**Regarding Central\_UART**

Central\_UART using NCS SDK 2.4.0.

Board：nrf5340dk\_nrf5340\_cpuapp

Configuration：prj.conf

Extra CMake arguments：-DSHIELD=nrf21540\_ek

#-----------------------------------------------------------------------------------------

// Add the following configuration to nrf5340dk\_nrf5340\_cpuapp.overlay.

/ {

    chosen {

        nordic,nus-uart = &uart0;

    };

};

&pinctrl {

    uart0\_default: uart0\_default {

        group1 {

            psels = <NRF\_PSEL(UART\_TX, 0, 20)>,

                <NRF\_PSEL(UART\_RX, 0, 22)>;

        };

        group2 {

            psels = <NRF\_PSEL(UART\_RX, 0, 22)>;

        };

    };

    uart0\_sleep: uart0\_sleep {

        group1 {

            psels = <NRF\_PSEL(UART\_TX, 0, 20)>,

                <NRF\_PSEL(UART\_RX, 0, 22)>;

            low-power-enable;

        };

    };

};

&qspi {

    status = "disabled";

};

&i2c0 {

    status = "disabled";

};

&uart0 {

    status = "okay";

    current-speed = <115200>;

    pinctrl-0 = <&uart0\_default>;

    pinctrl-1 = <&uart0\_sleep>;

    pinctrl-names = "default", "sleep";

};

/{

    nrf\_radio\_fem: nrf21540\_fem {

    compatible = "nordic,nrf21540-fem";

    tx-en-gpios = <&gpio0 21 GPIO\_ACTIVE\_HIGH>;

    rx-en-gpios = <&gpio0 27 GPIO\_ACTIVE\_HIGH>;

    pdn-gpios = <&gpio0 26 GPIO\_ACTIVE\_HIGH>;

    //ant-sel-gpios = <&gpio1 5 GPIO\_ACTIVE\_HIGH>;

    mode-gpios = <&gpio1 4 GPIO\_ACTIVE\_HIGH>;

    spi-if = <&nrf\_radio\_fem\_spi>;

    supply-voltage-mv = <3000>;

    };

};

fem\_spi: &spi4 {

    status = "okay";

    pinctrl-0 = <&spi0\_default\_alt>;

    pinctrl-1 = <&spi0\_sleep\_alt>;

    pinctrl-names = "default", "sleep";

    cs-gpios = <&gpio1 6 GPIO\_ACTIVE\_LOW>;

    nrf\_radio\_fem\_spi: nrf21540\_fem\_spi@0 {

            compatible = "nordic,nrf21540-fem-spi";

            status = "okay";

            reg = <0>;

            spi-max-frequency = <8000000>;

    };

};

&pinctrl {

    spi0\_default\_alt: spi0\_default\_alt {

        group1 {

            psels = <NRF\_PSEL(SPIM\_SCK, 1, 15)>,

                <NRF\_PSEL(SPIM\_MOSI, 1, 8)>,

                <NRF\_PSEL(SPIM\_MISO, 0, 25)>;

        };

    };

    spi0\_sleep\_alt: spi0\_sleep\_alt {

        group1 {

            psels = <NRF\_PSEL(SPIM\_SCK, 1, 15)>,

                <NRF\_PSEL(SPIM\_MOSI, 1, 8)>,

                <NRF\_PSEL(SPIM\_MISO, 0, 25)>;

            low-power-enable;

        };

    };

};

/ {

    aliases {

            nrf21540-spi = &spi4;

    };

};

#-----------------------------------------------------------------------------------------

// Add the following configuration to prj.conf.

# Enable the UART driver

CONFIG\_UART\_ASYNC\_API=y

CONFIG\_NRFX\_UARTE0=y

CONFIG\_SERIAL=y

CONFIG\_CONSOLE=y

CONFIG\_UART\_CONSOLE=y

# Enable the BLE stack with GATT Client configuration

CONFIG\_BT=y

CONFIG\_BT\_CENTRAL=y

CONFIG\_BT\_SMP=y

CONFIG\_BT\_GATT\_CLIENT=y

# Enable the BLE modules from NCS

CONFIG\_BT\_NUS\_CLIENT=y

CONFIG\_BT\_SCAN=y

CONFIG\_BT\_SCAN\_FILTER\_ENABLE=y

CONFIG\_BT\_SCAN\_UUID\_CNT=1

CONFIG\_BT\_GATT\_DM=y

CONFIG\_HEAP\_MEM\_POOL\_SIZE=2048

# This example requires more workqueue stack

CONFIG\_SYSTEM\_WORKQUEUE\_STACK\_SIZE=2048

# Enable bonding

CONFIG\_BT\_SETTINGS=y

CONFIG\_FLASH=y

CONFIG\_FLASH\_PAGE\_LAYOUT=y

CONFIG\_FLASH\_MAP=y

CONFIG\_NVS=y

CONFIG\_SETTINGS=y

# Config logger

CONFIG\_LOG=y

CONFIG\_USE\_SEGGER\_RTT=y

CONFIG\_LOG\_BACKEND\_RTT=y

CONFIG\_LOG\_BACKEND\_UART=n

CONFIG\_LOG\_PRINTK=n

CONFIG\_ASSERT=y

### ### 32MHz internal capacitance (XC1 and XC2) settings include: ### ###

CONFIG\_SOC\_ENABLE\_LFXO=y

CONFIG\_SOC\_LFXO\_CAP\_INT\_7PF=y

CONFIG\_SOC\_HFXO\_CAP\_INTERNAL=y

# Internal capacitors 7~20pF,step0.5pF (set Value 14~40,step1)

# 22 to get 11 pF

CONFIG\_SOC\_HFXO\_CAP\_INT\_VALUE\_X2=25

CONFIG\_BUILD\_WITH\_TFM=y

#-----------------------------------------------------------------------------------------

// Contents of the main.c program.

. . . . .

#include <hal/nrf\_gpio.h>

#include <zephyr/drivers/gpio.h>

#include <zephyr/logging/log.h>

#include <zephyr/drivers/uart.h>

. . . . .

#define BT40N\_V6 405    //5340+21540

#ifdef BT40N\_V6

    #define DCDC\_VOLTAGE

#else

    #define MONITOR\_TXPOWER\_SETTING 21

#endif

/////

#define client\_sent\_test

static bool is\_connect = false;

#ifdef client\_sent\_test

    struct k\_timer my\_timer;

    #define TIMER\_INTERVAL\_MSEC 500

    uint8\_t count\_val = 0;

#endif

/////

. . . . .

static void ble\_data\_sent(struct bt\_nus\_client \*nus, uint8\_t err,

                    const uint8\_t \*const data, uint16\_t len)

{

    ARG\_UNUSED(nus);

}

. . . . .

static void configure\_gpio(void)

{

. . . . .

#elif BT40N\_V6

    #define APP\_ANTSEL\_PIN  37      //P1.05

    nrf\_gpio\_cfg\_input(5,GPIO\_PULL\_UP);

    nrf\_gpio\_cfg\_output(28);

    nrf\_gpio\_cfg\_output(29);

    nrf\_gpio\_cfg\_output(30);

    nrf\_gpio\_cfg\_output(31);

    nrf\_gpio\_cfg\_output(APP\_ANTSEL\_PIN);

#endif

. . . . .

    uint8\_t rc2 = nrf\_gpio\_pin\_read(5);

    if(rc2 == 0){

        nrf\_gpio\_pin\_clear(APP\_ANTSEL\_PIN);

        printk("Set ANT\_SEL(P1.05) low \n");

    }else{

        nrf\_gpio\_pin\_set(APP\_ANTSEL\_PIN);

        printk("Set ANT\_SEL(P1.05) high \n");

    }

#endif

}

/////

#ifdef client\_sent\_test

uint8\_t hextoascii(uint8\_t hex\_byte)

{

    char result;

    if((hex\_byte>=0)&&(hex\_byte<=9))

        result = hex\_byte + 0x30;

    else if((hex\_byte >= 10)&&(hex\_byte <= 15))

        result = hex\_byte + 0x37;

    else

        result = 0xff;

    return result;

}

void central\_sample\_event(struct k\_timer \*timer\_id){

    if(is\_connect)

    {

        Dummy\_cnt++;

        if(Dummy\_cnt > 0x39)

            Dummy\_cnt = 0x30;

        Dummy\_tx\_buf[3]=Dummy\_cnt;

        printk("Dummy\_tx\_buf = %c\n",Dummy\_cnt);

    }else{

        return 0;

    }

}

K\_TIMER\_DEFINE(my\_timer, central\_sample\_event, NULL);

#endif

/////

/////

#ifdef BT40N\_V6

    #ifdef DCDC\_VOLTAGE

int change\_gpio\_voltage(uint32\_t target\_voltage)

{

    int err = 0;

    #if defined(CONFIG\_BOARD\_NRF5340DK\_NRF5340\_CPUAPP)

    uint32\_t regout = NRF\_UICR->VREGHVOUT;

    LOG\_INF("REGOUT = 0x%08x", regout);

    if ((regout & UICR\_VREGHVOUT\_VREGHVOUT\_Msk)  != target\_voltage) {

        LOG\_INF("Target voltage not set. Configuring");

        // Set NVMC in write mode:

        NRF\_NVMC->CONFIG = NVMC\_CONFIG\_WEN\_Wen << NVMC\_CONFIG\_WEN\_Pos;

        while (NRF\_NVMC->CONFIG != NVMC\_CONFIG\_WEN\_Wen << NVMC\_CONFIG\_WEN\_Pos) {

            // Wait...

        }

        // Write the actual UICR Register:

        NRF\_UICR->VREGHVOUT = (target\_voltage | ~UICR\_VREGHVOUT\_VREGHVOUT\_Msk);

        while (NRF\_NVMC->READY == NVMC\_READY\_READY\_Busy) {

            // Wait...

        }

        // Set NVMC back in read mode:

        NRF\_NVMC->CONFIG = NVMC\_CONFIG\_WEN\_Ren << NVMC\_CONFIG\_WEN\_Pos;

        while (NRF\_NVMC->CONFIG != NVMC\_CONFIG\_WEN\_Ren << NVMC\_CONFIG\_WEN\_Pos) {

            // Wait...

        }

        // Check whether it was set successfully:

        if ((NRF\_UICR->VREGHVOUT & UICR\_VREGHVOUT\_VREGHVOUT\_Msk) != target\_voltage) {

            err = 0;

        }

        // Reset if success. Config will remain on future reboots.

        if (err == 0) {

            NVIC\_SystemReset();

        }

    }

    return err;

    #else

    // implement for other chips

    return err;

    #endif

}

    #endif

#endif

/////

int main(void)

{

. . . . .

configure\_gpio();

. . . . .

/////

#ifdef client\_sent\_test

    #if 1

    k\_timer\_start(&my\_timer, K\_MSEC(TIMER\_INTERVAL\_MSEC), K\_MSEC(TIMER\_INTERVAL\_MSEC));     //start one shot timer that expires after 500 ms

    k\_timer\_status\_sync(&my\_timer);

    #endif

#endif

/////

#ifdef BT40N\_V6

    printk("------------------------------\n");

    uint8\_t \* cap\_xc12 = (uint8\_t \*) 0x500045C4;

    LOG\_INF("0x500045C4 = 0x%x%x%x%x\n",cap\_xc12[3],cap\_xc12[2],cap\_xc12[1],cap\_xc12[0]);

    printk("0x500045C4 = 0x%x%x%x%x\n",cap\_xc12[3],cap\_xc12[2],cap\_xc12[1],cap\_xc12[0]);

    uint8\_t \* int\_cap = (uint8\_t \*) 0x500046D0;

    LOG\_INF("0x500046D0 = 0x%x%x%x%x\n",int\_cap[3],int\_cap[2],int\_cap[1],int\_cap[0]);

    printk("0x500046D0 = 0x%x%x%x%x\n",int\_cap[3],int\_cap[2],int\_cap[1],int\_cap[0]);

    #ifdef DCDC\_VOLTAGE

    change\_gpio\_voltage(UICR\_VREGHVOUT\_VREGHVOUT\_3V3);

    uint8\_t \* int\_reg = (uint8\_t \*) 0x00FF8010;     //VREGHVOUT

    LOG\_INF("0x00FF8010 = 0x%x%x%x%x\n",int\_reg[3],int\_reg[2],int\_reg[1],int\_reg[0]);

    printk("0x00FF8010 = 0x%x%x%x%x\n",int\_reg[3],int\_reg[2],int\_reg[1],int\_reg[0]);

    uint8\_t \* int\_voltge = (uint8\_t \*) 0x50004B00;      //VREGH.DCDCEN:DC/DC enable register for VREGH

    LOG\_INF("0x50004B00 = 0x%x%x%x%x\n",int\_voltge[3],int\_voltge[2],int\_voltge[1],int\_voltge[0]);

    printk("0x50004B00 = 0x%x%x%x%x\n",int\_voltge[3],int\_voltge[2],int\_voltge[1],int\_voltge[0]);

    uint8\_t \* int\_voltge2 = (uint8\_t \*) 0x50004704;     //VREGMAIN.DCDCEN:DC/DC enable register for VREGMAIN

    LOG\_INF("0x50004704 = 0x%x%x%x%x\n",int\_voltge2[3],int\_voltge2[2],int\_voltge2[1],int\_voltge2[0]);

    printk("0x50004704 = 0x%x%x%x%x\n",int\_voltge2[3],int\_voltge2[2],int\_voltge2[1],int\_voltge2[0]);

    printk("------------------------------\n");

    #endif

#endif

            Dummy\_tx\_buf[0]='T';

            Dummy\_tx\_buf[1]='X';

            Dummy\_tx\_buf[2]=':';

            Dummy\_tx\_buf[3]=Dummy\_cnt;

            Dummy\_tx\_buf[4]=0x0d;

            Dummy\_tx\_buf[5]=0x0a;

for (;;) {

            err = bt\_nus\_client\_send(&nus\_client, Dummy\_tx\_buf, 6);

            if (err) {

                LOG\_WRN("Failed to send data over BLE connection"

                "(err %d)", err);

            }

            err = k\_sem\_take(&nus\_write\_sem, NUS\_WRITE\_TIMEOUT);

            if (err) {

                LOG\_WRN("NUS send timeout");

            }

    }

}

**Regarding Peripheral\_UART**

Peripheral\_UART using NCS SDK 2.4.0.

Board：nrf5340dk\_nrf5340\_cpuapp

Configuration：prj.conf

Extra CMake arguments：-DSHIELD=nrf21540\_ek

//----------------------------------------------------------------------------------------

// Add the following configuration to app.overlay.

/{

    nrf\_radio\_fem: nrf21540\_fem {

    compatible = "nordic,nrf21540-fem";

    tx-en-gpios = <&gpio0 21 GPIO\_ACTIVE\_HIGH>;

    rx-en-gpios = <&gpio0 27 GPIO\_ACTIVE\_HIGH>;

    pdn-gpios = <&gpio0 26 GPIO\_ACTIVE\_HIGH>;

    //ant-sel-gpios = <&gpio1 5 GPIO\_ACTIVE\_HIGH>;

    mode-gpios = <&gpio1 4 GPIO\_ACTIVE\_HIGH>;

    spi-if = <&nrf\_radio\_fem\_spi>;

    supply-voltage-mv = <3000>;

    };

};

fem\_spi: &spi4 {

    status = "okay";

    pinctrl-0 = <&spi0\_default\_alt>;

    pinctrl-1 = <&spi0\_sleep\_alt>;

    pinctrl-names = "default", "sleep";

    cs-gpios = <&gpio1 6 GPIO\_ACTIVE\_LOW>;

    nrf\_radio\_fem\_spi: nrf21540\_fem\_spi@0 {

            compatible = "nordic,nrf21540-fem-spi";

            status = "okay";

            reg = <0>;

            spi-max-frequency = <8000000>;

    };

};

&pinctrl {

    spi0\_default\_alt: spi3\_default\_alt {

        group1 {

            psels = <NRF\_PSEL(SPIM\_SCK, 1, 15)>,

                <NRF\_PSEL(SPIM\_MOSI, 1, 8)>,

                <NRF\_PSEL(SPIM\_MISO, 0, 25)>;

        };

    };

    spi0\_sleep\_alt: spi3\_sleep\_alt {

        group1 {

            psels = <NRF\_PSEL(SPIM\_SCK, 1, 15)>,

                <NRF\_PSEL(SPIM\_MOSI, 1, 8)>,

                <NRF\_PSEL(SPIM\_MISO, 0, 25)>;

            low-power-enable;

        };

    };

};

/ {

    aliases {

            nrf21540-spi = &spi4;

    };

};

&pinctrl {

    uart0\_default: uart0\_default {

        group1 {

            psels = <NRF\_PSEL(UART\_TX, 0, 20)>,

                <NRF\_PSEL(UART\_RX, 0, 22)>;

        };

        group2 {

            psels = <NRF\_PSEL(UART\_RX, 0, 22)>;

        };

    };

    uart0\_sleep: uart0\_sleep {

        group1 {

            psels = <NRF\_PSEL(UART\_TX, 0, 20)>,

                <NRF\_PSEL(UART\_RX, 0, 22)>;

            low-power-enable;

        };

    };

};

&qspi {

    status = "disabled";

};

&i2c0 {

    status = "disabled";

};

&uart0 {

    status = "okay";

    current-speed = <115200>;

    pinctrl-0 = <&uart0\_default>;

    pinctrl-1 = <&uart0\_sleep>;

    pinctrl-names = "default", "sleep";

};

/ {

    chosen {

        nordic,nus-uart = &uart0;

    };

};

/delete-node/ &{/pin-controller/pwm0\_default/group1/};

/delete-node/ &{/pin-controller/pwm0\_sleep/group1/};

&arduino\_adc {

    io-channel-map = <0 &adc 0>, <1 &adc 1>, <2 &adc 2>, <3 &adc 3>, <4 &adc 4>, <5 &adc 5>;

};

#-----------------------------------------------------------------------------------------

// Add the following configuration to prj.conf.

# Enable the UART driver

CONFIG\_UART\_ASYNC\_API=y

CONFIG\_NRFX\_UARTE0=y

CONFIG\_SERIAL=y

CONFIG\_GPIO=y

# Make sure printk is printing to the UART console

CONFIG\_CONSOLE=y

CONFIG\_UART\_CONSOLE=y

CONFIG\_HEAP\_MEM\_POOL\_SIZE=2048

CONFIG\_BT=y

CONFIG\_BT\_PERIPHERAL=y

CONFIG\_BT\_DEVICE\_NAME="Nordic\_UART\_Service"

CONFIG\_BT\_DEVICE\_APPEARANCE=833

CONFIG\_BT\_MAX\_CONN=1

CONFIG\_BT\_MAX\_PAIRED=1

# Enable the NUS service

CONFIG\_BT\_NUS=y

# Enable bonding

CONFIG\_BT\_SETTINGS=y

CONFIG\_FLASH=y

CONFIG\_FLASH\_PAGE\_LAYOUT=y

CONFIG\_FLASH\_MAP=y

CONFIG\_NVS=y

CONFIG\_SETTINGS=y

# Enable DK LED and Buttons library

CONFIG\_DK\_LIBRARY=y

# This example requires more workqueue stack

CONFIG\_SYSTEM\_WORKQUEUE\_STACK\_SIZE=2048

# Config logger

CONFIG\_LOG=y

CONFIG\_USE\_SEGGER\_RTT=y

CONFIG\_LOG\_BACKEND\_RTT=y

CONFIG\_LOG\_BACKEND\_UART=n

CONFIG\_LOG\_PRINTK=n

CONFIG\_ASSERT=y

CONFIG\_BT\_DEVICE\_NAME\_DYNAMIC=y

CONFIG\_BT\_DEVICE\_NAME\_MAX=20

CONFIG\_LOG\_BUFFER\_SIZE=4096

# Some command handlers require a large stack.

CONFIG\_MAIN\_STACK\_SIZE=4096

CONFIG\_MPSL\_WORK\_STACK\_SIZE=4096

### ### 32MHz internal capacitance (XC1 and XC2) settings include: ### ###

CONFIG\_SOC\_ENABLE\_LFXO=y

CONFIG\_SOC\_LFXO\_CAP\_INT\_7PF=y

CONFIG\_SOC\_HFXO\_CAP\_INTERNAL=y

# Internal capacitors 7~20pF,step0.5pF (set Value 14~40,step1)

# 22 to get 11 pF

CONFIG\_SOC\_HFXO\_CAP\_INT\_VALUE\_X2=25

CONFIG\_BUILD\_WITH\_TFM=y

#-----------------------------------------------------------------------------------------

// Contents of the main.c program.

. . . . .

#include <zephyr/logging/log.h>

#include <zephyr/drivers/gpio.h>

#include <hal/nrf\_gpio.h>

#include <zephyr/sys/printk.h>

. . . . .

#define BT40N\_V6 405        //5340+21540

#ifdef BT40N\_V6

    #define DCDC\_VOLTAGE

#else

    #define MONITOR\_TXPOWER\_SETTING

#endif

#define Factory\_TC

#ifdef Factory\_TC

    static bool is\_connect = false;

    #define BLE\_ADV\_NAME

    #ifdef BLE\_ADV\_NAME

    static bool command\_pending = false;

    static char name\_tH = '?';

    static char name\_tL = '?';

    static char name\_mH = '?';

    static char name\_mL = '?';

    #endif

#endif

. . . . .

/////

#ifdef BLE\_ADV\_NAME

static const struct bt\_data ad[] = {

    BT\_DATA\_BYTES(BT\_DATA\_FLAGS, (BT\_LE\_AD\_GENERAL | BT\_LE\_AD\_NO\_BREDR))

};

#else

static const struct bt\_data ad[] = {

    BT\_DATA\_BYTES(BT\_DATA\_FLAGS, (BT\_LE\_AD\_GENERAL | BT\_LE\_AD\_NO\_BREDR)),

    BT\_DATA(BT\_DATA\_NAME\_COMPLETE, DEVICE\_NAME, DEVICE\_NAME\_LEN),

};

#endif

/////

static void configure\_gpio(void)

{

. . . . .

#elif BT40N\_V6

    #define APP\_ANTSEL\_PIN  37      //P1.05

    nrf\_gpio\_cfg\_input(5,GPIO\_PULL\_UP);

    nrf\_gpio\_cfg\_output(28);

    nrf\_gpio\_cfg\_output(29);

    nrf\_gpio\_cfg\_output(30);

    nrf\_gpio\_cfg\_output(31);

    nrf\_gpio\_cfg\_output(APP\_ANTSEL\_PIN);    //P1.05

#endif

. . . . .

    uint8\_t rc2 = nrf\_gpio\_pin\_read(5);

    if(rc2 == 0){

        //gpio\_pin\_set(led\_dev1, 05, 1);

        nrf\_gpio\_pin\_clear(APP\_ANTSEL\_PIN);

        printk("Set ANT\_SEL(P1.05) low \n");

    }else{

        //gpio\_pin\_set(led\_dev1, 05, 0);

        nrf\_gpio\_pin\_set(APP\_ANTSEL\_PIN);

        printk("Set ANT\_SEL(P1.05) high \n");

    }

#endif

}

/////

#ifdef BT40N\_V6

    #ifdef DCDC\_VOLTAGE

int change\_gpio\_voltage(uint32\_t target\_voltage)

{

    int err = 0;

    #if defined(CONFIG\_BOARD\_NRF5340DK\_NRF5340\_CPUAPP)

    uint32\_t regout = NRF\_UICR->VREGHVOUT;

    LOG\_INF("REGOUT = 0x%08x", regout);

    if ((regout & UICR\_VREGHVOUT\_VREGHVOUT\_Msk)  != target\_voltage) {

        LOG\_INF("Target voltage not set. Configuring");

        // Set NVMC in write mode:

        NRF\_NVMC->CONFIG = NVMC\_CONFIG\_WEN\_Wen << NVMC\_CONFIG\_WEN\_Pos;

        while (NRF\_NVMC->CONFIG != NVMC\_CONFIG\_WEN\_Wen << NVMC\_CONFIG\_WEN\_Pos) {

            // Wait...

        }

        // Write the actual UICR Register:

        NRF\_UICR->VREGHVOUT = (target\_voltage | ~UICR\_VREGHVOUT\_VREGHVOUT\_Msk);

        while (NRF\_NVMC->READY == NVMC\_READY\_READY\_Busy) {

            // Wait...

        }

        // Set NVMC back in read mode:

        NRF\_NVMC->CONFIG = NVMC\_CONFIG\_WEN\_Ren << NVMC\_CONFIG\_WEN\_Pos;

        while (NRF\_NVMC->CONFIG != NVMC\_CONFIG\_WEN\_Ren << NVMC\_CONFIG\_WEN\_Pos) {

            // Wait...

        }

        // Check whether it was set successfully:

        if ((NRF\_UICR->VREGHVOUT & UICR\_VREGHVOUT\_VREGHVOUT\_Msk) != target\_voltage) {

            err = 0;

        }

        // Reset if success. Config will remain on future reboots.

        if (err == 0) {

            NVIC\_SystemReset();

        }

    }

    return err;

    #else

    // implement for other chips

    return err;

    #endif

}

    #endif

#endif

/////

#ifdef BLE\_ADV\_NAME

static void Command\_Pro(){

        int err = bt\_le\_adv\_stop();

        if (err) {

           LOG\_INF("Advertising failed to stop (err %d)\n", err);

        }else{

           printk("OK\n");

        }

        char temp\_name[] = {'M','9','9',' ','9','9'};

        temp\_name[1] = name\_tH;

        temp\_name[2] = name\_tL;

        temp\_name[4] = name\_mH;

        temp\_name[5] = name\_mL;

        struct bt\_data adname[] = {

        BT\_DATA\_BYTES(BT\_DATA\_FLAGS, (BT\_LE\_AD\_GENERAL | BT\_LE\_AD\_NO\_BREDR)),

        BT\_DATA(BT\_DATA\_NAME\_COMPLETE, temp\_name, 6),

        };

        err = bt\_le\_adv\_start(BT\_LE\_ADV\_CONN, adname, ARRAY\_SIZE(adname), sd, ARRAY\_SIZE(sd));

        if (err) {

           printk("Advertising failed to start (err %d)\n", err);

        }else{

           printk("Advertising restart \n");

        }

}

#endif

int main(void)

{

. . . . .

configure\_gpio();

. . . . .

#elif BT40N\_V6

    uint8\_t rc1 = nrf\_gpio\_pin\_read(5);

    char temp\_name[] = {'B','T','4','0','N','1'};

    if(rc1 == 0){

        nrf\_gpio\_pin\_clear(APP\_ANTSEL\_PIN);

        LOG\_INF("Set ANT\_SEL(P1.05) low! \n");

        printk("Set ANT\_SEL(P1.05) low! \n");

    }else{

        nrf\_gpio\_pin\_set(APP\_ANTSEL\_PIN);

        LOG\_INF("Set ANT\_SEL(P1.05) high! \n");

        printk("Set ANT\_SEL(P1.05) high! \n");

        temp\_name[5] = '2';

    }

    if (err) {

        LOG\_INF("Set Name error:(err %d)\n", err);

    }

#else

    err = bt\_set\_name("Unknow device");

#endif

. . . . .

#ifdef BLE\_ADV\_NAME

    struct bt\_data adname1[] = {

        BT\_DATA\_BYTES(BT\_DATA\_FLAGS, (BT\_LE\_AD\_GENERAL | BT\_LE\_AD\_NO\_BREDR)),

        BT\_DATA(BT\_DATA\_NAME\_COMPLETE, temp\_name, sizeof(temp\_name)),

        };

    err = bt\_le\_adv\_start(BT\_LE\_ADV\_CONN, adname1, ARRAY\_SIZE(adname1), sd, ARRAY\_SIZE(sd));

    if (err) {

        printk("Advertising failed to start (err %d)\n", err);

    }else{

        LOG\_INF("Advertising start \n");

        printk("Advertising start \n");

    }

#elif ADV\_NAME

    err = bt\_le\_adv\_start(BT\_LE\_ADV\_CONN\_NAME, ad, ARRAY\_SIZE(ad), sd,

                  ARRAY\_SIZE(sd));

    if (err) {

        printk("Advertising failed to start (err %d)\n", err);

    }else{

        printk("Advertising start \n");

    }

#else

    err = bt\_le\_adv\_start(BT\_LE\_ADV\_CONN, ad, ARRAY\_SIZE(ad), sd,

                  ARRAY\_SIZE(sd));

    if (err) {

        LOG\_ERR("Advertising failed to start (err %d)", err);

        return 0;

    }else{

        printk("Advertising start \n");

    }

#endif

#ifdef BT40N\_V6

    printk("------------------------------\n");

    uint8\_t \* cap\_xc12 = (uint8\_t \*) 0x500045C4;

    LOG\_INF("0x500045C4 = 0x%x%x%x%x\n",cap\_xc12[3],cap\_xc12[2],cap\_xc12[1],cap\_xc12[0]);

    printk("0x500045C4 = 0x%x%x%x%x\n",cap\_xc12[3],cap\_xc12[2],cap\_xc12[1],cap\_xc12[0]);

    uint8\_t \* int\_cap = (uint8\_t \*) 0x500046D0;

    LOG\_INF("0x500046D0 = 0x%x%x%x%x\n",int\_cap[3],int\_cap[2],int\_cap[1],int\_cap[0]);

    printk("0x500046D0 = 0x%x%x%x%x\n",int\_cap[3],int\_cap[2],int\_cap[1],int\_cap[0]);

    #ifdef DCDC\_VOLTAGE

    change\_gpio\_voltage(UICR\_VREGHVOUT\_VREGHVOUT\_3V3);

    uint8\_t \* int\_reg = (uint8\_t \*) 0x00FF8010;     //VREGHVOUT

    LOG\_INF("0x00FF8010 = 0x%x%x%x%x\n",int\_reg[3],int\_reg[2],int\_reg[1],int\_reg[0]);

    printk("0x00FF8010 = 0x%x%x%x%x\n",int\_reg[3],int\_reg[2],int\_reg[1],int\_reg[0]);

    uint8\_t \* int\_voltge = (uint8\_t \*) 0x50004B00;      //VREGH.DCDCEN:DC/DC enable register for VREGH

    LOG\_INF("0x50004B00 = 0x%x%x%x%x\n",int\_voltge[3],int\_voltge[2],int\_voltge[1],int\_voltge[0]);

    printk("0x50004B00 = 0x%x%x%x%x\n",int\_voltge[3],int\_voltge[2],int\_voltge[1],int\_voltge[0]);

    uint8\_t \* int\_voltge2 = (uint8\_t \*) 0x50004704;     //VREGMAIN.DCDCEN:DC/DC enable register for VREGMAIN

    LOG\_INF("0x50004704 = 0x%x%x%x%x\n",int\_voltge2[3],int\_voltge2[2],int\_voltge2[1],int\_voltge2[0]);

    printk("0x50004704 = 0x%x%x%x%x\n",int\_voltge2[3],int\_voltge2[2],int\_voltge2[1],int\_voltge2[0]);

    printk("------------------------------\n");

    #endif

#endif

for (;;) {

        if(command\_pending){

            command\_pending = false;

            Command\_Pro();

        }

        struct uart\_data\_t \*buf = k\_fifo\_get(&fifo\_uart\_rx\_data,

                             K\_FOREVER);

        bt\_nus\_send(NULL, buf->data, buf->len);

        k\_sleep(K\_MSEC(RUN\_LED\_BLINK\_INTERVAL));

    }

}

#-----------------------------------------------------------------------------------------

//Path：C:\ncs\_v240\v2.4.0\zephyr\samples\bluetooth\hci\_rpmsg/nrf5340dk\_nrf5340\_cpunet.overlay

//Add the following configuration to nrf5340dk\_nrf5340\_cpunet.overlay.

/{

    nrf\_radio\_fem: nrf21540\_fem {

    compatible = "nordic,nrf21540-fem";

    tx-en-gpios = <&gpio0 21 GPIO\_ACTIVE\_HIGH>;

    rx-en-gpios = <&gpio0 27 GPIO\_ACTIVE\_HIGH>;

    pdn-gpios = <&gpio0 26 GPIO\_ACTIVE\_HIGH>;

    //ant-sel-gpios = <&gpio1 5 GPIO\_ACTIVE\_HIGH>;

    mode-gpios = <&gpio1 4 GPIO\_ACTIVE\_HIGH>;

    spi-if = <&nrf\_radio\_fem\_spi>;

    supply-voltage-mv = <3000>;

    };

};

fem\_spi: &spi0 {

    status = "okay";

    pinctrl-0 = <&spi0\_default\_alt>;

    pinctrl-1 = <&spi0\_sleep\_alt>;

    pinctrl-names = "default", "sleep";

    cs-gpios = <&gpio1 6 GPIO\_ACTIVE\_LOW>;

    nrf\_radio\_fem\_spi: nrf21540\_fem\_spi@0 {

            compatible = "nordic,nrf21540-fem-spi";

            status = "okay";

            reg = <0>;

            spi-max-frequency = <8000000>;

    };

};

&pinctrl {

    spi0\_default\_alt: spi0\_default\_alt {

        group1 {

            psels = <NRF\_PSEL(SPIM\_SCK, 1, 15)>,

                <NRF\_PSEL(SPIM\_MOSI, 1, 8)>,

                <NRF\_PSEL(SPIM\_MISO, 0, 25)>;

        };

    };

    spi0\_sleep\_alt: spi0\_sleep\_alt {

        group1 {

            psels = <NRF\_PSEL(SPIM\_SCK, 1, 15)>,

                <NRF\_PSEL(SPIM\_MOSI, 1, 8)>,

                <NRF\_PSEL(SPIM\_MISO, 0, 25)>;

            low-power-enable;

        };

    };

};

/ {

    aliases {

            nrf21540-spi = &spi0;

    };

};

#-----------------------------------------------------------------------------------------

//Path：C:\ncs\_v240\v2.4.0\zephyr\samples\bluetooth\hci\_rpmsg/prf.conf

//Add the following configuration to prf.conf.

CONFIG\_IPC\_SERVICE=y

CONFIG\_MBOX=y

CONFIG\_HEAP\_MEM\_POOL\_SIZE=8192

CONFIG\_MAIN\_STACK\_SIZE=512

CONFIG\_SYSTEM\_WORKQUEUE\_STACK\_SIZE=512

CONFIG\_BT=y

CONFIG\_BT\_HCI\_RAW=y

CONFIG\_BT\_HCI\_RAW\_RESERVE=1

CONFIG\_BT\_MAX\_CONN=16

# Workaround: Unable to allocate command buffer when using K\_NO\_WAIT since

# Host number of completed commands does not follow normal flow control.

CONFIG\_BT\_BUF\_CMD\_TX\_COUNT=10

# Enable and adjust the below value as necessary

# CONFIG\_BT\_BUF\_EVT\_RX\_COUNT=16

# CONFIG\_BT\_BUF\_EVT\_RX\_SIZE=255

# CONFIG\_BT\_BUF\_ACL\_RX\_SIZE=255

# CONFIG\_BT\_BUF\_ACL\_TX\_SIZE=251

# CONFIG\_BT\_BUF\_CMD\_TX\_SIZE=255

##### FEM Config #####

CONFIG\_MPSL\_FEM\_DEVICE\_CONFIG\_254=y

CONFIG\_FEM=y

CONFIG\_MPSL=y

CONFIG\_MPSL\_FEM=y

CONFIG\_MPSL\_FEM\_NRF21540\_GPIO=y

CONFIG\_MPSL\_FEM\_NRF21540\_GPIO\_SPI=n

CONFIG\_MPSL\_FEM\_NRF21540\_TX\_GAIN\_DB=10

CONFIG\_MPSL\_FEM\_NRF21540\_TX\_GAIN\_DB\_POUTA=10

CONFIG\_MPSL\_FEM\_NRF21540\_TX\_GAIN\_DB\_POUTB=10

##### TX Power Config #####

CONFIG\_BT\_CTLR\_TX\_PWR\_DYNAMIC\_CONTROL=y

CONFIG\_BT\_CTLR\_TX\_PWR\_ANTENNA=17