PROJECT REPORT

DMA

The DMA provided a significant speed up in memory transfers and it's clear when large bytes of data is being moved.

The option of sending a 1 byte sized transfer vs a 4 byte sized transfer is useful because for small transfers 1 byte transfer is faster and for large transfers 4 byte is faster.

The beagle bone in comparison to frdm is way faster because of a powerful processor and it's very clear when we compare the profiled times.

Indeed, when compared to standard library and our version of memory transfer, DMA is significantly faster especially for large bytes of memory transfer.

SPI, RTC and NRF module

We are logging data which we get from various registers on nrf module we called different write read functions for different register

Each read/write function will start with cheap enable and should end with cheap disable this is because write / read command should be able to differentiate between incoming data and command.

Also we are reading status after each register to avoid buffer overrun condition so that it won't block next incoming data

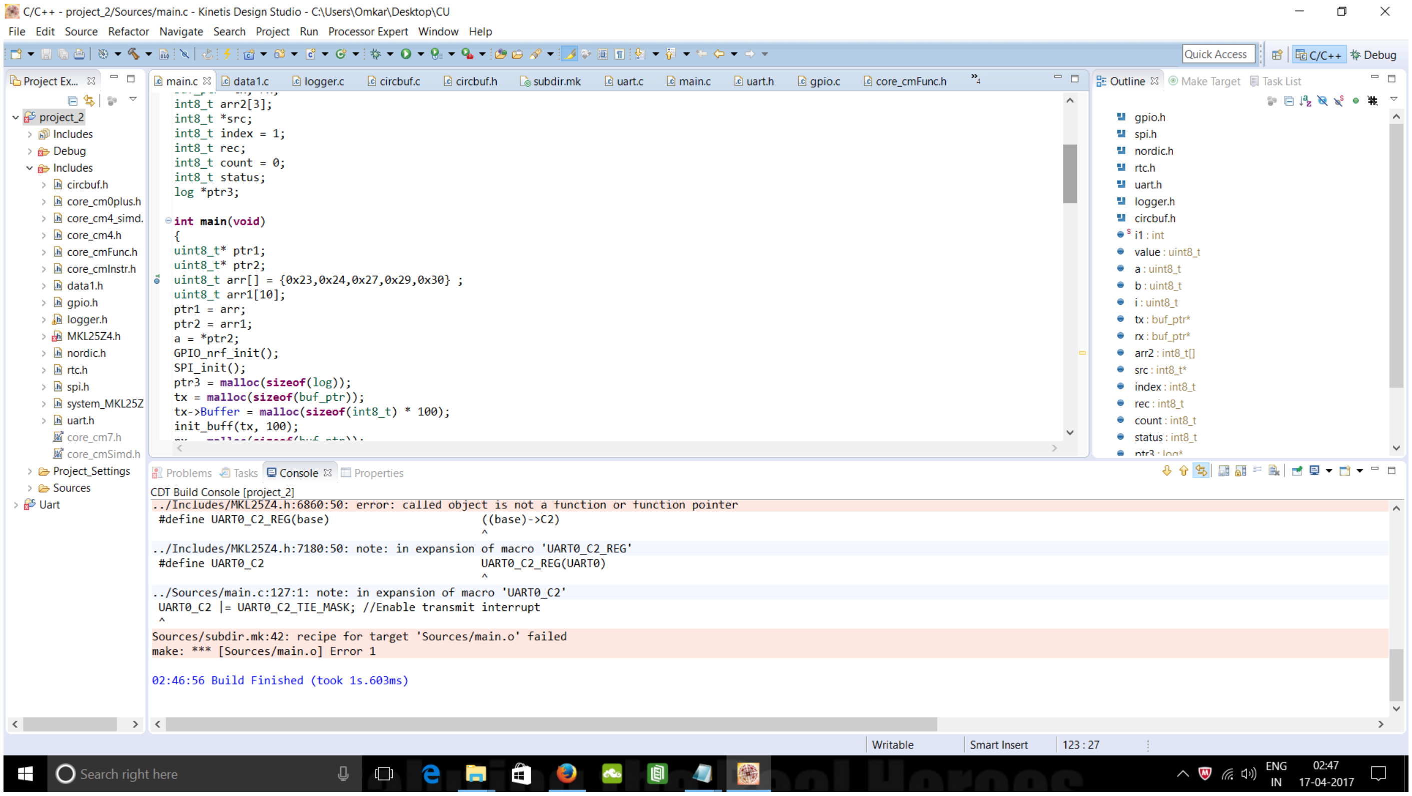
Before each read and write we have to wait for transmission buffer empty flag to set or receive buffer empty flag to set to know buffers are empty and avoid overwriting

In main at that end and before while loop we are logging data from these read and writes and printing it using UART

Once we empty buffer for the first time RTC interrupt will get set and will log data which will get information about time register here we will enable transmit interrupt of uart to print that data and after printing we are disabling uart so that it will be enabled and will print data at every second as timer interrupt is the one to generate it and disable as well.

We have added '(shell +%s)' command to settings to get epoch time and will print that time every second.

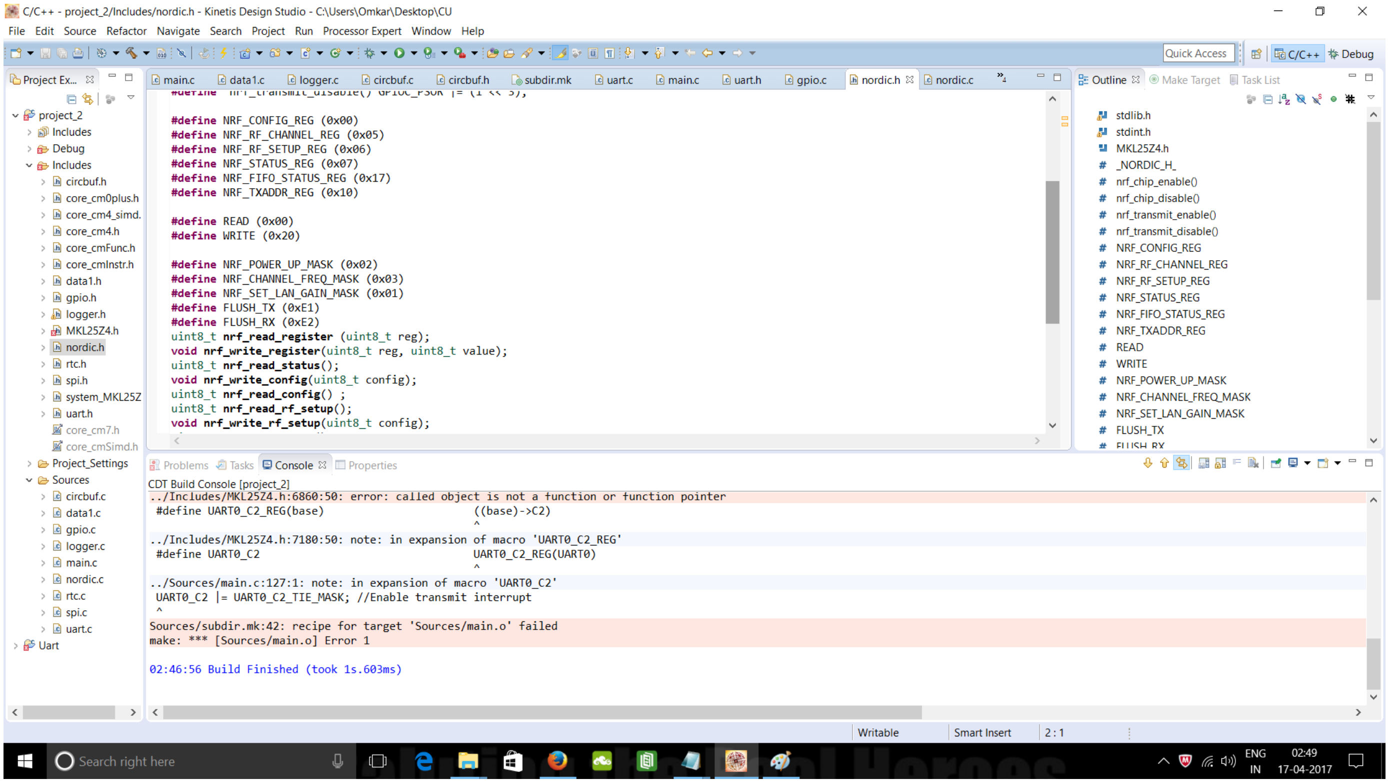
Here is our input to nrf module register which is array arr for tx addr and different mask mentioned below for different registers


C/C++ - project_2/Sources/main.c - Kinetis Design Studio - C:\Users\Omkar\Desktop\CU

```
int8_t arr2[3];
int8_t *src;
int8_t index = 1;
int8_t rec;
int8_t count = 0;
int8_t status;
log *ptr3;

int main(void)
{
uint8_t* ptr1;
uint8_t* ptr2;
uint8_t arr[] = {0x23,0x24,0x27,0x29,0x30} ;
uint8_t arr1[10];
ptr1 = arr;
ptr2 = arr1;
a = *ptr2;
GPIO_nrf_init();
SPI_init();
ptr3 = malloc(sizeof(log));
tx = malloc(sizeof(buf_ptr));
tx->Buffer = malloc(sizeof(int8_t) * 100);
init_buff(tx, 100);
```

CDT Build Console [project_2]

```
../Includes/MKL25Z4.h:6860:50: error: called object is not a function or function pointer
 #define UART0_C2_REG(base)                       ((base)->C2)
                                                  ^
../Includes/MKL25Z4.h:7180:50: note: in expansion of macro 'UART0_C2_REG'
 #define UART0_C2                                 UART0_C2_REG(UART0)
                                                  ^
../Sources/main.c:127:1: note: in expansion of macro 'UART0_C2'
 UART0_C2 |= UART0_C2_TIE_MASK; //Enable transmit interrupt
 ^
Sources/subdir.mk:42: recipe for target 'Sources/main.o' failed
make: *** [Sources/main.o] Error 1

02:46:56 Build Finished (took 1s.603ms)
```

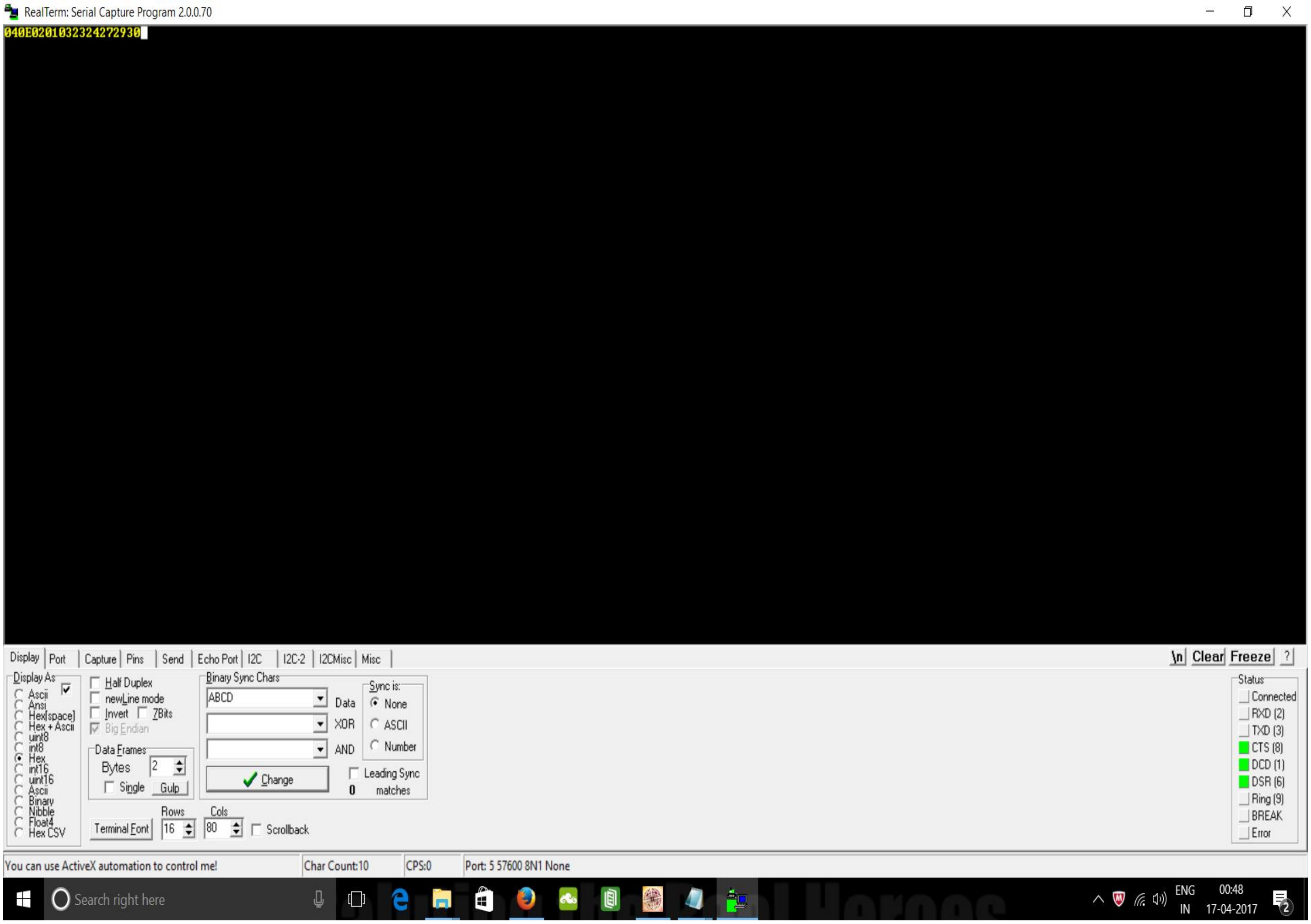

C/C++ - project_2/Includes/nordic.h - Kinetis Design Studio - C:\Users\Omkar\Desktop\CU

```
#define NRF_CONFIG_REG (0x00)
#define NRF_RF_CHANNEL_REG (0x05)
#define NRF_RF_SETUP_REG (0x06)
#define NRF_STATUS_REG (0x07)
#define NRF_FIFO_STATUS_REG (0x17)
#define NRF_TXADDR_REG (0x10)

#define READ (0x00)
#define WRITE (0x20)

#define NRF_POWER_UP_MASK (0x02)
#define NRF_CHANNEL_FREQ_MASK (0x03)
#define NRF_SET_LAN_GAIN_MASK (0x01)
#define FLUSH_TX (0xE1)
#define FLUSH_RX (0xE2)
uint8_t nrf_read_register (uint8_t reg);
void nrf_write_register(uint8_t reg, uint8_t value);
uint8_t nrf_read_status();
void nrf_write_config(uint8_t config);
uint8_t nrf_read_config() ;
uint8_t nrf_read_rf_setup();
void nrf_write_rf_setup(uint8_t config);
```

CDT Build Console [project_2]

```
../Includes/MKL25Z4.h:6860:50: error: called object is not a function or function pointer
 #define UART0_C2_REG(base)                       ((base)->C2)
                                                  ^
../Includes/MKL25Z4.h:7180:50: note: in expansion of macro 'UART0_C2_REG'
 #define UART0_C2                                 UART0_C2_REG(UART0)
                                                  ^
../Sources/main.c:127:1: note: in expansion of macro 'UART0_C2'
 UART0_C2 |= UART0_C2_TIE_MASK; //Enable transmit interrupt
 ^
Sources/subdir.mk:42: recipe for target 'Sources/main.o' failed
make: *** [Sources/main.o] Error 1

02:46:56 Build Finished (took 1s.603ms)
```

And here is the output



RealTerm: Serial Capture Program 2.0.0.70

```
040E0201032324272930
```

Display As
○ Ascii ☑
○ Ansi
○ Hex[space]
○ Hex + Ascii
○ uint8
○ int8
● Hex
○ int16
○ uint16
○ Ascii
○ Binary
○ Nibble
○ Float4
○ Hex CSV

□ Half Duplex
□ newLine mode
□ Invert □ 7Bits
☑ Big Endian

Data Frames
Bytes 2
□ Single Gulp

Binary Sync Chars
ABCD Data
XOR
AND

Sync is:
● None
○ ASCII
○ Number

✓ Change
□ Leading Sync
0 matches

Terminal Font Rows 16 Cols 80 □ Scrollback

Status
□ Connected
□ RXD (2)
□ TXD (3)
🟩 CTS (8)
🟩 DCD (1)
🟩 DSR (6)
□ Ring (9)
□ BREAK
□ Error

You can use ActiveX automation to control me! Char Count:10 CPS:0 Port: 5 57600 8N1 None