API Programming Guide

SeongJi Industrial

December 1, 2019

Contents

Document revision history

Date	File name	Updated
2018.04.23	[WISOL]AppNote_LOM202A_API_20180423_kr.pdf	First release
2018.07.01	[WISOL]AppNote_LOM20XA_API_20180701_en.pdf	English version
2018.08.01	[WISOL]AppNote_LOM20XA_API_20180801_en.pdf	Add function
2018.09.01	[WISOL]AppNote_LOM20XA_API_20180901_en.pdf	Fixed example (3.24,3.25…)
2019.01.04	[WISOL]AppNote_LOM20XA_API_20190104.en.pdf	Add P2P API
2019.03.05	[WISOL]AppNote_LOM20XA_API_20190305_en.pdf	Revise the contents
2019.06.03	AppNote_LOM20XA_API_20190603_en.pdf	Wisol->SeongJi Industrial
2019.12.01	AppNote_LOM20XA_API_20191201_en.pdf	Fixed Channel mask

Application Model

Model
LOM202A02 (API)
LOM204A02 (API)

※ Library file and Keil compiler preference files are provided separately for each country.

※ API product : LOM202A02, LOM204A02
CLI product : LOM202A00, LOM202A01, LOM202A10, LOM202A20, LOM202A30, LOM202AZ0
LOM204A01

※ Do not use API software for CLI product..

1. Pin configuration

1) Module Pin Map & CPU Pin map

MCU to Pin map LOM202A / LOM204A

Pin No.	Pin name	MCU pin name	Module function	SWDLOM202_SRD0K_V200 API
1	NC			
2	PB7	PB7	GPIO IN	GPIO,LPTIM1 IN2, COMP2 INP,VREF PVD Boot Loader Enable(High Active at boot)
3	USART5_TX	PB3	UART5 TX	for Debug and control
4	USART5_RX	PB4	UART5 RX	for Debug and control
5	GND			
6	USART1_TX	PA9	UART1 TX	GPIO, UART1 TX, I2C1 SCL To download Firmware through UART (At boot olny)
7	USART1_RX	PA10	UART1 RX	GPIO, UART1 RX, I2C1 SDA To download Firmware through UART (At boot olny)
8	SWCLK	PA14	SWCLK	To download Firmware and debug through STLink
9	SWDIO	PA13	SWDIO	
10	GND			
11	SPI2_MOSI	PB15	GPIO OUT	GPIO or SPI2 MOSI, I2S2 SD If "LED3_SLEEP" is defined (GPIO OUT) Sleep state : Low, normal/wakeup :High
12	SPI2_SCK	PB13		GPIO or SPI2 SCK, I2S2 CK, I2C2 SCL,TIM21
13	SPI2_MISO	PB14		GPIO or SPI2 MISO,I2S2 MCK, RTC OUT I2C2 SDA,TIM21 CH2
14	SPI2_NSS	PB12	GPIO IN	GPIO or SPI2 NSS, I2S2 WS, I2C2 SMBA Payload data bit 3 (In case of Cycle On)
15	GND			
16	VREF+	VREF+		
17	PH0	OSC_IN		Not connection
18	VDD			
19	GND			
20	PB10	PB10	GPIO IN	GPIO,TIM2 CH2, LPUART1 TX/RX, SPI2 SCK, I2C2 SCL

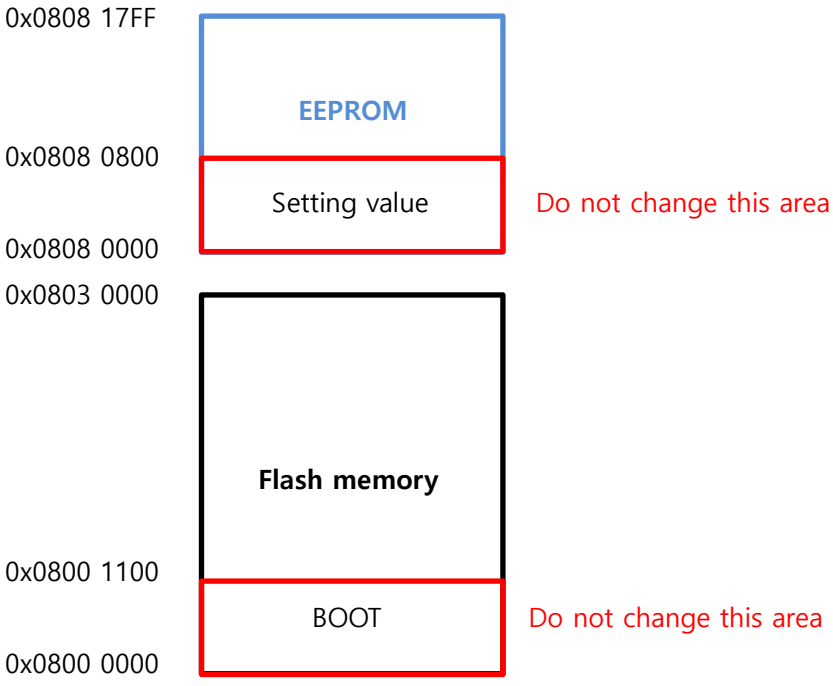
				Payload data bit 0 (In case of Cycle On)
21	PB11	PB11	GPIO IN	GPIO,EVENTOUT, TIM2 CH4, LPUART1 RX./TX, I2C2 SDA Payload data bit 1 (In case of Cycle On)
22	PA2	PA2	GPIO OUT	GPIO,TIM21 CH1, TIM2 CH3, USART2 TX, LPUART1 TX, COMP2 OUT/INM,ADC IN2 If "RS485_USE" is defined RS485 Control Pin
23	PA3	PA3	ADC IN	GPIO,TIM21 CH2,TIM2 CH4,USART2 TX,LPUART1 RX,COMP2 INP,ADC IN3 Payload data bit 4 ~ 15 (In case of Cycle On)
24	GND			
25	GND			
26	NRST	NRST	RST	Reset
27	VDD_RF			
28	GND			
29	PC2	PC2	ADC IN	Battery Level 12 bit ADC
30	WKUP1	PA0_WKUP1	INT IN	Wake up: Risiging Edge,(To wakeup CPU) Payload data bit 2 (In case of Cycle On)
31	ADC_IN1	PA1		GPIO,TIM2 CH2,USART2 RTS_DE, TIM21 ETR, USART4 RX,COMP1 INP,ADC IN1
32	NC			
33	PC0	PC0		GPIO,LPTIM1 IN1,EVENTOUT,LPUART1 RX I2C3 SCL, ADC IN10
34	I2C1_SCL	PB8		GPIO, I2C1 SCL
35	I2C1_SDA	PB9		GPIO, I2C1 SDA, EVENTOUT, SPI2 NSS, I2S2 WS
36	WKUP2	PC13		GPIO,RTC OUT, WKUP2
37	BOOT0	BOOT0		Open
38	GND			
39	GND			
40	NC			
41	GND			
42	NC			

43	GND			
44	NC			
45	GND			
46	GND			
47	RF			
48	GND			

User-configurable GPIO Pin

2) The following pins are used inside the module (User can not use)
   - MCU Pin name : PA4,PA5,PA6,PA7,PA11,PA12,PA15,NRST,PB0,PB1,PB2,PB5,PB6,PC14,PC15
   - PC1 : This pin is opened, It is not connected to the module output pin

2. Memory map



※ The LOM20xA02 is an API version and does not have read/write protection set in memory so that user can program it.

To prevent duplication and hacking of the product, after downloading the user program, you can secure the system by setting protection in the option bytes setting menu of the STM32 ST-Link utility program.

3. WAN API discription

3.1. CLI mode

Discription	Full functionality with LoRaWAN protocol When used as a modem to control and debug with UART This function is executed infinitely. Do not use with other modes(P2P_MODE(), or API_MODE())
Function	CLI_MODE()
Return value	Void

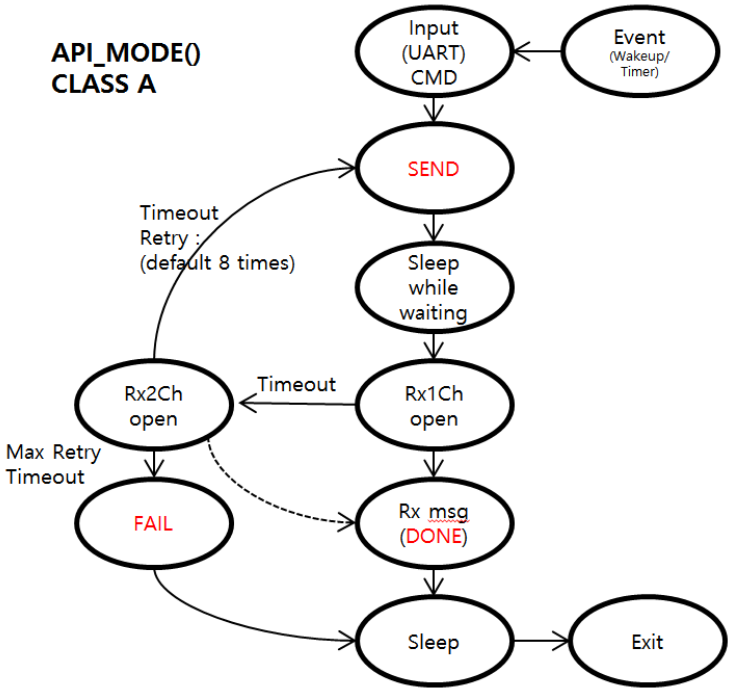
Example	int main(void)
	{
	Start_Init(); // variable IO, UART, ADC etc setting
	CLI_MODE();
	}

3.2. API mode

Discription	Only the unit functions for transmit and receive are executed LoRaWAN protocol Do not use with other modes(P2P_MODE(), or CLI_MODE())
Function	API_MODE()
Return value	DONE: 1, FAIL : 2 (uint8)

Example	int main(void)
	{
	Start_Init(); // IO, UART, ADC etc setting
	setClass(0); // CLASS A setting
	JOIN_START(); // Start Join process
	// This function calls not only OTAA but also ABP
	while(1)
	{
	CLI_Command_Process(); // Set to receive UART input
	API_MODE(); // Run Tx/Rx according to event
	Enable_enter_stop_mode (); // Enable to enter Stop mode
	Device_State_Sleep_Fn(); // Enter Stop mode
	}
	}

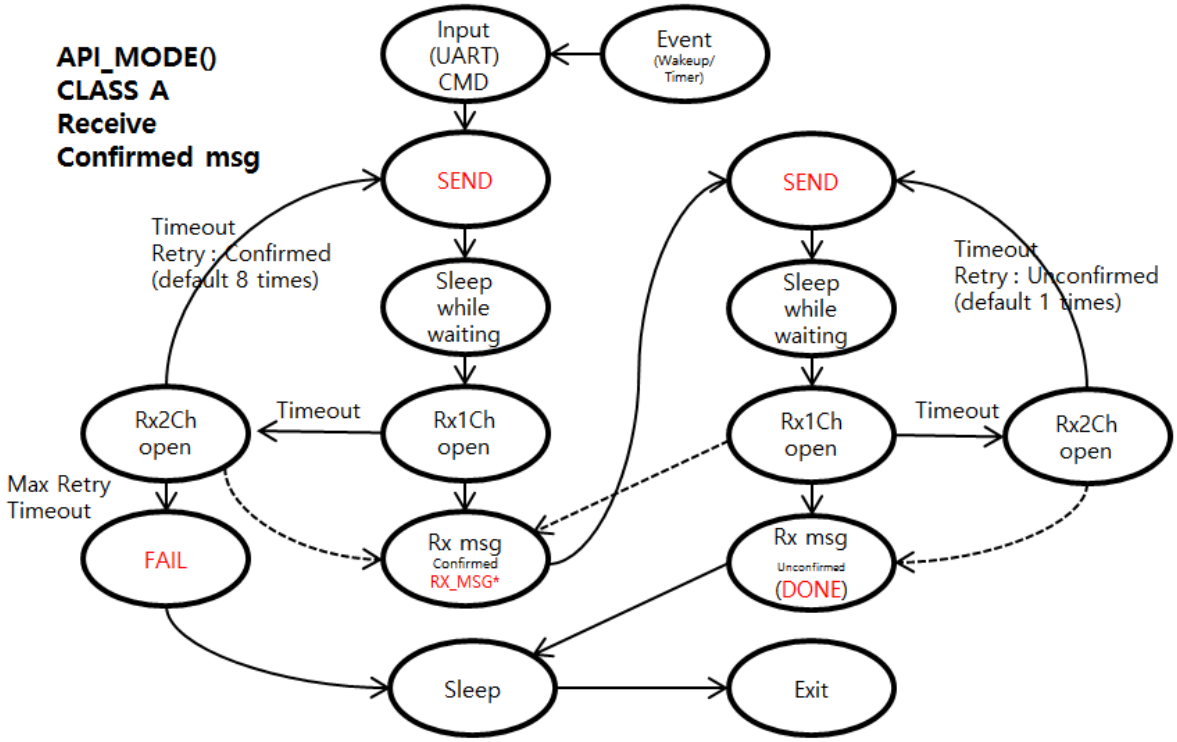
Function State Diagram (CLASS A)



※ Sleep has Stop mode and Low power run mode.

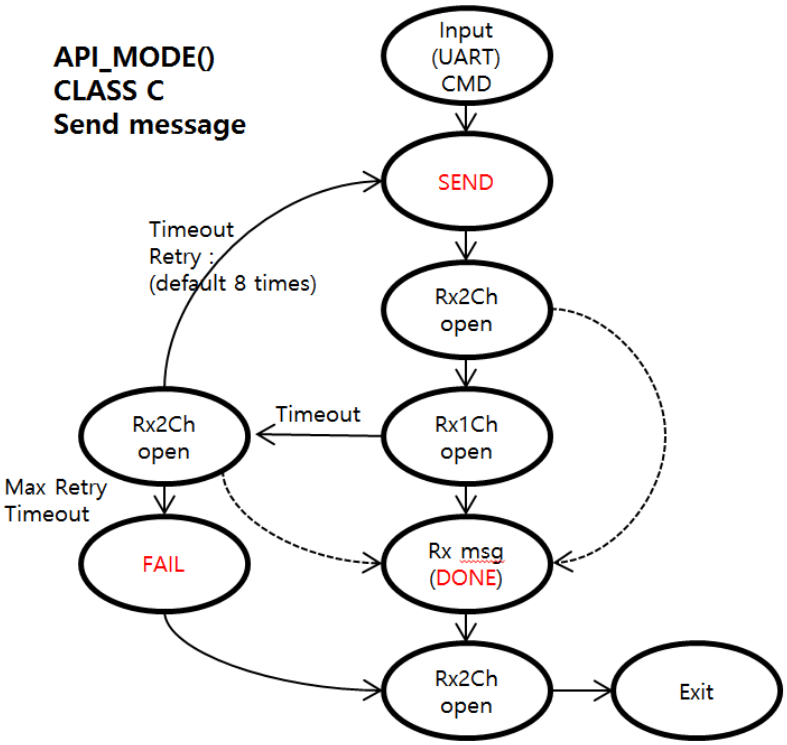
※ CLI_MODE() enter Stop mode in CLASS A mode.

※ The CPU wakes up after the event and then If there is no input CMD, it is in sleep state.

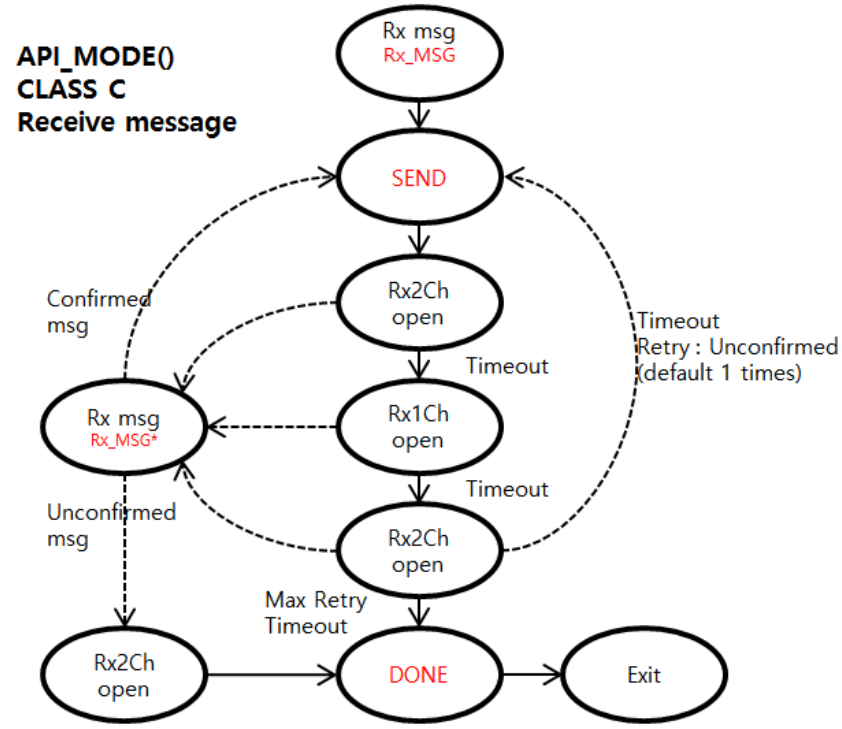


Rx_MSG* : If Rx msg is a MAC command , RX_MSG is not displayed
If Rx msg is a generic message , RX_MSG is displayed.

Function State Diagram (CLASS C, Send message)



Function State Diagram (CLASS C , Receive message)



Rx_MSG* : If Rx msg is ACK, RX_MSG is not displayed
If Rx msg is a MAC command , RX_MSG is not displayed
If Rx msg is a generic message , RX_MSG is displayed.

3.3. Start Init

Discription	System Clock ,NV, Peripheral , UART, SPI, ADC, RTC, Timer, IO setting
Function	Start_Init()
Return value	void
Example	Start_Init(); // First use when running main function

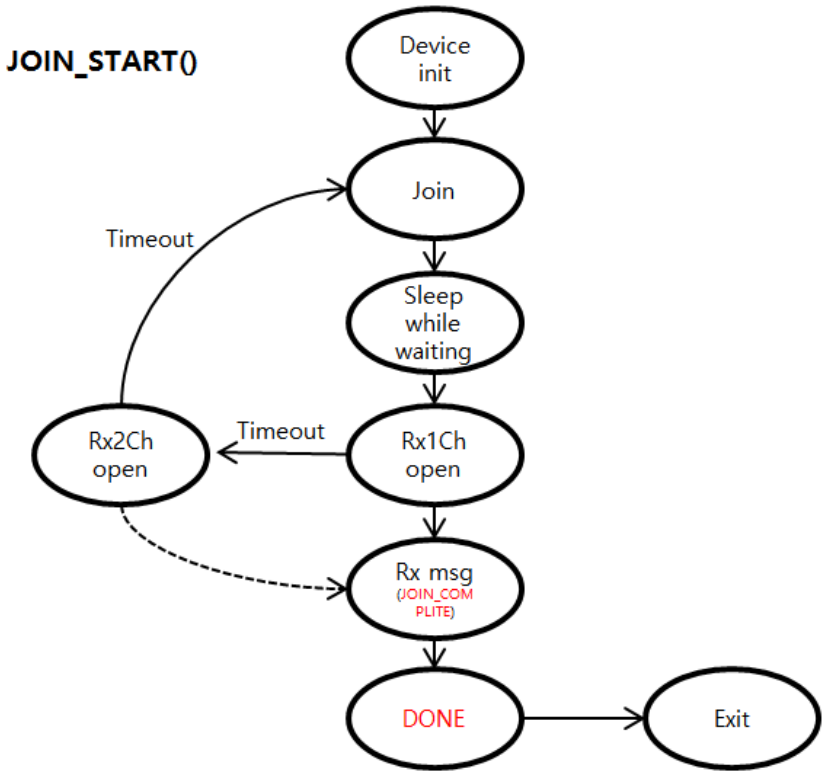
3.4. SetClass(CLASS)

Discription	Class A or B setting
Function	setClass(uint8_t LoRaClas)
parameter	Class A : 0, Class C : 2
Return value	Success : 0, Fail : 1
Example	setClass(0); // Class A

3.5. JOIN_START()

Discription	Join start, Run only once at the start of LoRaWAN This function must be run at start time in ABP mode As well as OTAA mode.
Function	JOIN_START()
Return value	void
Example	JOIN_START();

Function State Diagram



3.6. Enable Stop Mode

Discription	In Class A mode, it is possible to enter into Stop mode after sending / receiving message
Function	Enable_enter_stop_mode()
Return value	void
Example	Enable_enter_stop_mode(); // This is used before Device_State_Sleep_Fn();

3.7. Enable Low Power Run Mode

Discription	In Class A mode, it is possible to enter into Low power run mode after sending / receiving message ※ Not abailable in CLI mode
Function	Enable_enter_lprun_mode();
Return value	void
Example	Enable_enter_lprun_mode(); // This is used before Device_State_Sleep_Fn();

3.8. Enter Sleep mode

Discription	The module enters the stop mode or low power run mode in CLASS A mode. It have to be enabled by selecting the entry mode with Enable_enter_lprun_mode() / Enable_enter_stop_mode() before executing this API function.
Function	Device_State_Sleep_Fn ()
Return value	void
Example	Device_State_Sleep_Fn ();

3.9. Timer start(system tick 1ms)

Discription	Start 1ms system tick timer It is not executed when entering Stop mode.. API_MODE() or CLI_MODE() can be called from within the function to Stop mode enter. Refer to state diagram.		
Function	Timer_1ms_Start(unsigned short id, unsigned short interval, unsigned char reload, void (*Func)(void))		
Return value	void		
Parameter	Parameter	Value(range)	Description
	Id	TRX_LED USER1 USER2 USER3	Timer ID
	Interval	2 ~ 65535	Time out setting(Unit: ms)
	Reload	1 ~ 255	Retry number (255: Infinite repeat)
	Func		Function to execute
Example	Timer_1ms_Start(TRX_LED,300,6,LED3_BLINK); // Run LED3_BLINK function 6 times every 300ms using TRX_LED timer		

3.10. Timer stop(system tick 1ms)

Discription	1ms system tick timer stop
Function	Timer_1ms_Stop(unsigned short id)
Parameter	Id : TRX_LED, USER1, USER2, USER3 (User-available ID)
Return value	void
Example	Timer_1ms_Stop(TRX_LED); // Stop TRX_LED timer execution

3.11. Join Request

Discription	Join Request start In API_MODE(), change state to start from Join.
Function	JoinRequest_Tx(void)
Return value	SUCCESS : 0, BUSY : 2 (uint8)
Example	JoinRequest_Tx(); API_MODE();

3.12. data Tx

Discription	Transmit data setting. Actual transmission is executed by executing API_MODE();
Function	data_Tx(uint8_t dataForm, uint8_t dataType, uint8_t fPort, uint8_t dataLen, uint8_t * data)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2 (uint8)
Parameter	See table below

Example	data_Tx(1,1,1,10,"1234567890");
	// Transmits confirmed data in ASCII format to 10 bytes of data on
	// port 1
	API_MODE();

Parameter	Value(range)	Description
dataForm	0	Binary Data
	1	ASCII Data
dtatType	0	Unconfirmed
	1	Confirmed
fPort	1~0xDD	Frame port
dataLen	SF7 : 0~0xF2 SF8 : 0~0xF2 SF9 : 0~0xF2 SF10 : 0~0xF2 SF11 : 0~0x97 SF12 : 0~0x41	Payload size ※ See the LoRaWAN Regional Parameters document - Max payload size varies by country.
data	Ascii/Binary	Payload data

3.13. Link Check Request

Discription	Set to send a Link Check Request. Actual transmission is executed by executing API_MODE().
Function	linkCheckRequest_Tx(void)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2 (uint8)
Example	linkCheckRequest_Tx (); API_MODE();

3.14. Device Time Request

Discription	Set to send a Device Time Request. Actual transmission is executed by executing API_MODE().
Function	deviceTimeRequest_Tx(void)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2 (uint8)
Example	deviceTimeRequest_Tx(); API_MODE();

3.15. Get Device EUI

Discription	Get Device EUI.
Function	getDeviceEUI(void)
Return value	Device EUI (uint8*)

Example	uint8_t* res;
	res = getDeviceEUI();
	for(i=0; i<8; i++)
	PRINTF("%02x",res[i]);

3.16. Set Otaa Application EUI

Discription	Set the Application EUI in Otaa mode
Function	setOtaaAppEUI(uint8_t * AppEUI, uint8_t count)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	AppEUI : Application EUI (8byte hexa) Count : 16
Example	uint8_t *str1=" 900000100000010A"; setOtaaAppEUI(str1, 16);

3.17. Get Otaa Application EUI

Discription	Get the Application EUI in Otaa.
Function	getOtaaAppEUI(void)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2 (uint8)
Example	getOtaaAppEUI();

3.18. Set Otaa (Pseudo) Application KEY

Discription	Set the (Pseudo) Application KEY in Otaa mode
Function	setOtaaPseudoAppKey(uint8_t * PseudoAppKey, uint8_t count)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	PseudoAppKey : (Pseudo) Application KEY (16byte hexa) Count : 32
Example	uint8_t* PAppKey=" 8613561b702547C47151298039b0cb6e"; setOtaaPseudoAppKey (PAppKey, 32);

3.19. Get Otaa Pseudo Application KEY

Discription	Get the Pseudo Application KEY in Otaa mode For KR(SKT) only
Function	getOtaaPseudoAppKey(void)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2 (uint8)
Example	getOtaaPseudoAppKey ();

3.20. Set ABP Network ID

Discription	Set Network ID in ABP mode
Function	setAbpNetworkID(uint32_t abpNetID)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	abpNetID: Network ID (3bytes)
Example	setAbpNetworkID (0x123456);

3.21. Get ABP Network ID

Discription	Get Network ID in ABP mode
Function	getAbpNetworkID (void)
Return value	Network ID value (uint32)
Example	uint32_t NetID;

	NetID = getAbpNetworkID ();

3.22. Set ABP Device Address

Discription	Set Device Address in ABP mode
Function	setAbpDeviceAddress(uint32_t abpDevAddr)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	abpDevAddr: Device Address (4bytes)
Example	setAbpDeviceAddress (0x12345678);

3.23. Get ABP Device Address

Discription	Get Device Address in ABP mode
Function	getAbpDeviceAddress (void)
Return value	uint32_t
Example	uint32_t DevAdd; DevAdd = getAbpDeviceAddress();

3.24. Set ABP Network Session Key

Discription	Set Network Session Key in ABP mode
Function	setAbpNwkSKey(uint8_t * abpNwkSKey, uint8_t count)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	abpNwkSKey: Network Session Key (16bytes ASCII) Count : 32
Example	uint8_t *NSKey = "00000000000000000000000000000007"; setAbpNwkSKey (NSKey,32);

3.25. Set ABP Application Session Key

Discription	Get Application Session Key in ABP mode
Function	setAbpAppSKey(uint8_t * abpAppSKey, uint8_t count)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	abpAppSKey: Application Session Key (16bytes ASCII) Count : 32
Example	uint8_t *ASKey = "00000000000000000000000000000008"; setAbpAppSKey (ASKey,32);

3.26. Get Join Complite Status

Discription	Check whether the join is complete
Function	getJoinDoneStatus(void)
Return value	Join is complited: 0, Join is not complited : 1, BUSY : 2
Example	getJoinDoneStatus();

3.27. Get Firmware Version

Discription	Check the Firmware version
Function	getFirmwareVersion(void)
Return value	Join is complited: 0, Join is not complited : 1, BUSY : 2 (uint8)
Example	getFirmwareVersion();

3.28. Set Activation

Discription	Set Activation mode (OTAA or ABP)
Function	setActivation(uint8_t Activation)
Return value	Join is complited: 0, Join is not complited : 1, BUSY : 2
Parameter	Activation: Activation mode OTAA : 0 , ABP : 1
Example	setActivation (0); // OTAA

3.29. Get Activation

Discription	Get Activation mode
Function	getActivation(void)
Return value	OTAA: 0, ABP: 1, BUSY : 2 (uint8)
Example	getActivation ();

3.30. Set Class

Discription	Set Class A or C.
Function	setClass(uint8_t LoRaClass)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	Class A : 0, Class C : 2
Example	setClass (0); // Class A setting

3.31. Get Class

Discription	Check Class A or C.
Function	getClass (void)
Return value	Class A : 0, Class C : 2 (uint8)
Example	getClass ();

3.32. Set Adr enable

Discription	Set enable ADR.
Function	setAdrEnable(uint8_t Enable);
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	off : 0, on : 1
Example	setAdrEnable (0); // ADR off setting

3.33. Get Adr enable

Discription	Check ADR
Function	getAdrEnable (void)
Return value	off : 0, on : 1 (bool)
Example	getAdrEnable ();

3.34. Set Repeater Support

Discription	Set Repeater support
Function	setRepeaterSupportEnable(uint8_t Enable)
Return value	off : 0, on : 1
Parameter	Support off : 0, on : 1
Example	setRepeaterSupportEnable (0); // Repeater off

3.35. Get Repeater Support

Discription	Get Repeater support
Function	getRepeaterSupportEnable (void)
Return value	off : 0, on : 1 (uint8)
Example	getRepeaterSupportEnable ();

3.36. Set GMT

Discription	Set GMT.
Function	setGMT(int8_t Gmt)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	off : 0, on : 1
Example	setGMT (9); // Korea : set 9 hours

3.37. Get GMT

Discription	Check GMT
Function	getGMT (void)
Return value	GMT value (int32)
Example	getGMT();

3.38. Set confirmed ReTx count

Discription	Set the ReTx count value for the confirmed message.
Function	setReTxCntConfirmed(uint8_t ReTxCntCon)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	1~8 (Number of retransmissions) (uint8)
Example	setReTxCntConfirmed (8); // default 8

3.39. Get confirmed ReTx count

Discription	Check the ReTx count value for the confirmed message
Function	getReTxCntConfirmed (void)
Return value	Number of retransmissions (uint8)
Example	getReTxCntConfirmed ();

3.40. Set unconfirmed ReTx count

Discription	Set the ReTx count value for the unconfirmed message.
Function	setReTxCntUnconfirmed(uint8_t ReTxCntUncon)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	1~8 (Number of retransmissions)
Example	setReTxCntUnconfirmed (1); // default 1

3.41. Get unconfirmed ReTx count

Discription	Check the ReTx count value for the unconfirmed message
Function	getReTxCntUnconfirmed (void)
Return value	Number of retransmissions (uint8)
Example	getReTxCntUnconfirmed();

3.42. Set Debug Message

Discription	Set display level for the Debug message.
Function	setDebugMessage(uint8_t Enable)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	0 : off (compact message) , 1: on (full message), 2: none(minimize)
Example	setDebugMessage (0); // default 0

3.43. Get Debug Message

Discription	Get display level for the Debug message.
Function	getDebugMessage (void)
Return value	0 : off (compact message) , 1: on (full message), 2: none(minimize)
Example	getDebugMessage ();

3.44. Set AntennaGain

Discription	Set Antenna Gain value.
Function	setAntennaGain(uint8_t AntGain)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	0 ~ 5 (dBi)
Example	setAntennaGain (1); // default 0

3.45. Get AntennaGain

Discription	Get Antenna Gain value.
Function	getAntennaGain (void)
Return value	antenna gain value (uint8)
Example	getAntennaGain ();

3.46. Set Uart Baudrate

Discription	Set Uart Baudrate.
Function	setUartBaudRate(uint32_t BaudRate)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	19200, 38400,57600, 115200
Example	setUartBaudRate (115200); // default 115200 bps

3.47. Get Uart Baudrate

Discription	Get Uart Baudrate.
Function	getUartBaudRate (void)
Return value	Baudrate value (uint8)
Example	getUartBaudRate ();

3.48. Set Join Accept Delay1

Discription	Set Join Accept Delay1
Function	setJoinAcceptDelay1(uint32_t JoinAcceptDelay_1)
Return value	SUCCESS : 0, BUSY : 2
Parameter	JoinAcceptDelay_1 : 100 ~ 6000 (ms)
Example	setJoinAcceptDelay1(5000); // default 5000ms

3.49. Get Join Accept Delay1

Discription	Get Join Accept Delay1.
Function	getJoinAcceptDelay1(void)
Return value	Join Accept Dealy1 value (uint32)
Example	getJoinAcceptDelay1();

3.50. Set Rx Delay1

Discription	Set Rx Delay1
Function	setRxDelay1(uint32_t RxDelay1)
Return value	SUCCESS : 0, BUSY : 2
Parameter	RxDelay1: 100 ~ 6000 (ms)
Example	setRxDelay1 (1000); // default 1000ms

3.51. Get Rx Delay1

Discription	Get Rx Dealy1.
Function	getRxDelay1(void)
Return value	Rx Dealy1 value (uint32)
Example	getRxDelay1();

3.52. Set Channel Tx power For SKT(KR)

Discription	Set Channel Tx power for SKT
Function	setChTxPowerForSktKR(uint8_t ChId, int8_t ChTxPower)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	Chid : 1~8 ChTxPower : 0~14
Example	setChTxPowerForSktKR (1, 13);

3.53. Get Channel Tx power For SKT(KR)

Discription	Get Channel Tx power for SKT
Function	getChTxPowerForSktKR (void)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Example	getChTxPowerForSktKR ();

3.54. Set Channel Tx power

Discription	Set Channel Tx power
Function	setChTxPower(int8_t ChTxPower)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	ChTxPower : 0~14 (MAX ~ MIN , power index value) ※ See the LoRaWAN Regional Parameters document
Example	setChTxPower();

3.55. Get Channel Tx power

Discription	Get Channel Tx power.
Function	getChTxPower(void)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2 // for SKT (KR)

	Channel Tx power ID value (0~ 14) // All except SKT Error : 100 // All except SKT
Example	getChTxPower();

3.56. Set Data Rate

Discription	Set TX Data Rate
Function	setTxDataRate(int8_t TxDataRate)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	TxDataRate: 0~5 (MIN ~ MAX , datarate index value) ※ See the LoRaWAN Regional Parameters document

Example	setTxDataRate(0); // default 0 for EU,US,KR,SKT,AS
	// default 2 for AU

3.57. Get Data Rate

Discription	Get TX Data Rate.
Function	getTxDataRate(void)
Return value	Tx datarate ID value (0~ 5) Error : 100
Example	getTxDataRate ();

3.58. Get Last RSSI

Discription	Get the RSSI value received recently.
Function	getLastRssi(void)
Return value	Rssi value (int16)
Example	getLastRssi ();

3.59. Get Last SNR

Discription	Get the SNR value received recently.
Function	getLastSnr (void)
Return value	SNR value (int8)
Example	getLastSnr ();

3.60. execute system reset

Discription	Execute reset .
Function	systemReset(void)
Return value	none
Example	systemReset ();

3.61. Set sleep mode

Discription	Forced entry into sleep(stop)mode. It is a function for current measurement. Enter forced stop mode. It is applied without joining. CLASS A is applied after joining, but not in TX / RX And since the operation may be abnormal, this function is called Join Applies only for current measurement before completion.
Function	setSleepMode(void)
Return value	none
Example	setSleepMode ();

3.62. Set Channel Mask

Discription	Set Channel Mask
Function	setChannelMask(char* opt1, char* opt2, char* opt3, char* opt4, char* opt5, char* opt6)
Return value	SUCCESS : 0, FAIL : 1, BUSY : 2
Parameter	opt1 ~ opt6 : 0000 ~ FFFF (opt1: ch0~ch15, opt2: ch16~31, opt3 : ch32~ch47, opt4: ch48~ch63, opt5: ch64~80) ※ See the LoRaWAN Regional Parameters document

Example	Char* opt1="00ff";
	Char* opt2="0000";
	Char* opt3="0000";
	Char* opt4="0000";
	Char* opt5="00ff";
	Char* opt6="0000";
	setChannelMask (opt1); // use only 8 channels (ch0 ~ ch7)
	setChannelMask(opt1,opt2,opt3,opt4,opt5,opt6);
	// use ch0~ch7,ch64~ch71

3.63. Get Channel Mask

Discription	Get Channel Mask
Function	getChannelMask(uint8_t ch)

Return value	Channel Mask value (uint16)
Parameter	ch : channel number ※ See the LoRaWAN Regional Parameters document
Example	getChannelMask (0);

3.64. Executes an Alarm or Interrupt

Discription	When a timer alarm or a GPIO interrupt occurs, It allows the event to be processed. It is used to LoRaWAN protocol. It is used to wake up the timer alarm or GPIO interrupt in sleep mode (ex: When the press WAKEUP1 button to wake up in sleep mode When an RTC timer event occurs)
Function	Alarm_GPIO_Wakeup_Fn (void)
Return value	void
Example	Enable_enter_stop_mode(); Device_State_Sleep_Fn(); Alarm_GPIO_Wakeup_Fn();

3.65. Wake up Timer

Discription	This is RTC timer It operates in normal mode and sleep(stop, low power run) mode.. Wake up after (Basic_time + Var_time)
Function	void Wakeup_Timer(TimerEvent_t *obj, void (*callback)(void),uint32_t basic_time, uint32_t var_time)
Return value	void
Parameter	(see table below)
Example	Wakeup_Timer(&TxUser,User_fn,60000,0U); // Wakes up every 60 seconds and runs User_fn.

Parameter	Value(range)	Description
TimerEvent_t	TxUser	User RTC timer ID
callback	function	User function
Basic_time	30~4294967000	Wakeup after Basic_time (millisec)
Var_time	0 ~ 4294967 Var_time < Basic_time	Variable time It changes within the set value

3.66. Write data to NVM

<table>
<tr><td>Discription</td><td>This function is for writing EEPROM data for the user.
※This function operates normally with the voltage within the operating Voltage. Therefore, it is necessary to check the voltage before using this function.
If the voltage is too low, the EEPROM data may be corrupted.</td></tr>
<tr><td>Function</td><td>eNVErr_t nvm_write_user(const uint32_t lba, const void* data, const uint32_t len)</td></tr>
<tr><td>Return value</td><td>Enum (NVERR_NONE:0, NVERR_BUSY:1,NVERR_NOSPACE:2, NVERR_INVALID:3, NVERR_WRITE_FAIL:4)</td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><th>Parameter</th><th>Value(range)</th><th>Description</th></tr>
<tr><td>lba</td><td>NVM_USER (0x8080800) ~(0x80817FF)</td><td>User-accessible EEPROM Address area</td></tr>
<tr><td>data</td><td>const void*</td><td>User data</td></tr>
<tr><td>len</td><td>1 ~ 0x1000</td><td>User data size
(unit: byte)</td></tr>
</table>
</td></tr>
<tr><td>Example</td><td>typedef struct _LORA_EEPROM_tag
{
uint32_t test1;
uint8_t test2[16];
} lora_eeprom_t;
lora_eeprom_t eeprom;
eeprom.test1 = 12;
nvm_write_user(NVM_USER + offsetof(lora_eeprom_t, test1),(void *)(&eeprom.test1), 4);.</td></tr>
</table>

3.67. Read NVM data

<table>
<tr><td>Discription</td><td>This function reads NVM data.
※This function operates normally with the voltage within the operating Voltage. Therefore, it is necessary to check the voltage before using this function.</td></tr>
<tr><td>Function</td><td>eNVErr_t nvm_read_user(const uint32_t lba, void* data, const uint32_t len);</td></tr>
<tr><td>Return value</td><td>Enum (NVERR_NONE:0, NVERR_BUSY:1,NVERR_NOSPACE:2, NVERR_INVALID:3, NVERR_WRITE_FAIL:4)</td></tr>
</table>

<table>
<tr><td>Parameter</td><td>
<table>
<tr><th>Parameter</th><th>Value(range)</th><th>Description</th></tr>
<tr><td>lba</td><td>NVM_USER (0x8080800)
~(0x80817FF)</td><td>User-accessible EEPROM Address area</td></tr>
<tr><td>data</td><td>Void*</td><td>User data</td></tr>
<tr><td>len</td><td>1 ~ 0x1000</td><td>User data size
(unit: byte)</td></tr>
</table>
</td></tr>
<tr><td>Example</td><td>typedef struct _LORA_EEPROM_tag
{
uint32_t test1;
uint8_t test2[16];
} lora_eeprom_t;
lora_eeprom_t eeprom;

nvm_read_user(NVM_USER + offsetof(lora_eeprom_t, test1), (void *)(&eeprom.test1), 4);</td></tr>
</table>

4. P2P API discription

4.1. P2P_Init

Discription	Initialize P2P mode - Set the SX1276 to LoRa operation mode. - Initialize the variables used in P2P mode. - Load P2P mode setting value. - Set to receive mode
Function	P2P_Init(void)
Return value	Void
Parameter	None
Example	P2P_Init(); // This function is used after execution of Start_Init()

4.2. P2P_Mode

Discription	Only the unit functions for transmit and receive are executed SeongJi Industrial P2P protocol This function is set to RX mode in idle state. Do not use with other modes(API_MODE() or CLI_MODE())
Function	P2P_MODE(void)
Return value	Void

Parameter	None
Example	P2P_MODE(); // This function is used after execution of P2P_Init()

4.3. Check P2P Tx Done

Discription	This function checks whether transmission is completed after sending a message.
Function	Check_P2P_TX_Done(void)
Return value	Uint8_t Tx done: 1, else: 0
Parameter	None

Example	P2P_AT_CMD("AT+LORA_MSGW=2,1234567890₩r₩n₩0");
	P2P_MODE();
	If(Check_P2P_TX_Done ()) // If the transmission is completed
	{
	PRINTF("The TX is completed");
	}

4.4. P2P AT Command

Discription	It is an internal function that can use the same command to input through UART. You must add "₩r₩n₩0" to the end of the message See the "AppNote_AT_command_V3.2_en.pdf" file
Function	P2P_AT_CMD(uint8_t *atcmd)
Return value	void
Parameter	string
Example	Start_Init(); P2P_Init(); --- Ex1) P2P_AT_CMD("AT+LORA_MSGW=2,1234567890₩r₩n₩0"); Ex2) uint8_t str[] = {0x41, 0x54, 0x2B, 0x42, 0x49, 0x4E, 0x57, 0x3D, 0x03, 0xAA, 0xEE, 0xFF, 0xBB, 0x0D, 0x0A, 0x00}; P2P_AT_CMD(str); Ex3) P2P_AT_CMD("AT+LORA_S=₩r₩n₩0"); --- P2P_MODE();

4.5. Check P2P Rx Message

Discription	Check for received a message When the return value is 1 , you can use rf_rx structure to use the received message.
Function	Check_P2P_Rx_Msg(void)
Return value	Uint8_t Received message: 1, else: 0
Parameter	None

Example	P2P_MODE();
	if(Check_P2P_Rx_Msg()) // If rececive data is.
	{
	PRINTF("ID : %d₩r₩n",rf_rx.id);
	PRINTF("SN : 702C1F%X%X%X₩r₩n",rf_rx.sn[1],rf_rx.sn[2],rf_rx.sn[3]);
	if(rf_rx.bin ==0) // if rx data is ASCII
	{
	PRINTF("data = %s₩r₩n",rf_rx.data); // print rx data
	}
	else // if rx data is Binary
	{
	HAL_UART_Transmit_IT(&debugHandle,rf_rx.data,rf_rx.data_size);
	// print rx data
	DelayMs(rf_rx.data_size/10 +1);
	// The PRINTF command does not work normally while data is being output
	}
	}

5. Define Setting

1. STM32L071xx : CPU Chipset define (default)
2. USE_HAL_DRIVER : HAL driver define (default)
3. USE_BAND_920 : Korea band define (option)
4. AU915 : AU915 band define (option)
5. LOW_POWER_MODE_ENABLE : Low power enable define (default)
6. USE_DEBUGGER : Debug message display define (default)
7. TRACE : Debug trace define (default)
8. USE_BOOTLOADER : Boot loader define (default)
9. RS485_USE : RS485 board define (option)
10. LED3_TRX : when Tx/Rx, LED Blink define (option)
11. LA915A: LA915A band define (option) : This option should be used with the AU915
12. USE_TCXO: Applies to LOM204A module produced after 2019
13. P2P : Whe using P2P mode (option) There are two kinds of libraries
    (for WAN , for WAN+P2P)