample the voltage on a specified analog input. The function will block until the conversion is completed. The conversion is performed using 10 bit resolution and returns the average result of 16 samples. The operation will take approximately 640us.

```
void adc_read(nrf_saadc_input_t adc_input, nrf_saadc_gain_t gain, int16_t * buffer)
```

Parameter	Description	Example
adc_input	ADC input pin	NRF_SAADC_INPUT_AIN0
gain	ADC voltage gain multiplier	SAADC_CH_CONFIG_GAIN_Gain1_6
buffer	Result destination buffer	

Table 5: *adc_read* function argument

Software TWI

The TWI peripheral on NRF devices has a minimum clock frequency of 100KHz. Devices like the SHT temperature sensor may not operate at this frequency. A GPIO-based software TWI driver is prepared that operates at ~40KHz. The TWI transaction is performed in a blocking method.

```
bool twi_master_init(uint8_t scl, uint8_t sda);
```

Parameter	Description
scl	GPIO pin for TWI SCL
sda	GPIO pin for TWI SDA
return	true on success, otherwise false

Table 6:Argument of *twi_master_init* function

```
bool twi_master_transfer(uint8_t address, uint8_t *data, uint8_t data_length, bool issue_stop_condition);
```

Parameter	Description
address	Device address on TWI bus (includes read/write bit)
data	Buffer for data to be transferred
data_length	Byte length of buffer
issue_stop_condition	Send STOP after transfer is complete
return	true on success, otherwise false

Adding New Sensors

Beacon Packet Structure

When adding a new sensor that is not in the KFEI Sensor Kit beacon format, it is necessary to set the data of the sensor to be added to the packet. Figure 18: Set new sensor data in Sensor Data shown in beacon packet. In the source code, set new sensor data in the location shown in Figure 19 in the maic.c file.

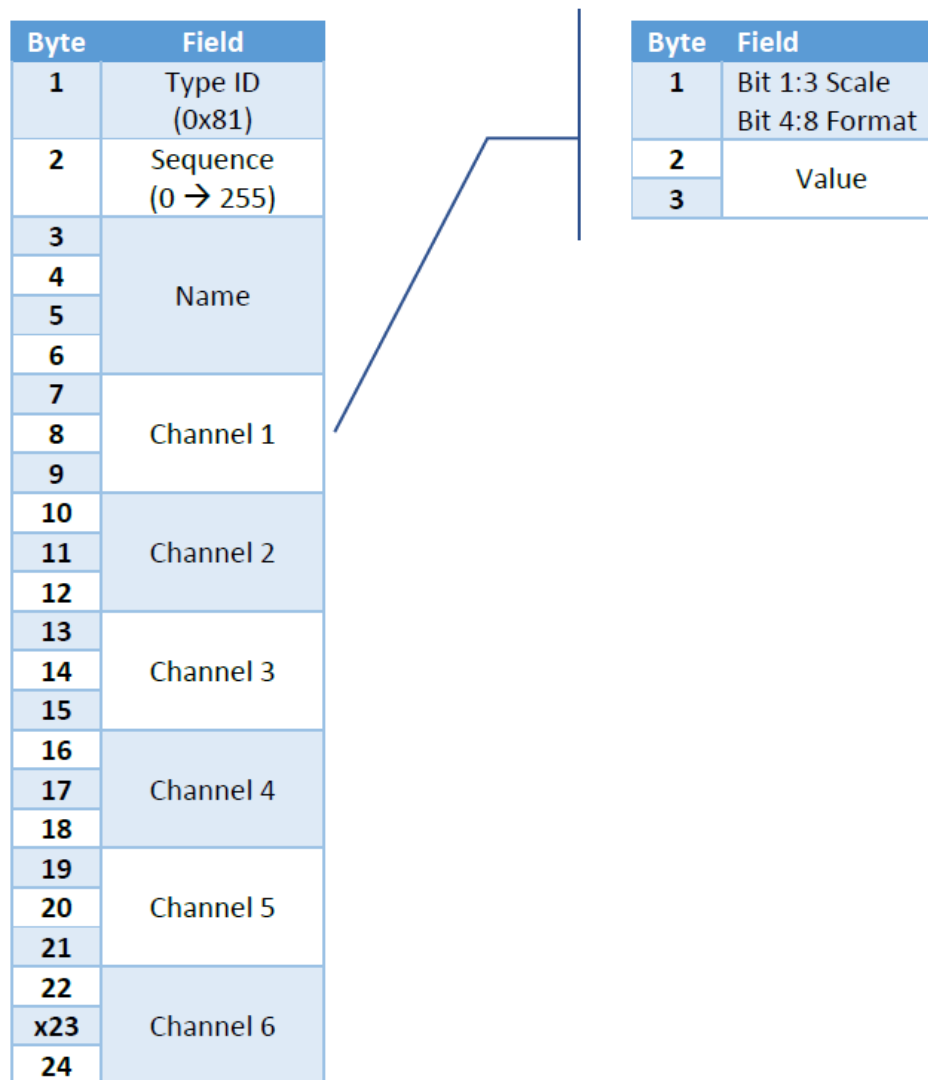


Figure 16: Beacon packet

```

266  manuf_specific_data.data.p_data = (uint8_t *) &custom_data; //(uint8_t *) m_beacon_info;
267  manuf_specific_data.data.size   = sizeof(ble_custom_adv_data_t); //APP_BEACON_INFO_LENGTH;

```

Figure 17: Setting location of sensor data

- To use sensors in your application, the driver must include at least the following functions: Also, since the function created in the following procedure is called from main.c, also create a header file that defines the added function (see
- Figure 20: Example of driver header).
- Also, to maintain portability between different board types, the pin assignment and ADC port are passed from the upper application to the driver in the current structure. Pins and ports are selected based on the connector and board selected for compilation.
- Also, the sample code is included in the folder (sdk / examples / peripheral) of Nordic SDK downloaded in “1.3 Nordic SDK Settings”, so please use it as a reference for the driver to be created.

[structure of .c file]

1. Initializer – prepare the peripheral, and preconfigure the sensor as required (e.g. set any control registers or enable sampling)
2. Read – read a data sample from the sensor

3. [optional] Converter – convert the data sample into a format suitable for the beacon. This can also be included as part of the read

```
1 #ifndef MAGNETIC_SENSOR__
2 #define MAGNETIC_SENSOR__
3
4 #include <stdint.h>
5
6 void magnetic_sensor_init(uint32_t hi_pin, uint32_t lo_pin);
7 int16_t magnetic_sensor_get_data(void);
8
9
10 #endif // MAGNETIC_SENSOR__
11
```

Figure 18: Driver header example

Event Handler

- As described in Section 0, A background timer fires at regular intervals.
- This timer calls an event handler on_timer_evt which updates the BLE advertising data.
- An associated event handler function for each connected sensor should be included in this timer event.
- This function should read the latest sensor value and store it in the advertising data.
- While the sensor data should be stored in the most appropriate positional field according to the KFEI Sensor Kit format described in Section 1.4.3, it is up to the developer's discretion over which field is used in the case where the sensor does not match one of the preset placeholders.

```
649 /**@brief Callback function for the sensor timer
650 *
651 * @details Updates the advertising data values
652 */
653 static void on_timer_evt(void *context)
654 {
655     NRF_LOG_INFO("Event Reached, Updating data");
656
657     // Sequence number
658     static uint16_t _sequ_num = 0;
659     _sequ_num += 1;
660
661     custom_data.sequence_num = _sequ_num;
662     custom_data.typenum = TYPE_ID;
663
664     /** Insert relevant sensor update calls here */
665     // on_temperature_update();
666     // on_humidity_update();
667     // on_light_update();
668     // on_pressure_update();
669     on_magnetic_update();
670     // on_co2_update();
671     // on_accelerometer_update();
672     on_current_update();
673     // on_txpower_update();
674
675     update_adv_data();
676 }
677
678
679 /**@brief Fetches the current data if sensor is connected
680 *
681 * @details Read and update the 'voltage_data' with the latest current
682 */
683 void on_current_update(void){
684
685     // Current data
686     int16_t _current_data = 0;
687     nrf_saadc_value_t current_val;
688
689     sensor_get_adc(&current_val);
690     NRF_LOG_INFO("ADC Buffer Data : %d", current_val);
691
692     /* Convert current value from the adc value read */
693     float calculated_val = sensor_convert_to_current(current_val);
694
695     // Update advertising data field
696     _current_data = calculated_val*100;
697     custom_data.current_data = _current_data;
698
699     NRF_LOG_INFO("Current Data : %d", _current_data);
700 }
```

Figure 19: Timer event preparing a new set of advertising data. The magnetic sensor and current sensor are connected in this case.

Add to Project

- When adding a sensor driver file to a project, it is easier to manage by placing the file in the path

Application/sensors / new_driver_folder / driver_name.c / h according to the structure of the project file.

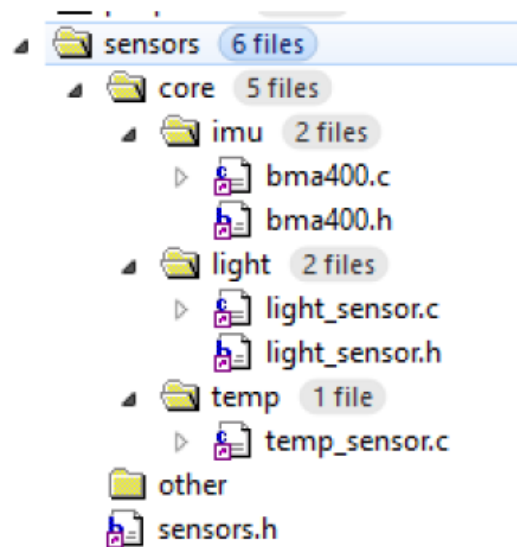


Figure 20:Project file structure

- After that, include the new header file to src/sensors/sensors.h in order to get included in the project.

```
8 // acceleration
9 #include "sensors/core/acceleration/bma400.h"
10
11 // light
12 #include "sensors/core/light/light_sensor.h"
13
14 // temperature
15 #include "sensors/core/temp/temp_sensor.h"
16
```

Figure 21: Including Header Files


Adding to Application

To utilize the new sensor in the application:

1. Include the driver's pin configuration and any required DEFINE labels to the user_config.h
2. Add the driver's init function to the sensor_init function called during main().
3. Add the event handler's function call to the on_timer_evt() function

KAGA FEI Co., Ltd. 20-Jan.2024Ver1.1 20/20 Copyright 2024 KAGA FEI Co., Ltd.

Documents / Resources

	<p>KAGA FEI EVBnRF528101.2 Multi Sensor Board Beacon Firmware [pdf] User Guide EVBnRF52810ES2810AA2, EVBnRF528101.2, EVBnRF528101.2 Multi Sensor Board Beacon Firmware, EVBnRF528101.2, Multi Sensor Board Beacon Firmware, Sensor Board Beacon Firmware, Board Beacon Firmware, Beacon Firmware, Firmware</p>
--	--

References

- [nRF5 SDK downloads - nordicsemi.com](#)
- [User Manual](#)

Manuals+, Privacy Policy

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.