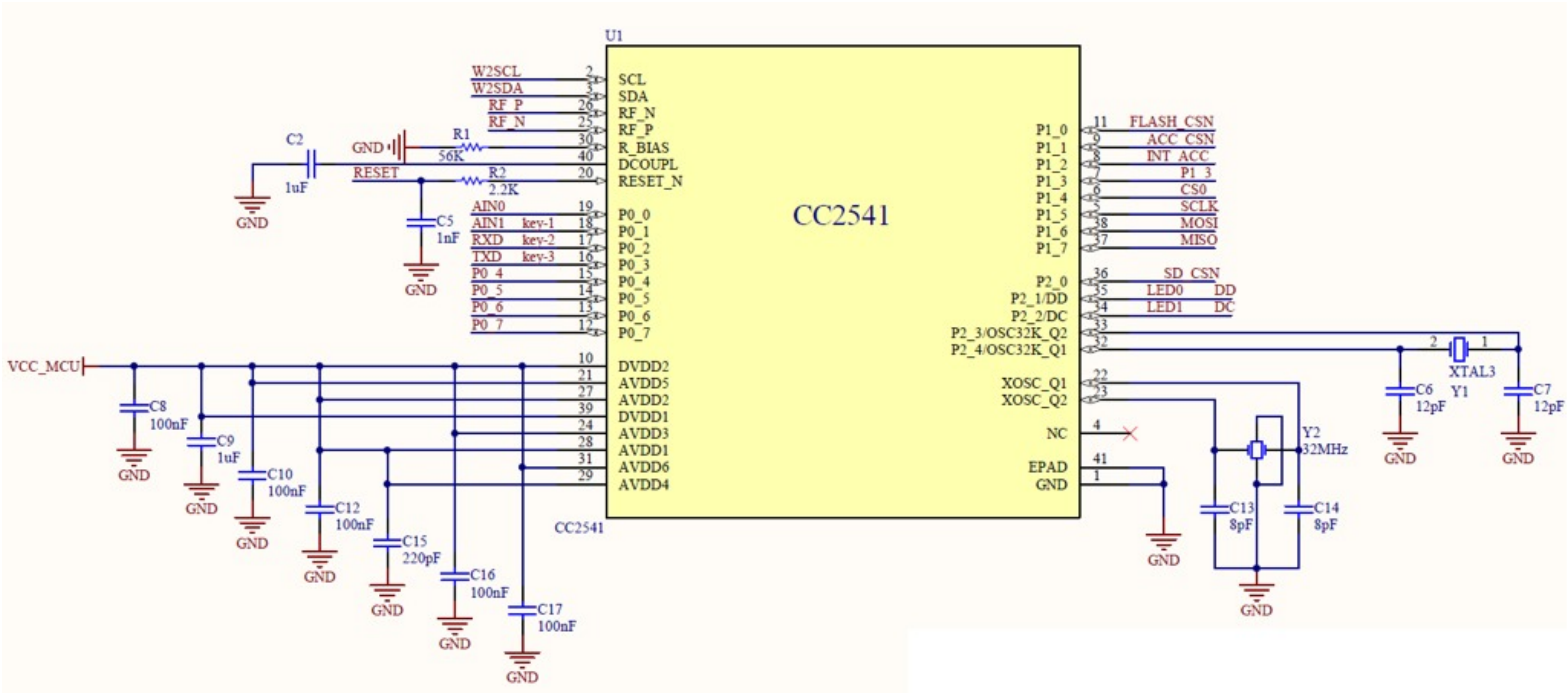
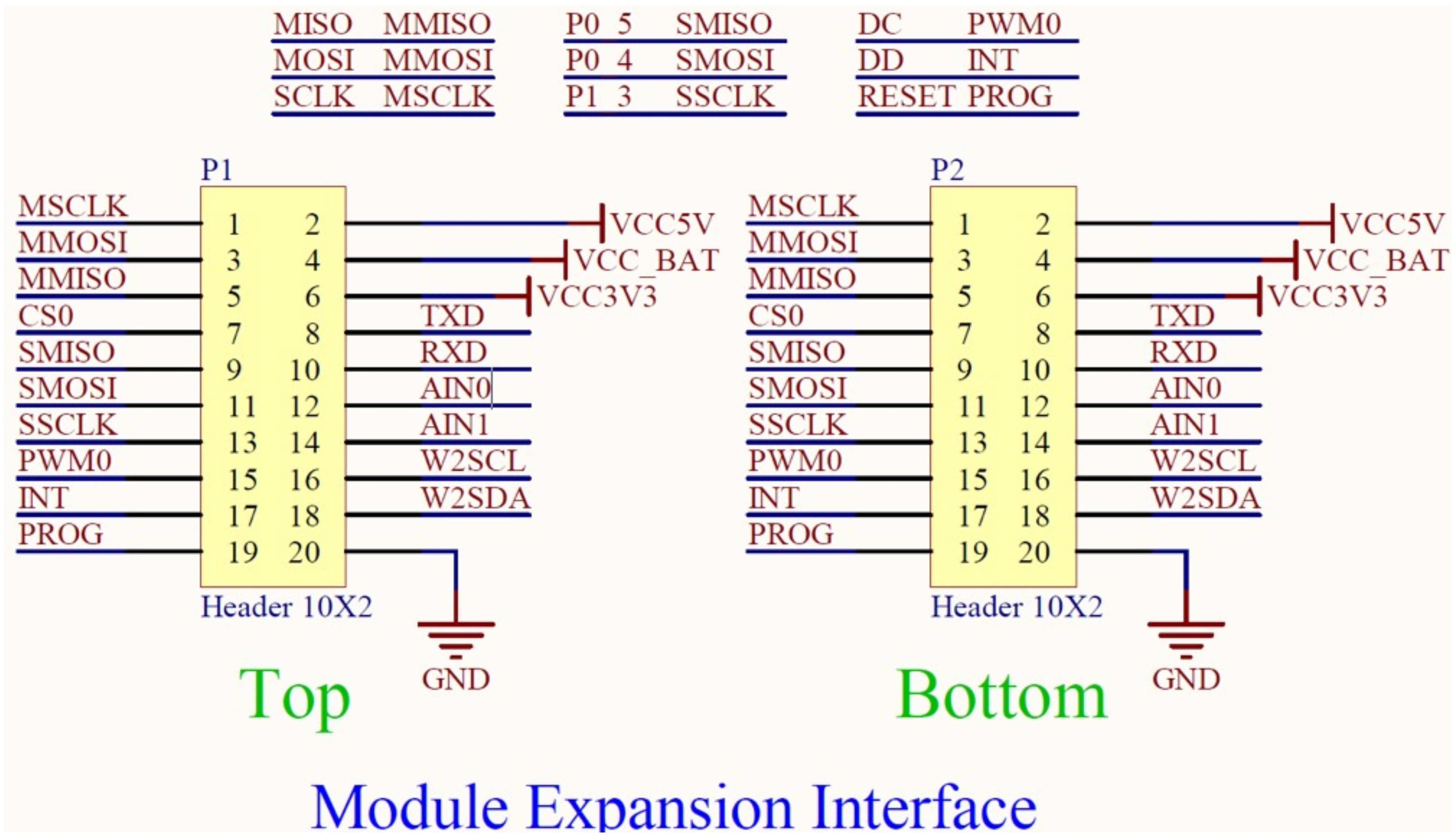
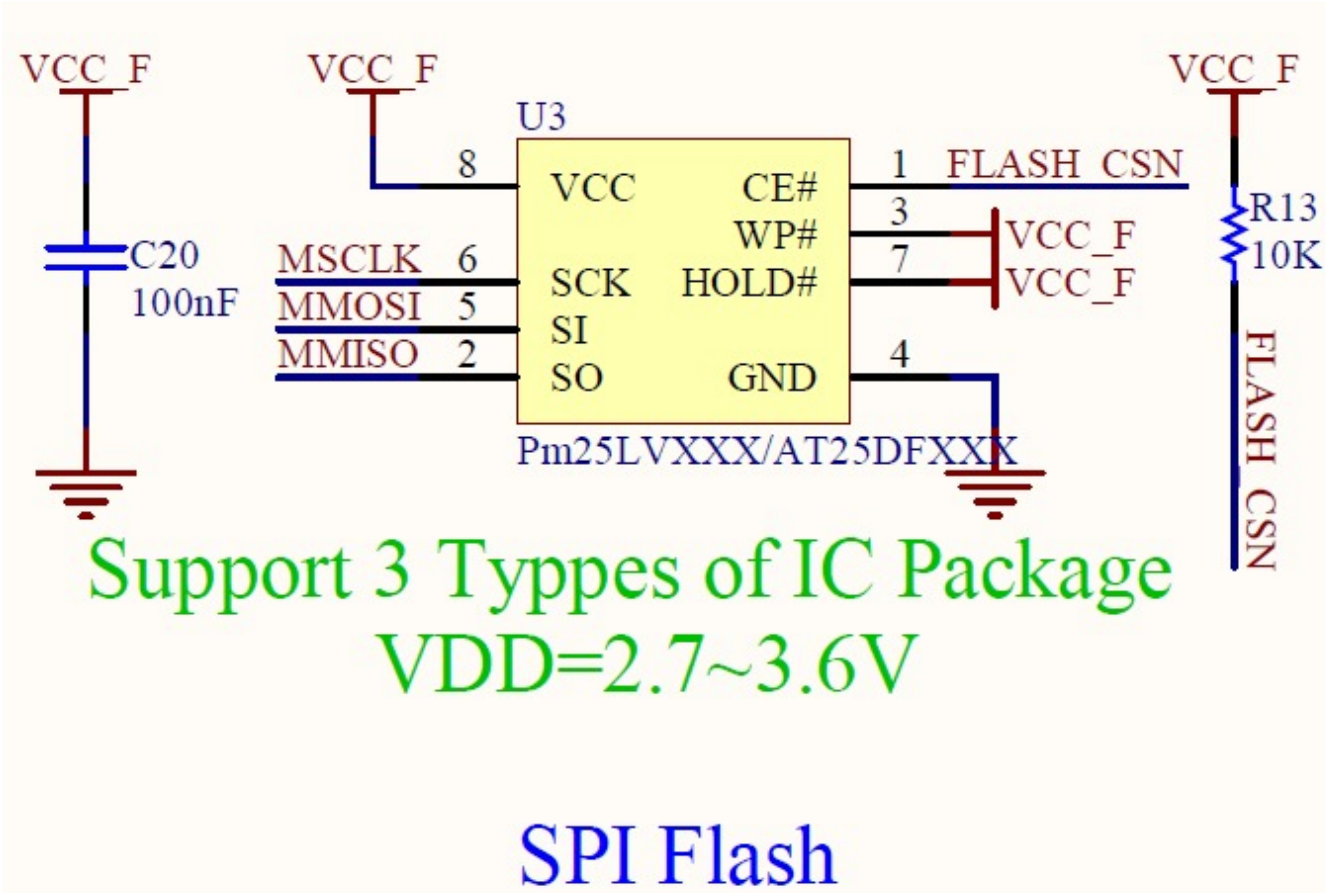
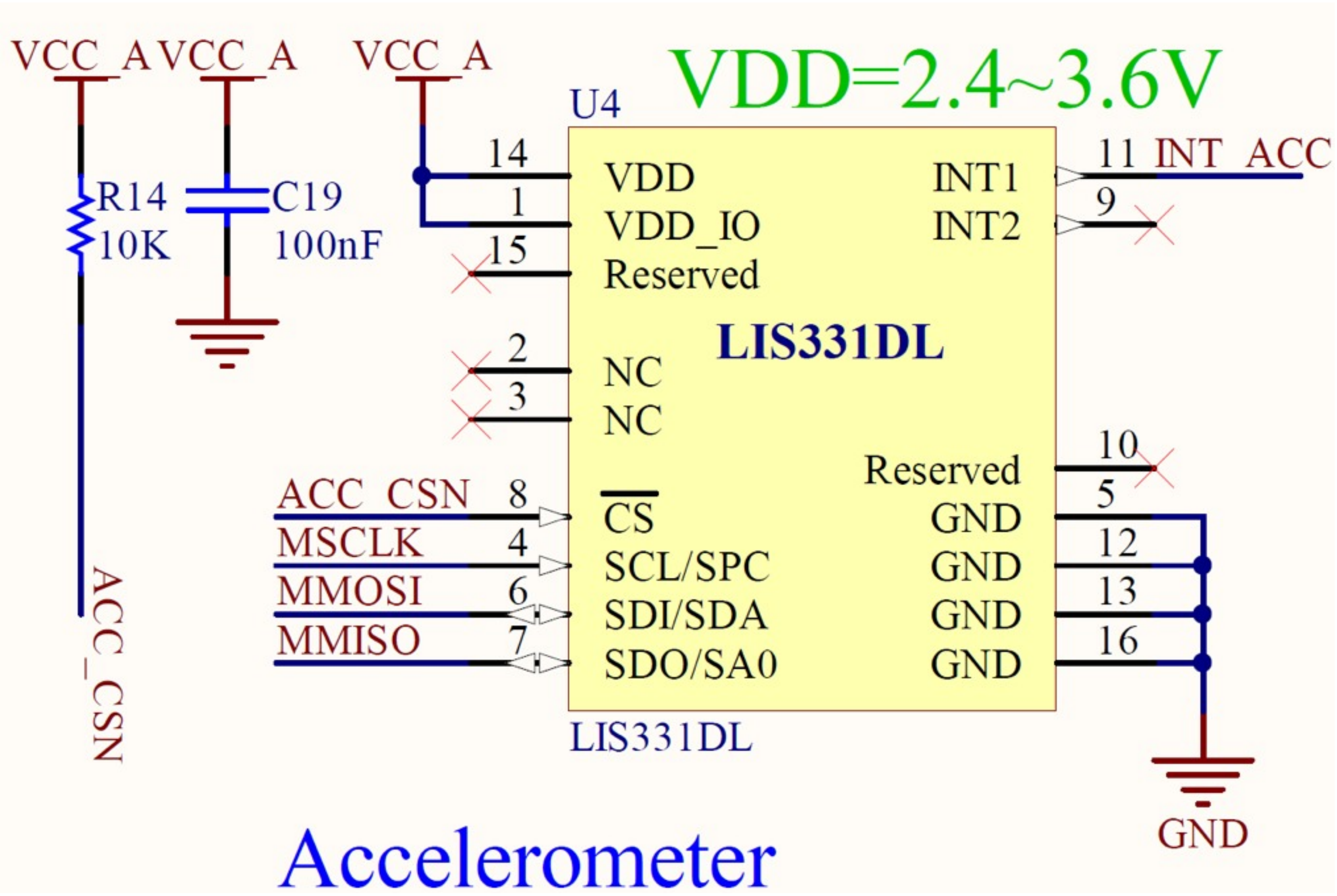
BLE Programming on CC254x

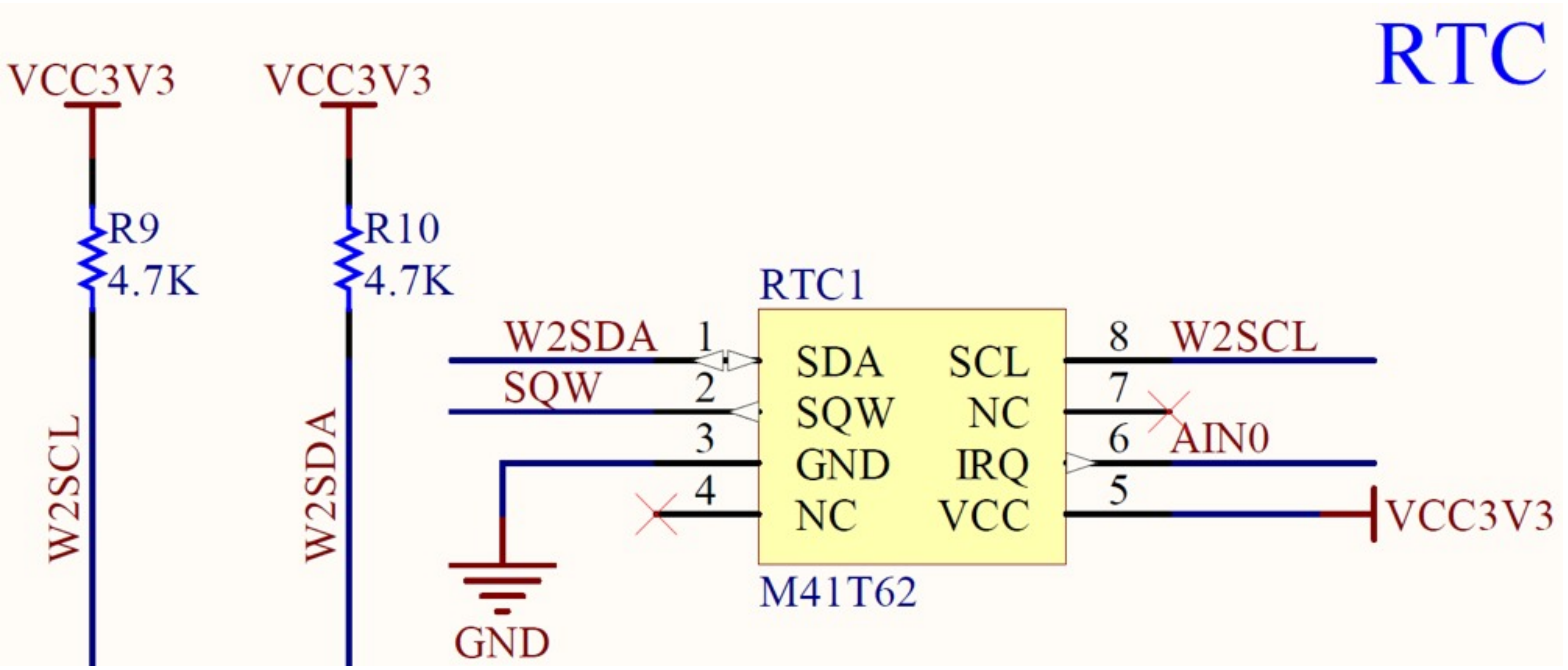

U1
CC2541
W2SCL
W2SDA
RF_P
RF_N
SCL
SDA
RF_N
RF_P
R_BIAS
DCOUPL
RESET_N
C2
1uF
GND
R1
56K
RESET
R2
2.2K
C5
1nF
AIN0
AIN1 key-1
RXD key-2
TXD key-3
P0_4
P0_5
P0_6
P0_7
P0_0
P0_1
P0_2
P0_3
P0_4
P0_5
P0_6
P0_7
VCC_MCU
DVDD2
AVDD5
AVDD2
DVDD1
AVDD3
AVDD1
AVDD6
AVDD4
C8 100nF
C9 1uF
C10 100nF
C12 100nF
C15 220pF
C16 100nF
C17 100nF
P1_0
P1_1
P1_2
P1_3
P1_4
P1_5
P1_6
P1_7
FLASH_CSN
ACC_CSN
INT_ACC
P1_3
CS0
SCLK
MOSI
MISO
P2_0
P2_1/DD
P2_2/DC
P2_3/OSC32K_Q2
P2_4/OSC32K_Q1
SD_CSN
LED0 DD
LED1 DC
XTAL3
Y1
C6 12pF
C7 12pF
XOSC_Q1
XOSC_Q2
NC
EPAD
GND
Y2
32MHz
C13 8pF
C14 8pF

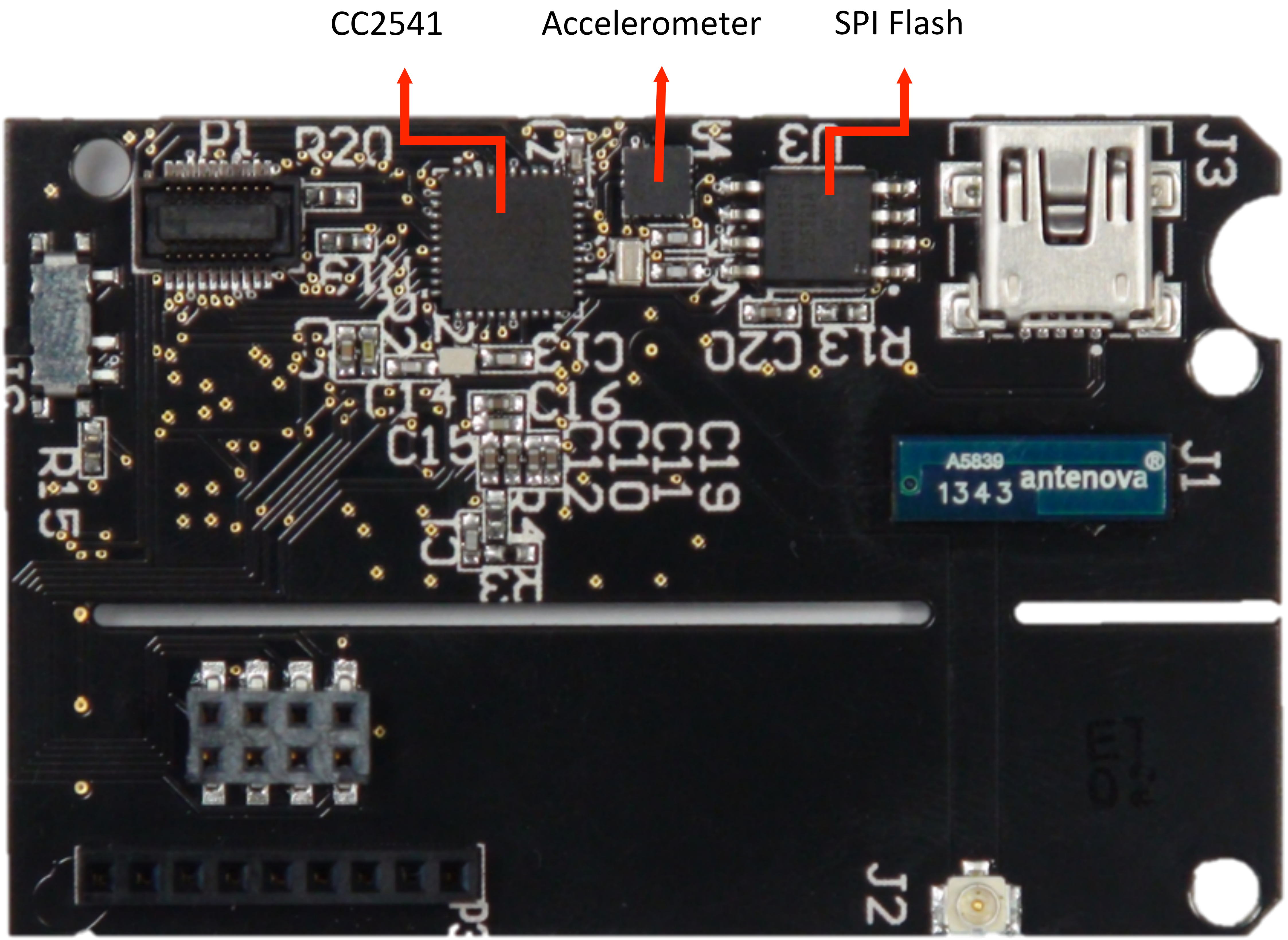


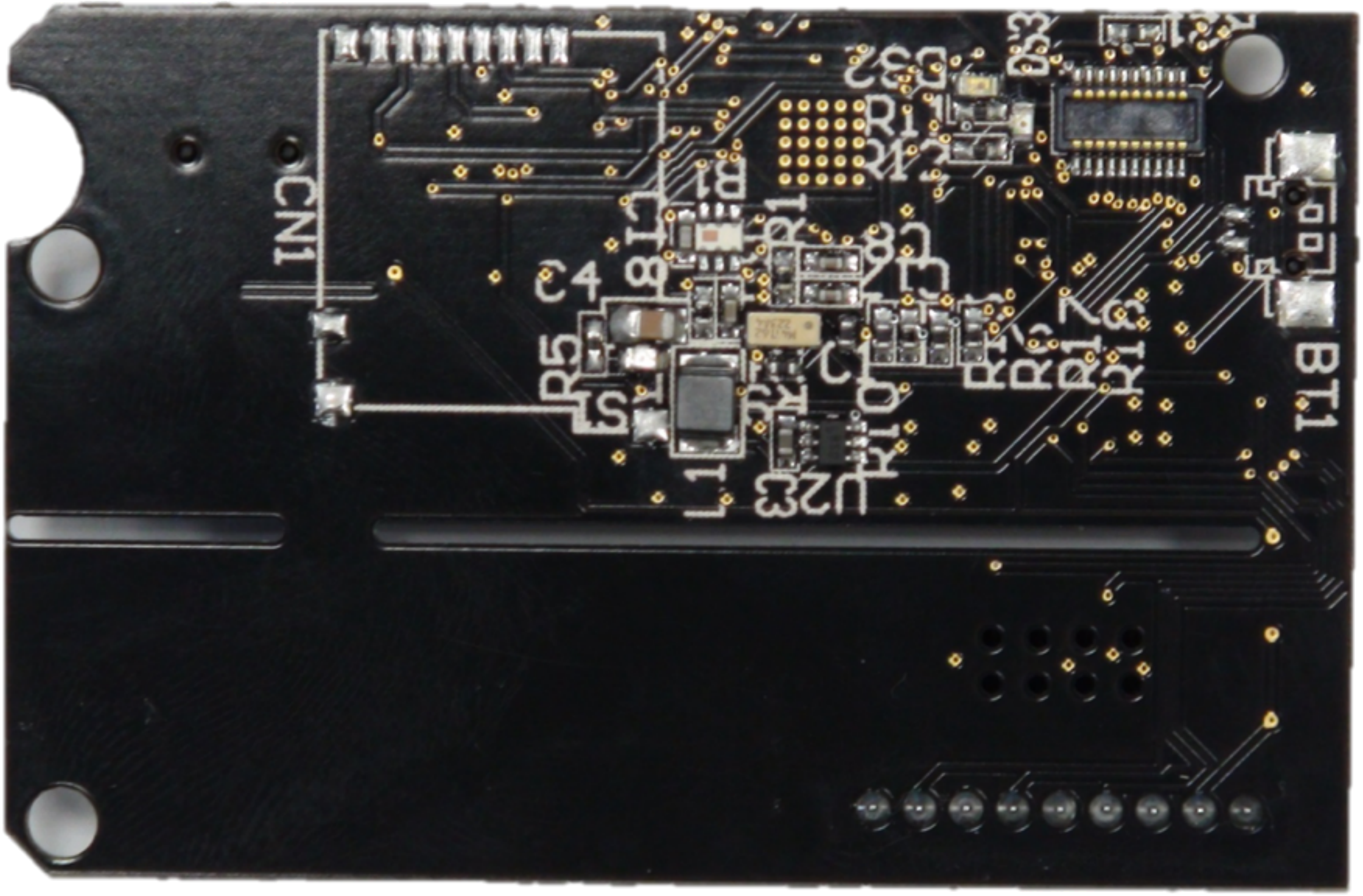
Module Expansion Interface


VCC_F
VCC_F
VCC_F
U3
8
VCC
CE#
1
FLASH_CSN
3
WP#
VCC_F
7
HOLD#
VCC_F
R13
10K
C20
100nF
MSCLK
6
SCK
MMOSI
5
SI
MMISO
2
SO
4
GND
Pm25LVXXX/AT25DFXXX
FLASH_CSN
Support 3 Typpes of IC Package
VDD=2.7~3.6V
SPI Flash


VCC_A
VCC_A
VCC_A
VDD=2.4~3.6V
U4
R14
10K
C19
100nF
14
VDD
1
VDD_IO
15
Reserved
2
NC
3
NC
LIS331DL
11 INT_ACC
INT1
9
INT2
10
Reserved
5
GND
12
GND
13
GND
16
GND
ACC_CSN 8
CS
MSCLK 4
SCL/SPC
MMOSI 6
SDI/SDA
MMISO 7
SDO/SA0
ACC_CSN
LIS331DL
GND
Accelerometer


RTC
VCC3V3
VCC3V3
R9
4.7K
R10
4.7K
W2SCL
W2SDA
RTC1
W2SDA
SQW
1
2
3
4
SDA
SQW
GND
NC
SCL
NC
IRQ
VCC
8
7
6
5
W2SCL
AIN0
VCC3V3
M41T62
GND


CC2541
Accelerometer
SPI Flash
A5839
1343 antenova


CN1
BT1
C4
L1
U23
R5
C1
D32
D33
R11
R12

Reference Document

- Texas Instruments CC2540/41 Bluetooth® Low Energy Software Developer's Guide v1.3.2
  - http://www.ti.com/lit/ug/swru271f/swru271f.pdf
- Content
  - Runtime Support: OSAL, HAL
  - Protocol Stack: GAP, GATT
  - Profiles: GAP Role, GAP Bond Mgr, Device Info,...
  - Application

CC2540 and CC2541

- CC2540
  - Has USB 2 (slave), no I2C hardware
  - Tx power can be 4dBm max
- CC2541
  - No USB2, but has I2C hardware
  - Tx power can be 0 dBm max
- Flash size: 128 KB or 256 KB

Circuit Board

- Different boards using same or diff chip
- EcoBT
  - some may have no RTC or serial flash
  - some may be missing accelerometer
- TI's boards
  - may have LCD, different accelerometer, no RTC
  - May have buttons for input

HAL: Hardware Abstraction Layer

- Want to have uniform view of hardware
    - e.g., access accelerometer whether or not it is connected to SPI or I2C
    - access RTC whether using real or emulated I2C
    - want to blink LED no matter which GPIO pin is used
- Essentially want a "driver" interface
- http://e2e.ti.com/cfs-file.ashx/__key/CommunityServer-Discussions-Components-Files/155/8117.HAL-Driver-API.pdf

Example: ADC

- void HalAdcInit();
- uint16 HalAdcRead(uint8 ch, uint8 res);
- #define HAL_ADC_CHANNEL_0 ..
  - macro names for channel number
- #define HAL_ADC_RESOLUTION_8
  - macro names for ADC resolution

Example: HAL LED

- void HalLedInit(void);
- void HalLedSet(uint8 led, uint8 mode);
  - led is bitmask of (up to 4) LEDs to be turned on
  - mode is on, off, blink, flash, toggle
- void HalLedBlink(uint8 leds, uint8 numBlinks, uint8 percent, uint16 period);
- HalLedEnterSleep(), HalLedExitSleep();

OSAL: Operating System Abstraction Layer

- Task loop: (8-bit task ID)
- Messaging (16-bit length, destTaskID)
- Events: (16-bit bitmap, 8-bit code)
- ISR Registration and interrupt mgmt
- dynamic memory management
- Timer-based task triggering
- Power Management

OSAL Task Scheduling

- ID from 0 .. 255
  - lower ID => higher priority
- Priority driven, cooperative (nonpreemptive)
  - link layer => highest priority (timing sensitive)
  - User tasks usually has lowest priority
- Each task provides two callbacks
  - _Init(), _ProcessEvent()

Events in OSAL

- 16-bit bitmask, one bit per event
- user application can define its own events
- Any layer of software can "set" (emit) an event for any task, including itself
  - osal_set_event(uint8 taskID, uint16 eventFlag);
  - osal_start_timerEx(uint8 taskID, uint16 eventID, uint32 timeout);
- Example: START_DEVICE_EVT,

OSAL-supported Message Events

- #define SYS_EVENT_MSG 0x8000
- allows one task to send a message to another task
    - Sender: osal_msg_allocate(), then osal_msg_send();
    - Receiver: osal_msg_receive(), then osal_msg_deallocate();
- More useful when you have larger data to pass

BLE Stack

- GAP: Generic Access Profile
  - Role: central/peripheral, broadcaster/observer
  - Discovery, link establishment and termination
  - Connection interval, slave latency
- GATT: Generic Attribute Profile
  - Role: Client/Server
  - Services: Device/vendor name (supports GAP), and any "characteristic" (named values)

GAP Roles

- Paired mode
  - Central (master) vs. Peripheral (slave)
- Unpaired mode
  - Broadcaster (sender) vs. Observer (receiver)
- Can combine roles
  - e.g., Peripheral AND Broadcaster

Peripheral Role

- States:
  - Init, started, advertising, waiting, waiting after timeout, connected, error
- GAP Role is a task
  - ..._Init(), ..._ProcessEvent()
- peripheral.c contains functions that define the device GAP behavior
  - Connection interval, slave latency, supervision timeout

GATT Server

- Provides "service(s)"
  - (GAP) Device/Vendor name, product ID
  - (GATT) primary service type, characteristic declaration, enable notification, and descriptor
- Accessing Characteristic Values
  - By either UUID (standard) or handle (address)
  - Read, Discover, Write, Write of Characteristic Value or Descriptor