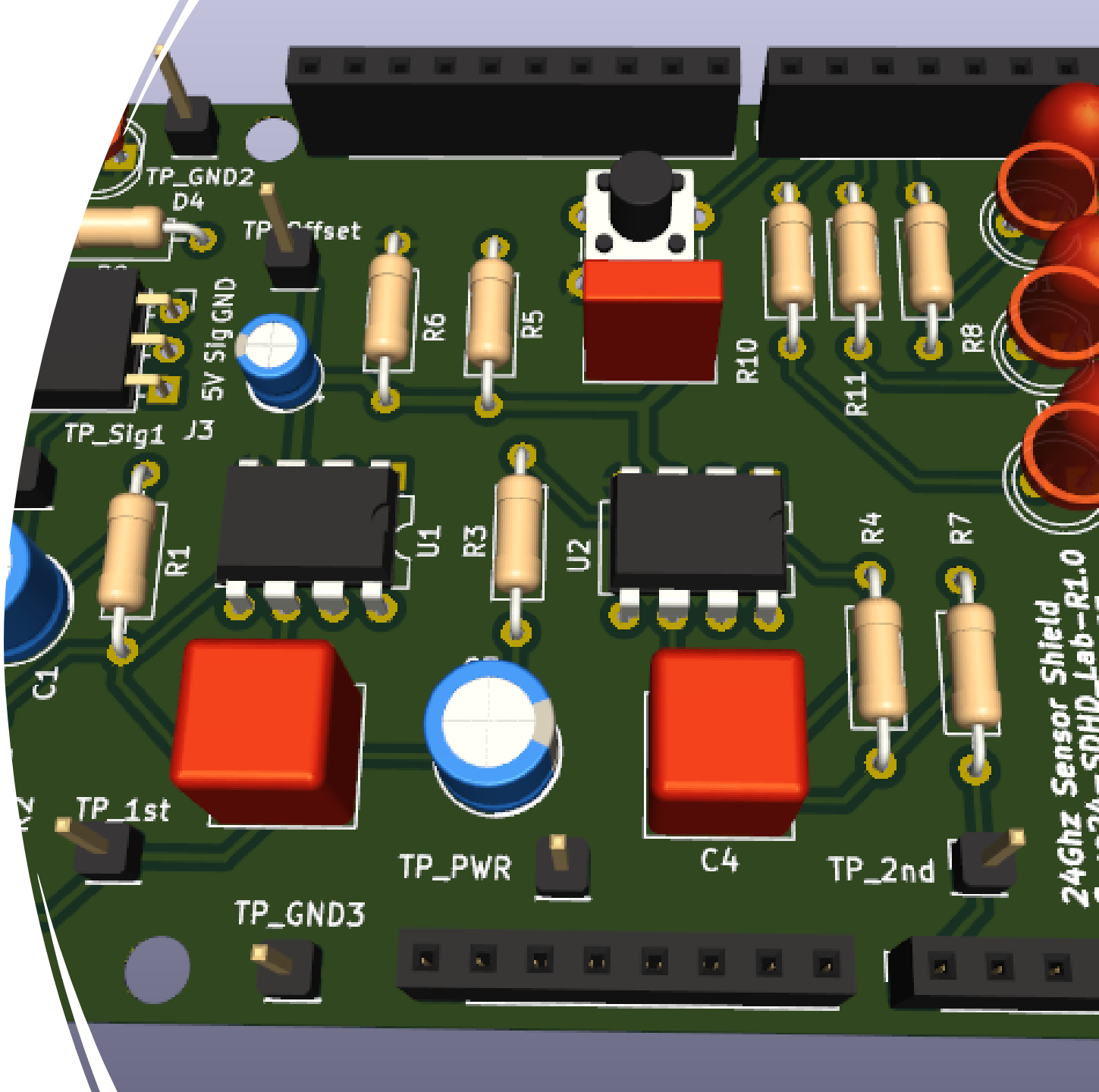
Lab 3: Software

Signal Hardware Driven Design

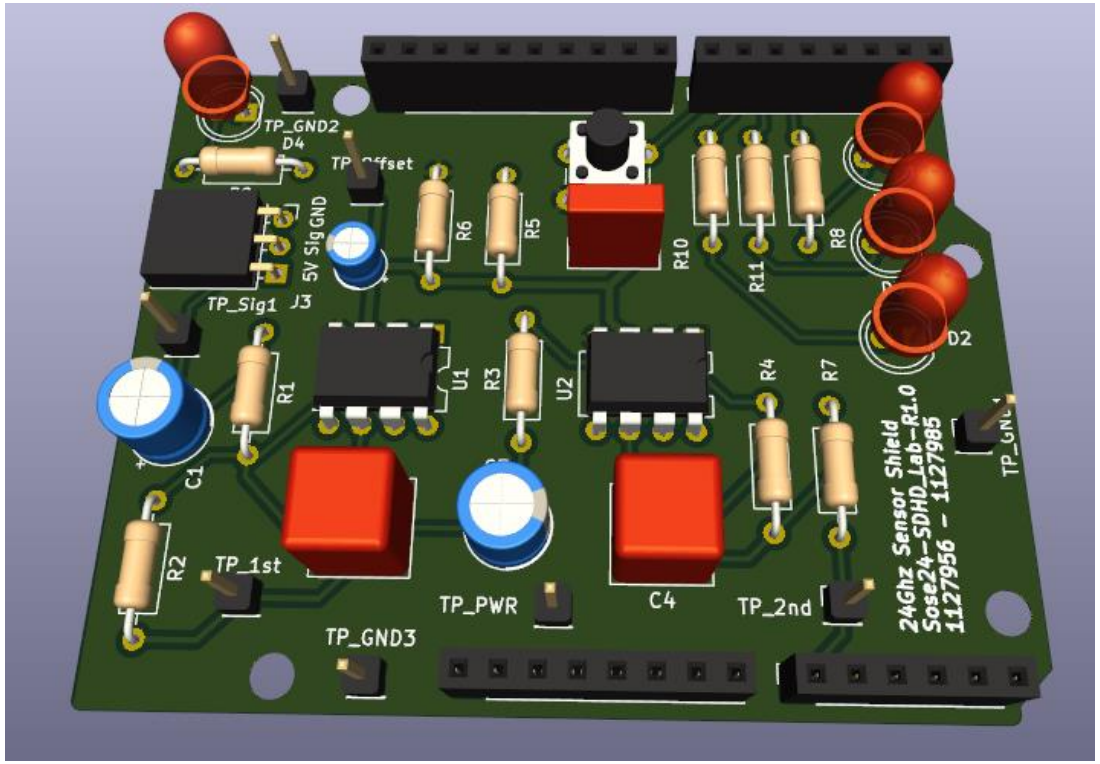
Nguyen Tien Anh Ha - 1127956

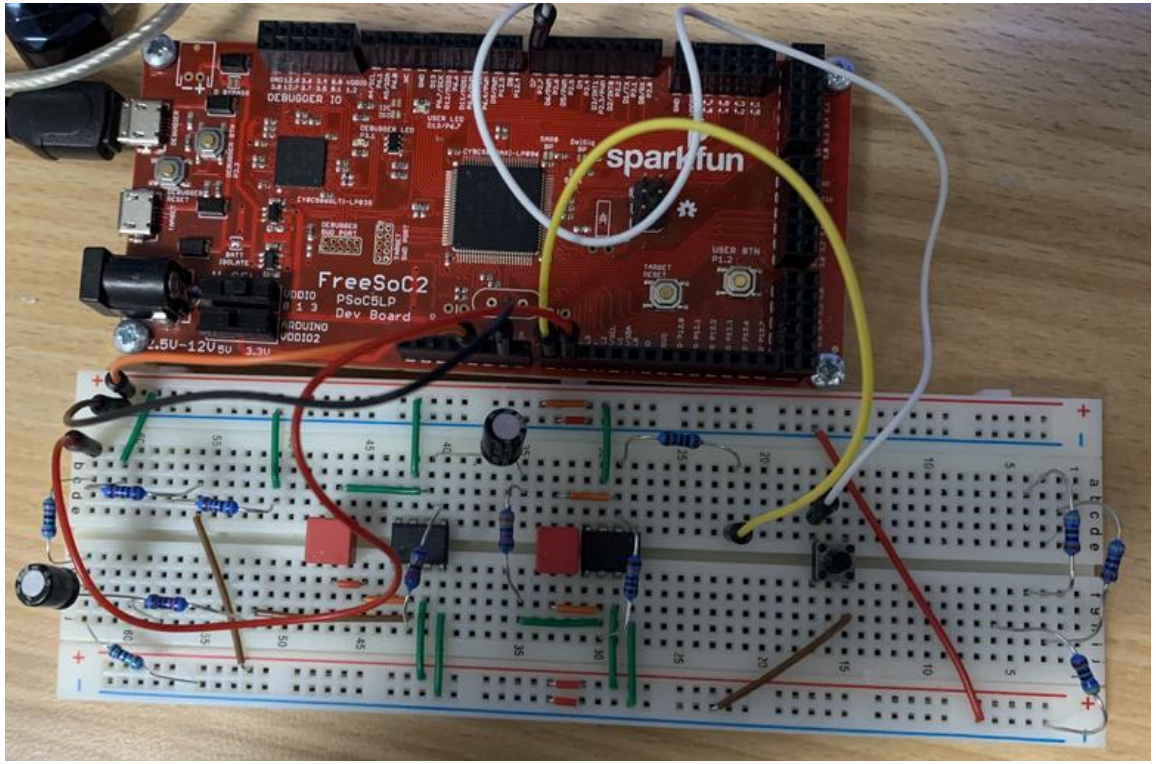
Table of content

- Hardware
- Top Level Design
- DMA Peripheral to Memory
- DMA Wizard
- State Machine
- UART Transfer
- Result

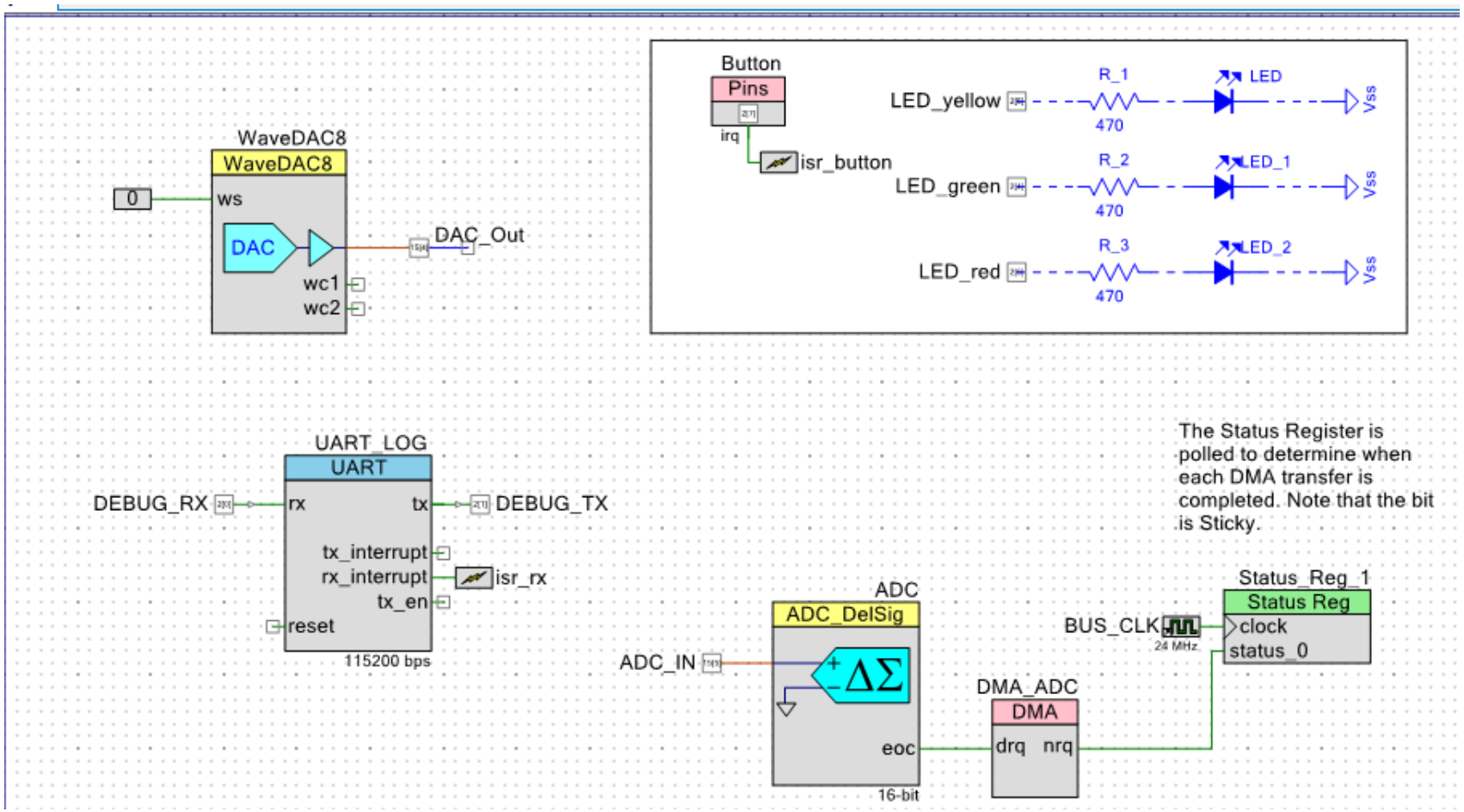


Hardware


TP_GND2
D4
TP_Offset
5V Sig GND
R6
R5
R10
R11
R8
TP_Sig1
J3
D2
R1
U1
R3
U2
R4
R7
C1
TP_GND1
R2
TP_1st
TP_PWR
C4
TP_2nd
TP_GND3
24Ghz Sensor Shield
Sose24-SDHD_Lab-R1.0
1127956 - 1127985


sparkfun
FreeSoC2
PSoC5LP
Dev Board

Top Level Design



DMA Peripheral to Memory

Task: transfer 2048 bytes(1024 x 16 bit data) from ADC and stored into memory location

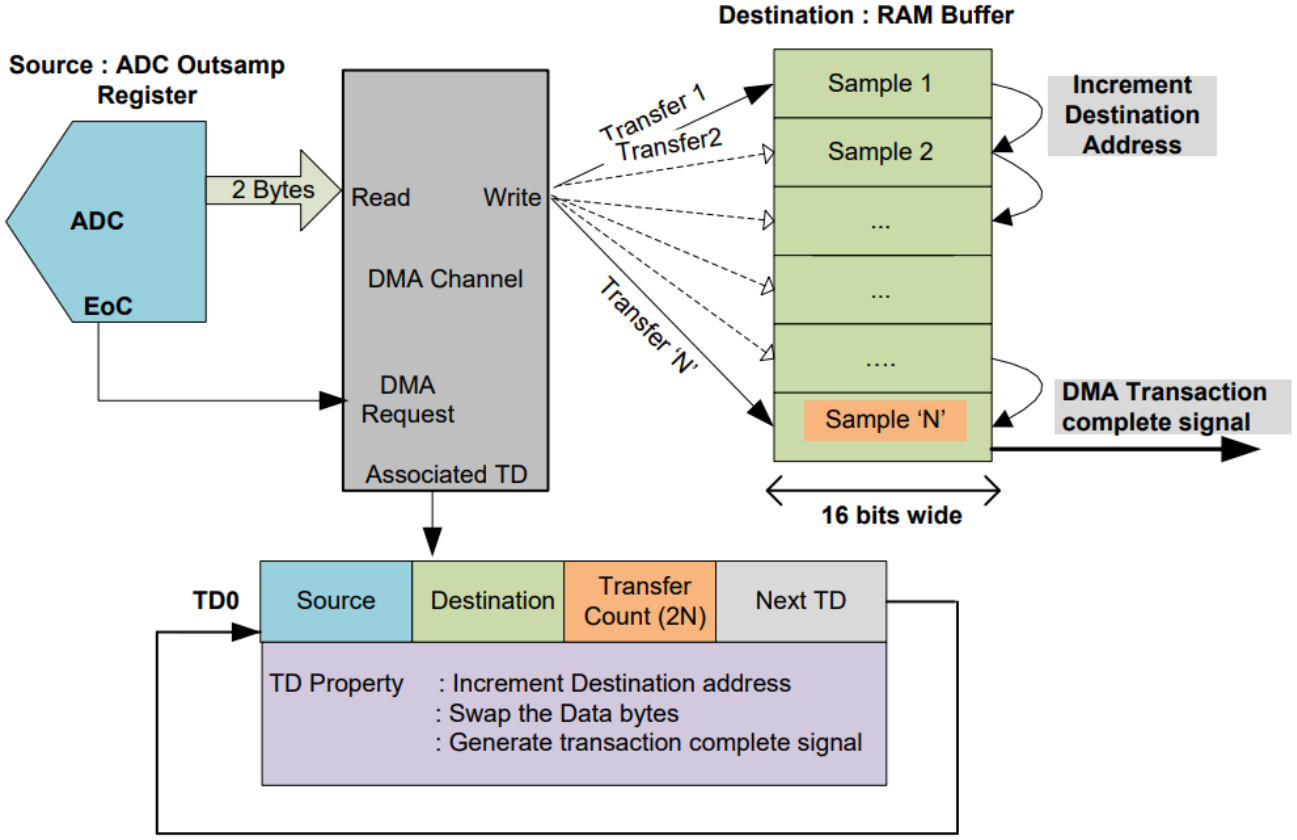
After each conversion, EOC generates interrupt that trigger DMA.

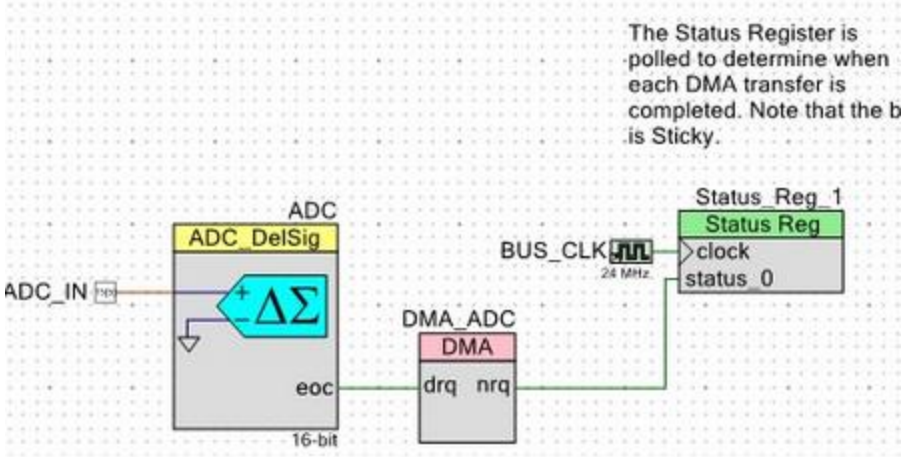
After storing first sample 2 bytes(16 bits). DMA increases the destination address to next sample and wait for EOC signal to start again.

When 2048 Bytes are received and written to memory, DMA generates interrupt and write to status register

DMA configuration is done by using DMA wizard

==> Reduce processor load , when polling is not performed or receiving interrupt





DMA Wizard

DMA Wizard

Generated Code

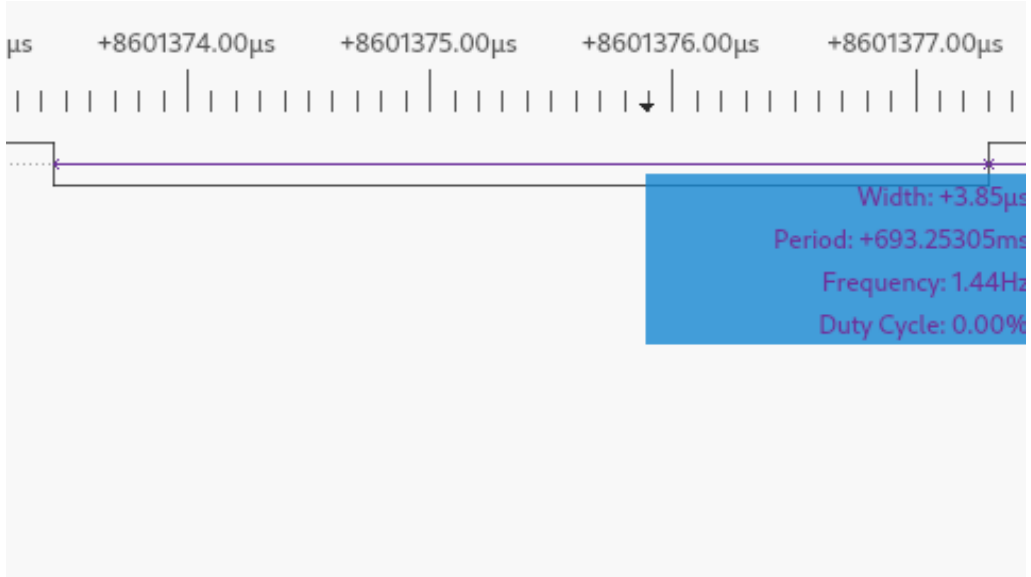
Copy and paste the code below into your design.

Invalid data was used to generate the code. Use the Back button to fix the errors.

```
/* Defines for DMA_ADC */
#define DMA_ADC_BYTES_PER_BURST 2
#define DMA_ADC_REQUEST_PER_BURST 1
#define DMA_ADC_SRC_BASE (CYDEV_PERIPH_BASE)
#define DMA_ADC_DST_BASE (CYDEV_SRAM_BASE)

/* Variable declarations for DMA_ADC */
/* Move these variable declarations to the top of the function
uint8 DMA_ADC_Chan;
uint8 DMA_ADC_TD[1];

/* DMA Configuration for DMA_ADC */
```

Copy to Clipboard | < Back | Finish | Cancel

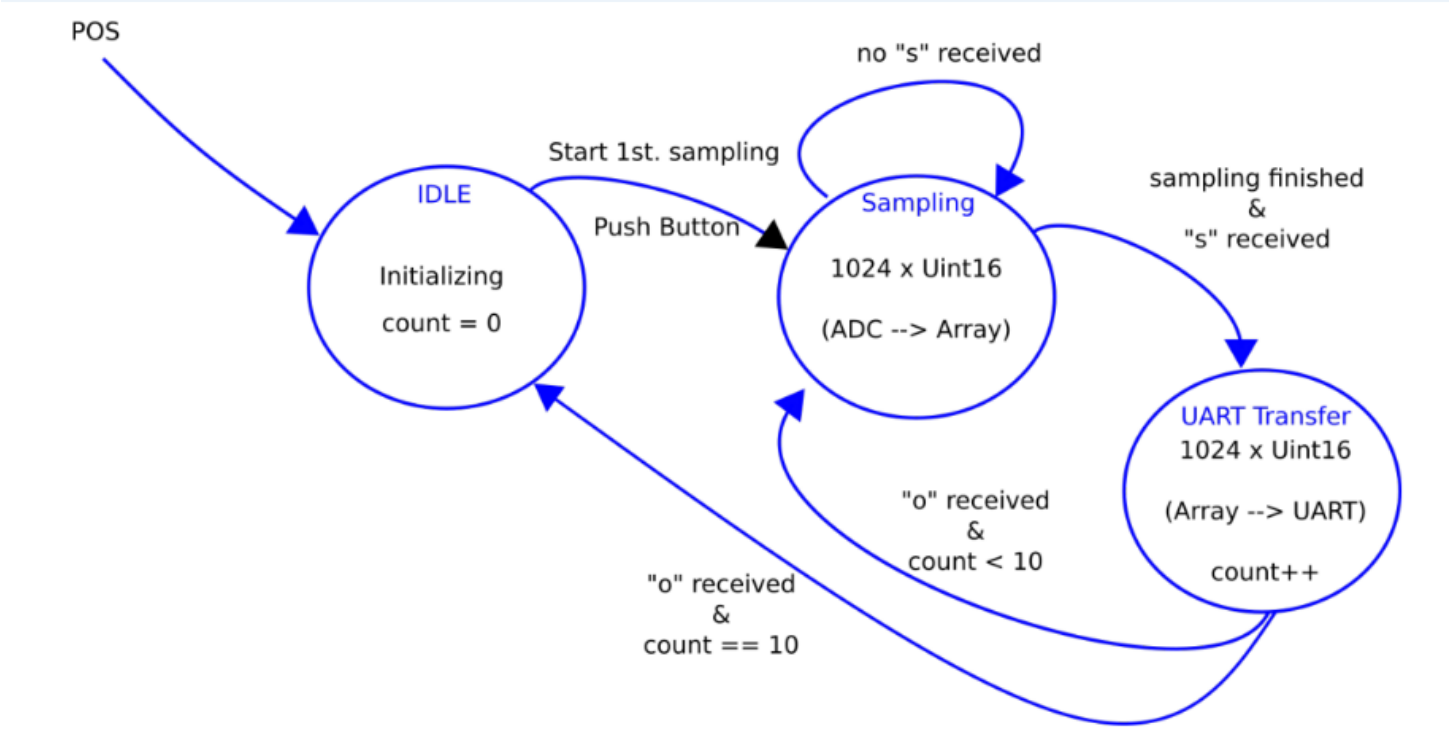
DMA Speed



DMA Transfer

ADC Polling End of Conversion

- State machine design



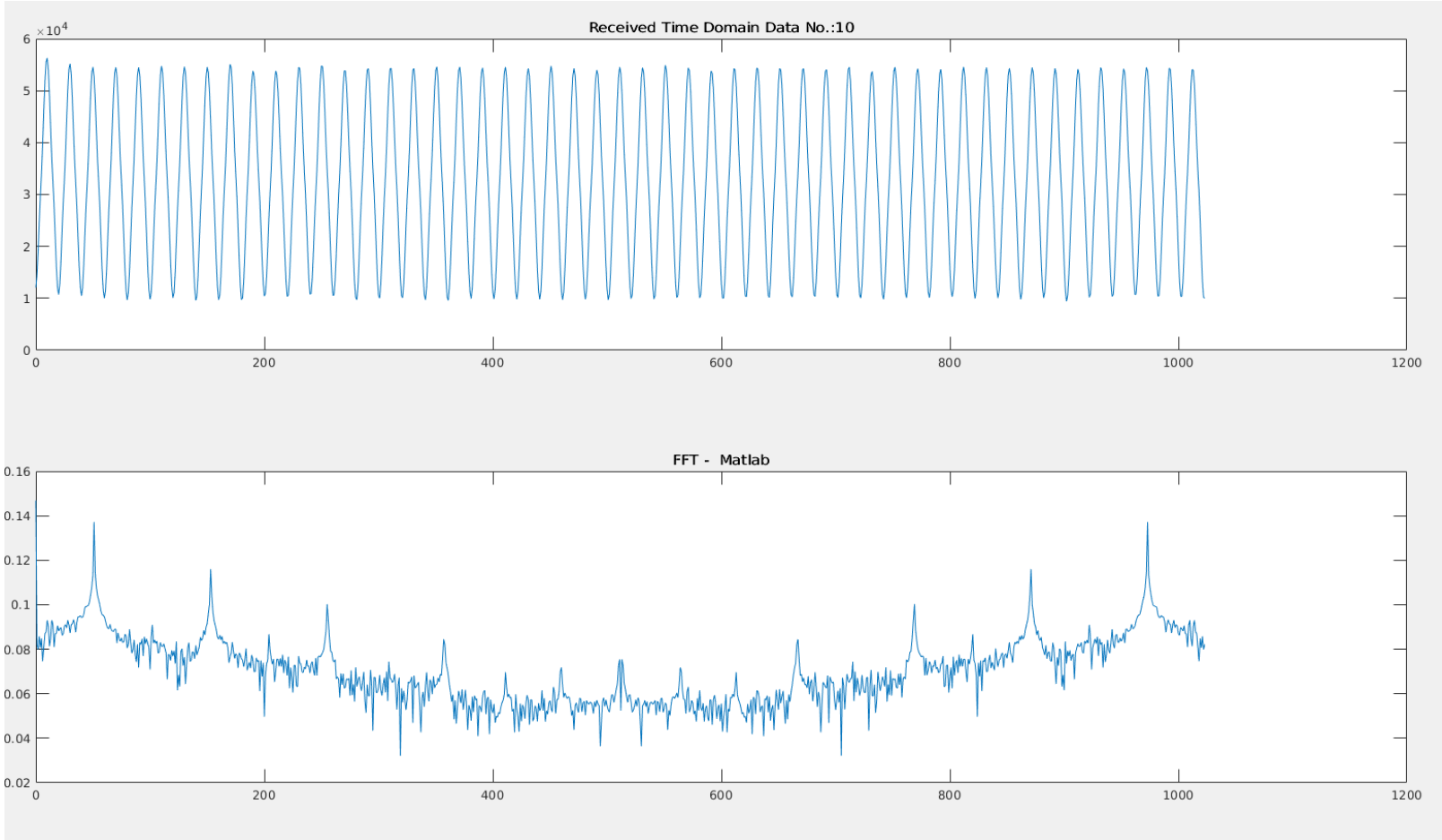
Shield data

```c
#define NUMBER_OF_TRANSFER 10
#define NUMBER_OF_SAMPLE 1024
typedef enum{
    IDLE,
    SAMPLING,
    UART_TRANSFER
}state_t;

typedef enum{
    NONE,
    RX_SAMPLING,
    RX_DONE,
}uartStatus_t;


typedef struct
{
    uint8_t count;
    int16_t sensor_data[NUMBER_OF_SAMPLE];
    state_t currentState;
    uartStatus_t uartStatus;
}shield_data_t;
```

UART Transfer to Matlab

```
for(int i = 0; i < NUMBER_OF_SAMPLE; i++){
    // Transfer lowest 8 bit
    UART_LOG_PutChar(shieldData.sensor_data[i] & 0xFF);
    // Shift to the right 8 bit, and write the rest to UART
    UART_LOG_PutChar(shieldData.sensor_data[i]>>8);
}
```

Result


Received Time Domain Data No.:10
×10^4
6
5
4
3
2
1
0
0
200
400
600
800
1000
1200
FFT - Matlab
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
200
400
600
800
1000
1200