List of API functions

```c
void control_CE(unsigned char input)
    //Enable or Disable chip enable signal.
    //Input = 1 → enables the signal, input = 0 → disables the signal
    //Users don’t have to worry about this function.

void control_CSN(unsigned char input)
    //Enable or Disable CSN signal. CSN controls the start and end of the communication between the MCU and the RF module
    //Input = 1 → enables the signal, input = 0 → disables the signal
    //Users don’t have to worry about this function.

void power_down(void)
    //You have to power down the RF module whenever it switches mode(PTX or PRX)
    //Users don’t have to worry about this function.

void power_up(void)
    //You have to power up first before you set the RF module in PTX or PRX mode
    //You cannot set the device in PWR_UP and PRX or PTX mode at once by writing to CONFIG register only one time.
    //Users don’t have to worry about this function.

unsigned char read_RF_register(unsigned char addr)
    //reads one byte value from any register that is one byte.
    //Don’t use this unless user is fully aware of the register structure of the RF module
    //addr = any register address in the RF chip
    //Reads and returns a value of register(1 byte) from the RF chip
    //Users don’t have to worry about this function. Maybe used for debugging purpose

void write_RF_register(unsigned char addr, unsigned char value)
    //reads one byte value from any register that is one byte.
    //Don’t use this unless user is fully aware of the register structure of the RF module
```
void clear_IRQs(void)

//Clears all the interrupt flags set in the RF module.

unsigned char TXmode(unsigned char input)

//Configures the RF module in PRX or PTX mode
//input = 0  ➔ PRX mode, 1  ➔ PTX mode
//For invalid input, returns 1 without issuing command, otherwise, returns 0 after correct operation

unsigned char setup_addr_width(unsigned char input)

//sets up Address Width of the data pipe
//input = 1  ➔ 3 byte, 2  ➔ 4 byte, 3  ➔ 5 byte
//Saves address width (in values 1, 2, 3 to represent 3, 4, 5 bytes respectively) in MCU variable for future use
For invalid input, returns 1 without issuing command, otherwise, returns 0 after correct operation

unsigned char setup_RF_frequency(unsigned char input)

//Sets up the base frequency for RF signal
//input = 0~125
//Fo = (2400 + input)[Mhz]
//Possible to set between 2.400GHz to 2.525 GHz
//For invalid input, returns 1 without issuing command, otherwise, returns 0 after correct operation

unsigned char setup_RF_data_rate(unsigned char input)

//Sets up data rate
//input = 0  ➔ 250kbps, 1  ➔ 1Mbps, 2  ➔ 2Mbps
//For invalid input, returns 1 without issuing command, otherwise, returns 0 after correct operation
unsigned char setup_auto_retr(unsigned char delay, unsigned char count)

//Sets up the Auto-Retransmission feature in case of missing NOACK
//delay = (0 ~ 15) 0 ==> 250uS delay, 15 ==> 4ms delay
// count = (0~15) 0 ==> no retransmission, 15==> IRQ generated after failing 15 retransmissions
//For invalid input, returns 1 without issuing command, otherwise, returns 0 after correct operation

void read_status(void)

//Issues NOP command just to read status
//just like all the functions, saves the value of status register in the RF module in MCU variable status

void set_RX_addr_PX(unsigned char MSByte, unsigned long LSBytes){

//PX_corresponds to pipe number(P0~P5)
//sets up 5 byte(maximum) RX pipe address
//Only lower bytes will be used if user sets Address width that is lower than 5 bytes
//Address is arranged as {MSByte[7..0], LSbytes[31..0]}

void set_TX_addr(unsigned char MSByte, unsigned long LSBytes){

//sets up 5 byte(maximum) TX pipe address
//Only lower bytes will be used if user sets Address width that is lower than 5 bytes
//For one to one transmission, TX address should be equal to RX address

unsigned char set_payload_length(unsigned char input){

//Sets the the number of Bytes in RX payload
//input = 0~32; 0==> datapipe not used, 1~32 ==>payload length
//for now, we only use datapipe0
//For invalid input, returns 1 without issuing command, otherwise, returns 0 after correct operation
unsigned char read_RX_payload(void) {
    // reads RX payload and saves in RX_payload array
    // RX_payload_array[0] stores the first data that arrived in RX_FIFO
    // This function is to be executed when the RX receive IRQ is asserted
}

unsigned char write_TX_payload(noack) {
    // no_ack = 0 ==> the module do not wait for ACK signal
    // no_ack = 1 ==> the module waits for ACK signal and generates
    // Sends data from TX_payload_array to TX FIFO
    // TX_payload_array goes into TX FIFO first and therefore is sent on air first.
}

void flush_TX_FIFO(void) {
    // Flushes TX_FIFO
    // Flush TX_FIFO is full and the user wants to send the new data as soon as possible.
}

void flush_RX_FIFO(void) {
    // Flushes RX_FIFO
    // When the read RX_payload is greater than 32 bytes, the user must execute this command as received packet is not valid.
}

unsigned char read_RX_payload_width(void) {
    // return RX payload width when using DPL feature.
}

void enable_TX_NOACK(void) {
    // Enables TX_PAYLOAD_NOACK command
    // If TX_PAYLOAD_NOACK command is issued, the transmitter does not wait for ACK statement.
}

unsigned char check_MAX_RT(void) {
    // returns 1 if MAX RT interrupt flag inside the RF chip is set, returns 0 otherwise.
    // before returning, it clears all the interrupt sources.
}

unsigned char check_TX_DS(void)
//returns 1 if TX_DS interrupt flag inside the RF chip is set, returns 0 otherwise.
//before returning, it clears all the interrupt sources.

unsigned char check_RX_DR(void)

//returns 1 if RX_DR interrupt flag inside the RF chip is set, returns 0 otherwise.
//before returning, it clears all the interrupt sources.

• Functions Not yet debugged.

void enable_all_pipes(void)

//Enables transmissions from all the pipes.

void enable_pipe(unsigned char input)

//enables only one pipe

//input = 0~5 which specifies pipe 0~5.