

Technical Document

ELICIT-R1 USER GUIDE ver.2.6

RF Module for Industrial IoT



Feature List

1. Operation Condition

- Supply voltage: 1.8 ~ 3.8V
- Operating current (Tx): 87mA @ +20dBm
- Operating current (Rx): 8.2mA
- Deep-Sleep current: < 1µA
- Operating temperature: -40°C to +85°C

- Operating frequency: KOR USN 940.1 ~ 944.3MHz
- Operating RF channel: 19
- Maximum Tx power: +20dBm
- LBT(CSMA/CA), ECH(ELICIT-R1+ only)
- 2 PHY mode

Ultra High Data rate (UHDR) mode (only ELICIT-R1+)	
Modulation	MSK
RF Bit rate	120kbps
Rx sensitivity	-105dBm
High Data rate (HDR) mode	
Modulation	MSK
RF Bit rate	80kbps
Rx sensitivity	-105dBm
Long Range (LR) mode	
Modulation	2FSK
RF Bit rate	20kbps
Channel coding	Convolutional coding(K=7)
Rx sensitivity	-110dBm

2. Interface

- 2x UART
- 8x GPIO
- 2x Button Dedicated Pin
- 2x Status LED Dedicated Pin
- 1x I2C
- 1x 12bit ADC, 1x 12bit DAC

3. General Specification

- Form factor: Through-hole(1.27mm), surface-mount
- Size: 20x27x3(mm)
- Antenna Connector: U.FL

4. Wireless Characteristics

5. Certification

ELICIT-R1: KC (R-R-ELQ-ELICIT-R1)
 ELICIT-R1+: KC (R-R-ELQ-ELICIT-R1-PLUS)

Revision History

Author	Description of Changes	Version	Date
ELIOT	Initial Draft		Mar. 2022
ELIOT	Button, Event 설명 추가, 양산 제품으로 사진 변경, 인증 정보 추가		Sep. 2022
ELIOT	Command 변경, Linked event 추가	1.00	Mar. 2023
ELIOT	ELICIT R1 ver1.00에서 ver2.00으로 update - ver1.00과 ver2.00 간에는 무선 연결이 지원되지 않습니다 - EVENT slot 수 조정 - PANID 크기 및 사용방법 수정 - AT command 추가(FJOIN, IND)	2.00	Mar. 2024
ELIOT	새로운 AT Command 형식을 추가(Format Type 1)	2.1	Mar. 2024
ELIOT	Test version	2.2	May. 2024
ELIOT	JOIN RSP Button 기능 변경 IO INT 기능 수정 LED default mode 수정	2.3	May. 2024
ELIOT	ELICIT-R1+ information 추가(ECH 추가) AT command 수정 & 추가	2.4	Nov. 2024
ELIOT	AT command 수정 & 추가 - PARAMS command 수정, NAME command 추가 - 자신의 address를 "THIS"라고 입력 가능 - command 부분 소문자 입력 가능	2.5	Mar. 2025
ELIOT	ELICIT-R1+ 기능 추가 - GPIO pin & AUX pin ADC 기능 추가	2.6	Apr. 2025

Disclaimer

This work contains information supplied by ELIOT SYSTEM INC. ("ELIOT SYSTEM"). All such information is supplied without liability for errors or omissions. No part may be reproduced, disclosed, or used except as authorized by contract or by other written permission by ELIOT SYSTEM. The copyright and the foregoing restrictions on reproduction and use extend to all media in which the information may be embodied.

엘리엇시스템(주)는 본 사용 설명서와 관련된 특허권, 상표권, 저작권 기타 지적 소유권 등의 권리를 가지고 있습니다. 본 사용 설명서의 모든 내용은 엘리엇시스템 주식회사의 사전 승인 없이 어떠한 형식이나 수단으로도 복사 또는 수정하여 사용할 수 없습니다. 승인 없이 문서의 일부 또는 전체 내용을 사용할 경우 처벌을 받을 수도 있습니다.

엘리엇시스템 주식회사

<https://www.eliot-system.com/>

Copyright © 2021 ELIOT SYSTEM Inc.

ELIOT SYSTEM CONFIDENTIAL PROPRIETARY



Contents

1 Technical Specification	10
1.1 General Operating Conditions	10
1.2 Absolute Maximum Ratings	10
1.3 RF Characteristics	10
1.4 PHY Mode 0 – Long range mode	11
1.5 PHY Mode 1 – High data rate mode.....	11
1.6 PHY Mode 2 – Ultra High data rate mode.....	11
1.7 HW Pin Description.....	12
1.8 ELICIT R1/R1+ Dimension.....	14
2 Elicit.....	15
2.1 산업용 사물인터넷	15
2.1.1 940MHz 대역.....	15
2.1.2 100mW 출력.....	16
2.1.3 Proprietary Stack.....	16
2.1.4 Industrial IO.....	16
2.1.5 Low Power.....	16
3 Elicit Protocol Stack	17
3.1 Packet Frame Architecture	17
3.2 다중 접속(Multiple Access).....	18
3.2.1 LBT 모드	18
3.2.2 ECH 모드	19
3.3 Elicit Network 구성	20
3.3.1 ELICIT PAN.....	20
3.3.2 Public Network	22
3.4 Network Status	23
3.4.1 Alone State	23
3.4.2 Joinable State.....	23
3.4.3 Join State	23
3.5 PAN Join 절차	24

3.5.1 주소(Address).....	27
4 LINK MODE	28
4.1 LBT.....	28
4.1.1 Tree Topology.....	28
4.1.2 Tree Routing	29
4.1.3 Broadcast	30
4.1.4 Hopping Mode 1	30
4.1.5 Hopping Mode 2	31
4.1.6 Hopping Mode 3	31
4.1.7 Hopping Mode 4	32
4.2 ECH.....	33
4.2.1 ECH 특징	33
4.2.2 Channel hopping list	34
4.2.3 ACH(Adaptive channel hopping)	34
4.2.4 ECH 구조	34
4.2.5 비콘(Beacon)	35
4.2.6 Slot 할당.....	35
4.2.7 노드 호출(Pending address).....	36
4.2.8 Join in ECH	36
4.2.9 ECH 운영 예제.....	36
5 Radio 설정.....	38
5.1 RF Channel.....	38
5.2 TX power.....	38
5.3 PHY mode	38
5.4 Channel Scan	38
6 IO	40
6.1 GPIO.....	41
6.2 DI	41
6.3 DO	42
6.3.1 Push-pull out.....	42
6.3.2 Open-drain out.....	43

6.3.3 Open-source out	43
6.4 AUX Input	44
6.5 RESET	45
6.6 ADC	45
6.7 DAC	45
6.8 Button	45
6.8.1 버튼으로 Join	47
6.8.2 버튼으로 초기화(Initialize)	47
6.8.3 버튼으로 사용자 명령 수행(Button Linked Event).....	47
6.8.4 버튼으로 Sleep mode 해제	47
6.9 LED	48
7 Serial Data Interface.....	49
7.1 UART	49
7.2 User data 전송	50
7.2.1 SEND 명령	50
7.2.2 UART1 자동 전송 모드(Pass through).....	51
7.2.3 UART2 자동 전송 모드	52
7.2.4 Example: 무선 UART 콘솔	54
7.2.5 Example: 무선 MODBUS 장비	55
7.3 I2C	56
8 Sleep mode	57
8.1 Normal sleep	57
8.2 Deep sleep	58
8.3 다시 잠들기(Back to sleep again).....	58
8.3.1 LBT mode에서 Event 수행 완료 후 sleep timing.....	58
8.3.2 ECH mode에서 다시 잠들기.....	58
9 Event	60
9.1 Indication message	60
9.2 Report to AP	60
10 Linked Event.....	61
10.1 Periodic timer linked event (PTEVT).....	61

10.2 Sleep timer linked event (STEVT).....	61
10.3 DI Linked event (DIEVT).....	61
10.4 Button Linked event (BTEVT)	62
10.5 Wake up from deep sleep linked event (DSEVT).....	62
10.6 Wake up from normal sleep linked event (NSEVT).....	62
11 기타 부가 기능	63
11.1 디바이스 초기화	63
11.2 펌웨어 업데이트	63
11.3 PER(Packet error rate) test.....	63
11.4 PARAMS SAVE & LOAD	64
12 AT command	65
12.1 AT Command 전용 UART1.....	65
12.2 표기 방식	65
12.3 AT 명령 구조	66
12.4 명령 형식	67
12.5 응답/수신 형식	68
12.6 응답/수신 형식 유형(AT Format Type)	69
12.6.1 형식 유형의 차이점	69
12.7 AT 명령어 예제	71
12.7.1 설정, 실행, 전달 명령 예제(Set/Execute/Send Examples)	71
12.7.2 읽기, 문의 명령 예제(Get/Querry Examples)	72
12.7.3 수신, 알림 메시지 예제(Receive/Indication Examples)	72
13 AT Command 리스트	74
13.1 AT+HELP	74
13.2 AT+FORMAT	74
13.3 AT+VER	75
13.4 AT+SN	75
13.5 AT+BOOT	76
13.6 AT+RESET	76
13.7 AT+FINIT	77
13.8 AT+ADDR.....	78

13.9 AT+PANID	79
13.10 AT+HOPMODE.....	79
13.11 AT+ROLE.....	80
13.12 AT+OPMODE	81
13.13 AT+PHY.....	81
13.14 AT+JOINSTS	82
13.15 AT+PARAMS	83
13.16 AT+JOINREQ.....	84
13.17 AT+JOINRSP	84
13.18 AT+PJOIN	85
13.19 AT+FJOIN	85
13.20 AT+EXIT	86
13.21 AT+ASADDR.....	86
13.22 AT+ECHO	87
13.23 AT+BAUD.....	88
13.24 AT+SEND	89
13.25 AT+U1SND	90
13.26 AT+U2SND	92
13.27 AT+RFCH	94
13.28 AT+TXPWR.....	95
13.29 AT+TONE	96
13.30 AT+BTN.....	97
13.31 AT+LED	97
13.32 AT+IO	98
13.33 AT+AUX	100
13.34 AT+DI	100
13.35 AT+DO	101
13.36 AT+VCC.....	102
13.37 AT+ADC	102
13.38 AT+VDAC.....	103
13.39 AT+I2C	104

13.40 AT+TIME	105
13.41 AT+XTAL.....	106
13.42 AT+DIEVT	106
13.43 AT+BTEVT.....	107
13.44 AT+PTEVT.....	108
13.45 AT+STEVT	109
13.46 AT+NSEVT	110
13.47 AT+DSEVT	111
13.48 AT+DSWUP	112
13.49 AT+SLEEP.....	113
13.50 AT+PER	114
13.51 AT+PER.TX.....	115
13.52 AT+PER.RX	115
13.53 AT+PERSTOP.....	116
13.54 AT+SCAN	116
13.55 AT+TIMEOUT	117
13.56 AT+REPORT.....	118
13.57 AT+TXRTY.....	119
13.58 AT+NAME.....	120
13.59 AT+LINKMODE	120
13.60 AT+BCN	121
13.61 AT+ECHLIST	122
13.62 AT+ACH	123
13.63 AT+PEND	124
13.64 AT+IND	125

1 Technical Specification

1.1 General Operating Conditions

Parameter	Min	Typ	Max	Unit
Operating temperature	-40	-	85	°C
Supply voltage (VCC)	1.8	3.3	3.8	V
Operating current (Tx), +20dBm ^{#1,2}	85	87	90	mA
Operating current (Tx), +14dBm ^{#1,2}	40	45	50	mA
Operating current (Tx), +10dBm ^{#1,2}	33	35	40	mA
Operating current (Rx), Listen ^{#1,2}	7.1	8.2	9.1	mA
Stand-by current (Sleep) ^{#1,2}	5	-	6	µA
Stand-by current (Deep-Sleep) ^{#1,2}	0.5	-	1	µA
Voltage on IO(GPIO, AUX, I2C, UART)	-0.3	VCC	VCC+0.3	V
Input Low voltage of GPIO	-	-	0.3*VCC	V
Input High voltage of GPIO	0.7*VCC	-	-	V
Input voltage of ADC	-0.3		VCC	V

#1 VCC = 3.3V

#2 PHY mode = High data rate mode (MSK 80kbps)

1.2 Absolute Maximum Ratings

Parameter	Min	Typ	Max	Unit
Storage temperature	-50	-	+150	°C
Voltage on supply pin (VCC)	-0.3	-	3.8	V
Voltage on GPIO pin	-0.3	-	VCC+0.3	V
Voltage on RESETn pin	-0.3	-	3.8	V
Absolute voltage on RF connector	-0.3	-	1.2	V
Current per GPIO pin (Sink)	-	-	50	mA
Current per GPIO pin (Source)	-	-	50	mA
Current for all GPIO pins (Sink)	-	-	200	mA
Current for all GPIO pins (Source)	-	-	200	mA

1.3 RF Characteristics

Parameter	Min	Typ	Max	Unit
RF Frequency ^{#1}	940.1	-	944.3	MHz
Maximum Tx Power	17.7	20	21.4	dBm
Minimum Tx Power	-0.8	0	0.2	dBm
Adjustable RF Power Step	-	0.1	-	dBm
Spurious emissions1 (718~915MHz) ^{#2}			-78	dBm
Spurious emissions2 (949.3~962MHz) ^{#2}			-85	dBm
Spurious emissions3 (over 1GHz) ^{#2}			-41	dBm

#1 간접회피 또는 간접경감기술로 940.1~944.3MHz 주파수 대역의 전파를 사용하는 USN 용 무선설비 기준

#2 RF CH 2 ~ 20, TX Power 10dBm

1.4 PHY Mode 0 – Long range mode

Parameter		Unit
Modulation	2FSK	-
Deviation	10	kHz
RF bit rate	20	kbps
Symbol (Data) rate	10	kbps
Channel spacing	200	kHz
Forward error correction (FEC)	Convolutional Code (Constraint k = 7)	-
Rx sensitivity	-110	dBm

1.5 PHY Mode 1 – High data rate mode

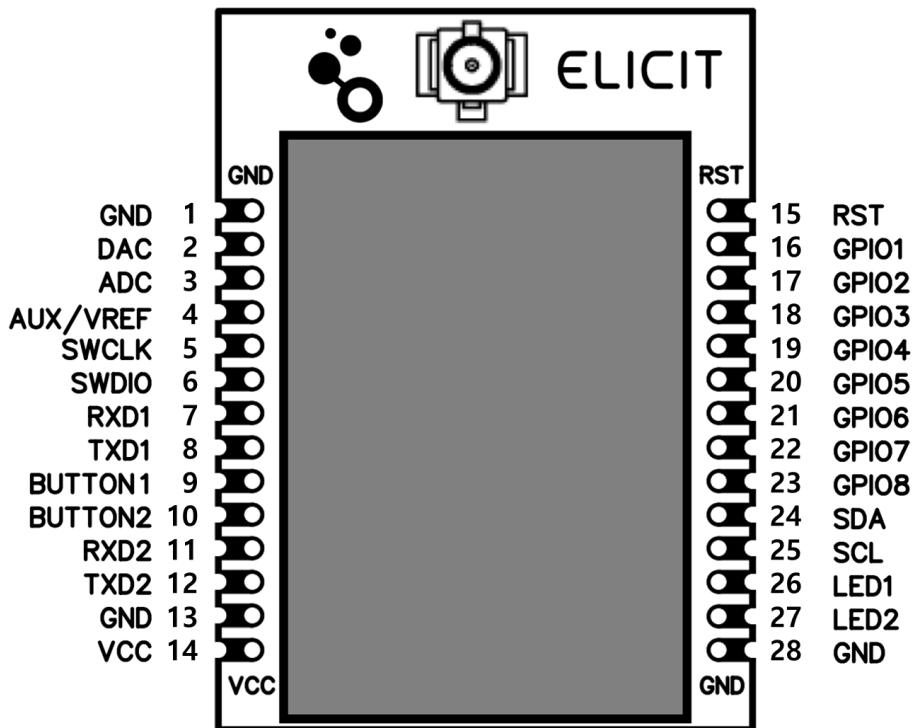
Parameter		Unit
Modulation	MSK	-
Deviation	20	kHz
RF bit rate	80	kbps
Symbol (Data) rate	80	kbps
Channel spacing	200	kHz
Channel coding	N/A	-
Rx sensitivity	-105	dBm

1.6 PHY Mode 2 – Ultra High data rate mode

(Only ELICIT-R1+)

Parameter		Unit
Modulation	MSK	-
Deviation	30	kHz
RF bit rate	120	kbps
Symbol (Data) rate	120	kbps
Channel spacing	200	kHz
Channel coding	N/A	-
Rx sensitivity	-105	dBm

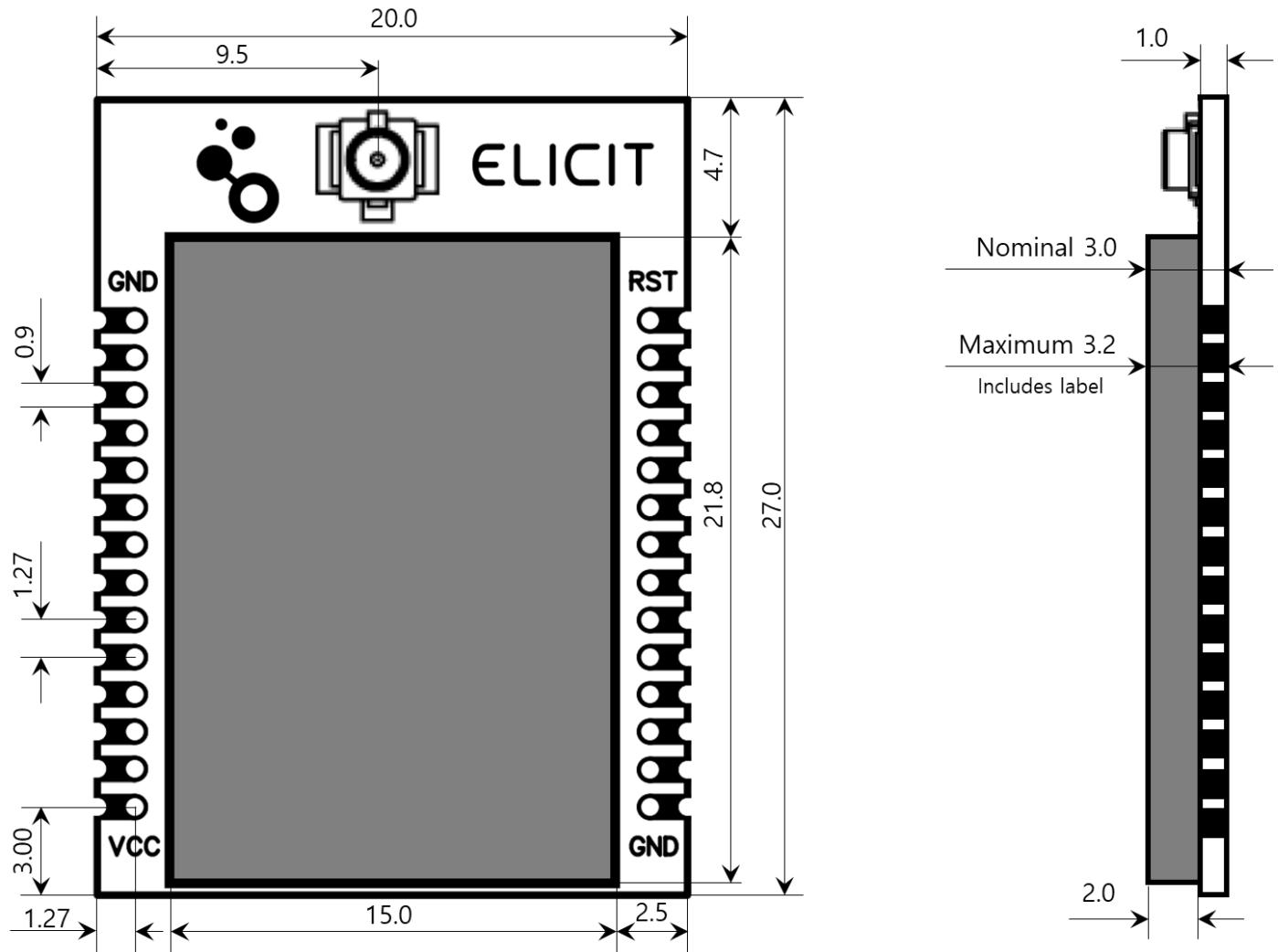
1.7 HW Pin Description



PIN NO.	Name	Description
1	GND	Ground
2	DAC	Digital Analog Converter Output
3	ADC	Analog Digital Converter Input
4	AUX	Auxiliary Input, could wake up from normal sleep
5	SWCLK	SWD CLOCK
6	SWDIO	SWD DATA
7	RXD1	UART1 RX, for host interface (AT command)
8	TXD1	UART1 TX, for host interface (AT command)
9	BUTTON1	External Button Input1, wakes up from Deep/normal sleep
10	BUTTON2	External Button Input2, wakes up from normal sleep
11	RXD2	UART2 RX, for user device
12	TXD2	UART2 TX, for user device
13	GND	Ground
14	VCC	Power supply (1.8 ~ 3.8V). default 3.3V
15	RST	Reset input, active low, internal pulled up.
16	GPIO1	GPIO (DI/DO, wakes up from Deep-sleep)
17	GPIO2	GPIO (DI/DO)
18	GPIO3	GPIO (DI/DO, wakes up from Deep-sleep)
19	GPIO4	GPIO (DI/DO, DI linked event)
20	GPIO5	GPIO (DI/DO, DI linked event)
21	GPIO6	GPIO (DI/DO, DI linked event)
22	GPIO7	GPIO (DI/DO, DI linked event, wakes up from normal sleep)

23	GPIO8	GPIO (DI/DO, DI linked event, wakes up from normal sleep)
24	SDA	I2C DATA
25	SCL	I2C CLOCK
26	LED1	Status LED1(Push-pull/Open drain output)
27	LED2	Status LED2(Push-pull/Open drain output)
28	GND	Ground

1.8 ELICIT R1/R1+ Dimension



단위: mm

PCB 두께: 1mm

Shield can 높이: 2mm

Through hall diameter: 0.6mm

2 Elicit

Elicit[엘리시트]은 엘리엇시스템 주식회사(ELIOT SYSTEM Inc.)에서 자체 개발한 IoT 를 위한 Proprietary wireless network protocol 이자, 무선 모뎀 제품명이다. Elicit protocol 은 저전력으로 동작하며, 보안과 안정성, 확장성이 높은 무선 네트워크를 위해 개발되었으며, 무선 네트워크에 대한 전문적인 지식을 습득하지 않아도 현장 엔지니어들이 빠르게 무선 네트워크를 구축하고 운영할 수 있도록 사용자 인터페이스를 제공한다.

Elicit	무선 IoT 네트워크를 위한 Proprietary Protocol Stack
ELICIT-R1/R1+	Elicit protocol 이 탑재된 940MHz 대역의 무선 모뎀

2.1 산업용 사물인터넷

산업용 사물인터넷(Industrial Internet of Things, IIoT)은 제조, 건축, 에너지 관리, 환경 모니터링 등 다양한 산업 분야에서 혁신적인 변화를 이끌고 있다. 이러한 IIoT 네트워크는 수많은 디바이스가 연결되어 데이터를 주고받으며, 실시간으로 모니터링하고 제어할 수 있어야 한다. 이를 효과적으로 구현하기 위해서는 안정적이고, 확장 가능하며, 에너지 효율적인 통신 솔루션이 요구된다.

ELICIT 모듈은 산업용 사물인터넷 무선 네트워크를 위해 설계, 개발되었다. 다양한 산업용 설비, 장비와 쉽게 연결할 수 있도록 8 개의 GPIO, 2 개의 UART(MODBUS 지원), I2C, 12bit ADC/DAC 를 가지고 있으며, Input/Timer 이벤트 등과 연동하여 사용자의 명령을 수행할 수 있도록 설정 가능하여 사용자가 원하는 IoT 서비스를 별도의 프로그래밍 없이 빠르게 제공할 수 있다. 일반 IoT 서비스를 위한 무선 기술들은 주로 가정의 범위(Home area)안에서 사용되지만, 산업용 IoT 는 보다 넓은 범위의 사업장을 대상으로 신뢰할 수 있는 무선기술이 필요하다.

ELICIT 모듈이 산업 IoT 네트워크를 구성하는 데 필수적인 이유는 다음과 같다.

2.1.1 940MHz 대역

ELICIT 모듈은 940.1~944.3MHz 대역을 사용한다. 이 대역은 대한민국에서 USN(Ubiqitous Sensor Network) 용도로 새롭게 지정된 주파수로, 기존 USN 용 주파수 대비 채널 혼잡이 적다. ELICIT 모듈은 총 21 개의 무선 채널을 제공하며, 데이터 충돌과 간섭을 최소화하여 안정적인 네트워크 환경을 제공한다.

신규 940MHz 지정 이유: 기존의 LoRa, Wi-SUN 등 LPWA(Sub-Giga) 기술과 RFID 가 함께 사용하는 917MHz 대역은 산업과 민간에서 IoT 디바이스 수요가 늘어나면서 채널 부족과 낮은 출력(25mW)¹ 제한이라는 문제가 있었다. 이를 해결하기 위해 정부는 940.1~944.3MHz 대역을 새롭게 USN(Ubiqitous Sensor Network) 용도로 지정하고, 이 대역에서 최대 200mW 출력이 가능한 21 개 채널을 제공하도록 하였다. (Figure 1 940MHz 대역의 기술 기준 - 출력기준(과학기술정보통신부)).

¹ 917MHz 대역에서 허용하는 200mW 출력은 실외 고정형 점대다점 기기로, 기지국 장비등이 이에 해당하며, 일반 무선 디바이스는 최대 25mW 로 제한된다.

신고하지 아니하고 개설할 수 있는 무선국용 무선설비의 기술기준

⑥ 간선회피 또는 간선포경기술로 940.1 ~ 944.3MHz 주파수대역의 전파를 사용하는 USN용 무선설비의 기술기준은 다음 각 호와 같다.

4. 안테나절대이득을 포함한 복사전력은 200mW 이하일 것

Figure 1 940MHz 대역의 기술 기준 - 출력기준(과학기술정보통신부)

2.1.2 100mW 출력

ELICIT 모듈은 최대 출력 100mW(20dBm)를 지원한다. 이를 통해 넓은 통신 범위를 제공하며, 기존 저출력 장치에 비해 거리 제약을 줄일 수 있다. 넓은 영역에서 다양한 잡음이 존재하는 산업현장과 같은 열악한 환경에서도 신뢰성 높은 데이터 전송이 가능하다.

2.1.3 Proprietary Stack

ELICIT 모듈은 독자적으로 개발된 무선 네트워크 스택인 [Elicit, 엘리시]을 사용한다. 이 스택은 IIoT 환경에서 요구되는 효율성과 유연성을 제공하며, 복잡한 네트워크 설정 없이도 안정적이고 간편한 통신을 지원한다.

2.1.4 Industrial IO

ELICIT 모듈은 다양한 IO 인터페이스를 제공하여 외부 디바이스와 쉽게 통합할 수 있도록 설계되었다. 입력 이벤트(Digital Input) 및 Timer 이벤트와 연동하여 사용자 정의 명령을 실행할 수 있는 기능을 지원한다. 이러한 기능은 추가 마이크로컨트롤러(MCU) 없이도 복잡한 동작을 수행할 수 있도록 하며, 시스템 설계의 복잡성을 낮출 수 있다.

2.1.5 Low Power

ELICIT 모듈은 초 저전력 설계로 에너지 효율을 극대화하였다. Deep Sleep 모드에서는 1μA 미만의 전류만 소모하도록 설계되었다. 이를 통해 배터리 수명을 대폭 늘릴 수 있으며, 유지보수 비용을 절감할 수 있다.

산업용 IoT 네트워크를 위한 ELICIT-R1/ELICIT-R1+

- USN 용도로 신규 지정된 940MHz 대역 사용, 혼잡하지 않은 21 개의 무선 채널 운영
- 최대 출력 100mW(20dBm), 보다 넓은 통신 범위 제공
- 독자 개발한 IIoT 를 위한 무선 네트워크 스택(Elicit)을 적용
- 다양한 IO Interface 제공, 입력 이벤트, Timer 이벤트와 연동하여 사용자 지정 명령 실행 가능
- 초 저전력(Deep sleep mode: < 1μA)

3 Elicit Protocol Stack

3.1 Packet Frame Architecture

Elicit Radio Frame Format

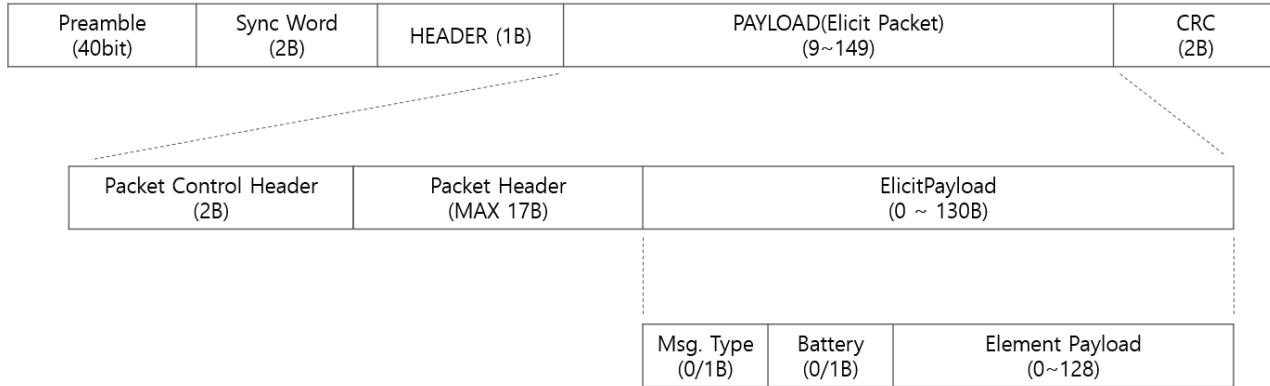


Figure 2 Elicit Radio Packet frame format

Elicit 프로토콜의 무선 패킷(Radio packet)은 위 그림과 같이 구성된다. preamble 과 Sync word 를 제외하고 한 번에 송신하는 패킷의 데이터는 최대 152byte 이다.

HEADER 는 물리계층의 정보를 담고 있으며, Radio Packet 의 마지막에는 16bit CRC(Cyclic redundancy checking)가 자동으로 덧붙여져서 전송된다. CRC 는 네트워크 통신에서 데이터의 무결성을 보장하기 위해 많이 사용하는 기술로, 데이터의 신뢰성을 높일 수 있다.

PAYLOAD 는 보안을 위해 엘리엇시스템이 개발한 Elicit Cipher 를 사용하여 매 패킷마다 새로 암호화한다. 즉, 동일한 데이터를 연속으로 전송해도 각각의 패킷은 서로 다르게 암호화되어 전송되기 때문에 도/감청 등의 보안 이슈에 매우 강력하게 대처할 수 있다.

PAYLOAD 는 다시 Packet Control Header, Packet Header, Elicit Payload 로 구성된다. Packet Control Header, Packet Header 는 메시지의 종류, 암호화 정보, 라우팅 정보 등이 포함되며, 프로토콜에서 자동으로 생성되는 정보이다. Elicit Payload 는 사용자 데이터인 Element payload 와 사용자 메시지의 형식, 현재 전원의 전압(배터리 전압)을 조합하여 구성한다. 사용자의 데이터(User data)는 Elicit Payload 안의 Element payload 에 할당되며, **최대 128Byte** 를 한 번에 전송할 수 있다.

수신한 패킷은 CRC 를 통해 수신한 데이터가 유효한지 판단하고, 오류가 있으면 해당 Packet 은 버리고 정상 수신된 Packet 만 사용한다. 이런 동작은 프로토콜에서 자동으로 처리되기 때문에 사용자는 별도로 수신한 데이터를 확인할 필요가 없다.

무선 전송의 신뢰도 향상을 위해 Elicit 에서는 상대방의 주소를 명시하여 전송할 때는 항상 ACK²를 요청하여 전송확인을 한다. 수신한 패킷이 ACK 요청 패킷이고, CRC 오류가 없으면 즉시 ACK 를 송신한다. 메시지 송신이 완료된 이후 20ms 이내에 ACK 가 오지 않으면 설정된 횟수[AT+TXRTY]만큼 재송신을 하고, 이 후에도 ACK 를 수신하지 못하면 해당 패킷은 송신 실패로 처리한다. 다수의 디바이스에게 동시에 전송(broadcasting)하는

² ACK(Acknowledgement, 확인응답): 수신 측에서 예상 없이 정상적으로 수신했을 때 송신 측으로 응답하는 것.

경우에는 ACK 를 요청하지 않는다. 사용자는 메시지를 전송한 후, 송신 성공이면 상대방에게 메시지가 전달된 것이므로, 별도의 전송 확인을 하지 않아도 된다.

신뢰성 높은 무선 네트워크를 위한 Elicit 프로토콜

- 무선 패킷을 암호화하는 Elicit Cipher 는 동일 데이터를 전송해도 매 패킷마다 다르게 변조
- CRC 를 통해 자동으로 수신 데이터의 유효성 확인
- 사용자가 한 번에 전송할 수 있는 최대 데이터 크기는 128byte
- ACK 를 통해 전송 확인

3.2 다중 접속(Multiple Access)

ELICIT 프로토콜은 다중 접속(다수의 디바이스가 한정된 무선 채널을 공유하는 방법, Multiple access)을 위해 LBT(Listen before talk)와 ECH(Elicit Channel Hopping)의 두가지 방식을 지원한다. ELICIT에서는 이러한 다중 접속 방식을 **Link mode** 라는 이름으로 구분한다. Link mode 는 명령[AT+LINKMODE]로 설정할 수 있다. Link mode 별 자세한 내용과 네트워크 구성 및 운영은 [4 장 LINK MODE]을 참조한다.

3.2.1 LBT 모드

LBT (Listen Before Talk) 모드는 데이터를 송신하기 전에 무선 채널의 사용 가능 여부를 확인한 후 송신을 수행하는 방식이다. ELICIT 네트워크에서는 송신 전에 채널 상태를 확인하는 **CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)**³ 알고리즘을 사용하여 다수의 디바이스가 무선 채널을 효율적으로 공유할 수 있도록 한다. 채널 상태 확인은 **Energy Detection CCA(Clear Channel Assessment)** 방식을 활용하며, 기준 임계값(Threshold)은 -65dBm 으로 설정되어 있다. 즉, 송신 전 측정한 채널의 수신 신호 세기(RSSI, Received Signal Strength Indicator)가 -65dBm 이하일 경우에만 송신이 이루어진다.

LBT 모드는 ECH 모드에 비해 송신 절차가 간소화되어 응답 속도가 빠르고, 필요한 데이터만 전송하기 때문에 전력 소모를 최소화할 수 있는 장점이 있다. LBT 모드는 **Tree 구조**의 네트워크 토플로지를 지원하며, 최대 4 단계의 **Multi-hop 통신**을 구현할 수 있다. Tree 구조를 사용함으로써 단순하고 안정적인 네트워크 구성이 가능하며, 특히 정적(static) 환경에서 효과적인 통신 방식을 제공한다.

³ CSMA(Carrier Sense Multiple Access)/CA(Collision avoidance): 반송파 감지 다중 접속/충돌 회피 기술, 송신하기 전에 반송파(주파수 채널)를 누군가 사용 중인지 확인후 사용 중이면 임의의 시간 동안 기다린 후(Back-off)에 다시 확인하고, 사용 중이 아닐 때 송신하는 방법.

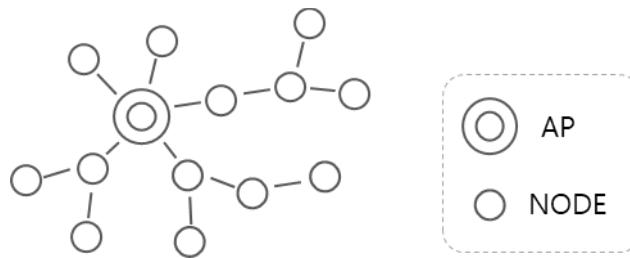


Figure 3 LBT: Tree Topology

LBT 모드는 설치와 운용이 간단하면서도 효율적인 통신 방식을 제공하여, 저전력 환경이나 대규모 디바이스 간 통신이 필요한 응용 분야에 적합하다.

ELICIT LBT mode

- ECH mode 대비 간결한 송신 절차와 빠른 응답 속도
- CSMA/CA 방식을 사용
- ELICIT R1은 LBT mode만 사용 가능하며, ELICIT R1+는 선택 가능(default LBT mode)
- 최대 4 단계의 Multi-hop을 지원하는 Tree 구조 네트워크 토플로지

3.2.2 ECH 모드

ECH (Elicit Channel Hopping) 모드는 AP(Access Point)에서 주기적으로 송신되는 **Beacon Message**를 기준으로 네트워크 동기를 맞추고, 데이터 송신을 위해 AP로부터 **통신 슬롯(time slot)**을 할당 받아 지정된 채널과 시간에 송신을 수행하는 방식이다. ECH 모드는 **Multi-channel** 및 **Time Slot** 기반 통신을 제공하며, ELICIT 네트워크에서 효율적이고 신뢰성 높은 데이터 송신을 가능하게 한다.

이 모드는 동기화된 **Beacon** 메시지를 기반으로 채널을 이동하며 각 노드가 서로 다른 시간에 통신을 수행하므로, 통신 신뢰성과 보안성을 동시에 강화할 수 있다. 특히, 주파수 간섭이 적은 환경을 유지하기 때문에 산업 환경이나 좁은 공간에서 다수의 노드가 동시에 동작해야 하는 상황에서 매우 유리하다.

ECH 모드는 **주파수 채널 간섭 회피**를 위해 주기적인 채널 변경(frequency hopping)을 수행하며, 이를 통해 동작 중 발생할 수 있는 간섭을 최소화한다. 이러한 방식은 **940MHz 대역**과 같이 상대적으로 혼잡도가 낮은 주파수 대역에서 높은 효율성을 발휘한다. 또한, **T SCH(Time Slotted Channel Hopping)** 기술에서 불필요한 요소를 제거하고, ELICIT 프로토콜에 최적화한 설계가 적용되어 프로토콜 간 호환성과 효율성을 높였다.

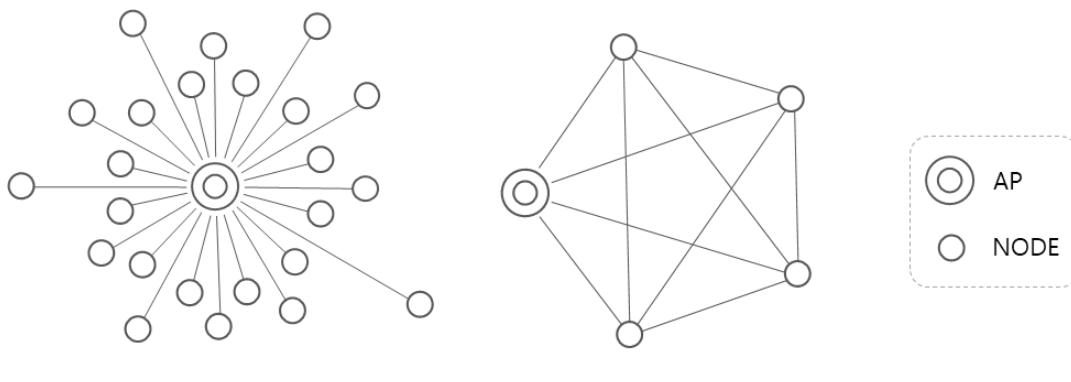


Figure 4 ECH: Star & Peer to peer

ECH 모드는 Tree 구조와 달리 **Multi-hop 통신**을 지원하지 않지만, 직접 통신이 가능한 **Peer to peer** 연결을 지원한다. ECH 는 **단일 흡 기반**의 통신 환경에서 최적의 성능을 제공하도록 설계되었다. 이러한 특성은 복잡한 네트워크 환경보다는 **단일 AP 와 다수 노드간 신뢰성 있는 통신**을 요구하는 응용 분야에 적합하다.

ELICIT ECH mode

- Beacon 동기화. Channel hopping & time slot 을 할당하여 송신
- 많은 node 들을 안정적으로 운영가능
- ELICIT R1+에서 사용 가능(default LBT mode)

3.3 Elicit Network 구성

무선 통신 기술의 한 범주인 **LPWAN(Low-Power Wide-Area Network)**은 낮은 전력으로 광범위한 지역을 커버하는 네트워크 기술이다. 주로 IoT 와 같은 저전력 디바이스가 넓은 지역에서 데이터를 전송하는 데 사용되며, LoRa, Wi-SUN 등이 대표적이다.

ELICIT 의 무선 특성과 활용 범위는 LPWAN 에 속하며, 저전력 설계와 뛰어난 무선 성능을 통해 넓은 지역에서 안정적인 데이터 통신을 제공한다. 이는 산업 시설 관리, 환경 모니터링 등 다양한 산업용 IoT 애플리케이션에 적합하다.

3.3.1 ELICIT PAN

ELICIT 네트워크는 **하나의 AP 와 다수의 노드(Node)로 구성된다.** ELICIT AP 를 중심으로 하는 개별 무선 네트워크를 **ELICIT PAN(Private Access Network)**이라고 정의한다. AP 는 하나의 PAN 을 구성하고, PAN 의 AP 는 유일하게 존재해야 한다. ELICIT 모듈은 별도의 HW 구분 없이 설정만으로 AP 와 NODE 의 역할을 지정할 수 있다.

다수의 PAN 이 존재할 경우, 각각의 PAN 은 메시지에 자동으로 포함되는 PAN ID 를 이용하여 구분한다. PAN ID 는 AP 마다 가지고 있는 고유한 값(Device ID)과 사용자가 지정한 값[User PAN ID(8Bit): [AT+PANID](#)]을 조합하여 구성한다. 사용자 지정 PNA ID 는 변경할 수 있지만, Device ID 는 변경할 수 없다. 따라서, **AP 마다 동일한 User PAN ID 를 설정해도 각각의 PAN 은 서로 통신할 수 없다.**

PAN 내부에서 각 디바이스를 구분하는 주소로는 16Bit SHORT ADDRESS(이하 SHORT ADDR)을 사용한다. PAN 내부에서 서로 메시지를 주고받을 때 SHORT ADDR 을 사용해서 발신자와 수신자를 구분한다.

Node 가 **PAN** 에 **가입하는 것을 Join** 이라고 정의하고, **PAN** 에 속하지 않은 상태를 **Alone**, **PAN** 에서 벗어나는 것을 **Exit** 라고 정의한다. Alone 상태일때의 사용자 PAN ID 는 0xFF, SHORT ADDR 은 0xFFFF 가 기본값으로 설정된다.

ELICIT PAN 을 인터넷이나 LTE 같은 다른 네트워크와 연결하려면, PAN 내 모든 노드와 통신이 가능한 AP 와 게이트웨이를 연결하여 구성할 수 있다.

AP 의 SHORT ADDR 은 항상 0x0000 으로 고정된다. ELICIT 모듈을 AP 로 설정하는 순간 SHORT ADDR 은 0x0000 으로 자동 설정된다. 디바이스 간 데이터를 전달하는 Multi-hop 모드인 경우, Hop mode 에 따라 주소의 단계를 구분할 수 있다. AP는 최상위 레벨의 주소이고, AP와 직접 통신할 수 있는 노드는 1단계 흡 레벨(1st HOP level)이라고 한다. AP와 연결되지 않고, 1단계와 연결된 노드를 2단계 흡 레벨이라고 하고, 계속해서 단계가 늘어날수 있다. Multi-hop 을 지원하는 LBT 모드에서는 최대 4 단계의 흡 레벨을 지원한다. 단계를 구분할 때 AP 쪽 단계를 상위 레벨, 면 쪽을 하위 레벨로 구분하여 설명한다.

Join 은 상위 level 에서 하위 level 로 확장하는 방식으로 구성한다. Node 는 Join 하면서 PAN 의 정보(PAN ID, RF CH, Data rate, Addressing mode 등)를 받아서 설정하고, SHORT ADDR 을 할당 받아서 PAN 의 구성원이 된다.

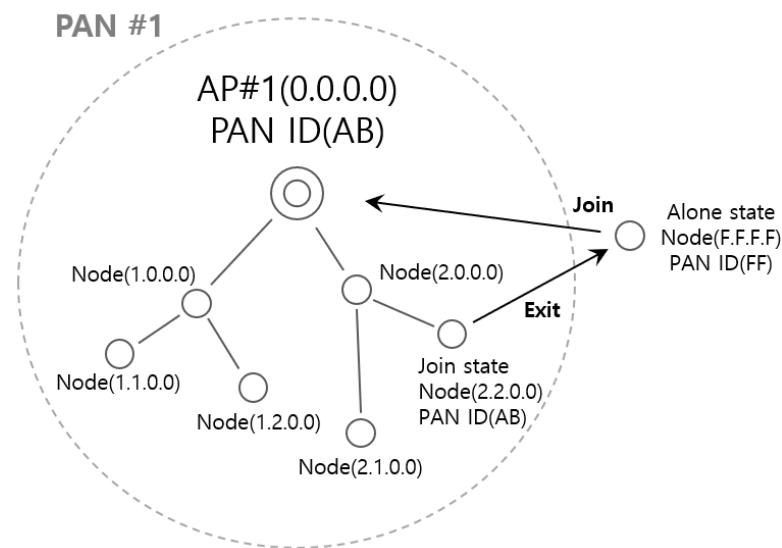


Figure 5 Elicit PAN – Multi-hop 모드(LBT mode)에서 Join/Exit

LONG ADDR 을 상호 알고 있는 디바이스들은 JOIN 여부와 상관없이 서로 직접 통신이 가능하다. Elicit Network 는 상호 통신이 가능한 모든 디바이스들의 통합된 네트워크를 의미한다.

Elicit Network

- 모든 디바이스는 자신의 고유 ID 를 가지고 있음
- ELICIT PAN: ELICIT AP 를 중심으로 구성한 네트워크
- PAN 내부에서는 16bit SHORT ADDR(SADDR)을 사용한다. AP 의 SADDR 은 항상 0000 으로 고정
- PAN 에 가입하는 것을 Join, PAN 에서 나가는 것을 Exit 라고 정의
- PAN 에 속하지 않은 상태는 Alone(Alone state)
- Alone 상태의 PAN ID 는 FF, SHORT ADDR 은 FFFF 이다.
- PAN ID, SHORT ADDR, 고유 ID(Serial Number)는 모두 16 진수(Hexadecimal)로 표기
- 디바이스간 LONG ADDR 을 알고 있으면 상호 직접 통신 가능([AT+SEND](#))
- Elicit Network: Elicit protocol 을 통해 서로 통신할 수 있는 모든 디바이스 네트워크

AT Command

CMD(Ref.)	Description	Set/Get	Parameters	Default/Initial
AT+ROLE	ROLE	Set/Get	NA(Nothing), NODE, AP	NA
AT+ADDR	Set/Get short address	Set/Get	[ADDR]	0xFFFF
AT+PANID	Set/Get PAN ID	Set/Get	[PANID]	0xFF
AT+RFCH	Set/Get Rf channel	Set/Get	[CHANNEL]	15
AT+HOPMODE	Set/Get Hop mode	Set/Get	[HOPMODE]	1
AT+PHY	Set/Get Date rate mode	Set/Get	[PHY]	H(high mode)
AT+LINKMODE	Set/Get Link mode	Set/Get	[LINL MODE]	0(LBT mode)
AT+JOINSTS	Join status	Get	(A) Alone, (J) Join	A

3.3.2 Public Network

ELICIT 네트워크에 Join 하지 않고, Alone 상태의 디바이스간 통신이 가능하도록 설정하고 네트워크를 구성할 수 있다. 이렇게 구성한 네트워크는 Public network 라고 하고, 다음의 특징이 있다.

Public network 의 특징(Alone 디바이스간 통신)

- Link mode LBT 만 가능(ECH 설정 불가능)
- Multi-hop 통신 불가능
- 다른 PAN 에 속한 디바이스와 통신 불가능
- 다른 Public network 와 혼신 가능성 있음

Alone 상태에서는 AP 의 정보를 받지 않은 상태이기 때문에 어떠한 PAN 과도 통신을 할 수 없다. 또한 Tree 구조를 구성하지 않았기 때문에 다른 디바이스를 경유하여 메시지를 전달하는 Multi-hop 을 사용할 수 없다. 단지 알고 있는 SHORT ADDR 을 통해 메시지를 전달할 수 있다.

Public network 를 구성하기 위해 필요한 네트워크 정보는 User PAN ID, SHORT ADDR, RF Channel, Data rate(PHY Mode), Link mode 이다. 이 정보만 일치하면 서로 통신이 가능하다. 즉 다른 사용자가 동일한 설정을 한다면 상호 통신이 가능하기에 보안이 없다. Public network 는 테스트용도로만 사용하고, 실제 서비스를 위해서는 ELICIT 네트워크를 구성하여 사용해야 한다.

아래 AT 명령어는 기본적인 네트워크 설정 관련 명령 리스트이다. Public network 를 구성한다면 모든 디바이스를 동일하게 설정해야 하고, ELICIT network 를 구성할 때는 먼저 AP 를 원하는 대로 설정하면, 이후 Join 하는 Node 는 AP 와 동일하게 네트워크 설정이 된다.

AT Command

CMD(Ref.)	Description	Set/Get	Parameters	Default/Initial
AT+ADDR	Set/Get short address	Set/Get	[ADDR]	0xFFFF
AT+PANID	Set/Get PAN ID	Set/Get	[PANID]	0xFF
AT+RFCH	Set/Get Rf channel	Set/Get	[CHANNEL]	15
AT+PHY	Set/Get Date rate mode	Set/Get	[PHY]	H(high mode)
AT+LINKMODE	Set/Get Link mode	Set/Get	[LINL MODE]	0(LBT mode)

3.4 Network Status

Elicit network 상태는 네트워크에 속하지 않은 Alone 상태와 네트워크에 속한 Join 상태로 구분할 수 있다. Alone 상태에서 Join 하기 위해서는 우선 접속가능 상태(Joinable state)로 변경된 후 Join 과정이 성공적으로 이루어진 다음에 Join state 가 된다.

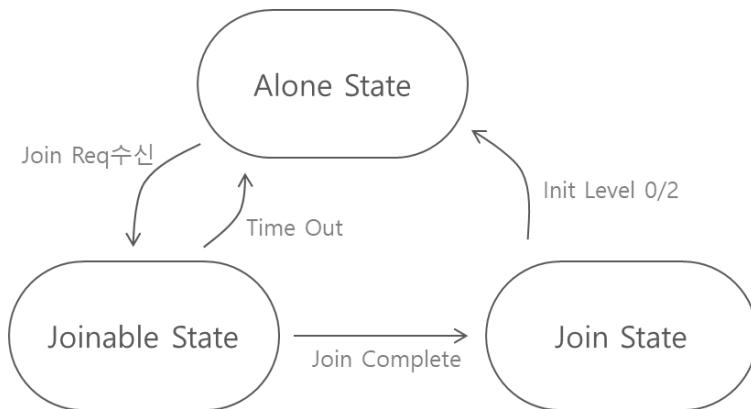


Figure 6 Network State Transition

3.4.1 Alone State

- 출고 후 기본 상태, Elicit Network 에 속하지 않은 상태
- PAN ID: 0xFF, SHORT ADDR: 0xFFFF
- Alone 상태에서 조인 요청(Join Request) 메시지를 수신하면 Joinable 상태로 변경

3.4.2 Joinable State

- PAN 에 JOIN 하는 과정 중인 상태
- 2 초마다 LED 1 과 2 가 동시에 2 회씩 점멸하여 현재 상태를 표시
- **Joinable 상태에서 1분 이내에 Join 허가 메시지를 받지 못하면 Alone 상태로 복귀**
- Join Response 메시지는 AT 명령[\[AT+JOINRSP\]](#)이나 버튼 1 을 이용하여 전송
- Join Response 메시지를 보낸 후, Join 허가 메시지[\[AT+PJOIN\]](#)를 수신하면 Join State 로 변경됨
- Join 허가 메시지에 포함된 PAN 정보에 따라 자동으로 네트워크 설정
- 네트워크 설정: Hopping Mode, PAN ID, SHORT ADDR 변경

3.4.3 Join State

- ELICIT PAN 에 포함된 상태
- Join 상태에서 Joinable 상태로 변경될 수 없음
- Join 상태에서 Alone 상태로 변경은 초기화 명령[\[AT+FINIT\]](#) 또는 EXIT 명령[\[AT+EXIT\]](#)을 사용

AT Command				
CMD(Ref.)	Description	Set/Get	Parameters	Default/Initial
AT+FINIT	Initialize the parameters	Set	[INIT LEVEL: 0/1/2]	-
AT+JOINSTS	Network Status	Get	-	-

3.5 PAN Join 절차

PAN 을 구성하는 절차는 AP 를 먼저 설정하고, 여기에 노드(Node)를 추가하는 방식으로 PAN 을 구성한다. 즉, 상위 Node 부터 차례로 하위 Node 를 Network 에 Join 시키는 과정이다. AP 에서 1st level Node 를 Join 시키고 Join 된 Node 는 자신의 하위에 새로 Node 를 Join 시킬 수 있다.

ELICIT PAN 은 AP 의 network 파라메터 설정대로 구성된다. 따라서 PAN 을 구성하기 전에 AP 의 PAN ID, Data rate, Link mode, RF Channel(이하 RFCH), Hopping mode 등을 미리 설정해야 한다.

Join 과정은 상위 노드에서 Join 요청 메시지[[AT+JOINREQ](#)]를 전송함으로써 시작한다. 이 메시지를 받은 Alone 상태의 모든 노드들은 Join 가능 상태로 변경된다. 변경된 상태를 쉽게 구분하기 위해 **LED 1, 2 가 2초마다 2회 짧게 점멸한다.**

ELICIT 모듈은 최대 출력이 20dBm(100mW)로 매우 넓은 통신 범위를 가지고 있기 때문에 Join 대상이 아닌 노드들(예를 들어, 다른 사용자의 디바이스)도 Join 가능 상태가 된다. 이러한 원치 않는 접속을 차단하기 위해서 Elicit network 에 Join 하려는 디바이스는 반드시 사용자가 버튼을 누르거나 AT Command 를 입력할 수 있어야 한다.

Join 가능 상태의 디바이스에서 Join 요청 메시지에 대한 응답(Join Response)을 보낼 수 있다. Join 응답은 AT 명령[[AT+JOINRSP](#)]을 통해서 전송하거나 버튼 1 을 누르면 전송된다. 응답 메시지를 전송하면 LED 점멸은 중단된다. Join 가능 상태가 된 이후 **1 분 이내에 Join 되지 않으면 다시 Alone 상태로 변경된다.**

Join 요청 메시지를 전송한 상위 디바이스는 다수의 디바이스로부터 Join 응답을 받을 수 있다. Join 응답 메시지에는 어떤 디바이스한테 join 요청 메시지를 받았는지 알기 위해 join 요청을 송신한 ADDR 와 응답한 디바이스의 LONG ADDR 이 포함되어 있다. 사용자는 응답한 디바이스 중에서 **LONG ADDR** 을 확인하고 원하는 디바이스를 선택하여 네트워크에 Join 시킨다. 따라서 사용자는 개별 디바이스의 LONG ADDR 을 사전에 알고 있어야 한다.

상위 디바이스는 Join 을 허락할 때 새로운 SHORT ADDR 을 할당하는데, 자동으로 할당[[AT+PJOIN](#)]하거나, 사용자가 임의의 SHORT ADDR 을 지정[[AT+FJOIN](#)] 할 수 있다. 자동으로 주소를 할당할 때는 내부에 ASADDR(Assigned address counter) 변수의 값을 사용하고, 할당할 때 마다 하나씩 증가시킨다. 이 값은 [[AT+ASADDR](#)] 명령을 통해 읽고 쓸 수 있다.

Join 요청 메시지를 전송했으나, 1 분 동안 Join 허락 메시지를 받지 못한 디바이스는 다시 Alone 상태로 전환된다. 따라서 사용자는 Join 요청 메시지를 수신한 다음 1 분 이내에 허락 메시지를 전송해야 한다.

Join 응답에 대해 허락 메시지를 수신한 디바이스는 수신한 정보로 네트워크를 설정하고, 상태를 JOIN 으로, ROLE 을 NODE 로 변경한다. 마지막으로 Join 완료 메시지를 상위 디바이스에게 자동으로 전송하고 Join 절차는 종료된다. Join 완료 메시지는 항상 AP 까지 전송된다.

다음 [Figure 7 Join Sequence Diagram]은 AP 와 Host 가 UART1 을 통해 연결되고, 신규로 Node 를 Elicit network 에 Join 시키는 과정을 나타낸 흐름도이다.

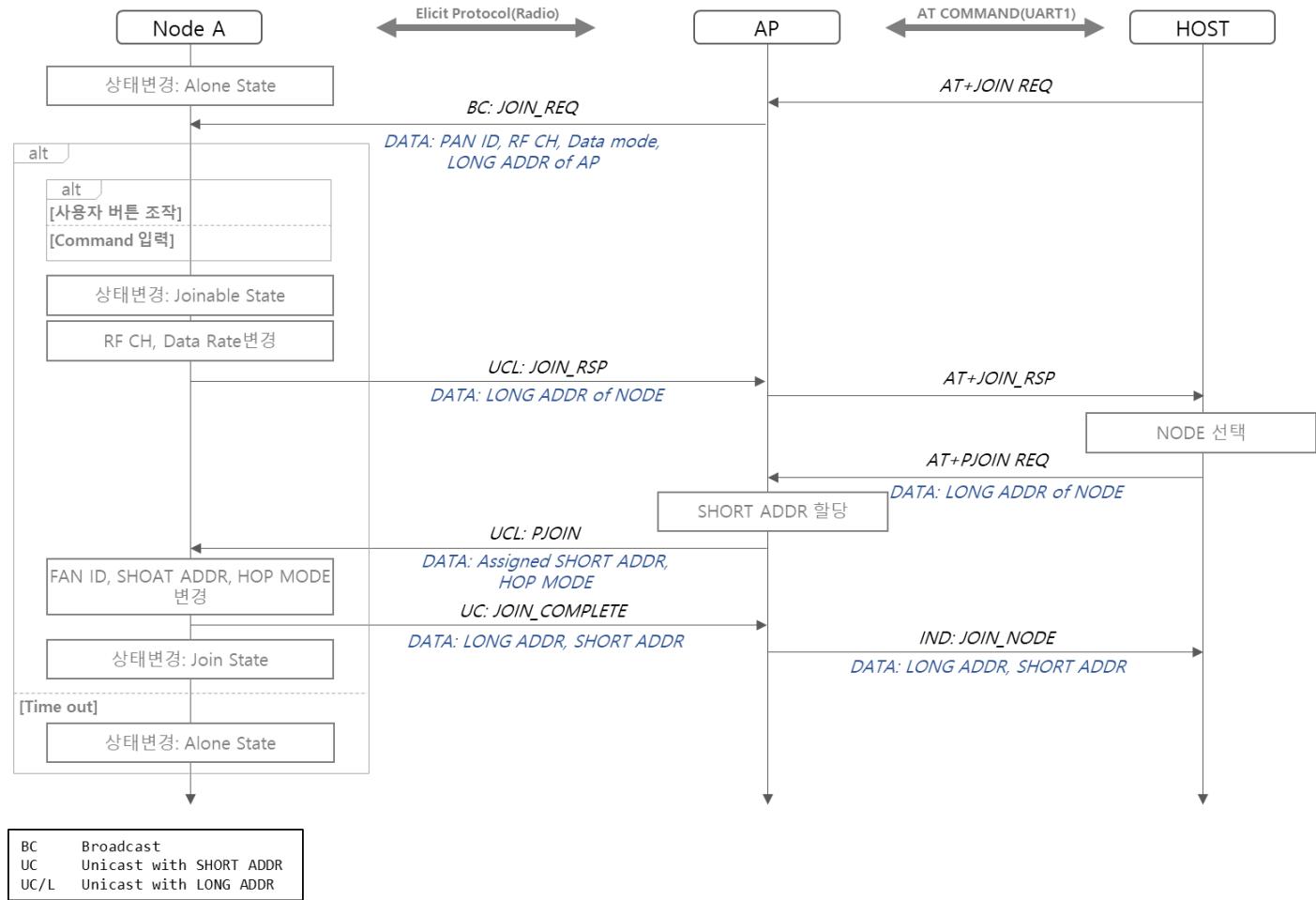


Figure 7 Join Sequence Diagram

Elicit 네트워크를 구성하는 절차의 상세한 순서와 내용을 다음 [표 1 Elicit Network 구성 절차]와 같다.

AT Command 설정과 사용 방법은 [AT command]을 참조

표 1 Elicit Network 구성 절차

단계	내용
1	네트워크를 구성할 ELICIT-R1 혹은 ELICIT-R1+ 디바이스 준비 <ul style="list-style-type: none"> - Elicit 네트워크 구성을 위해 최소 2 개 이상의 디바이스를 준비하고 Host 등과 연결 - AP로 설정할 디바이스는 반드시 UART1을 통한 AT Command 인터페이스를 가지고 있어야 한다 - 각 ELICIT 디바이스를 초기화[11.1 디바이스 초기화]
2	AP 지정 및 네트워크 파라미터 설정 <ul style="list-style-type: none"> - AP로 운영할 ELICIT 디바이스의 UART1과 Host(PC, Gateway 등)를 연결 - Host의 UART 속도와 동일하도록 AP의 Baud rate를 설정[AT+BAUD] - ELICIT-R1/R1+ 모듈을 AP로 설정(초기화 이후 기본 SHORT ADDR은 FFFF) AT+ROLE=FFFF AP - AP로 설정한 후에는 SHORT ADDR이 0000(ZERO)으로 자동 변경됨.
3	AP에서 Join Request(JOINREQ) 메시지를 송출 <ul style="list-style-type: none"> - 모든 channel에 있는 ELICIT R1에게 송신 AT+JOINREQ=0

	<ul style="list-style-type: none"> - 지정된 channel에 있는 ELICIT R1에게 송신(아래 예제는 채널 10으로 송신) AT+JOINREQ=0 10
4	<p>Node에서 JOINREQ 메시지에 대해 응답</p> <ul style="list-style-type: none"> - Node에서 JOINREQ를 수신하면 [Joinable State]로 변경됨 - LED 1, 2가 2초마다 동시에 2회씩 짧게 점멸 - 버튼이나 AT Command 중 하나를 사용하여 응답 - Joinable 상태에서 1분동안 응답하지 않으면 다시 Alone 상태로 변경됨 - 응답 이후 LED 점멸 중지됨 <p>[버튼으로 응답]</p> <ul style="list-style-type: none"> - Host 없이 디바이스를 구성하였을 때 버튼을 사용하여 Join 과정을 수행할 수 있음 - 버튼 1을 누르면, 자동으로 Join 응답 메시지[JOIN_RSP]를 전송 <p>[AT Command로 응답]</p> <ul style="list-style-type: none"> - AT Command [AT+JOINRSP]를 통해 응답 메시지 전송 AT+JOINRSP=FFFF
5	<p>AP에서 JOINRSP 확인 후 해당 Node 선택</p> <ul style="list-style-type: none"> - Node가 전송한 JOINRSP(Node의 LONG ADDR 정보 포함)를 AP에서 확인 - Alone 상태에서 JOINREQ를 수신한 Node는 모두 응답할 수 있으므로 다수의 JOINRSP를 수신할 수 있음 - 수신한 JOINRSP 중에서 등록하고자 하는 Node의 LONG ADDR을 확인 후 등록허가 메시지 전송 [AT+PJOIN 혹은 AT+FJOIN] Ex) LONG ADDR이 0012345600abcd12 일 때 AT+PJOIN=0 0012345600abcd12 - PJOIN은 SHORT ADDR을 AP 내부에서 자동으로 할당한다. - FJOIN은 SHORT ADDR을 사용자가 할당한다.
6	<p>네트워크 구성 확인</p> <ul style="list-style-type: none"> - Node는 PJOIN/FJOIN 메시지 수신 후 자동으로 네트워크 정보를 변경하고, JOIN 상태로 변경 - Node는 자동으로 네트워크 등록 완료 메시지인 JOIN CONFIRM을 AP로 전송 - AP는 IND 형식으로 JOIN CONFIRM 수신 AT+IND=[ADDR] [RSSI] [BATT] JOIN [NEW JOINED ADDR] [NEW JOINED LONG ADDR] Ex) JOIN 한 Node의 SHORT ADDR이 0001이고, LONG ADDR이 0012345600abcd12 일 때 AT+IND=0001 -60 33 JOIN 0001 0012345600abcd12 OK AP에 연결된 Host는 JOIN CONFIRM IND 메시지를 이용하여 신규 Node를 관리할 수 있다

AT Command

CMD(Ref.)	Description	Set/Get	Parameters	Default/Initial
AT+JOINREQ	Request to Join	Set	[channel: 1 ~ 21]	
AT+JOINRSP	Response to Join	Set	[SHORT ADDR]	
AT+PJOIN	Join 허락(자동 주소할당)	Set		
AT+FJOIN	Join 허락(사용자 주소할당)			
AT+IND	Indication of Event	Ind	[EVENT TYPE]	
AT+ASADDR	Count of Assigned Addr	Set/Get	[ASSIGNED ADDR COUNT]	0

3.5.1 주소(Address)

PAN에서 다른 디바이스로 메시지를 전달할 때 상대방의 주소를 통해 개별적으로 직접 전달(unicast)할 수도 있고, 다수의 디바이스에게 메시지를 동시에 전달(broadcast)할 수도 있다.

PAN이 구성될 때 각 디바이스에게 부여되는 주소는 16bit의 Short Address이다. 이 주소를 통해 PAN 내부에서 서로를 구분하고, 메시지를 전달할 수 있다. 따라서 PAN 안에서 Short Address는 중복되지 않고 유일하게 할당되어야 한다. [\[AT+FJOIN\]](#)을 사용할 때는 사용자가 직접 주소를 할당하기 때문에 주소가 중복되지 않도록 각별히 주의해야 한다.

[\[AT+PJOIN\]](#)은 상위 디바이스에서 하위 레벨로 Join하는 디바이스의 SHORT ADDR을 자동으로 할당한다. 자동으로 할당하는 주소의 관리는 [\[AT+ASADDR\]](#)로 할 수 있다. 이 값은 초기값이 0이며, 신규 할당할 때마다 1씩 자동으로 증가시킨다. ASADDR의 값은 사용자가 직접 변경 가능하고, 이를 이용하여 할당하려는 주소를 조정할 수 있는데, 주소가 겹치지 않도록 각별히 주의하여야 한다.

LBT 모드는 Tree 구조 네트워크를 구성한다. 네트워크 구성을 하면서 주소를 할당하기 때문에, Short address를 통해 Tree 네트워크에서 디바이스의 위치를 알 수 있다. 주소를 알면 Tree의 가지(branch), Hop level을 할 수 있다.

모든 ELCIT 디바이스는 각각 고유한 ID(Serial number)를 가지고 있다. 이 고유 ID를 이용해서 서로 통신이 가능한데, 이렇게 고유 ID를 주소처럼 사용할 수 있기 때문에 고유 ID를 Long Address라고 한다.

Long Address를 사용한 통신은 네트워크를 구성하지 않고도 할 수 있다. 이 때는 사용자 메시지(User data)만 전달할 수만 있다. 즉 [\[AT+SEND\]](#) 명령만 사용 가능하고, 다른 AT 명령은 Long address를 이용해 전달할 수 없다.

또한, PAN을 구성할 때, 상위 디바이스는 Long address를 사용하여 디바이스를 구분하고 Join을 허락하기 때문에, Long address는 외부 노출이 되지 않도록 주의해야 한다.

디바이스 ID, Long address는 명령[\[AT+SN\]](#)을 통해 확인할 수 있다.

4 LINK MODE

LINK mode[AT+LINKMODE]는 무선 네트워크 접속 방식을 설정하는 mode로 ELICIT 네트워크를 환경에 따라 적합한 mode를 선택하여 효율적인 무선 통신을 구현할 수 있다. LINK MODE 종류는 LBT mode와 ECH mode 2 가지가 있다. ELICIT-R1은 LBT mode를 고정으로 사용하고, ELICIT-R1+는 LBT mode와 ECH mode를 선택하여 사용할 수 있다. (Default LINK MODE는 LBT mode이다)

4.1 LBT

4.1.1 Tree Topology

Elicit protocol의 LBT mode는 AP(Access point)를 중심으로 Node들이 나무가지 형식으로 연결된 Tree 구조(Tree topology)를 기본으로 개발되었다. 계층적으로 상호 연결된 각 디바이스(AP와 Node)들은 네트워크 내에서 서로 구분할 수 있는 주소(short address, SHORT ADDR)를 할당 받고, 이를 통해 정보를 교환한다.

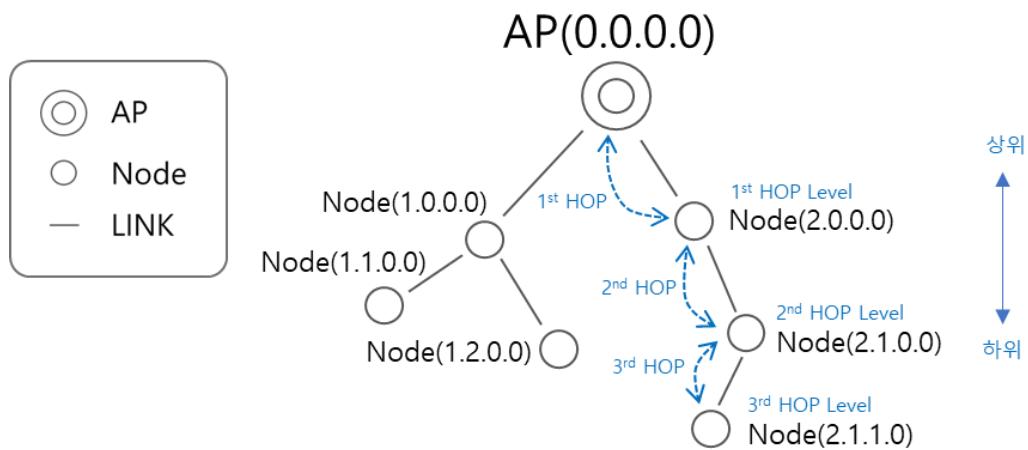


Figure 8 Elicit Tree Topology

최종 목적지까지 정보를 전송하는 경로는 연결된 가지를 통해서 이루어 진다. AP는 모든 가지의 중심으로 네트워크 내의 모든 Node와 통신이 가능하다. 위 [Figure 8 Elicit Tree Topology]에서 AP는 직접 연결된 Node(1.0.0.0), Node(2.0.0.0)와 통신이 가능하고, Node(1.0.0.0)은 AP, Node(1.1.0.0), Node(1.2.0.0)과 통신이 가능하다. 직접 연결되지 않은 AP와 Node(2.1.0.0) 사이의 통신은 중간의 Node(2.0.0.0)을 경유하여 이루어 진다. 이렇게 정보를 직접 전달하지 않고 다른 Node를 경유하는 단위를 Hop이라고 한다.

AP는 Hop level 0이며, AP와 첫 번째 Hop으로 연결된 Node들을 1st Hop level이라고 하고, 다시 1st Hop level에서 연결된 두 번째 Hop으로 연결된 Node들을 2nd Hop level이라고 한다. Elicit에서는 최대 4th Hop level 까지 지원한다.

Tree 네트워크 상의 임의의 Node를 기준으로 AP 쪽으로 연결된 Node를 상위 Node라고 하고, AP에서 Hop 단계를 더 많이 거치는 Node를 하위 노드라고 한다. 모든 Node들의 최상위 Node는 AP이다. Tree 구조에서 AP와 Node는 자신을 중심으로 형성된 가지의 모든 하위 Node들과 통신을 할 수 있으며, Node는 자신의 상위 가지에 있는 Node 및 AP와 통신할 수 있다.

Elicit 프로토콜 스택의 LBT mode 구조

- 최대 4 단계의 Hop 을 지원하는 Tree 구조의 프로토콜
- AP 는 모든 Node 의 정보를 수집할 수 있음.
- Node 는 자신이 속한 가지의 모든 Node 와 연결 가능
- 다른 가지에 있는 Node 로의 전송은 동일 레벨까지 Hopping 한 후 전달
- AP 와 Node 의 하드웨어 차이는 없으며, 사용자 설정으로 결정

4.1.2 Tree Routing

Elicit 은 고정된 주소 길이(16bit)를 Hopping mode 마다 할당할 수 있는 주소 범위를 구분하여 최대 4 Hop 까지 지원되는 4 가지의 Hopping mode 를 제공한다. 이를 통해 다양한 IoT 서비스 환경에 유연하게 적용할 수 있고, 간단한 구조를 가지고 있어 주소 자체가 Tree network 에서 Routing table 의 역할을 한다.

SHORT ADDR 를 표기할 때 Hop level 마다 점(dot)을 찍어 level 을 구분한다. 단일 Hop level 만 있는 Hopping mode1 은 점이 없이 (0001)와 같이 표기하고, 2Hop 을 지원하는 Hopping mode2 의 경우 (01.00), Hopping mode3 은 (01.0.0), Hopping mode4 는 (1.0.0.0)와 같은 형식으로 표기한다.

SHORT ADDR 을 할당할 때 상위 level 주소와 자기 주소는 유지하고, 하위 주소만을 신규로 할당한다. 이렇게 주소를 할당하기 때문에 주소만 보면 해당 주소가 Tree 구조에서 어디에 위치한 것인지 쉽게 파악이 가능하다.

Elicit protocol 에서는 동일한 Level 의 Node 들은 서로 통신이 가능한 범위에 위치한다고 가정하고 메시지를 송신한다. 다른 가지에 있는 Node 에게 정보를 전달할 때는 동일 Hop level 일 경우에는 직접 전달하고, 자신보다 상위 level 이면 자신의 가지에서 목적지 Node 의 level 과 동일한 level 까지 경유(hopping)한 후에 전달한다. 그리고 자신보다 하위 level 이면 동일한 level 의 Node 에게 먼저 전달하고 목적지 Node 에게 전달한다.

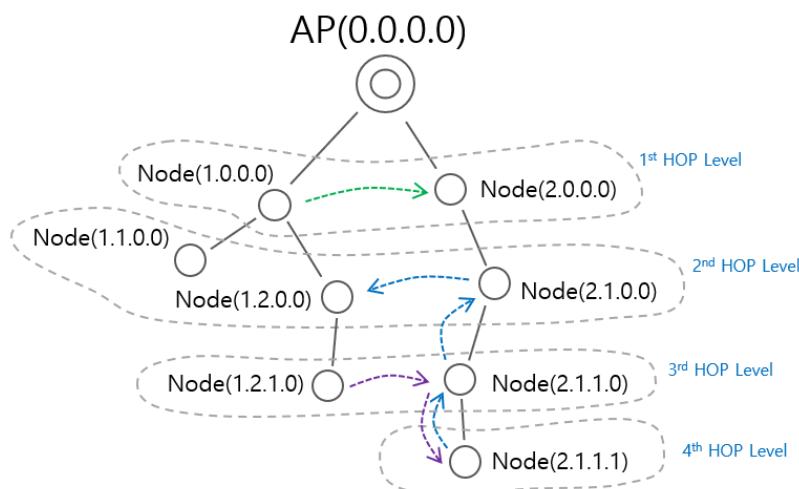


Figure 9 Packet Hopping

[Figure 9 Packet Hopping]에서 왼쪽 가지의 Node(1.0.0.0)에서 동일한 level 인 오른쪽 가지에 있는 Node(2.0.0.0)으로 메시지의 전달은 AP 를 경유하지 않고 직접 전달한다(녹색 화살표). 오른쪽 가지의 최하위 Node(2.1.1.1)에서 왼쪽 가지의 2 번째 level 에 위치한 Node(1.2.0.0)으로 메시지를 보낼 때, 우선 자신의 가지에서 상대방 Node 의 level 까지 hopping 한 후에 전달(파란색 화살표)한다. 왼쪽 가지의 3 번째 level 에 있는

Node(1.2.1.0)에서 오른쪽 가지의 4 번째 level 의 Node(2.1.1.1)로 메시지를 보낼 때는, 자신과 동일한 level 의 Node(2.1.1.0)에게 먼저 보내고 동일한 가지인 4 번째 level 의 Node(2.1.1.1)에게 Hopping 으로 전달한다. (보라색 화살표).

4.1.3 Broadcast

개별 Device 에 부여된 주소를 통해 특정한 단일 Device 와 통신하는 것을 Unicast 라고 한다. 동시에 다수의 Device 에게 전송하는 것을 Multicast 라고 하고, 모든 대상을 상대로 전송하는 것을 Broadcast 라고 한다. Elicit 에서 할당된 SHORT ADDR 로 전송하면 Unicast 로 동작하고, 자신의 하위 level 의 주소 범위를 F 로 채우면 해당 level 에 있는 모든 Node 에게 Broadcast(이하 BC)한다. Node(01.00)에서 (01.FF)로 전송하면 Node(01.00)하위의 모든 Node 에게 전달된다.

Broadcast 메시지는 다수의 Device 가 동시에 수신하므로, BC 메시지를 수신한 Device 는 ACK 를 송신하지 않는다.

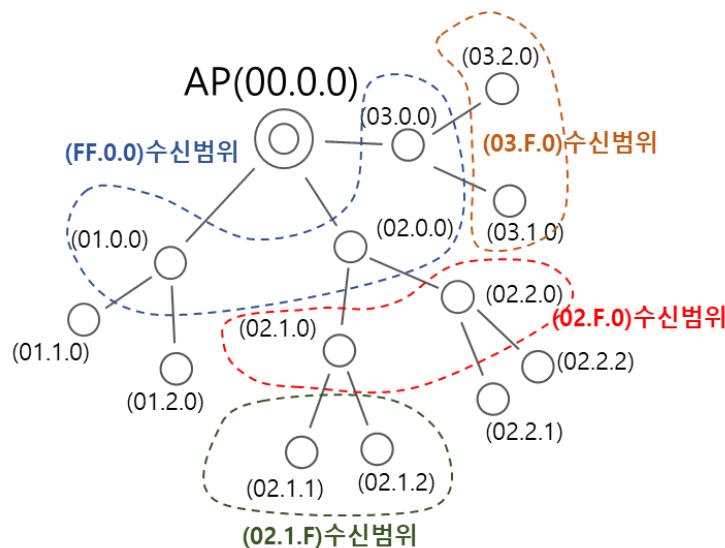
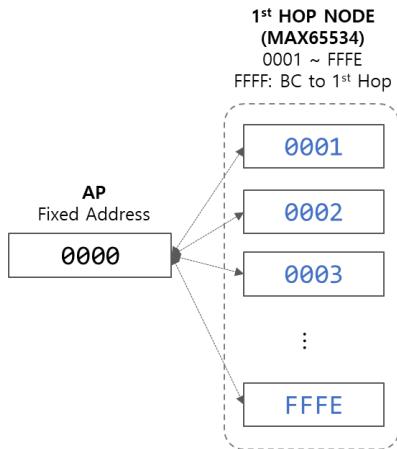


Figure 10 Broadcast 수신 범위

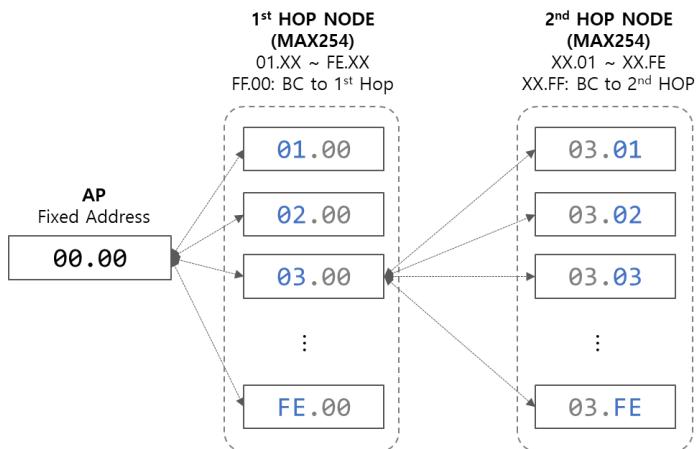
4.1.4 Hopping Mode 1

- AP 를 중심으로 모든 Node 가 연결된 Star 구조, 1Hop 만 지원.
- AP 에서 모든 Node 의 SHORT ADDR 을 할당
- 최대 65534 개의 Address 할당 가능
- Broadcast(BC) address 는 (FFFF)



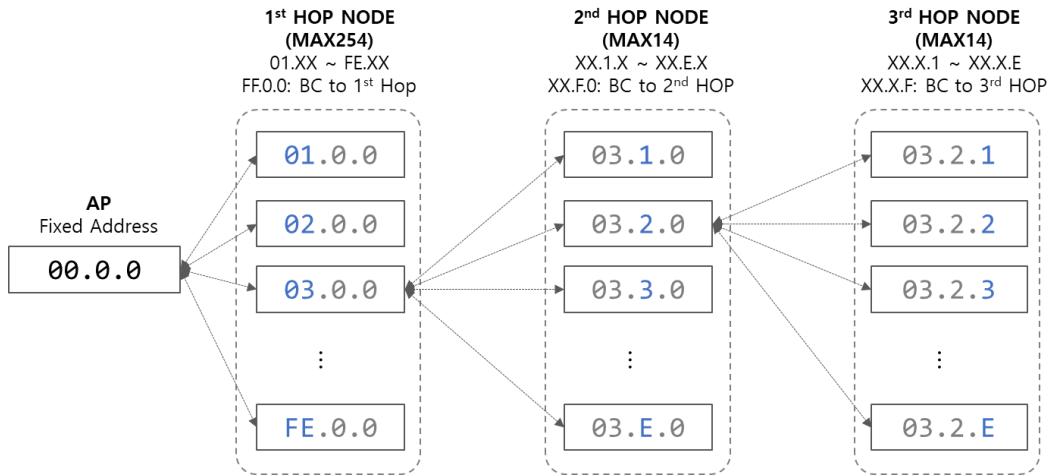
4.1.5 Hopping Mode 2

- Tree topology 2Hop 지원, 각 Hop 당 최대 254 개 주소 할당 가능
 - 1st Hop 주소 범위: (01.00) ~ (FE.00), 1st Hop 에 대한 Broadcast 주소: (FF.00)
 - 2nd Hop 주소 범위: (XX.01) ~ (XX.FE), 2nd Hop 에 대한 Broadcast 주소: (XX.FF)
 - 1st Hop Broadcast ADDR: (FF.00)
 - Broadcast ADDR to 2nd Hop Nodes: (XX.FF)
- 주) XX: 임의의 1Hop 주소



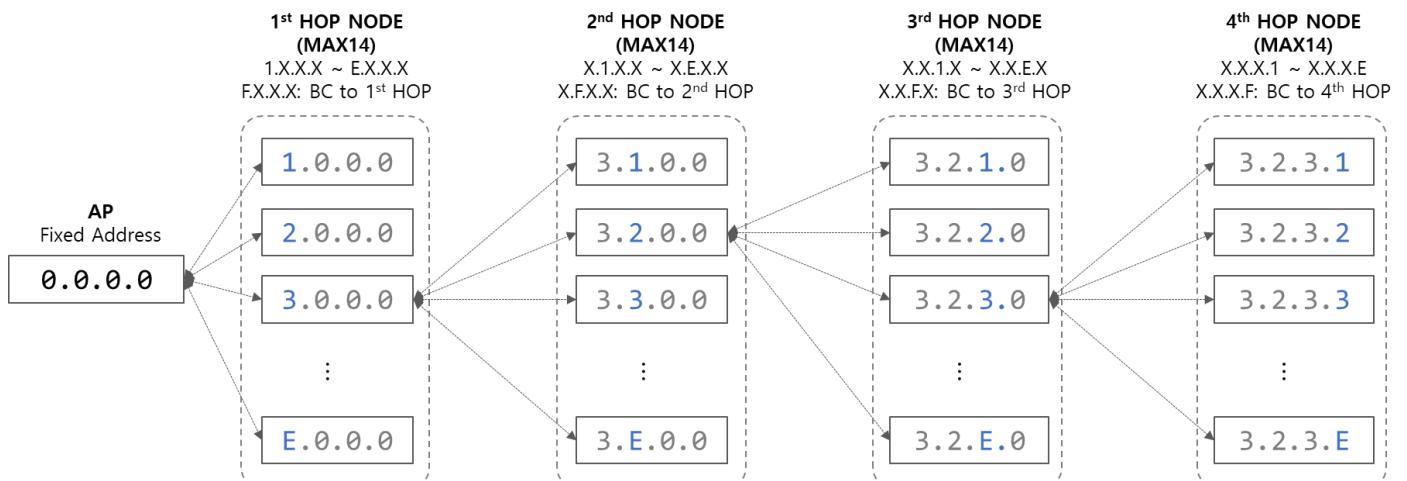
4.1.6 Hopping Mode 3

- Tree topology 3Hop 지원, 1st Hop 254 개, 2nd 와 3rd Hop 은 각각 14 개씩 주소 할당 가능
- 1st Hop 주소 범위: (01.0.0) ~ (FE.0.0), 1st Hop 에 대한 Broadcast 주소: (FF.00)
- 2nd Hop 주소 범위: (XX.1.0) ~ (XX.E.0), 2nd Hop 에 대한 Broadcast 주소: (XX.F.0)
- 3rd Hop 주소 범위: (XX.X.1) ~ (XX.X.E), 3rd Hop 에 대한 Broadcast 주소: (XX.X.F)



4.1.7 Hopping Mode 4

- Tree topology 4Hop 지원, 모든 Hop 은 각각 14 개씩 주소 할당 가능
- 1st Hop 주소 범위: (1.0.0.0) ~ (E.0.0.0), 1st Hop 에 대한 Broadcast 주소: (F.0.0.0)
- 2nd Hop 주소 범위: (X.1.0.0) ~ (X.E.0.0), 2nd Hop 에 대한 Broadcast 주소: (X.F.0.0)
- 3rd Hop 주소 범위: (X.X.1.0) ~ (X.X.E.0), 3rd Hop 에 대한 Broadcast 주소: (X.X.F.0)
- 4th Hop 주소 범위: (X.X.X.1) ~ (X.X.X.E), 4th Hop 에 대한 Broadcast 주소: (X.X.X.F.0)



4.2 ECH

ECH(ELICIT Channel Hopping)는 **TSCH(Time slotted channel hopping)** 기술을 기반으로 산업용 IoT에 최적화한 stack이다. ECH은 TSCH의 **시간 슬롯 기반 통신과 채널 호핑 기능**을 적용하여 안정적인 통신 성능과 신뢰성을 보장한다. 이를 통해 간섭이 많은 복잡한 산업 환경에서도 안정적인 네트워크 연결을 제공할 수 있다.

ECH은 표준 TSCH에서 불필요한 요소를 제거하고 기능을 단순화하여, 더 효율적이고 경량화된 통신 프로토콜이다. 특히, 기존 엘리엇시스템의 ELICIT 프로토콜과 완벽한 호환성을 제공해, 기존 ELICIT 무선 인프라에 손쉽게 통합할 수 있다.

ECH은 저전력, 장거리 통신, 시간 동기화 기반의 고신뢰성 통신, 네트워크 확장성과 같은 TSCH의 핵심 장점을 그대로 유지하면서도, 산업 IoT 환경에 맞춘 효율성을 극대화한 기술이다.

ECH mode는 **ELICIT-R1+만 설정할 수 있고 ELICIT-R1은 지원하지 않는다.**

ECH mode는 **Multi-hop**을 지원하지 않고, AP를 중심으로 연결된 **Star 구조**의 네트워크 구조이다. 하지만 각 노드는 각자의 통신 범위 안에서 서로의 주소를 통해 직접 통신할 수 있다(peer to peer).

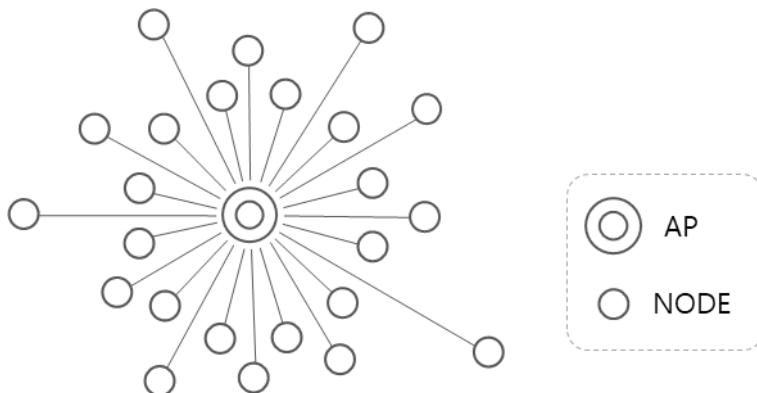


Figure 11 ECH Network

4.2.1 ECH 특징

■ 시간 슬롯 기반 통신

ECH은 시간 슬롯(Time Slot)을 기반으로 각 노드가 언제 데이터를 송수신할지 미리 예약된 시간에 맞춰 동작한다. 이를 통해 충돌 없는 통신을 보장하며, 네트워크 전반의 효율성을 높이기 때문에 **초고밀도 무선 네트워크**를 운영할 수 있다. 1분에 1회 64Byte의 데이터를 전송하는 센서 네트워크의 경우, 동일 네트워크에서 최대 약 1500개 이상의 노드를 운영할 수 있다.

■ 채널 호핑(Channel Hopping)

ECH은 일정한 주기로 사용 채널을 변경하는 채널 호핑 메커니즘을 사용한다. 이로 인해 간섭(Interference)이나 다중 경로 페이딩(Multipath Fading) 등의 문제를 회피할 수 있으며, 네트워크 신뢰성이 향상된다. 특히, 잡음이 많은 산업 환경에서 유리한 통신 방식이다.

■ 고신뢰성 통신

ECH 는 네트워크내의 모든 디바이스들이 시간 동기화를 유지하면서 데이터 전송을 수행하기 때문에, 시간에 민감한 애플리케이션에서도 신뢰할 수 있는 통신을 제공한다. 데이터 전송의 일정성과 예측 가능성을 보장해 패킷 손실을 줄일 수 있다. 이와 함께 특정 채널의 환경이 좋지 않을 경우, 자동으로 해당 채널을 회피하는 기능(Adaptive channel hopping)도 적용하였다. 이 기능을 통해 안정적으로 신뢰성 높은 무선 네트워크를 유지할 수 있다.

■ 저전력 운영

ECH 는 LPWA 모뎀의 에너지 효율성을 극대화할 수 있는 스택이다. 각 노드는 사전에 정해진 시간 슬롯에서만 동작하고 나머지 시간에는 절전 모드로 전환될 수 있기 때문에, 더 오랜 시간 동안 배터리로 운영할 수 있다. 이는 장거리 통신과 저전력이 중요한 IoT 애플리케이션에 매우 필요한 기술이다.

■ 확장성과 유연성

ECH 는 네트워크 규모에 따라 쉽게 확장할 수 있다. 유연한 시간 슬롯과 채널 호핑 옵션을 활용해 많은 수의 노드를 효과적으로 관리할 수 있어 대규모 IoT 네트워크에서 뛰어난 성능을 발휘할 수 있다. 또한 다양한 산업 애플리케이션에 유연하게 적용할 수 있다.

■ 보안성 강화

주기적인 채널 변경과 시간 동기화로 인해 매우 높은 보안성을 제공한다. 외부에서 네트워크를 감시하거나 방해하기 어렵기 때문에, 중요한 데이터를 다루는 산업 환경에서 안전한 통신을 보장할 수 있다.

4.2.2 Channel hopping list

AP 에서 Link mode 로 ECH 를 설정[AT+LINKMODE]할 때 18 개의 채널들을 임의의 순서로 조합하여 channel hopping list 를 생성한다. Hopping 순서를 확인하거나 다시 조합할 때는 [\[AT+ECHLIST\]](#) 명령을 사용한다.

18 개의 채널은 15 번 채널을 제외한 2 ~ 20 번 채널이다.

4.2.3 ACH(Adaptive channel hopping)

ECH 를 사용하면 18 개의 채널을 임의의 순서로 hopping 하며 사용한다. ACH(적응형 채널 호핑) 기능은 각 채널 별 상태를 검사하여 혼잡할 경우(누군가 채널을 점유하는 경우)에는 해당 채널을 사용하지 않는 기능이다. 이렇게 사용하지 않도록 설정한 채널의 상태가 좋아지면 다시 자동으로 사용 가능하도록 설정한다. ACH 기능은 [AT+ACH] 명령으로 설정할 수 있다.

현재의 Hopping 순서 리스트는 [\[AT+ECHLIST\]](#) 명령을 사용하여 확인할 수 있는데, ACH 기능으로 채널이 사용 불가능인 경우에는 채널번호에 100 을 더한 값으로 표기된다. 즉 7 번 채널이 혼잡하여 사용 불가능인 경우, 107 번으로 표시된다.

4.2.4 ECH 구조

ECH 모드에서 네트워크의 구조는 다음과 같다.

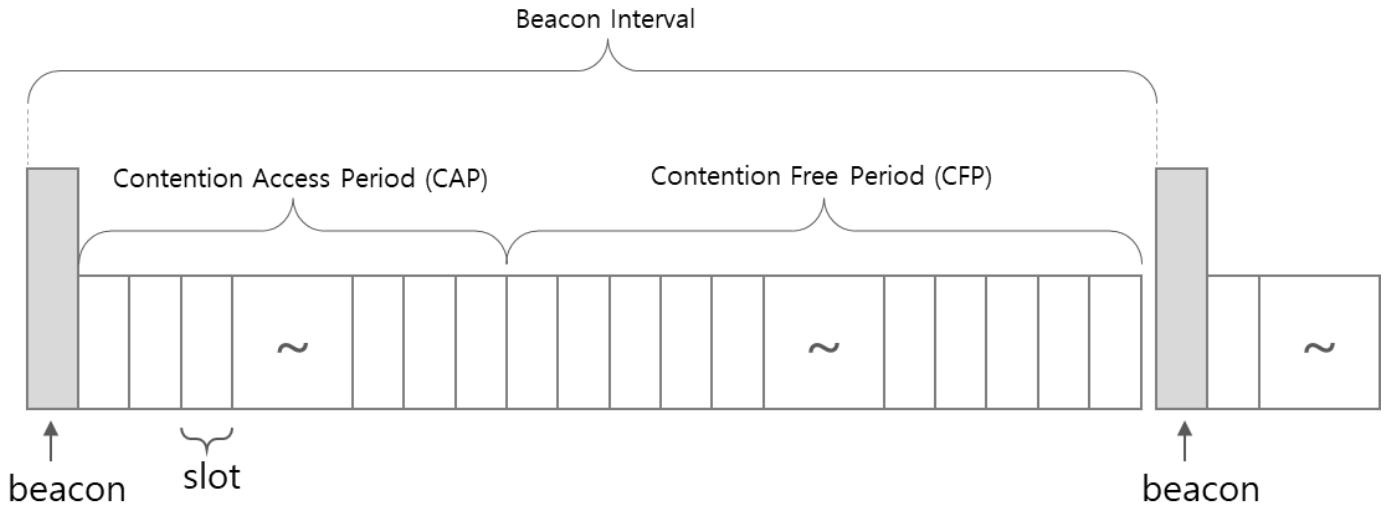


Figure 12 ECH 구조

ECH는 기본적으로 5ms 단위로 구분된 slot으로 구성된다. AP가 송신하는 beacon이 하나의 slot을 차지하며, 주기적으로 송신한다. Beacon이 송신되는 주기(Beacon interval)안에서 CAP(Contention Access Period) 구간과 CFP(Contention Free Period) 구간으로 구성된다.

4.2.5 비콘(Beacon)

ECH mode가 설정된 네트워크의 모든 노드들은 AP가 송신하는 비콘(beacon) 메시지를 수신하여 시간 동기화를 이룬다. AP는 일정한 주기마다 broadcast 형식으로 비콘을 송신한다. 비콘 메시지는 Beacon interval, Hopping channel, 노드 별 할당된 Slot, Node 호출(Pending address) 등의 정보로 구성된다.

비콘 송신 주기는 beacon interval로 정해진다. Beacon interval은 최소 100ms에서 최대 400ms 까지 설정[AT+BCN]할 수 있다. Beacon 송수신 주기가 짧아지면 운영할 수 있는 node의 수는 적어지지만, data를 송수신하는 기회가 빨리 돌아오기 때문에 data 송수신의 반응이 빨라진다. beacon 송수신 주기가 길어지면 송수신의 반응은 비교적 느려지지만, 많은 node를 운영할 수 있다.

Beacon interval mode list

- Mode 1 : 100ms
- Mode 2 : 200ms(Default)
- Mode 4 : 400ms

비콘 메시지를 받지 못한 노드는 [AT+IND] 메시지로 beacon time out을 알린다. ECH 네트워크에서 동기화된 노드가 다음 비콘을 수신하지 못할 경우, 다음 순서의 channel로 이동하여 비콘을 수신하기 위해 대기한다. 이후에도 비콘을 수신하지 못하면 채널 이동 없이 18 beacon interval 기간 동안 대기하며 beacon을 기다린다.

4.2.6 Slot 할당

ECH 모드에서는 네트워크의 모든 디바이스는 AP로부터 slot을 할당 받아야 자신의 메시지를 송신할 수 있다. 노드는 slot을 할당 받기 위해서 먼저 비콘을 통해 네트워크와 시간 동기화를 하고, CAP 구간안에서 임의의 시간에 CSMA/CA 방식을 사용해서 필요한 slot 만큼 AP로 할당 요청 메시지를 전송한다. AP는 이 할당 요청을 받고 다음 비콘 주기에 slot을 할당할 수 있으면, 다음 비콘 메시지에 해당 노드가 할당된 정보를 포함한다.

노드는 다음 비콘 메시지에서 자신에게 slot 이 할당되었는지를 확인한 후, 지정된 slot 에서 메시지를 송신한다. 만약 slot 을 할당 받지 못하면 자동으로 다음 CAP 구간에서 다시 slot 할당 요청을 보낸다.

ECH 에서 이런 할당 요청과 slot 할당의 과정은 모두 프로토콜 내부에서 자동으로 수행되는 과정이다. 따라서 사용자는 메시지 전송 명령을 보낸 후, 성공/실패만 확인하면 된다.

4.2.7 노드 호출(Pending address)

배터리로 동작하는 센서들은 전력 소모를 줄이기 위해 평상시에는 잠들어 있다가 필요할 때만 깨어나서 동작하도록 설정하게 된다. 이런 센서 디바이스들에게 명령이나 데이터를 전달하려고 할 때, 잠들어 있는 경우가 대부분이라 메시지를 전달하기 어렵다. 노드 호출은 이런 디바이스에게 메시지를 전달하기 위한 기능으로 AP 에서 수행한다.

노드 호출은 AP 가 전달하려는 명령과 해당 노드의 주소를 [AT+PEND] 명령을 사용하여 수행한다. PEND 명령을 통해 입력된 노드의 주소와 명령은 AP 내부에 저장되어, 비콘 메시지에 해당 노드의 주소를 포함하는데, 이를 Pending address 라고 한다.

잠들어 있던 노드가 깨어나서 수신한 비콘 메시지의 Pending address 에 자신의 주소가 포함되어 있으면 노드는 slot 할당을 요청하고, AP 는 해당 노드에게 slot 을 할당하고 메시지를 전송한다. Pending address 에 있던 노드와 통신이 이루어지면 AP 는 자동으로 Pending address list 에서 해당 노드의 주소와 명령을 삭제한다.

AP 에서 Pending address 는 최대 5 개까지 입력할 수 있다. 5 개 중 하나는 broadcast 전용이며, 이 명령은 노드에게 전달되더라도 자동으로 삭제되지 않는다. Pending address list 에는 동일한 address 를 중복하여 입력할 수 없다. Pending address list 는 reboot 하면 저장된 정보가 사라진다.

Pending address 수행 예제

- sleep mode 를 수행하고 있는 node 0x1 의 ADC command 수행을 명령할 때 :

 - 1) AP 에서 AT+PEND command 를 입력
 - AT+PEND=0 0x1 (ADC?0x1)
 - 2) sleep mode 에서 깨어난 node 0x1 은 beacon 동기화 진행
 - 3) beacon 동기화 완료 후, beacon 의 pending address list 에 0x1 을 확인
 - 4) node 0x1 에서 GTS request 송신, AP 가 저장된 command 를 송신

4.2.8 Join in ECH

ECH 에서 PAN 에 노드를 Join 하는 방법은 LBT 모드와 동일하다. 다만 Join 하려는 Node 는 반드시 초기화된 상태(초기화 level 0 혹은 2)이어야 한다. 초기화는 명령[AT+FINIT]을 통해서 수행할 수 있다.

ECH 에서 join 절차

- join 하려는 디바이스는 사전에 초기화(FINIT level 0/2)해야 한다.
- BCH 와 동일한 방법으로 Join 한다.

4.2.9 ECH 운영 예제

ECH 는 다수의 노드들이 전파 간섭 없이 지정된 시간에 통신을 하기 때문에 다수의 노드들을 동시에 통신할 수 있다. 네트워크가 수용할 수 있는 메시지의 양은 비콘 주기에 따라 조금씩 다른다. 주기가 짧으면 개별 노드의 응답속도가 빠른 반면, 한 주기에 수용할 수 있는 노드의 개수는 줄어든다. 반대로 비콘 주기가 길면 반응 속도는 느리지만 한 주기내에서 수용할 수 있는 노드의 개수는 증가한다. 매우 다음은 ECH 를 이용해서 네트워크를 구성할 때 최대 노드의 수, 혹은 최소 전송 주기를 정리한 표이다.

모든 노드가 각각 지정된 주기마다 1 회씩 전송할 경우, 최대 구성 가능한 노드의 수

64Byte 데이터를 전송하는 디바이스로 ECH 네트워크 구성

비콘주기 \ 전송주기	1 초	10 초	1 분
100ms	25 개	250 개	1500 개
200ms	29 개	291 개	1746 개
400ms	31 개	312 개	1872 개

100 개의 노드가 각각 지정된 크기의 데이터를 1 회씩 전송할 경우, 최소 전송 주기

100 개의 노드로 ECH 네트워크 구성할 때 최소 전송 주기

비콘주기 \ 데이터크기	16Byte	64Byte	128Byte
100ms	3 sec	4 sec	6 sec
200ms	2.8 sec	3.6 sec	4 sec
400ms	2.8 sec	3.2 sec	4 sec

5 Radio 설정

5.1 RF Channel

ELICIT-R1/R1+는 USN 용으로 할당된 940.1 ~ 944.3MHz 대역을 사용한다. 이 대역에서 사용하는 중심주파수는 [표 2 940MHz 대역의 중심주파수(Channel list)] 와 같이 총 21 개의 채널을 사용할 수 있다. 각 채널 당 점유주파수 대역폭은 200kHz 이다. 디바이스간 서로 송수신하기 위해서는 동일한 채널을 사용해야 한다.

표 2 940MHz 대역의 중심주파수(Channel list)

channel	주파수(MHz)	channel	주파수(MHz)	channel	주파수(MHz)
1	940.2	8	941.6	15	943.0
2	940.4	9	941.8	16	943.2
3	940.6	10	942.0	17	943.4
4	940.8	11	942.2	18	943.6
5	941.0	12	942.4	19	943.8
6	941.2	13	942.6	20	944.0
7	941.4	14	942.8	21	944.2

5.2 TX power

ELICIT-R1/R1+의 최대 송신 출력은 약 20dBm(17.7 ~ 21.4dBm)이다. 출력설정은 AT Command 로 0dBm ~ 21.0dBm 까지 0.1dBm 단위로 설정 가능하다. 출력이 높을수록 전력소모도 높다.

5.3 PHY mode

ELICIT-R1 은 두 가지의 PHY 를 지원한다. 명령[AT+PHY]으로 Long range mode 와 High data rate mode 를 설정할 수 있다. ELICIT-R1+는 추가로 Ultra-high data rate mode 까지 설정할 수 있다. 기본값은 High data rate mode 이다.

Long range mode 는 좀 더 높은 수신감도를 위해 RF bit rate 를 20kbps 로 낮추고, FEC(Forward error correction)를 적용하여 안정적으로 통신이 가능하도록 하였다. FEC 는 원데이터보다 많은 정보를 전송(2 배)하기 때문에 실제 사용자의 데이터 전송속도는 10kbps 이다.

High data rate mode 는 200kHz 씩 할당된 RF channel 을 최대한 효율적으로 사용하기 위해 MSK(Minimum shift keying)을 적용하고, 전송속도는 80kbps 이다. ELICIT-R1 을 운영하면서 전력소모가 가장 많을 때가 RF 전송할 때이다. 그래서 전송 속도가 높으면 동일한 데이터라도 더 짧은 시간동안 전송하기 때문에 전력소모가 더 적다.

표 3 PHY mode

PHY mode	Name	Modulation	RF Bit rate	FEC
L	Long range mode	FSK	20kbps	Convolutional code(K=7)
H	High data rate mode	MSK	80kbps	None
U	Ultra-high data rate mode	MSK	120kbps	None

5.4 Channel Scan

ELICIT-R1/R1+은 무선 채널의 상태를 확인할 수 있도록 다음과 같이 RF channel scan 기능을 제공한다.

- 일정 시간(time unit: second) 동안에 무선 채널을 scan 하여 RSSI 값을 read
- Min/Max/Avg RSSI 값을 출력
- Scan 을 진행 중일때에는 data 송수신 제한
- 모든 channel scan 을 실행하면, 각 channel 의 RSSI 값들과 Max RSSI 가 가장 좋은 channel 을 출력

AT Command

CMD	Description	Set/Get	Parameters	Default/Initial
<u>AT+SCAN</u>	Scan RF channels	Set	[channel: 1~21] [period: 1~60]	
<u>AT+TXPWR</u>	Tx Power	Set/Get	[Tx Power in 0.1dBm]	210

6 IO

ELICIT-R1/R1+는 8 개의 GPIO, I2C, ADC, DAC, Button 전용 DI 2 개, LED 전용 DO 2 개와 보조 DI(AUX) 1 개를 가지고 있다. 이를 활용하여 다양한 IoT Device 를 빠르게 개발할 수 있도록 각 IO 마다 사전 정의된 기능들이 탑재되어 있다.

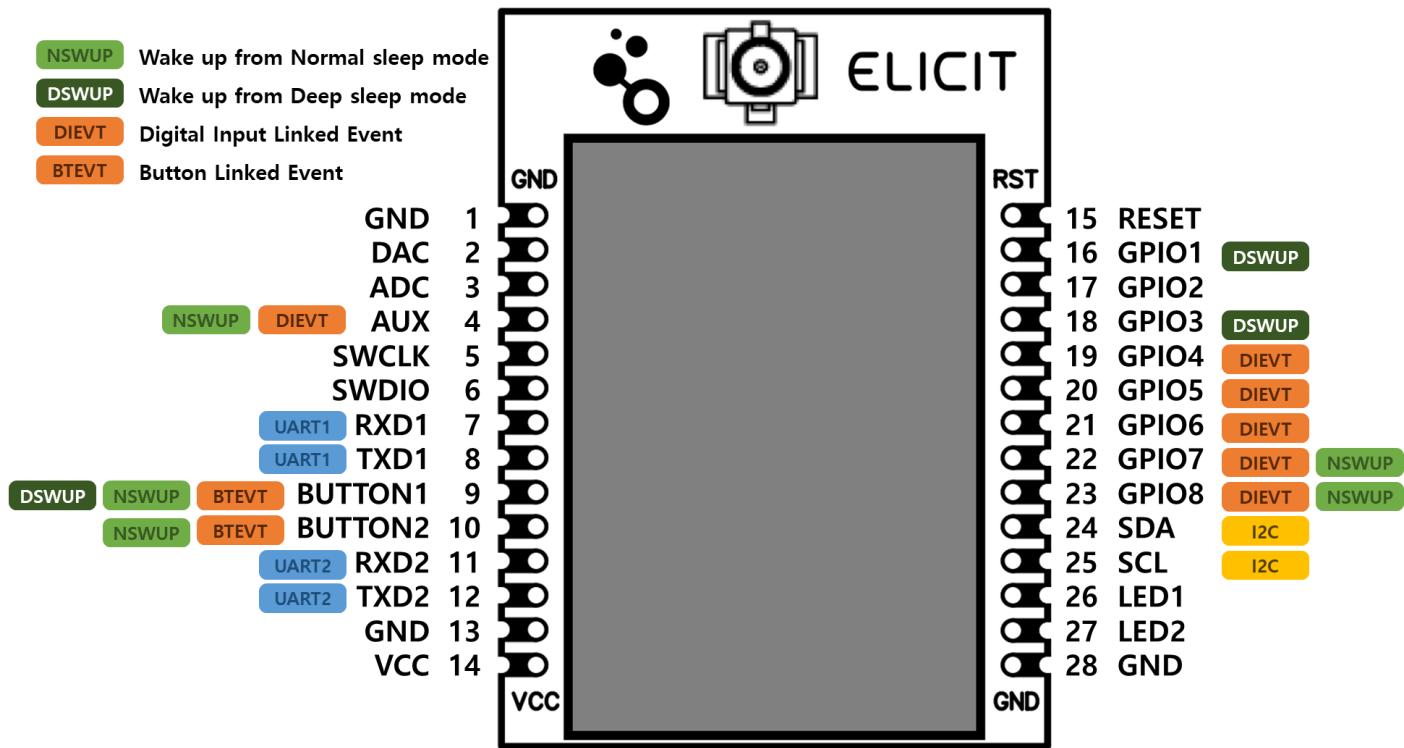


Figure 13 ELICIT-R1 IO

GPIO(Pad 16 ~ 23)는 모두 Input 이나 Output 으로 설정이 가능하다. GPIO4~GPIO8 은 입력 모드(Input mode)에서 입력 상태의 변화(High → Low, Low → High, Both)를 감지하여 이벤트를 발생시킬 수 있도록 설정할 수 있다. AUX(Pad 4)는 입력 전용으로 GPIO 의 Input mode 와 동일하게 사용할 수 있다.

ELICIT-R1+는 AUX 와 GPIO 모두 ADC mode 로 설정하여 ADC 기능을 사용할 수 있다.

BUTTON1,2(Pad 9,10)는 입력 전용 IO 로, 외부에 버튼을 연결하여 다양한 기능을 수행할 수 있다.

LED1,2(Pad 26,27)은 출력 전용 IO 로, 송/수신 상태나 내부 상태를 표시한다. 사용자가 직접 제어할 수는 없다.

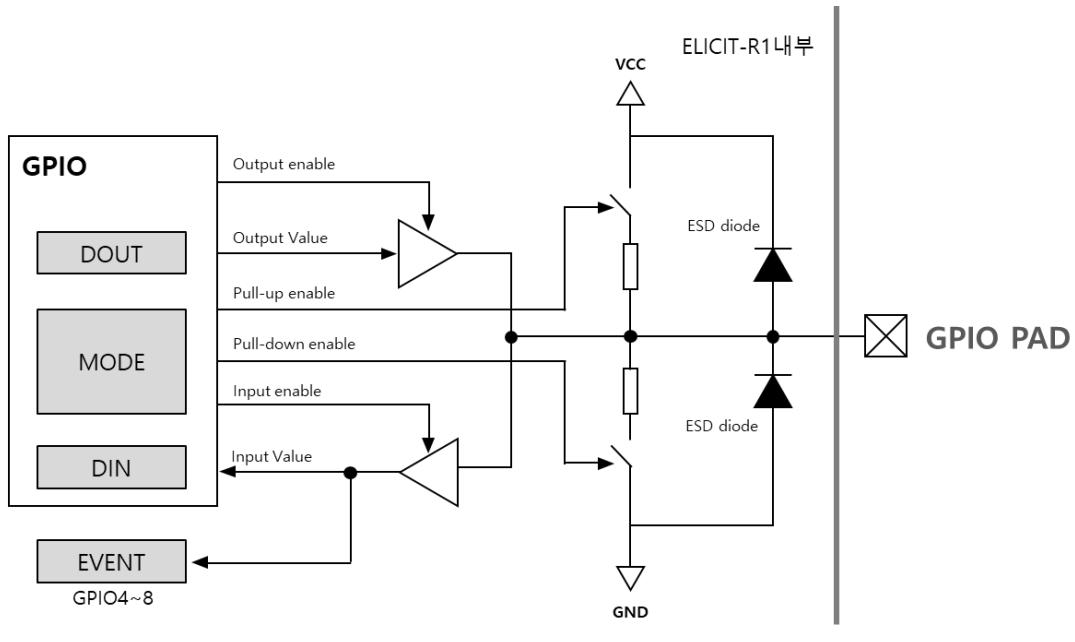
SDA(Pad 24), SCL(Pad 25)는 I2C 전용으로, 내부 $10k\Omega$ 으로 full-up 되어 있다. SWCLK, SWDIO(Pad 5,6)은 ELICIT-R1 의 MCU 디버깅용 Pad 이다. 사용자는 활용 불가능 하다.

RESET(Pad 15)은 최소 100ns 이상 Low($0.3 \times Vcc$) 상태를 유지하면 시스템이 Reset 된다. RESET Pad 는 내부에서 Pull-up 되어 있어서 외부에 별도의 전압을 걸지 않고 사용한다.

[Figure 13]에서 [NSWUP] 과 [DSWUP] 표시는 각각 Normal sleep 이나 Deep sleep 상태에서 외부 입력을 통해 깨어날 수 있는 IO 를 나타낸다. Button 은 사용자가 별도로 설정하지 않아도 누르면 Sleep 상태에서 깨어날 수 있도록 기본 설정되어 있고, 다른 IO 들은 해당 Sleep mode 에서 깨어나기 위해서는 별도로 설정해야 한다.

6.1 GPIO

ELICIT-R1/R1+은 GPIO(General purpose input output) 8 개를 제공한다(Pad 16~23). GPIO 의 내부 구조는 [Figure 14]과 같다. GPIO 의 In/Out, Pull-up/down, Event 등의 설정은 명령[[AT+IO](#)]을 사용하여 설정한다.



Internal Pull-up/Pull-down Resistance: 33 ~ 55kΩ

Figure 14 GPIO 의 내부 구조

6.2 DI

- 모든 GPIO 는 Digital Input 으로 설정 가능
- 각 DI 마다 개별적으로 입력모드 Pull-up, Pull-down, no pull 설정 가능
- 내부 Full up/down 저항: 33~55kΩ
- 명령[[AT+DI](#)]을 이용하여 해당 GPIO 의 DIN 값을 읽을 수 있음
- DIN 값은 logic high(PAD 전압 > 0.7*VCC)일 때 1 이고, logic low(PAD 전압 < 0.3* VCC)일 때 0 이다.
- GPIO 1, 3 은 Deep sleep 상태에서 깨어날 수 있도록 설정할 수 있음
- GPIO 7, 8 은 Normal sleep 상태에서 깨어날 수 있도록 설정할 수 있음
- GPIO 4~8 은 입력 상태의 변화(Positive edge/Negative edge/Both edge)에 대해 Event 설정 가능
- Positive edge event(HIGH)는 DIN 의 값이 logic low 에서 high 로 변할 때 발생.
- Negative edge event(LOW)는 DIN 의 값이 logic high 에서 low 로 변할 때 발생.
- Both edge event(BOTH)는 DIN 의 값이 변할 때 발생한다.
- 설정한 Event 가 발생하면, Event 메시지를 [[AT+IND](#)] 형식으로 출력하여 상태 변화 알림
- DI event 주기는 100ms 이상 설정을 권장
- 명령[[AT+REPORT](#)]를 이용하여 발생한 Event 메시지를 자동으로 AP 에게 전달할 수 있다.
- Event 와 연동하여 사용자가 지정한 AT command 를 수행 가능: Linked Event
- DI event 를 사용할 경우에는, switching noise(chattering)를 방지할 수 있도록 외부 회로를 구성해야 함

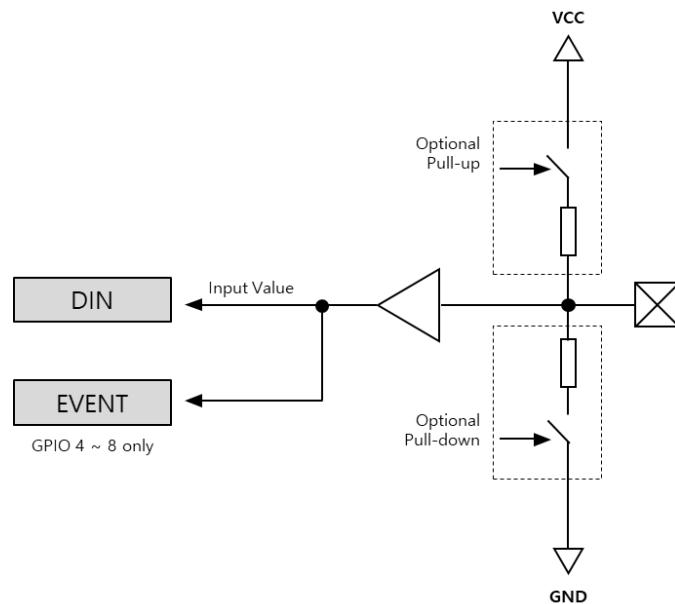


Figure 15 Digital Input 구조

6.3 DO

- 모든 GPIO 는 Digital Output 으로 사용 가능
- 각 DO 마다 개별적으로 출력 모드를 Push-pull, Open-drain, Open-source 중 하나로 설정 가능
- Open-drain 은 옵션으로 Pull up, Open-source 는 옵션으로 Pull down 설정 가능
- 내부 Full up/down 저항: 33~55kΩ
- 명령[AT+DO]을 이용하여 DOUT 값을 읽고, 쓸 수 있음
- GPIO Pad 의 출력은 DOUT 값과 출력 모드에 의해 결정
- 각 DO 마다 개별적으로 DOUT 값을 원하는 주기(time unit: 100ms)로 Toggle 가능

6.3.1 Push-pull out

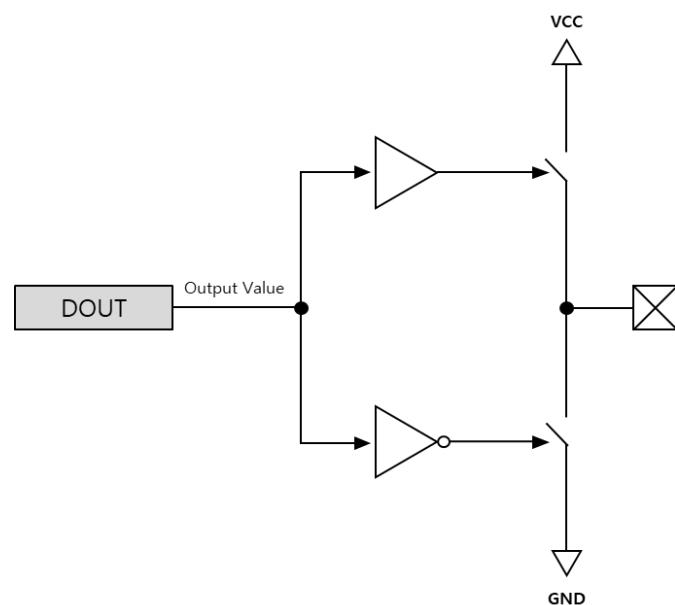


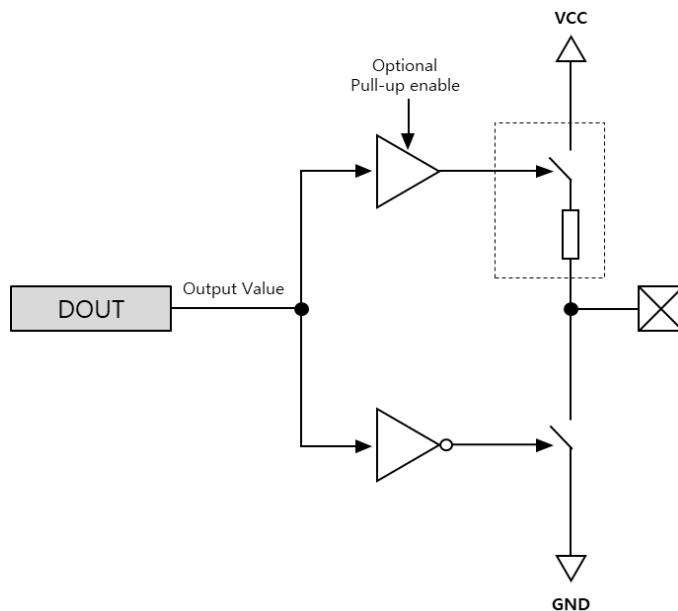
Figure 16 Digital Output(Push-Pull) 구조

- DOUT 값이 1 이면, Pad는 Vcc 와 연결(GND는 끊어짐)
- DOUT 값이 0 이면, Pad는 GND 와 연결(Vcc는 끊어짐)
- Push-pull은 내부 Pull up/down이 없기 때문에, 외부 회로를 구성할 때 단락(Short)되지 않도록 주의
- 단일 GPIO Pad의 최대 출력 전류는 50mA이고, 모든 출력의 합이 200mA를 넘지 않도록 해야함

표 4 Push-pull out table

DOUT	PAD STATUS
0	GND
1	Vcc

6.3.2 Open-drain out

**Figure 17 Digital Output(Open-drain) 구조**

- DOUT 값이 1 이면, Pad는 Floating 상태가 됨
- DOUT 값이 0 이면, Pad는 GND 와 연결
- Open-drain + Pull up 설정에서 DOUT 값이 1 이면, Pad는 Vcc 와 Pull up 연결
- Open-drain + Pull up 설정에서 DOUT 값이 0 이면, Pad는 GND 와 연결(Pull up disabled)

표 5 Open-drain out table

DOUT	Open-drain	Open-drain with pull-up
0	GND	GND
1	FLOAT (Any State)	Internal pull-up

6.3.3 Open-source out

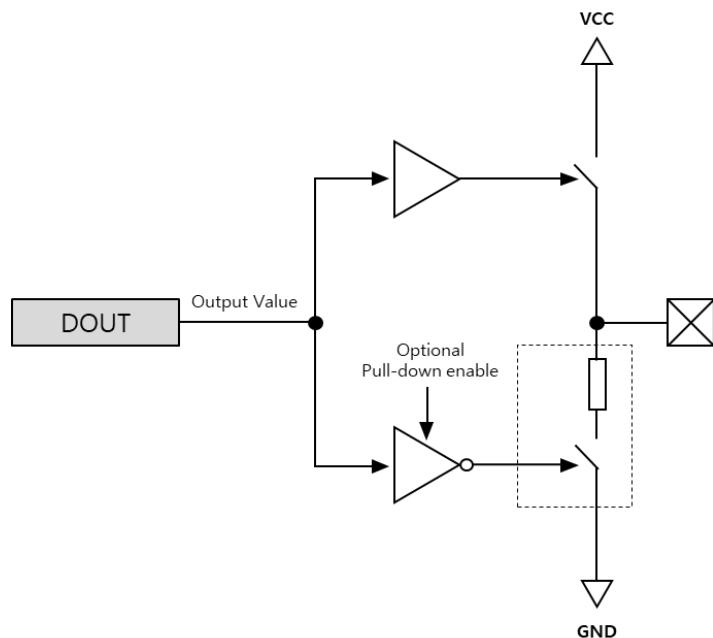


Figure 18 Digital Output(Open-Source) 구조

- DOUT 값이 1 이면, Pad는 Vcc 와 연결
- DOUT 값이 0 이면, Pad는 Floating 상태가 됨
- Open-source + Pull down 설정에서 DOUT 값이 1 이면, Pad는 Vcc 와 연결(Pull-down disabled)
- Open-source + Pull down 설정에서 DOUT 값이 0 이면, Pad는 GND 에 Pull-down 연결

표 6 Open-source out table

DOUT	Open-source	Open-source with pull-down
0	FLOAT (Any State)	Internal pull-down
1	Vcc	Vcc

6.4 AUX Input

- AUX(Pad4)는 보조 디지털 입력(Auxiliary digital input)
- GPIO 의 DI 설정과 동일하게 입력 모드와 Event 설정 가능
- Normal Sleep mode 에서 깨어날 수 있도록 설정 가능
- DI 와 동일하게 Linked event 설정
- DI event 를 사용할 경우에는, switching noise(chattering)를 방지할 수 있도록 외부 회로를 구성해야 함

AT Command

CMD	Description	Parameters	Notes
<u>AT+IO</u>	GPIO Config	<IO> <MODE> <TRIGGER>	AUX 포함
<u>AT+DI</u>	Digital Input	<GPIO>	GPIO 만 읽음
<u>AT+AUX</u>	AUX Input get		AUX 만 읽음
<u>AT+DO</u>	Digital Output	<GPIO> <DOUT> [<DOUT DEFAULT> <TGPERIOD>]	

6.5 RESET

- RESET Pad 를 외부에서 최소 100ns 이상 logic low($0.3 \times V_{CC}$)로 떨어뜨리면 Reset 을 수행한다
- RESET Pad 는 내부에서 Pull-up 되어 있음

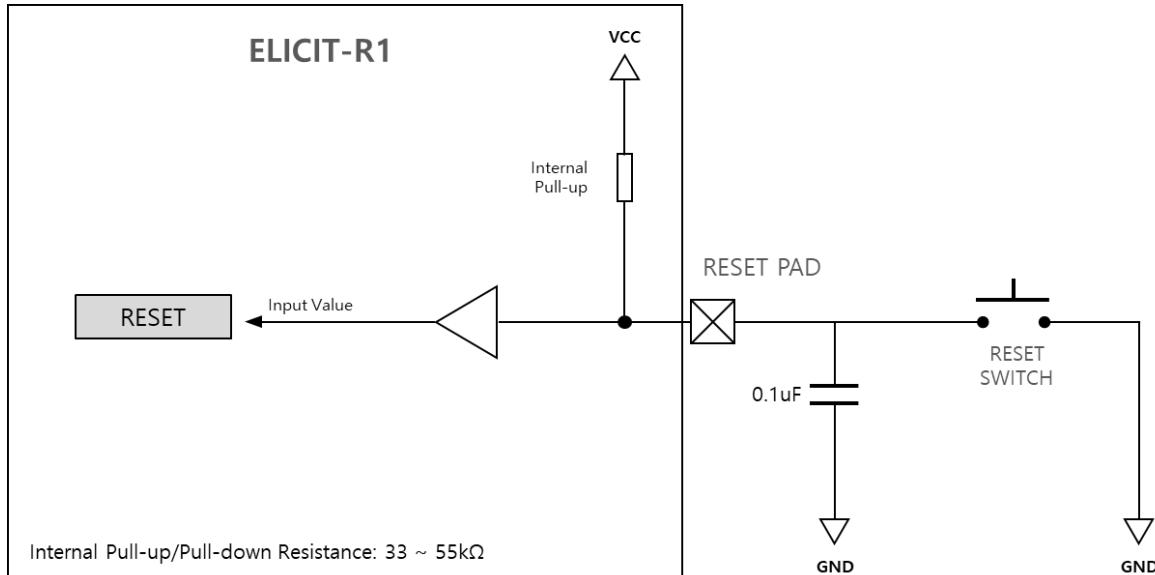


Figure 19 외부 Reset Switch 회로 구성

6.6 ADC

- 12bit ADC, 76.9ksps with x32 oversampling
- ADC(Pad3)와 전원(Vcc, Pad14)의 전압을 측정
- IO(1~8)와 AUX pin 의 mode 를 ADC 로 변경 가능(ELICIT-R1+ 전용)
- 내부의 1.21V 기준 전압을 사용하여 별도의 외부 기준 전압이 필요 없음
- 측정 범위: 0 ~ Vcc(1.8 ~ 3.8V), Vcc 를 넘는 전압은 측정할 수 없음
- Timer event 를 통해 주기적으로 ADC value read 가능
- 명령[[AT+ADC](#)]을 이용하여 ADC 측정을 하고, [AT+IND] 형식으로 결과(단위: mV)를 반환
- 명령[[AT+REPORT](#)]을 이용하여 자동으로 AP 에게도 전달할 수 있다.

6.7 DAC

- DAC(Pad2)는 mV 단위로 설정 가능한 전압 출력 전용 Pad
- 내부의 기준 전압을 사용하여 별도의 외부 기준 전압이 필요 없음
- 설정 범위: 0 ~ Vcc(1.8 ~ 3.8V) → Vcc 를 넘는 전압은 설정할 수 없음
- 명령 [[AT+VDAC](#)]을 이용하여 DAC 출력 값 설정

6.8 Button

- Button1(Pad9)과 Button2(Pad10)은 버튼 전용으로 할당된 IO, 일반 GPIO 로 사용할 수 없음
- Host 연결 없이 버튼으로 Join, 초기화, 사용자 지정 명령을 수행할 수 있음

- 입력모드로 Pull-up, Pull-down 설정 가능(내부 Full up/down 저항: 33~55kΩ)
- 버튼 눌림, 뗄 상태의 입력 값은 Pull-up/down에 따라 다름[표 7 Button 모드 별 입력 값]
- Pull-up/down 설정에 따라 내부에서 자동으로 버튼 눌림, 뗄 상태 변환
- Deep sleep mode에서 Button1을 누르면 깨어남
- Normal sleep mode에서 Button1이나 Button2를 누르면 깨어남
- [중요] 버튼은 Tactile switch(택트 스위치)와 같이 눌렀을 때 접점이 붙는 방식을 사용해야 함
- [중요] 반드시 Chattering 방지 회로(debounce)를 외부에 구성하여 사용해야 함[Figure 20][Figure 21]

표 7 Button 모드 별 입력 값

BUTTON INPUT VALUE	Pull-up	Pull-down
누름 (Press)	0	1
뗄 (Release)	1	0

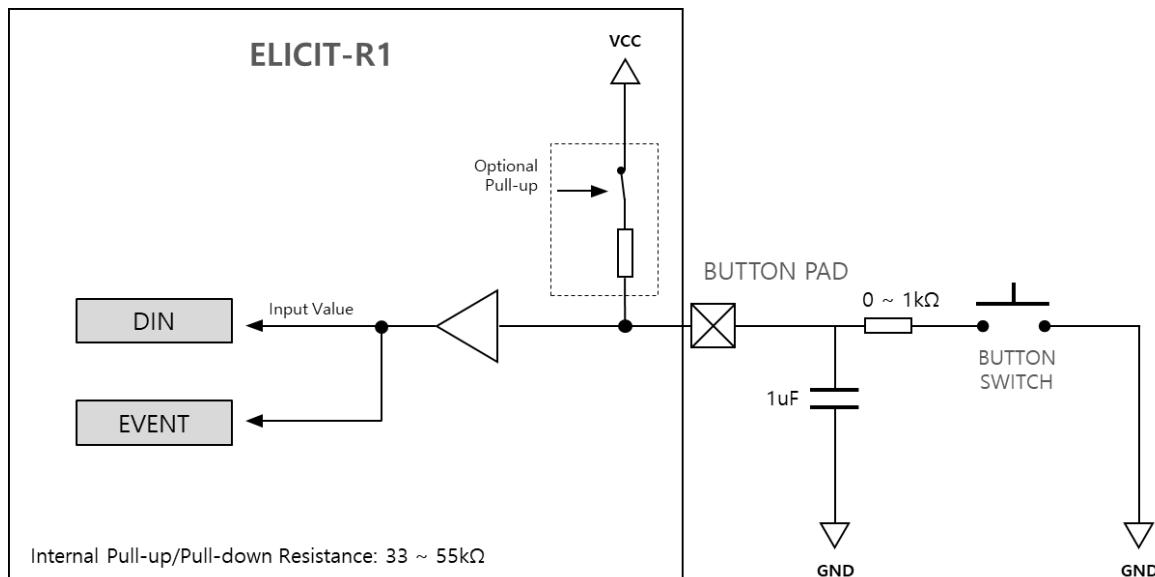


Figure 20 Button 회로(Pull-up, Debounce circuit)

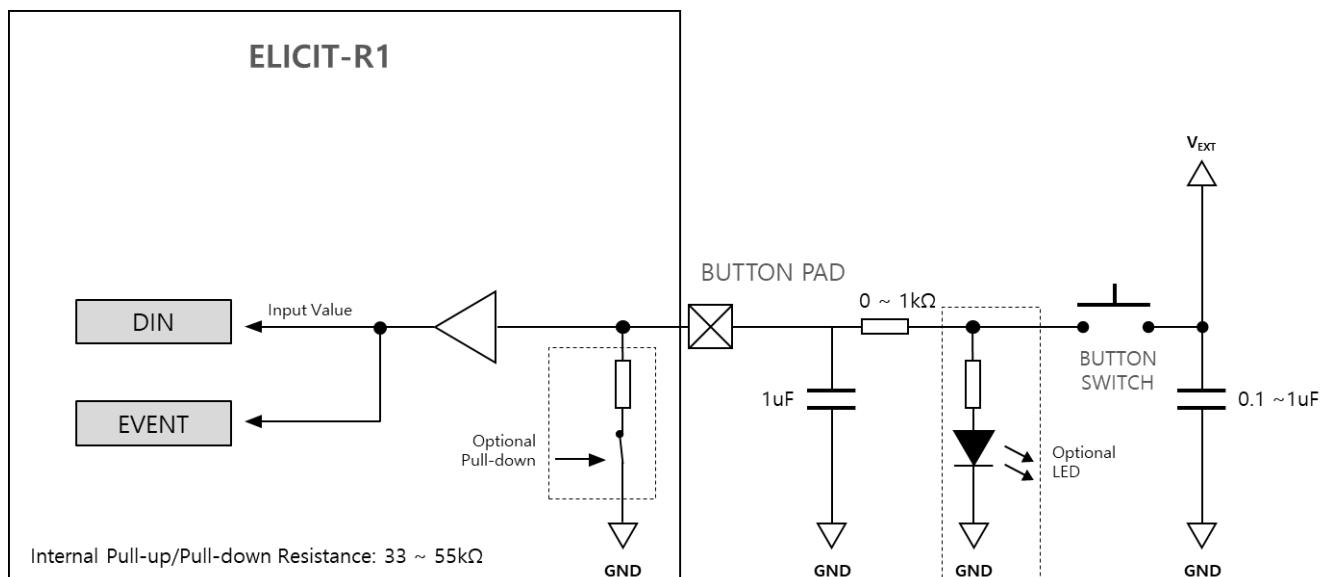


Figure 21 Button 회로(Pull-down, Debounce circuit)

6.8.1 버튼으로 Join

- Joinable 상태의 Node[3.4.2 Joinable State]는 Host 연결 없이 사용자가 직접 버튼으로 Join 가능
- Join by Button 절차
 1. AP에서 JOINREQ 명령 송신
 2. Join 하려는 Node에서 JOINREQ를 수신하면 Joinable 상태로 변경됨
(2초마다 LED1, 2가 동시에 짧게 2회 점멸)
 3. 버튼 1을 누르면 Join Response 메시지가 자동으로 전송된다[표 1 Elicit Network 구성 절차].
 4. 사용자가 AP에서 해당 Node의 LONG ADDR을 선택하여 허락 메시지[[AT+PJOIN](#)] 전송

6.8.2 버튼으로 초기화(Initialize)

- ELICIT-R1/R1+는 Host 연결 없이 사용자가 직접 버튼으로 초기화를 진행할 수 있음
- 버튼 초기화 모드 진입 절차
 1. button1, button2를 동시에 누른 상태에서 reset 실행
 2. reset 이후 3초 이내에 누르고 있던 button1, button2를 동시에 해제
 3. LED1, 2가 같이 켜지면 초기화 모드 진입
 4. 초기화 모드 진입 후 5초 이내에 초기화를 수행하지 않으면 초기화 모드 해제

버튼초기화 방법 (초기화 모드 진입 후 5초 이내 수행)

- 버튼 1을 5초간 누름: 네트워크 초기화(Init level 0)
- 버튼 2를 5초간 누름: IO, Event 초기화(Init level 1)
- 버튼 1과 2를 5초간 누름: 모든 설정 초기화(Init level 2)

6.8.3 버튼으로 사용자 명령 수행(Button Linked Event)

- ELICIT-R1/R1+는 은 Host 연결 없이 사용자가 직접 버튼으로 저장된 명령을 수행할 수 있음
- 버튼 1, 2마다 AT Command를 저장할 수 있음[[AT+BTEVT](#)]
- 버튼이 눌릴 때 수행

AT Command				
CMD	Description	Set/Get	Parameters	Default/Initial
AT+BTN	Button Config	Get	[Mode] [EVENT]	
AT+BTEVT	Button event set	Set/Get	[Button num] [ACT] [CMD]	

6.8.4 버튼으로 Sleep mode 해제

Sleep again mode로 진입한 디바이스는 Wake up event로 깨어나도 설정된 명령만 수행하고 다시 잠들기 때문에 메시지 수신도 불가능하고 사용자가 추가적인 명령할 수도 없다. 이런 경우에 버튼을 이용하여 Sleep again mode를 해제할 수 있다. 해제한 이후에는 Normal 상태로 동작한다. 다시 계속해서 잠들기 위해서는 Sleep again 옵션을 활성화하여 [[AT+SLEEP](#)] 명령을 수행해야 한다.

버튼 Sleep again mode 해제 방법(Normal/Deep sleep 공통)

- 버튼 2를 누른 상태에서 버튼 1을 1초간 누름

6.9 LED

LED1(Pad26)과 LED2(Pad27)은 현재 상태를 표시하기 위한 LED 전용 출력 패드이다. 다른 용도로 사용할 수 없다.

표 8 내부 상태별 LED 점등

상태	상태설명	LED
무선 전송(Tx)	무선으로 데이터를 전송 중인 상태	LED1 점등
무선 수신(Rx)	무선으로 데이터를 수신 중인 상태	LED2 점등
버튼 초기화 모드	초기화 설정 모드로 진입한 상태	LED1,2 동시 점등
Joinable State	네트워크 Join 요청(Join Req) 메시지를 수신한 후 접속 가능한 상태(joinable state)	LED 1, 2가 2초 주기로 동시에 2회씩 점멸

LED pad는 Push-pull로 고정되어 있다. 사용자는 이를 감안하여 LED를 다음 회로를 참고하여 연결(Cathode를 Pad 쪽으로 결선)한다.

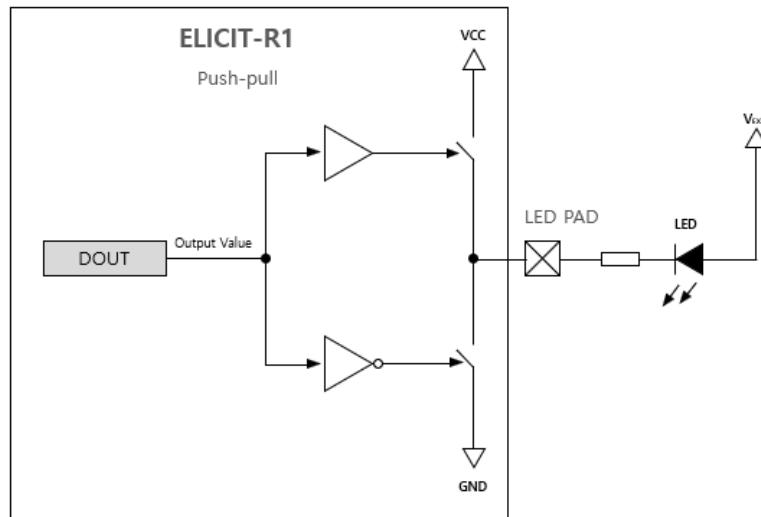


Figure 22 LED 연결(Push-pull)

7 Serial Data Interface

Elicit R1은 Serial data interface로 2개의 UART 전용 포트(UART1, UART2)와 1개의 I2C 전용 포트를 가지고 있다. SWD(Serial wire debug)용의 SWCLK, SWDIO는 디버깅 전용으로 사용자 지원은 하지 않는다.

IO 전압은 다른 IO와 같이 VCC(1.8~3.8V)에 연동된다.

7.1 UART

UART1은 사용자나 Host device 전용으로 AT command를 입력할 수 있다. UART1은 기본으로 Local Echo(사용자의 입력을 그대로 출력) 기능이 해제되어 있다. 그래서 UART 터미널(PuTTY, Tera term 등)을 연결했을 때 입력한 문자가 표시되지 않는다. 입력한 문자를 표기하려면 명령[AT+ECHO]을 이용하여 Echo 기능을 활성화해야 한다.

UART2는 사용자 디바이스를 연결하여 다양한 IoT 서비스를 수행할 수 있다.

UART1/2로 입력한 데이터는 설정에 따라 자신의 다른 UART 포트로 출력할 수 있을 뿐만 아니라, Elicit 네트워크 내의 다른 Node의 UART로 출력할 수 있다.

UART1/2의 기본 입력 버퍼 크기는 128byte이다. 따라서 이보다 더 큰 데이터는 나누어서 전송해야 한다.

UART

- 지원 속도(Baud rate) 1200 ~ 115200 bps
- 입력 버퍼 크기 128 Byte
- UART1 User console, Host Device, AT command
- UART2 User device, Modbus device

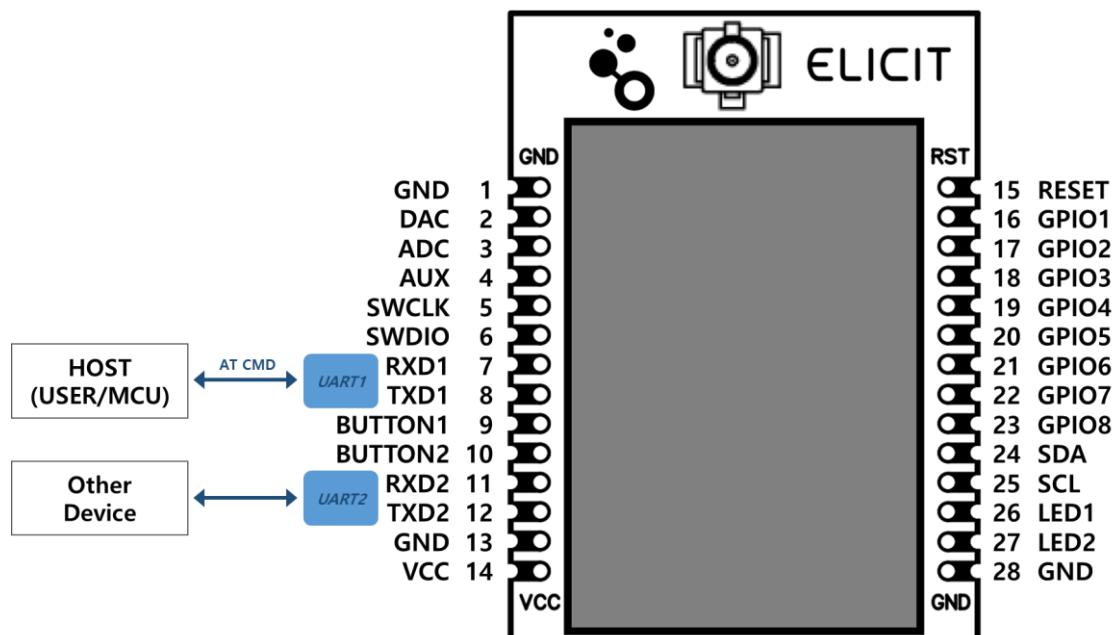


Figure 23 UART 연결

7.2 User data 전송

사용자의 데이터를 전달하는 방법은 AT command 를 이용하거나, UART port 에 입력되는 데이터를 자동으로 전송하는 방법의 두 가지가 있다.

7.2.1 SEND 명령

[[AT+SEND](#)] 명령을 이용하여 사용자 데이터를 목적지(최종 수신지), 출력할 UART port 와 형식을 지정하여 전송할 수 있다.

목적지 지정은 16bit SADDR 뿐만 아니라 64bit LONG ADDR(LADDR)을 사용할 수 있는데, AT 명령 중 유일하게 LADDR 을 사용할 수 있다. ALONE 상태의 디바이스라도 LADDR 을 알고 있으면 직접 메시지를 전달할 수 있다. 이때 PHY mode 와 RF CH 이 서로 일치해야 한다.

바이너리 데이터(binary data)를 목적지에서 출력하려고 할 때, 전송하려는 데이터는 16 진수(Hexadecimal)로 표현하여 입력한다. 16 진수 포맷은 접두어 '0x'을 생략할 수 있고, 0~9, A(a)~F(f)의 문자만으로 표현한다. 데이터 구분을 위해 공백을 추가할 때는 전체 데이터를 큰따옴표(")로 묶어서 전송한다. 데이터의 입력 단위는 최소 1byte, 최대 4byte 까지 입력한다.

출력형식으로 'Binary+CRC16' 옵션을 선택할 수 있다. 입력한 데이터에 대해 Modbus 에서 사용하고 있는 CRC16($X^{16}+X^{15}+X^2+1$)을 계산하여 Binary 데이터에 덧붙여서 출력한다. 이를 이용하여 Modbus RTU 를 지원하는 다양한 장비를 연결할 때 유용하게 사용할 수 있다.

사용자 데이터 전송 방법 ([\[AT+SEND\]](#) 명령)

- AT+SEND=<DADDR> <PORT> <FORMAT> <"USER DATA">
- DADDR 전송할 디바이스 선택, LADDR 사용가능
- PORT 출력할 UART 포트 선택
- FORMAT 출력 형식 선택, 문자열(String)이나 바이너리(Binary) 형식 등
- USER DATA 전달할 사용자 데이터, 최대

Hexadecimal 데이터 입력 형식(format 이 "B"나 "BM"일 경우)

- "0x"를 제외한 16 진수 형식 data 입력 "A0B11234"
- 16 진수로 입력한 data 를 binary 로 변환하여 송신 (MAX 64Byte(128 자)까지 입력 가능)
- Example) AT+SEND=100 1 B "A1243" → 0xA1234
- AT+SEND=100 1 B "1234567ABC" → 0x1234567ABC

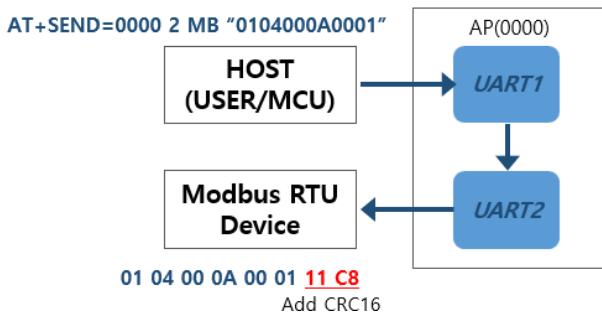


Figure 24 Send data to local UART2, Binary+CRC16 출력

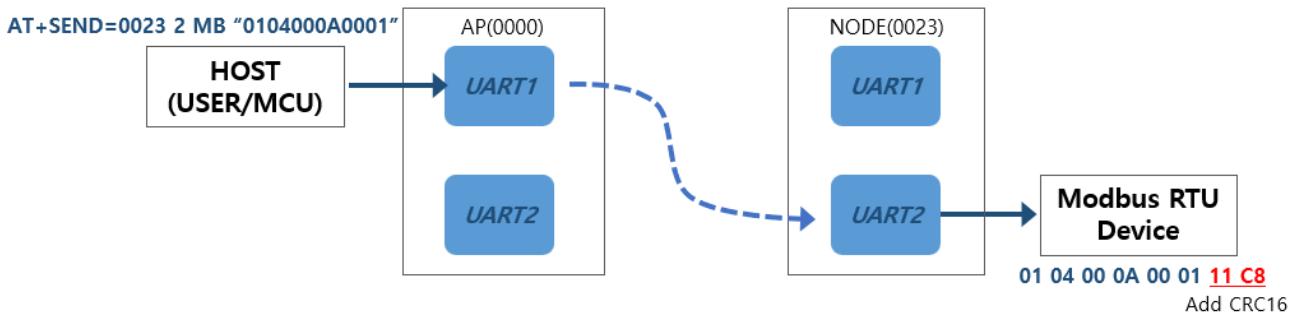


Figure 25 Send data to Other device, Binary+CRC16 출력

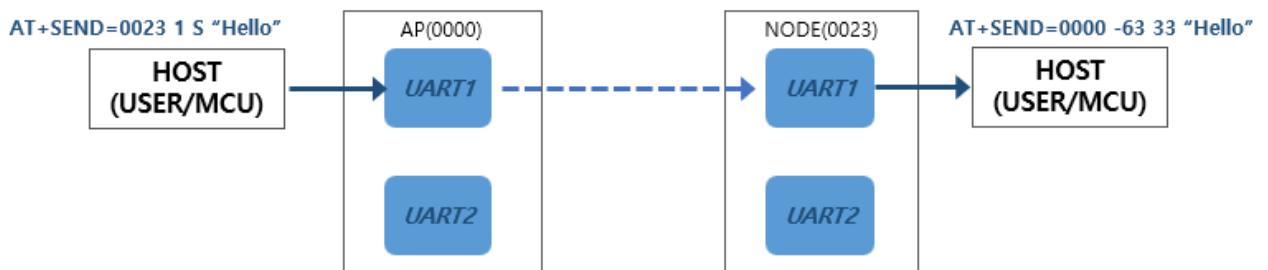


Figure 26 Send data to Other device, String 출력

7.2.2 UART1 자동 전송 모드(Pass through)

[[AT+U1SND](#)] 명령을 이용하여 UART1 으로 입력되는 모든 데이터를 자동으로 원하는 디바이스의 지정된 포트로 전송하는 모드를 설정할 수 있다. 이 모드가 설정되면 UART1 으로 일반 AT 명령을 입력해도 수행하지 않고, 입력된 데이터를 무조건 지정된 디바이스로 전송한다. 이 모드를 해제하려면 연속으로 "AT#"을 입력하거나, 다른 디바이스에서 전송 중지 명령을 보내야 한다.

전송은 데이터 입력이 2ms 이상 중단되거나, 연속으로 입력된 크기가 128Byte 이상이 되면 자동으로 입력된 데이터를 전송한다.

자동 전송 모드에서 데이터의 출력은 문자열(String) 형식이나, 16 진수(hexadecimal) 형식, 바이너리(Binary) 형식, 입력 데이터 그대로 출력(Pass-through)하는 형식 등으로 지정할 수 있다.

UART1 자동 전송 모드([AT+U1SND] 명령)

- 입력된 모든 데이터를 지정된 디바이스의 지정 포트, 지정한 형식으로 자동으로 출력
- 출력포트를 자기 자신의 UART1 으로 설정할 수 없음
- +SEND 명령과 달리 LADDR 을 지원하지 않음
- AT 명령도 수행하지 않고 전송하기 때문에 명령을 입력하기 위해서는 모드를 해제해야 한다.
- Binary 출력할 때 입력 데이터는 Hexadecimal 형식의 문자열로 입력한다.

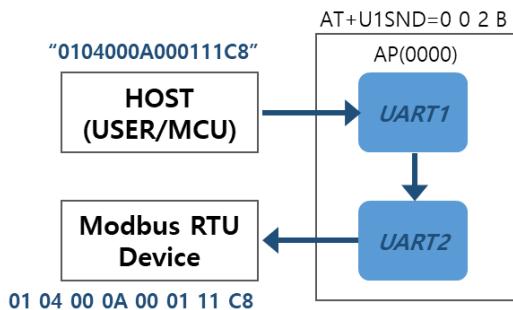


Figure 27 UART1 자동 전달 모드(UART1 to UART2, Binary)

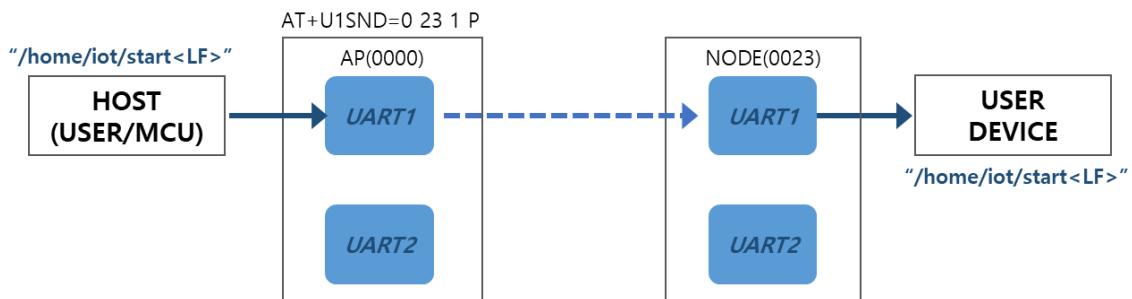


Figure 28 UART1 자동 전달 모드 (Pass-through to UART1)

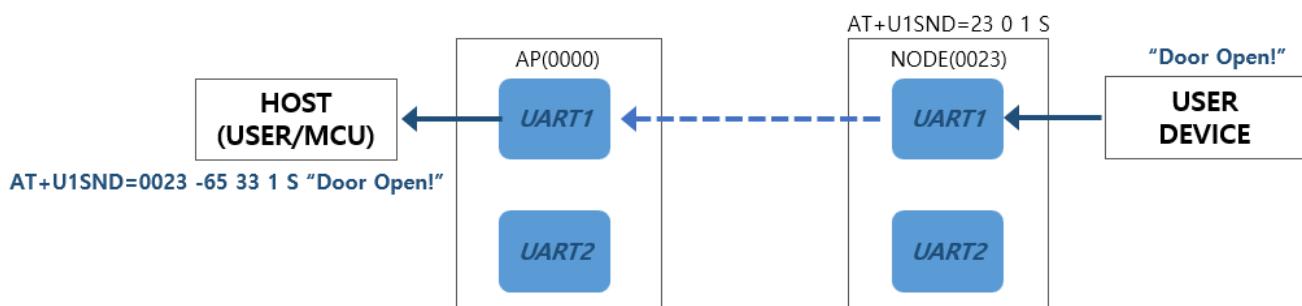


Figure 29 UART1 자동 전달 모드 (Send String data UART1)

7.2.3 UART2 자동 전송 모드

UART1 자동 전송 모드와 같이, [AT+U2SND]명령을 이용하여 UART2로 입력되는 모든 데이터를 원하는 디바이스로 자동 전송하여 지정된 포트로 출력할 수 있다.

전송 시점은 UART2로 데이터 입력이 일정 시간 이상 중단되거나, 연속으로 입력된 크기가 128Byte 이상이 되면 자동으로 입력된 데이터를 전송한다.

전송 시점을 결정하는 입력 delay는 baud rate 별 약 2Byte 입력 시간으로 계산되며, 9600 이상의 baud rate는 2ms로 설정된다.

자동 전송 모드에서 데이터의 출력은 문자열(String) 형식이나, 16 진수(hexadecimal) 형식, 바이너리(Binary) 형식, 입력 데이터 그대로 출력(Pass-through)하는 형식 등으로 지정할 수 있다.

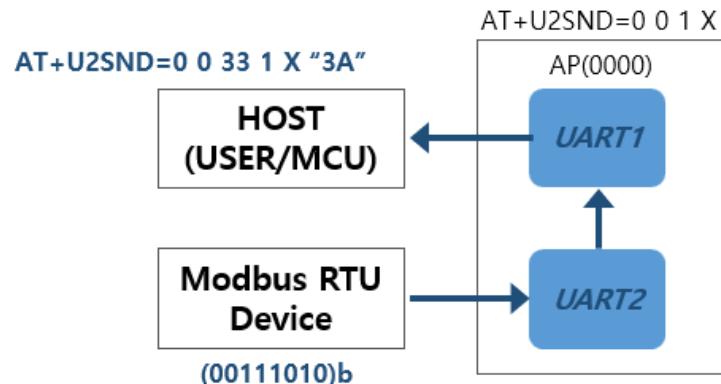


Figure 30 UART2 자동 전달 모드(UART2 to UART1, Hexadecimal)

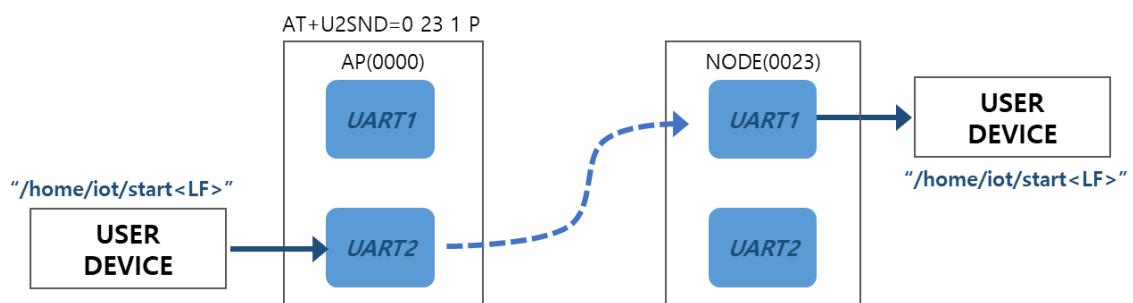


Figure 31 UART2 자동 전달 모드 (Pass-through to UART1)

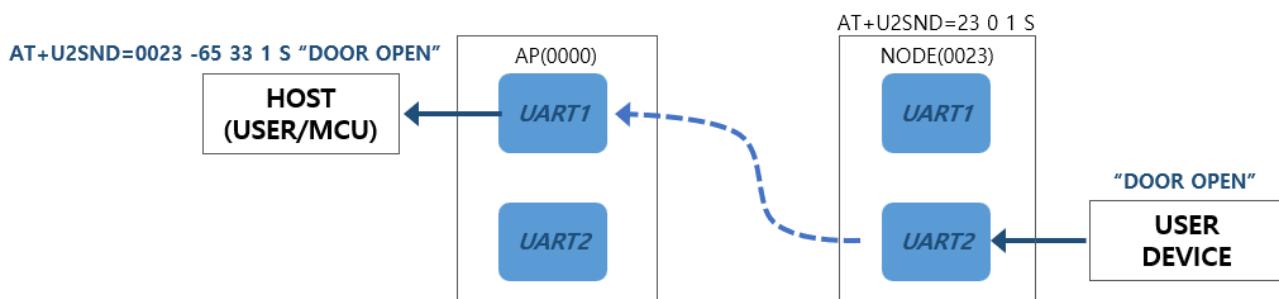


Figure 32 UART2 자동 전달 모드 (Send String data UART1)

ELICIT UART

- UART1 은 사용자 명령(AT command) 전용으로 사용
- UART2 는 다른 장비와 연결용으로 사용

Send User message

- AT 명령으로 사용자 메시지를 원하는 디바이스로 포트와 형식을 지정하여 전송 가능
- AT+SEND 명령은 유일하게 64bit LADDR 을 사용하여 직접 메시지를 전달할 수 있음

UART 자동 전송

- UART1 을 자동 전송 모드로 설정하면 AT 명령을 입력해도 수행하지 않음
- UART1 의 자동 전송 모드 해제: 연속으로 AT#입력(문자 사이에 시간 공백 없이 입력)

7.2.4 Example: 무선 UART 콘솔

장비의 UART 콘솔을 무선으로 연결하는 예제

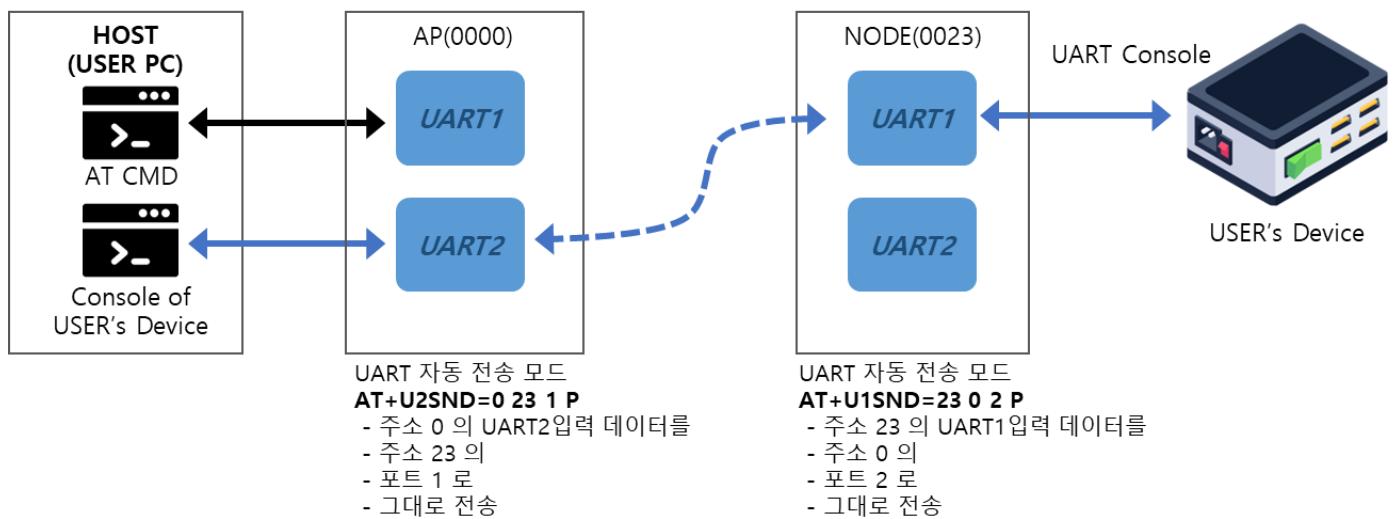


Figure 33 Example: UART Bypass

무선 UART 콘솔 구성 예제

- 원격으로 연결하려는 사용자 장비의 콘솔(Baud rate 38400)을 NODE(0023)의 UART1 과 연결
- AP(0000)에는 Host 용 PC의 UART1 과 2 를 모두 연결
- AP 의 UART1 과 연결된 터미널은 명령 입력용으로 사용
- AP 의 UART2 와 연결된 터미널은 장비의 원격 콘솔로 사용
- 콘솔의 baud rate 보다 무선 속도가 높아야 원활한 통신이 가능
- AP 의 UART2 baud rate 설정(38400): AT+BAUD=0 1 38400
- AP 의 UART2 자동 전달 모드 설정: AT+U2SND=0 23 1 P
- AP 에서 NODE 를 무선으로 설정(AP 의 UART1 과 연결된 터미널 사용)
- NODE(0023)의 UART1 baud rate 설정: AT+BAUD=23 1 38400
- NODE(0023)의 UART1 자동 전달 모드 설정: AT+U1SND=23 0 2 P
- 설정 이후에 AP 의 UART2 터미널과 사용자 장비가 직접 연결된 것과 같이 사용 가능

7.2.5 Example: 무선 MODBUS 장비

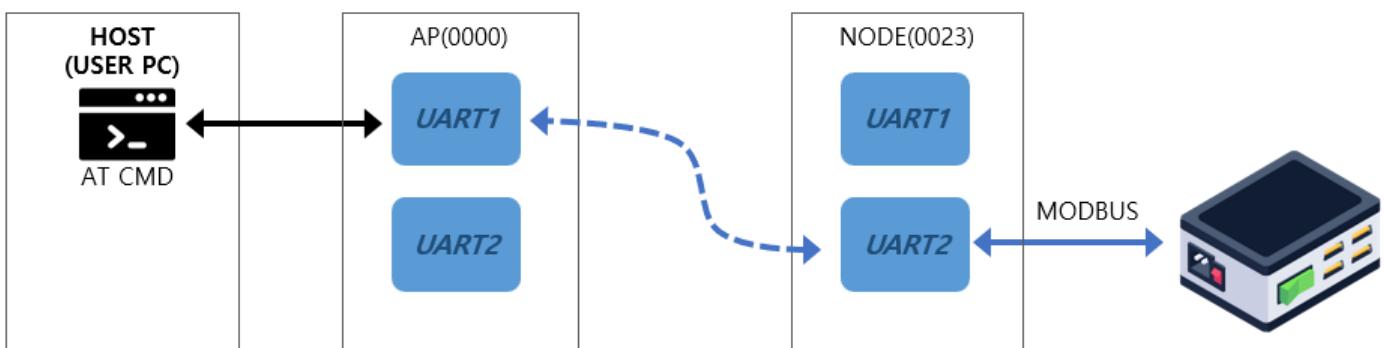
UART2에 Modbus 장비를 연결하여 사용하는 예제

MODBUS 명령을 AT CMD로 전송

AT+SEND=23 2 BM 0104000A0001

- Node(0023)의 UART 2로
- 입력데이터를 Binary 변환하고
- CRC16을 붙여서 출력하도록 명령

Binary + CRC16 출력
01 04 00 0A 00 01 11 C8



수신 메시지(MODBUS Data) 출력

AT+U2SND=23 -65 33 1 X "010402ABCD0795" OK

- 주소 23으로부터 UART2 입력 데이터를 수신
- 수신 세기 -65dBm
- 메시지 보낸 디바이스의 배터리 용량 100%
- 포트 1로 Hexadecimal 형식으로
- 수신 성공

UART 자동 전송 모드

AT+U2SND=23 0 1 X

- 주소 23 의 UART2입력 데이터를
- 주소 0 의
- 포트 1로
- Hexadecimal 문자열로 전송

Figure 34 Example: MODBUS 장비 연결

Modbus 장비 연결 구성 예제

- ◆ NODE(0023)의 UART2에 Modbus RTU를 지원하는 장비 연결
- ◆ Modbus 장비와 통신은 AP의 UART1 터미널에서 AT command로 수행
- ◆ NODE(0023)의 UART2 baud rate 설정(예 19200): AT+BAUD=23 2 19200
- ◆ NODE(0023)의 UART2 자동 전송 모드 설정: AT+U2SND=23 0 1 X
- ◆ AP에서 장비(국번:01, Function code 4,)주소 10(A)에서 1개 레지스터 읽기 명령을 전송
 - AT+SEND=23 2 BM 0104000A0001
 - NODE(0023)의 UART2에서 출력될 때 MODBUS 용 CRC가 자동을 추가된다
 - Modbus 장비에서 읽기 명령에 대한 응답으로 (010402ABCD0795)h 전송
- ◆ AP에서 수신한 Modbus 장비의 응답 데이터
 - AT+U2SND=23 -65 33 1 X "010402ABCD0795" OK

7.3 I2C

다양한 IoT 장비를 연결할 수 있도록 I2C 통신 기능을 제공한다. Master 모드만 지원하며, 명령[AT+I2C]을 이용하여 쉽게 통신을 할 수 있다. I2C 읽기 명령을 전송한 다음, 수신한 응답 데이터는 [AT+IND]형식으로 반환된다.

I2C 버스의 Master mode 지원

- Standard mode(100 kHz 클럭 속도) 지원
- SCL, SDA는 내부에서 10kΩ으로 Vcc에 Pull-up 연결됨
- I2C Write 명령으로 전송할 수 있는 최대 데이터는 4byte
- I2C Read 명령으로 읽을 수 있는 최대 데이터는 16byte
- I2C Read 명령으로 읽은 데이터는 [AT+IND]로 반환
- [I2C dev address]는 앞에 "0x"를 추가하여 입력 또는 생략하여 입력해도 16 진수로 인식

I2C Example

- RTC 모듈(DS1307)과의 통신. (device address는 0x68)
- AT+I2C=[address] [I2C dev address] [read flag(2)] [register address] [length]
- AT+I2C=[address] [I2C dev address] [write flag(1)] [register address] [length] [Data_1] ... [Data_n]
- READ :

AT+I2C=0 68 2 0 7

- WRITE :

AT+I2C=0 68 1 0 2 1 5

8 Sleep mode

저전력 운영을 위해 ELICIT-R1/R1+는 전력소모 레벨에 따라 Normal 과 Deep, 두 가지의 Sleep mode 를 제공한다. Sleep mode 에 진입한 상태에서는 RF block 과 Serial data interface(UART, I2C) block 의 전원을 차단하기 때문에 무선 송/수신과 UART, I2C 통신이 불가능하다. AP는 항상 AT command 수신과 무선 수신 대기 상태이어야 하기 때문에 Sleep mode 로 들어가지 않는다.

표 9 Sleep mode

Sleep mode	Normal Sleep	Deep Sleep
Current consumption	5~6uA	MAX 1uA (Typically 0.6uA)
Wake up event	Sleep Timer, AUX, Button1/2, GPIO7/8	Sleep Timer, Button1, GPIO1/3
Wake up point	Sleep point	Reboot
Wake up time	MAX 800usec	MAX 250msec

일정 시간동안 Sleep mode 로 들어갔다 깨어날 수 있는 Sleep timer 를 Normal/Deep sleep 에서 공통으로 사용할 수 있다. Sleep time event(STEVT)는 Sleep timer 의 Time out 에 의해 깨어났을 때 최대 4 개의 지정된 명령을 수행하는 기능이다.

명령[[AT+SLEEP](#)]을 통해 Normal sleep 이나 Deep sleep mode 로 진입할 수 있고, Sleep timer 와 Sleep 에서 깨어났을 때 다시 잠들 것(sleep again option)인지 여부를 설정할 수 있다. Sleep again 이 적용되면 외부 입력이나 Sleep Timer 에 의해 깨어난 후, 설정된 명령이 있으면 이를 수행하고 다시 잠들게 된다.

Sleep timer

- Sleep timer 의 설정 주기는 초(second) 단위이고, 0 으로 설정하면 Timer 는 동작하지 않는다.
- Sleep timer 는 최대 2,073,600 초(24 일)까지 주기를 설정할 수 있다.

8.1 Normal sleep

Normal sleep mode 를 시작할 때는 수행 중인 동작을 멈추고 Sleep 상태로 들어간다. 그리고 깨어날 때는 이전에 Sleep 상태로 들어가기 전 멈춘 시점에서 깨어난다. 만약 전송 대기 중(pending)인 메시지가 있을 때 Sleep 상태가 되면, 깨어난 이후에 전송하게 된다.

Button1 과 Button2 를 누르면 별다른 설정 없이도 Normal sleep 에서 깨어날 수 있다. Button 과 연결된 명령(Button linked event)이 있으면 이를 항상 수행한다. AUX 와 GPIO7, GPIO8 은 IO 설정 명령[[AT+IO](#)]으로 Event trigger 를 설정하면, Normal sleep mode 에서 DI event 가 발생했을 때 깨어날 수 있다.

Normal Sleep mode

- Normal sleep 은 메모리 내용을 유지하면서 정지된 상태, 깨어나면 정지된 시점에서 시작
- 깨어나서 다시 시작하기까지 약 800usec 소요됨
- Sleep timer 와 IO 에 의해 깨어날 수 있는 Event 와 명령을 연결할 수 있다

8.2 Deep sleep

Deep sleep 은 Sleep timer 와 외부 입력(Button1, GPIO1, GPIO3)에 의해서 깨어날 수 있다. 전력 소모를 최소화하기 위해 대부분의 기능을 정지시키고, 깨어날 때는 시스템을 Reset 하여 처음부터 시작한다. 이 때, ELICIT-R1/R1+는 어떤 Event 에 의해 Reset 이 되었는지 확인할 수 있다. Deep sleep 진입 전후, 깨어나서 Reset(Reboot)이 완료될 때까지 GPIO 의 상태는 계속 유지된다.

명령[[AT+DSWUP](#)]을 이용하여 Deep sleep mode 에서 깨어날 수 있는 외부입력을 설정할 수 있고, 깨어나서 수행할 AT command 지정은 [[AT+DSEVT](#)]명령을 사용한다. 깨어날 수 있는 외부입력이 하나라도 설정되어 있어야 Deep sleep mode 로 진입할 수 있고, 설정된 외부 입력이 없으면 Deep sleep mode 로 들어가지 않는다.

Button1 은 눌렀을 때 Deep sleep 에서 깨어나도록 기본 설정되어 있다. Button2 를 누른 상태에서 Button1 을 누르면 다시 잠들기 기능(Sleep again)이 해제된다.

Deep Sleep에서 깨어났을 때 RESET Indication 메시지

- button 1 AT+RESET=0100 0 33 1000
- GPIO1 AT+RESET=0100 0 33 1001
- GPIO3 AT+RESET=0100 0 33 1003
- Timeout AT+RESET=0100 0 33 1004

8.3 다시 잠들기(Back to sleep again)

SLEEP 명령의 옵션인 AGAIN 을 1(Enable)로 설정하면 Sleep 상태(Normal/Deep 공통)에서 깨어났을 때, Event 와 연결된 명령을 수행하고 다시 잠들게 된다.

Normal sleep 상태에서 Sleep timer 나 GPIO7/8 event 로 깨어나면 연결된 명령(Linked Event)을 수행하고 바로 Sleep mode 로 진입한다. Sleep timer 가 설정된 상태에서 Button 으로 깨어났을 경우에도 해당 Button 에 연결된 명령을 수행하고 다시 Sleep mode 로 진입한다.

Normal sleep 이나 Deep sleep 상태에서 깨어났을 때는 연결된 명령을 수행하고, 다시 잠든다. Button2 를 누른 상태에서 Button1 을 1 초 이상 누르면 다시 잠들기 기능(AGAIN)이 해제(disabled)된다.

AGAIN 을 0(Disable)으로 설정하면, Event 에 의해 깨어난 후, 계속 Normal mode 로 동작한다.

8.3.1 LBT mode에서 Event 수행 완료 후 sleep timing

LBT mode(normal mode)에서 SLEEP 명령의 옵션인 AGAIN 이 설정되어 있을 때, event(STEVT, NSEVT, DSEVT, BTEVT ...) 수행이 완료되면 바로 sleep mode 를 수행한다. Sleep mode 에서 깨어나 수행되는 event slot command 는 자신에게 내리는 local command 와 원격으로 setting 하는 전달 command 를 지원한다.

8.3.2 ECH mode에서 다시 잠들기

ECH mode 에서는 sleep 상태(Normal/Deep 공통)에서 깨어 났을 때, Beacon 동기화를 수행한다. Beacon interval 에 비례하여 2 cycle 의 시간이 지나도 beacon 동기화가 되지 않는다면 다시 sleep 상태로 진입한다. Beacon 동기화

진행과 동시에 event 명령을 수행한다. Event 가 local command 일 경우 바로 수행하고, remote command 일 경우, data queue 에 넣고 다음 event 를 수행한다. Remote command 를 실패할 경우 retry 횟수를 전부 수행하고 sleep 상태로 진입한다. Sleep mode 재진입은 event 를 모두 수행하고 beacon 을 수신하면 sleep mode 로 진입한다. ECH mode 에서도 Sleep mode 에서 깨어나 수행되는 event slot command 는 자신에게 내리는 local command 와 원격으로 setting 하는 전달 command 를 지원한다.

9 Event

ELICIT-R1/R1+ 에서 Event 는 외부 입력이나, 내부 Timer event 를 감지했을 때 생성하는 메시지이다. 생성된 event 는 Indication message 형태로 사용자에게 UART1 이나 네트워크를 통해 전달할 수 있다. 그리고 event 와 명령을 연결하여 상황에 맞는 다양한 동작을 수행할 수 있다.

표 10 Event 의 종류

Event	Name	Description
Wake up by IO(Deep sleep)	DSEVT	Deep sleep 상태에서 외부 입력에 의해 깨어났을 때 발생
Wake up by IO(Normal sleep)	NSEVT	Normal sleep 상태에서 외부 입력에 의해 깨어났을 때 발생
Wake up by Sleep timer	STEV	Sleep 상태에서 sleep timer 에 의해 깨어났을 때 발생
Periodic Timer	PTEVT	Normal mode 에서 지정한 주기마다 발생
GPIO(DI)/AUX Digital Input	DIEVT	GPIO(DI)/AUX 입력 상태가 변했을 때 발생
Button Press/Release	BTEVT	Button 을 누르거나 떼었을 때 발생
I2C Read	I2C	I2C 읽기 명령에 대한 응답을 수신했을 때 발생
ADC Read	ADC	ADC 읽기 명령에 대한 응답을 수신했을 때 발생
UART Receive	UART	UART 로 데이터를 수신했을 때 발생

9.1 Indication message

IND 메시지는 event 가 발생했을 때나 내부의 동작 상태 변화(Join Complete, Network Exit)를 알리는 용도로 사용한다. Host 는 IND message 를 통해 IO 의 변화나 동작 상태를 알 수 있다.

표 11 IND Message 를 생성하는 Event

IND Source	Description	Report
JOIN Complete	Elicit network 에 JOIN 을 완료할 때 IND 생성	Always
RELAY_FAIL	Hop Tx 를 진행하다 중간 node 에서 Tx 를 실패했을 경우 IND 생성	Always
SCAN result	SCAN 을 완료할 경우 IND 생성	Always
ADC Event	ADC 읽기 명령을 완료할 경우 IND 생성	O
DIEVT	GPIO(DI)/AUX Event 가 발생할 경우 IND 생성	O
PER result	PER test 를 진행을 완료할 경우 IND 생성	O
I2C Event	I2C 읽기 명령을 완료할 경우 IND 생성	O
ELICIT EXIT	Elicit network 에서 EXIT 했을 경우 IND 생성	O
INIT	Factory init 을 실행했을 경우 IND 생성	X
RESET	Power on, RESET, Wake up from Deep sleep 등 Reset 이후 발생	X
BCN_TIMEOUT	ECH network 에서 beacon 수신을 실패했을 경우 IND 생성	X

9.2 Report to AP

Node 에서 발생한 IND message 를 자동으로 AP 로 전송하는 기능으로, 명령[AT+REPORT]을 통해 IND 자동 전송 모드를 설정할 수 있다.

10 Linked Event

ELICIT 모듈의 주요 기능인 Linked Event 는 Event 와 사용자 지정 명령을 연결하여 해당 Event 가 발생했을 때 자동으로 지정된 명령을 수행하는 것이다. Event 마다 명령을 저장할 수 있는 공간(Slot)에 차이가 있다.

표 12 Linked Event 의 종류

Event name	Event	내용	R1 Slot	R1+ Slot
AT+PTEVT	Periodic Timer event	지정한 주기마다 저장된 명령 수행	4	8
AT+STEVT	Sleep timer event	Sleep 에서 깨어난 후 저장된 명령 수행	2	4
AT+DIEVT	DI event (GPIO/AUX)	DI 이벤트가 발생하면 저장된 명령 수행	5	8
AT+BTEVT	Button event	BUTTON 을 누르면 저장된 명령을 수행	2	2
AT+DSEVT	Wake up from deep sleep	DI 이벤트에 의해 Deep sleep 에서 깨어났을 때 저장된 명령을 수행	2	4
AT+NSEVT	Wake up from normal sleep	DI 이벤트에 의해 normal sleep 에서 깨어났을 때 저장된 명령을 수행	3	4

10.1 Periodic timer linked event (PTEVT)

설정한 주기마다 사용자가 지정한 명령을 수행하는 기능으로 Normal mode 에서 동작한다. ELICIT-R1 에서는 최대 4 slot, ELICIT-R1+는 최대 8 slot 까지 지원한다. Timer event 주기는 1 초 단위로 최대 2,073,600 초(24 일)까지 설정이 가능하다. 주기를 0 으로 설정한 Slot 은 수행하지 않는다.

Timer event 는 주기와 함께 offset 도 설정할 수 있어서 동일한 주기를 갖지만 실행하는 시점은 각기 다른 Event 를 설정할 수 있다. 내부 RTC 의 시간(초로 환산한 시간)에 Offset 을 더한 값을 주기로 모듈러 연산을 하여 Timer event 를 실행하는 시점을 결정한다.

Timer Event 발생 시점(t): (RTC + OFFSET) % PERIOD 가 0 이 되는 시간

[AT+PTEVT]명령으로 설정한다.

10.2 Sleep timer linked event (STEVT)

Normal/Deep sleep mode 에서 Timer 를 동작 시키고 Time out 이 발생했을 때 수행할 명령을 ELICIT-R1 은 2 slot, ELICIT-R1+는 4 slot 까지 지정할 수 있다. Sleep 에 진입하기위한 명령[[AT+SLEEP](#)]에 Timer 주기를 옵션으로 지정한다. 주기는 1 초 단위로 최대 2,073,600 초(24 일)까지 설정이 가능하다. 깨어났을 때 Slot 에 저장된 명령을 Slot 순서대로 모두 수행한다. 개별 명령은 10ms 간격으로 실행한다. [[AT+STEVT](#)]명령으로 설정한다.

10.3 DI Linked event (DIEVT)

DI Event 에 대해 Linked event 를 설정할 수 있다. 개별 DI Event 마다 다른 사용자 명령을 할당할 수 있다. GPIO 중 GPIO4 ~ GPIO8 만 Linked Event 를 지원한다. 명령[[AT+DIEVT](#)]을 이용하여 설정한다.

DI Event 를 사용하기 위해서는 사전에 [[AT+IO](#)]명령을 사용하여 해당 GPIO 를 입력으로 설정하고, Event Trigger 를 설정해야 한다.

10.4 Button Linked event (BTEVT)

Button1 과 Button2 에 대해 각각 Linked event 를 설정할 수 있다. 명령[\[AT+BTEVT\]](#)으로 설정된 Event 는 Button 이 눌릴 때 지정된 명령을 수행한다.

Button 은 항상 입력으로 설정되어 있고, 별도의 IO 설정을 할 필요는 없다.

10.5 Wake up from deep sleep linked event (DSEVT)

Deep sleep 은 외부 입력 중 Button1(기본값으로 설정됨)과 GPIO1, GPIO3 으로 깨어날 수 있다. 최대 2 개의 slot에 대해 각각 IO 와 수행 명령을 지정할 수 있다. [\[AT+DSEVT\]](#) 명령으로 설정한다.

DSEVT 를 사용하기 위해서는 [\[AT+DSWUP\]](#) 명령을 사용하여 어떤 GPIO 의 입력을 사용하여 Deep sleep 상태에서 깨어날 수 있는지 설정해야 한다.

10.6 Wake up from normal sleep linked event (NSEVT)

Normal sleep 은 외부 입력 중 GPIO7, GPIO8, AUX 입력 event 로 깨어날 수 있다. ELICIT-R1 은 최대 3 개의 slot, ELICIT-R1+는 최대 4 개의 slot 에 대해 각각 IO 와 수행 명령을 지정할 수 있다. [\[AT+NSEVT\]](#) 명령으로 설정한다.

DI Event 를 사용하기 위해서는 [\[AT+IO\]](#) 명령을 사용하여 해당 GPIO 를 입력으로 설정하고, Event Trigger 를 설정해야 한다.

11 기타 부가 기능

11.1 디바이스 초기화

ELICIT 모듈에 저장된 파라메터와 설정 값들의 초기화는 명령[[AT+FINIT](#)]이나 버튼으로 할 수 있다. 초기화는 3 가지 레벨로 구성된다.

Init level 0 (Network Initialize)

- 가입(Join)된 Elicit network 정보 초기화
- PAN ID, SADDR, JOINSTS, RFCH 등 초기화
- GPIO, AUX, UART, Event 정보 유지

Init level 1 (IO and Event Initialize)

- GPIO, AUX, UART 등의 설정 초기화
- Timer event, Sleep timer 주기, Sleep timer event 초기화
- Linked event 초기화
- Elicit network 정보 유지

Init level 2 (Factory Initialize)

- 모든 설정 초기화(Init level 0 + level 1)

11.2 펌웨어 업데이트

사용자는 UART1 을 이용하여 ELICIT 모듈의 펌웨어를 다운로드 할 수 있다. [[AT+BOOT](#)]명령으로 Bootloader 모드로 진입한 다음, 다운로드 메뉴를 선택할 수 있다. 다운로드는 XMODEM(128 Byte CRC) 프로토콜을 사용한다.

XMODEM 을 이용한 펌웨어 업데이트

- UART1 을 XMODEM 을 지원하는 터미널과 연결
- 명령[[AT+BOOT](#)]을 입력하여 Bootloader mode 로 재부팅
- Bootloader menu 다운로드(1), 재부팅(2) 중 다운로드 메뉴 선택(1 을 입력)
- 3 초 이내 재부팅(2)을 선택하지 않으면 자동으로 다운로드 모드(XMODEM Receive)로 변경
- XMODEM 프로토콜로 펌웨어 다운로드
- 다운로드가 완료되면 자동으로 업데이트후에 재 부팅
- 다운로드 모드에서 60 초 이내에 XMODEM 다운로드를 시작하지 않으면 재 부팅

11.3 PER(Packet error rate) test

두 디바이스 간 시험용 패킷을 주고받으며 무선 환경을 시험해 볼 수 있다. 명령[[AT+PER](#)]을 사용하여 시험을 진행할 상대방 디바이스, 전송 횟수, 시험데이터 크기, 패킷 전송 사이 간격(msc 단위) 등의 옵션을 설정할 수 있다. PER 시험을 시작하면, 송수신측 모두 진행 상황이 Indication 메시지로 RSSI, 송/수신 성공 카운트 등이 출력된다.

PER 시험

- Tx count (Max 65535), data length (Max 100), Tx interval(50 ~ 3000 ms)을 설정하여 시험한다.
- Test 완료 후 PER, Avg RSSI, Max RSSI, Min RSSI 등의 정보를 표시한다.

11.4 PARAMS SAVE & LOAD

ELICIT-R1/R1+의 parameter 를 setting 하고, setting 값들을 저장할 수 있다. 명령[[AT+PARAMS](#)]을 사용하여 명령을 사용한 현재 parameter 들을 저장하고, 원할 때 불러와 save 당시 parameter 를 현재 값으로 setting 할 수 있다. 이 기능으로 save point 를 만들어 유용하게 사용 가능하다. 이 명령은 AT command 로 설정할 수 있는 모든 parameter 들을 save 또는 load 한다.

ALL PARAMETER SAVE & LOAD

- 현재 적용중인 모든 parameter 들을 저장한다.
 - AT+PARAMS=0 SAVE
- Parameter 를 setting 하고, 저장했던 parameter 로 다시 돌아가고 싶으면 LOAD 한다.
 - AT+PARAMS=0 LOAD

12 AT command

12.1 AT Command 전용 UART1

ELICIT 모듈은 UART1을 통해 사용자 명령을 수행할 수 있다. UART2는 명령어 입력을 지원하지 않는다. UART1의 기본 설정은 (표 13)과 같다. UART의 속도 설정은 명령[[AT+BAUD](#)]을 이용하여 변경할 수 있다.

표 13 UART1의 기본 설정

Parameter	Value
속도(Baud rate)	1200 ~ 115200 bps (Default 115200)
데이터비트(Data bit)	8bit
패리티비트(Parity bit)	None
종료비트(Stop bit)	1bit
흐름제어(Flow control)	None

12.2 표기 방식

Elicit 명령어는 다음의 표기법을 사용한다. 괄호와 슬래시 같은 기호는 실제로 입력하지 않는다.

파라메터 간 구분은 공백문자(space, ASCII code 32(0x20))로 한다. 파라메터로 숫자를 입력할 때 '0x'를 붙이면 16 진수로 인식하고, 그렇지 않으면 10 진수로 인식한다. '0x'를 붙이지 않아도 'a(A)~f(F)'를 포함한 숫자는 자동으로 16 진수로 인식한다.

주소(SHORT ADDR/LONG ADDR)은 항상 "0x"를 제외한 Hexadecimal로 표기한다. 주소를 표기할 때 '0'으로 앞자리 채우기가 허용된다(000A, 00A, A 모두 동일한 주소로 인식)

CR	복귀문자(Carriage return character, ASCII code 13(0xD)), 명령어의 끝을 나타낸다.
LF	개행문자(Line feed return character, ASCII code 10(0xA)), 명령어의 끝을 나타낸다.
<...>	홑화살괄호(꺾쇠, angle brackets)는 생략할 수 없는 파라메터를 표현한다.
[...]	대괄호(square brackets)는 생략할 수 있는 옵션 파라메터를 표현한다.
(...)	소괄호(parentheses)는 명령어 내에서 명령어를 구분하기 위해 사용한다.
/	슬래시(slash)는 여러 파라메터 중 하나를 선택할 때 파라메터 구분용으로 사용한다.
'C'	작은 따옴표로 표기하는 것은 하나의 문자(ASCII code)를 나타낸다.
"string"	큰 따옴표는 문자열(string)을 나타낸다. 문자열은 공백을 포함할 수 있다.
SADDR	16bit 네트워크 주소(SHORT ADDR)
LADDR	64bit 주소, 고유 ID(LONG ADDR)
OD	발신지(Origin device), 메시지를 최초 송신한 디바이스.
OADDR	Address of origin device, OD의 주소. Hopping mode 구분용 점('Comma')은 생략한다.
DD	목적지(Destination device), 메시지의 최종 목적지 디바이스.
DADDR	Address of destination device, DD의 주소. Hopping mode 구분용 점은 생략한다.
HD	Hopping device, DD로 전송하는 경로에서 메시지를 전달하는 디바이스.
T>	송신 프롬프트. 명령 수행 예제에서 입력(전송)하는 것을 나타냄. 실제 입력하지 않음.
R<	수신 프롬프트. 명령 수행 예제에서 터미널로 출력된 내용을 나타냄. 실제로 출력되지 않음.

12.3 AT 명령 구조

Elicit 의 AT 명령 체계는 무선 IoT 네트워크(Elicit network)에 포함된 임의의 디바이스가 임의의 다른 디바이스에게 명령을 전달할 수 있고, 그 결과를 해당 디바이스로부터 수신하는 구조를 위해 만들어졌다. 이를 위해 명령마다 해당 명령을 수행할 디바이스의 주소를 입력해야 하고, 출력하는 모든 메시지에는 해당 메시지가 어느 디바이스로부터 왔는지 그 주소가 포함되어 있다. 또한 수신한 모든 메시지에는 수신 신호의 세기(RSSI)와 해당 디바이스의 전원 전압 값이 기본으로 포함되어, 네트워크를 유지보수하기 용이하도록 설계되었다.

Elicit 에서 사용하는 AT 명령의 기본 형식은 'AT+'와 COMMAND 를 이어서 입력하는 방식이다. 명령의 종류에 따라 명령어 다음에 '='나 '?'로 지정된 명령 구분자를 붙이고, 해당 명령을 수행할 디바이스의 주소를 덧붙여서 사용한다. 그 다음에는 명령 별로 각기 정의된 파라메터를 입력한다.

AT 명령의 형식은 크게 다음 3 가지 종류로 구분할 수 있다.

명령: 사용자가 Elicit 에 입력하는 메시지로, 실행/설정/읽기/전달/문의 등으로 구분할 수 있다.

응답: 명령에 대한 결과로, Elicit 이 출력하는 메시지

수신: 다른 디바이스로부터 수신한 메시지나, 내부 이벤트에 의해 발생한 알림 메시지를 출력하는 메시지.

표 14 AT 명령의 형식/종류

형식	종류	구분자	내용
명령	실행(Execute)	=	정의된 동작/기능을 실행
	설정(Set)		주어진 값을 적용(쓰기)
	전달(Send)		다른 디바이스로 메시지나 실행/설정 명령을 전달
	읽기(Get)	?	파라메터/설정 값을 읽기
	문의(Query)		다른 디바이스에게 파라메터/데이터를 문의
응답	응답(Response)	=/?	명령 수행 결과
수신	수신(Receive)	=/	내부/외부로부터 수신한 데이터/메시지
	알림(Indication)	#(수신 접두어)	내부/외부로부터 수신한 이벤트, 알림

실행/설정/읽기 명령은 사용자(호스트)와 직접 연결된 디바이스에서 명령이 수행되고, 전달/문의 명령은 무선 네트워크 상의 다른 디바이스로 해당 명령을 전송한다.

사용자(호스트)가 명령을 입력하면, Elicit 은 반드시 수행한 결과를 즉시 출력하는데, 이런 명령에 대한 결과를 출력하는 메시지를 **응답(Response)**이라고 한다. 응답에는 명령을 수행한 디바이스의 주소, 성공 여부와 기타 정보가 포함된다.

다른 디바이스로부터 수신한 메시지나, 내부 이벤트에 의해 발생한 메시지는 **수신(Receive)**이라고 한다.

응답과 수신은 Elicit 이 UART1 을 통해 출력하는 메시지로, 메시지를 생성한 디바이스의 주소와 전원 전압(배터리 전압), 수신한 신호의 세기(RSSI)가 기본으로 포함되어 출력된다.

응답과 수신은 Type 0 와 Type 1 의 두 가지 유형이 있다. Type 0 는 펌웨어 버전 1 과 동일하고, Type2 는 펌웨어 버전 2.0 부터 신규 추가되었다. 이 유형은 [\[AT+FORMAT\]](#) 명령으로 설정할 수 있다.

12.4 명령 형식

명령의 기본형식

AT+COMMAND[=/?] [DADDR] [PARAMETERS ...] <CR/LF/CRLF>

AT+	모든 명령은 "AT+"나 "at+"로 시작한다
COMMAND	명령어는 대문자로 입력한다. (펌웨어 버전 2.5 부터 소문자 입력도 가능하다)
=/?	해당 명령의 종류를 명시하는 구분자(separator). '=': 실행/설정/전달 구분자. '=' 다음에는 반드시 해당 명령을 수행할 디바이스의 주소가 있어야 한다. 자기 주소에서 실행하는 명령일 때 구분자와 주소를 모두 생략할 수 있다. '?': 읽기/문의 구분자.
DADDR	해당 명령을 수행할 최종 목적지 주소, 자기 주소를 사용할 수 있다. 자기 정보를 읽는(Get) 경우, 생략할 수 있다. 펌웨어 버전 2.5 부터 "THIS"를 입력하면 자기 주소로 인식한다.
PARAMETERS	명령마다 파라미터의 종류와 개수가 개별적으로 정의되어 있다.
CR/LF	AT 명령의 종료를 나타내는 문자, 둘 중의 하나만 입력되어도 되고, 둘 다 입력되어도 된다.

AT command 명령 형식 예제

[실행/전달 명령]

T>AT+RESET

명령만 사용하면 연결된 해당 디바이스에서 명령을 실행(Execute)한다.

T>AT+RESET=0001

주소를 사용. 자기 주소가 아니면 해당 주소로 명령을 전달(SEND)한다.

[정보 설정/전달 명령]

T>AT+TXPWR=0001 210

정보(설정 값/메시지)를 포함. 해당 주소로 정보를 전달(SEND)한다.

T>AT+SEND=0001 1 S "Hello"

해당 주소로 메시지를 전달(SEND)한다.

[읽기/문의 명령]

T>AT+TXPWR?

주소 없이 '?'만 사용하면 연결된 해당 디바이스의 정보를 반환한다.

T>AT+TXPWR?0001

주소를 사용. 자기 주소가 아니면 해당 주소로 정보를 요청(Querry)한다.

[DADDR에 "THIS" 입력]

T>AT+ECHO=THIS 1

자신의 address 대신 "THIS"를 입력하면 자신의 address로 인식한다.

[예외 형식]

T>AT

AT 시험 명령(AT Test Command). 'OK'만 반환한다.

T>AT#

Pass-Through Mode 해제 명령. 응답이 없는 명령.

12.5 응답/수신 형식

AT 명령에 대한 응답, 수신한 메시지의 기본 형식은 다음과 같다.

Format 0 응답/수신

AT+COMMAND=ADDR RSSI BATT [DATA ...] OK/FAIL [FAIL REASON]<LF>

Format 1 실행/쓰기/전달 응답

+COMMAND=ADDR [RSSI BATT DATA ...] OK/FAIL [FAIL REASON]<LF>

Format 1 읽기/문의 응답

+COMMAND?ADDR [RSSI BATT DATA ...] OK/FAIL [FAIL REASON]<LF>

Format 1 수신

#COMMAND=ADDR RSSI BATT [DATA ...] OK<LF>

AT+ 모든 명령에 대한 응답은 "AT+"로 시작한다. (Format 0)

유형 1(Format 1)은 AT 접두어를 생략하고 출력한다.

COMMAND 수행한 명령어.

=/? 명령과 주소 사이에 위치한 메시지의 종류를 구분할 수 있는 구분자.

유형 0(Format 0)은 응답과 수신 모두 항상 등호('=')를 구분자로 사용한다.

유형 1(Format 1)은 명령과 동일한 구분자로 응답한다

유형 1(Format 1)의 수신은 항상 우물정자('#')를 접두어로 사용한다.

ADDR 응답일 경우, 해당 명령을 수행한 디바이스의 주소.

수신일 경우, 해당 메시지/이벤트를 생성한 디바이스의 주소.

RSSI 수신신호 세기(Received signal strength indication, Unit: dBm).

ADDR 이 자신의 주소이면, 외부에서 수신한 것이 아니기 때문에 값은 0으로 반환한다.

ADDR 이 다른 디바이스이면 전송 경로 상의 첫 번째 디바이스로부터 수신한 신호의 세기.

BATT 해당 응답을 전달한 디바이스의 Vcc 전압 값. [100mV] 단위로 표기한다. 즉, 33 은 3.3V 이다.

DATA 읽은 파라메터(Get)나 요청한 상태에 대한 값, 명령마다 종류와 가짓수가 다르다.

OK/FAIL 해당 명령이 정상적으로 수행되면 OK, 실패하면 FAIL로 표기.

FAIL REASON 해당 명령이 실패했을 때 실패의 원인/이유.

LF 응답/수신은 항상 LF로 종료한다.

12.6 응답/수신 형식 유형(AT Format Type)

펌웨어 버전 2.0 부터 응답과 수신의 형식을 새로 추가하였다. AT 명령어 형식은 [AT+FORMAT] 명령으로 설정할 수 있다.

기존 응답/수신 형식	AT 형식 유형 0(AT Format Type 0)
신규 응답/수신 형식(권장)	AT 형식 유형 1(AT Format Type 1)

신규 유형 1은 명령과 응답, 수신 메시지를 명확히 구분할 수 있도록 개선하였다. 새로운 프로젝트를 시작하는 사용자는 유형 1을 사용하기 권장한다. 향후 버전부터는 유형 1만 지원할 예정이다.

표 15 AT 형식 유형 0: 응답/수신 종류

형식	종류	구분자	내용
응답	응답(Response)	=	명령 수행 결과
수신	수신(Receive)		내부/외부로부터 수신한 데이터/메시지
	알림(Indication)		내부/외부로부터 수신한 이벤트, 알림

표 16 AT 형식 유형 0의 기본 형식

성공응답	AT+COMMAND=ADDR RSSI BAT [. . .] OK
실패응답	AT+COMMAND=ADDR RSSI BAT FAIL [FAIL_REASON]
수신 메시지	AT+COMMAND=ADDR RSSI BAT [. . .] OK

형식 유형 0(Format Type 0)에서는 응답이나 수신의 형식이 동일하다.

표 17 AT 형식 유형 1: 응답/수신 종류

형식	종류	구분자	내용
응답	응답(Response)	=/?	명령 수행 결과. 명령과 동일한 구분자를 사용한다.
수신	수신(Receive)	#	내부/외부로부터 수신한 데이터/메시지
	알림(Indication)		내부/외부로부터 수신한 이벤트, 알림

표 18 AT 형식 유형 1의 기본 형식

실행/쓰기/전달 성공응답	+COMMAND=ADDR RSSI BAT [. . .] OK
실행/쓰기/전달 실패응답	+COMMAND=ADDR FAIL [FAIL_REASON]
읽기/문의 성공응답	+COMMAND?ADDR RSSI BAT [. . .] OK
읽기/문의 실패응답	+COMMAND?ADDR FAIL [FAIL_REASON]
수신 메시지	#COMMAND=ADDR RSSI BAT [. . .] OK

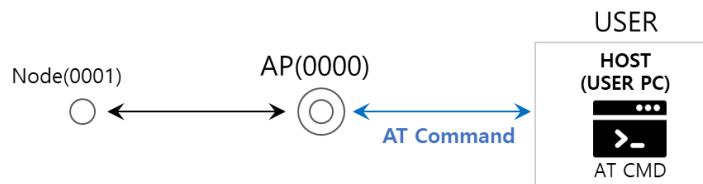
형식 유형 1(Format Type 1)에서는 AT 접두어가 생략된다. 그리고 명령이 실패한 경우, 응답에 데이터가 포함되지 않는다. 수신은 접두어 '#'를 사용한다.

12.6.1 형식 유형의 차이점

명령은 Format 0 와 Format 1 이 서로 동일하고, 응답과 수신의 형식이 다르다. Format 1 에서는 입력과 출력 메시지를 명확하게 구분하기 위해 모든 출력 메시지는 AT 를 제외하고 출력한다.

Format 1 에서는 구분자로 '=' 를 사용하는 명령에 대한 응답은 구분자로 '=' 를 사용하고, 구분자로 '?' 를 사용하는 명령에 대한 응답은 구분자로 '?' 를 사용한다. 또한, 외부로부터 수신한 모든 메시지와 이벤트 메시지는 구분자 '#' 으로 시작한다. 그리고 실행/설정/전달 명령의 응답에는 설정값/메시지 등의 정보가 포함되지 않는다.

다음은 Format 0 와 Format 1 의 사용 예제이다. 모든 명령은 AP 에 입력한다.



Format0/1	설정명령	AT+TXPWR=0000 210
Format0	설정응답(성공)	AT+TXPWR=0000 0 33 210 OK
Format0	설정응답(실패)	AT+TXPWR=0000 0 33 FAIL [INVALID_PARAMETER]
Format1	설정응답(성공)	+TXPWR=0000 0 33 OK
Format1	설정응답(실패)	+TXPWR=0000 FAIL [INVALID_PARAMETER]
Format0/1	읽기명령	AT+TXPWR?0000
Format0	읽기응답(성공)	AT+TXPWR=0000 0 33 210 OK
Format0	문의응답(실패)	AT+TXPWR=0000 0 33 FAIL [TX NO ACK]
Format1	읽기응답(성공)	+TXPWR?0000 -70 33 210 OK
Format1	문의응답(실패)	+TXPWR?0000 FAIL [TX NO ACK]
Format0/1	전달명령	AT+SEND=0001 1 S "Hello"
Format0	전달응답(성공)	AT+SEND=0001 -70 33 OK
Format0	전달응답(실패)	AT+SEND=0001 0 33 FAIL [TX NO ACK]
Format1	전달응답(성공)	+SEND=0001 -70 33 OK
Format1	전달응답(실패)	+SEND=0001 FAIL [TX NO ACK]
Format0	수신메시지	AT+SEND=0000 -70 33 1 S "Hello" OK
Format1	수신메시지	#SEND=0000 -70 33 1 S "Hello" OK
다른 디바이스가 설정을 변경할 경우(설정 명령을 전달받은 경우, Node[0001]의 출력)		
Format0	수신메시지	AT+TXPWR=0001 -70 33 210 OK
Format1	수신메시지	#TXPWR=0000 -70 33 210 OK

Format 0 에서 유의할 사항

- 설정과 읽기 명령에 대한 응답이 동일한 형식이다. 다른 디바이스가 설정을 변경할 경우도 동일한 형식으로 출력한다. 즉, 직접 설정한 경우와 다른 디바이스가 설정한 경우를 구분할 수 없다.
- SEND 명령(전달)에 대한 성공 응답과 수신메시지는 동일한 형식이나, 수신에만 메시지가 포함되어 있다.

12.7 AT 명령어 예제

다음 예제에서 형식 유형 0은 R0로 표기하고, 유형 1은 R1으로 표기한다.

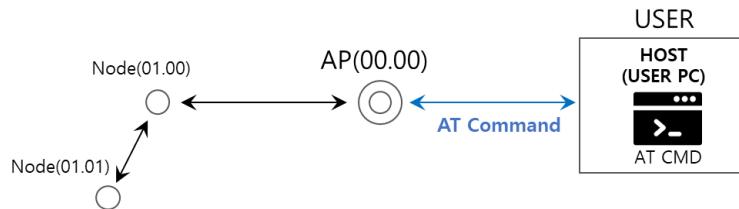
시험용 네트워크의 구성은 다음과 같다.

Hopping mode: 2

AT Command 연결: AP(00.00)

12.7.1 설정, 실행, 전달 명령 예제(Set/Execute/Send Examples)

설정/실행/전달 명령 예제



[실행/전달 명령]

T>AT+RESET

주소 없이 명령만 사용하면 연결된 해당 디바이스에서 명령을 실행(Execute)한다.

T>AT+RESET=0100

주소를 사용. 자기 주소가 아니면 해당 주소로 명령을 전달(SEND)한다.

[정보 설정/전달 명령]

T>AT+TXPWR=0100 210

해당 주소로 정보를 전달(SEND)한다.

[성공 응답]

R0<AT+PWR=0100 -70 33 OK

명령 전달 성공.

R1<+PWR=0100 -70 33 OK

수신 신호 크기 -70dBm, Node[0100]의 전원 전압 3.3V

[실패 응답]

R0<AT+PWR=0100 0 33 FAIL [NO_ACK]

명령 전달 실패. 수신확인을 받지 못함.

R1<+PWR=0100 FAIL [NO_ACK]

Format1은 실패응답에서 RSSI와 BAT 정보를 표시하지 않는다.

[전달 명령]

AT+SEND=0101 1 S "Hello"

Node[0101]로 메시지 전달.

AP[0000]에서 Hop 전달(0000 → 0100 → 0101)

[성공 응답]

R0<AT+SEND=0101 -70 33 OK

명령 전달 성공(0100으로 전달 성공).

R1<+SEND=0101 -70 33 OK

RSSI, BATT은 0100의 값이다.

[실패 응답]

R0<AT+SEND=0101 0 33 FAIL [NO_ACK]

명령 전달 실패. 수신확인을 받지 못함.

R1<+SEND=0101 FAIL [NO_ACK]

[예외 형식]

T>AT

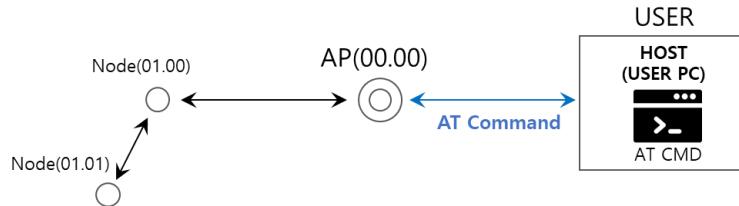
AT 시험 명령(AT Test Command). 'OK'만 반환한다.

T>AT#

Pass-Through Mode 해제 명령. 응답이 없는 명령.

12.7.2 읽기, 문의 명령 예제(Get/Querry Examples)

읽기/문의 명령 예제



[읽기 명령]

T>AT+TXPWR?

자신의 정보를 요청할 때 주소를 생략할 수 있다.

[성공 응답]

R0>AT+TXPWR=0000 0 33 210 OK

자신의 정보를 요청한 경우의 응답. RSSI는 0으로 고정됨.

R1>+TXPWR?0000 0 33 210 OK

Format1은 읽기 명령과 동일하게 ‘?’를 응답의 구분자로 쓴다.

[문의 명령]

T>AT+TXPWR?0101

Node[0101]의 정보를 요청: Hop 전달(0000 → 0100 → 0101)

[성공 응답]

R0>AT+TXPWR=0101 -70 33 210 OK

문의에 대한 응답. RSSI, BAT는 0100으로부터 받은 신호.

R1>#TXPWR?0101 -70 33 210 OK

[실패 응답]

R0>AT+TXPWR=0100 0 33 FAIL [NO_ACK]

명령 문의 실패. 수신확인을 받지 못함.

R1>+TXPWR?0100 FAIL [NO_ACK]

12.7.3 수신, 알림 메시지 예제(Receive/Indication Examples)

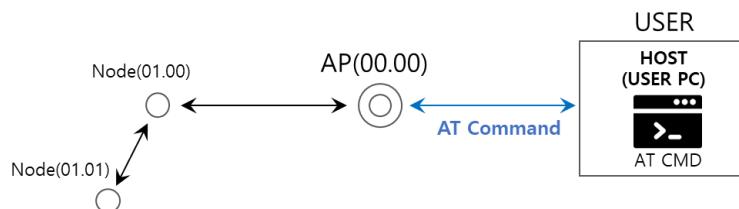
디바이스에서 발생한 알림과 다른 디바이스로부터 전달받은 모든 메시지를 출력하는 형식.

내부/외부로부터 발생한 이벤트 메시지를 수신한 경우 출력.

외부로부터 메시지를 수신(SEND, U1SND, U2SND)한 경우 출력.

외부로부터 정보(값)가 있는 설정 메시지를 수신한 경우 출력.

알림/메시지 수신 예제



[메시지 수신]

R0>AT+SEND=0101 -70 33 1 S “Door Open” Node[0101]이 송신한 메시지 수신. RSSI, BATT은 0100의 값이다.

R1>#SEND=0101 -70 33 1 S “Door Open”

R0>AT+U1SND=0100 -70 33 1 S “Hello” Node[0100]이 송신한 메시지 수신. RSSI, BATT은 0100의 값이다.

R1>#U1SND=0100 -70 33 1 S “Hello”

[알림 수신]

R0>AT+IND=0100 -70 33 DIEVT 4 1 OK

Node[0100]이 송신한 DI 이벤트 알림.

R1>#IND=0100 -70 33 DIEVT 4 1 OK

R0>AT+IND=0000 0 33 ADC 2456 OK AP[0000]의 ADC 측정 이벤트 알림.

R1>#IND=0000 0 33 ADC 2456 OK

[Node[0100]이 AP[0000]로 설정 메시지 송신, Node[0100]의 터미널에서 입력하고 수신한 예제]	
T>AT+DO=0000 1 1 0 0	NODE[0100]에서 입력한 명령
R0>AT+DO=0000 -72 33 1 1 0 0 OK	NODE[0100]에서 수신한 응답
R1>+DO=0000 -72 33 OK	Format1에서 +로 시작하는 메시지는 자신이 보낸 명령에 대한 응답
[AP[0000]가 Node[0100]으로부터 설정 메시지 수신, AP[0000]의 터미널 수신 메시지]	
R0>AT+DO=0100 -70 33 1 1 0 0 OK	NODE[0100]에서 수신한 메시지
R1>#D0=0100 -70 33 1 1 0 0 OK	Format1에서 #로 시작하는 메시지는 다른 디바이스가 보낸 명령에 대한 응답

13 AT Command 리스트

13.1 AT+HELP

AT command list 를 출력

실행(Execute)만 지원, 명령 구분자('?'/'='), DADDR 없이 명령 단독 실행

[Execute format]

AT+HELP<CR/LF/CRLF>

[Execute Response]

R< -----

R< Get: AT+<CMD>?<[DADDR]>

R< Set: AT+<CMD>=<DADDR> <PARAMS> [<PARAMS> ...]

R<-----

R<HELP - print all commands List

R<SN - print serial number. Usage: AT+SN?<ADDR>

...

13.2 AT+FORMAT

AT command 메시지의 포맷을 설정

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+FORMAT=<DADDR> <FORMAT><CR/LF/CRLF>

FORMAT 0 Default. Response 에 AT+가 출력됨

1 local response 에는 +, remote response 에는 #0| 출력

[Set/Send Response]

AT+FORMAT=<DADDR> <RSSI> <BATT> <FORMAT> OK<LF>

DADDR 목적지 SADDR

RSSI 수신신호세기[dBm]

BATT 배터리용량[100mV]

FORMAT FORMAT 모드

[Get/Query format]

AT+FORMAT?[DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+FORMAT=<OADDR> <RSSI> <BATT> <FORMAT> OK<LF>

OADDR 응답을 보낸 디바이스 주소

RSSI 수신신호세기[dBm]

BATT 배터리용량[100mV]

FORMAT FORMAT 모드

[Example format 0]

[Get]

T>AT+FORMAT?0100<LF>

```
R<AT+FORMAT=0100 0 33 0 OK<LF>          # FORMAT mode = 0 (Default)
[Set]
T>AT+FORMAT=0100 1<LF>
R<+FORMAT=0100 0 33 1 OK<CR>          # FORMAT mode = 1
[Example format 1]
[Get]
T>AT+FORMAT?0100<LF>
R<+FORMAT?0100 0 33 1 OK<LF>          # FORMAT mode = 1
[Set]
T>AT+FORMAT=0100 1<LF>
R<AT+FORMAT=0100 0 33 0 OK<CR>          # FORMAT mode = 0
```

13.3 AT+VER

ELICIT-R1/R1+의 버전 출력
읽기/문의(Get/Query) 지원

[Get format]

AT+VER? [DADDR]<CR/LF/CRLF>

DADDR 목적지 주소, 생략하면 자기 정보를 반환

[Get Response]

AT+VER=<DADDR> <RSSI> <BATT> <HW VER> <SW VER> <PRDCT TIME> <LOT NUM> OK<LF>

RSSI 수신신호세기[dBm]

BATT 배터리용량[100mV]

HW VER HW Revision version, 10(1.0) ~ : ELICIT-R1 / 20(2.0) ~ :ELICIT-R1+

SW VER SW version, 100: 1.00

PRDCT TIME Production time, 제품 검사 일시("2022/10/31")

LOT NUM Production LOT, 제품 생산 Lot, “L22Q301” : Lot '22, 3 분기 1st 생산분

[example format 0]

T>AT+VER?<LF>

R<AT+VER=FFFF 0 33 10 100 "2022/10/31 14:0:0" "L22Q301" OK<LF>

[example format 1]

T>AT+VER?<LF>

R<+VER?FFFF 0 33 10 100 "2022/10/31 14:0:0" "L22Q301" OK<LF>

13.4 AT+SN

ELICIT 모듈의 고유 ID(LONG ADDR) 출력

읽기(Get)만 지원, 다른 디바이스에게 문의(Query) 불가

16bit DADDR 지정 없음

[Get format]

AT+SN?<CR/LF/CRLF>

「Get Response」

AT+SN=<SADDR> <RSSI> <BATT> <LADDR> OK<LF>

SADDR	자신의 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
LADDR	LONG ADDR

[Example format 0]

```
T>AT+SN?<LF>
R<AT+SN=FFFF 0 33 1234567890ABCDEF OK<LF>
```

[Example format 1]

```
T>AT+SN?<LF>
R<+SN?FFFF 0 33 1234567890ABCDEF OK<LF>
```

13.5 AT+BOOT

Bootloader mode로 재부팅

실행(Execute)만 지원, 명령 구분자('?'/'='), DADDR 없이 명령 단독 실행

[Execute format]

```
AT+BOOT<CR/LF/CRLF>
```

[Execute Response format]

부트로더(loader) 모드로 재부팅(Reboot to bootloader mode)

펌웨어 다운로드(1)나 재부팅(2) 중 선택 가능

3초 이내 재부팅(2)을 선택하지 않으면 자동으로 다운로드 대기(XMODEM Receive) 상태로 변경

60초 이내 XMODEM 다운로드를 개시하지 않으면 자동으로 재부팅

[Execute Example]

```
T>AT+BOOT<CR>
```

```
ELICIT R1 Bootloader
-----
1. Download
2. Reboot
Begin download(Download within 1 minute)
cccccccccccc|
```

13.6 AT+RESET

Reset ELICIT 모듈

실행/전달/알림(Execute/Send/Indication) 지원

[Execute/Send format]

```
AT+RESET[=DADDR]<CR/LF/CRLF>
```

[Execute/Send Response]

```
AT+RESET=<DADDR> <RSSI> <BATT> OK<LF>
```

DADDR	목적지 SADDR
-------	-----------

RSSI	수신신호세기[dBm]
------	-------------

BATT	배터리 용량[100mV]
------	---------------

[Execute Example]

```
T>AT+RESET<CR>
R<AT+RESET=0000 0 33 OK<CR>
R< ... Booting messages ...
```

[Send Example]

#AP(0)에서 Node(0100)에게 Reset 명령 전달
T>AT+RESET=0100<CR>
#Reset 으로 Booting 하면서 Reset 의 원인을 IND 메시지로 알림
R<AT+IND=0100 0 33 RESET 40 OK<CR>

[Reset Indication format]

AT+IND=<DADDR> <RSSI> <BATT> <EVENT:"RESET"> <REASON> <CR/LF/CRLF>

REASON	Reset 원인
0x1:	Power on Reset
0x2:	Pin Reset (RESET Pad)
0x3:	Power on Reset + Pin Reset
0x4:	Wake up from Deep sleep
0x8:	Watchdog 0
0x10:	Watchdog 1
0x40:	Reset command(AT+RESET)
0x80:	VCC Fail(Digital Block)
0x100:	VCC Fail(Digital LE Block)
0x200:	VCC Fail(Digital DEC Block)
0x400:	VCC Fail(Analog Block)
0x800:	VCC Fail(IO Block)
0x1000:	Wake up from Deep sleep by BUTTON1
0x1001:	Wake up from Deep sleep by GPIO 1
0x1003:	Wake up from Deep sleep by GPIO 3
0x1004:	Wake up from Deep sleep by timeout

13.7 AT+FINIT

Initialize parameters

실행(Execute) 지원, 쓰기/전달/읽기/문의(Set/Send/Get/Query) 불가

[Execute format]

AT+FINIT=<DADDR> <INITLEVEL><CR/LF/CRLF>

INITLEVEL	Initialize level<0/1/2>
0	Network Initialize(PANID, SADDR, JOINSTS, RFCH, PHY) IND(EXIT) 알림 메시지를 자동 생성한다
1:	IO/EVENT Initialize(IO, Timer, Sleep, LINK EVENT) IND(INIT) 알림 메시지를 자동 생성한다
2:	Initialize all(Factory Initialize) IND(INIT) 과 IND(EXIT) 알림 메시지를 자동 생성한다

[Execute Response]

AT+FINIT=<SADDR> <RSSI> <BATT> OK<LF>

SADDR	자신의 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Example format 0]

T>AT+FINIT=0100 2<LF>

R<AT+FINIT=FFFF 0 33 OK<LF>

[Example format 1]

T>AT+FINIT=0100 2<LF>

R<+FINIT=FFFF 0 33 OK<LF>

13.8 AT+ADDR

Parameter: Short Address(SADDR), Elicit network address

읽기/쓰기(Get/Set) 지원, 다른 디바이스로 전달/문의(Send/Query) 불가

Role 이 AP 인 디바이스는 변경 불가

[Get format]

AT+ADDR?<CR/LF/CRLF>

[Get Response]

AT+ADDR=<SADDR> <RSSI> <BATT> <SADDR> OK<LF>

SADDR	자신의 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
SADDR	자신의 SADDR

[Set format]

AT+ADDR=<SADDR><CR>/<LF>

SADDR	변경하려는 SHORT ADDR
-------	------------------

[Set Response]

AT+ADDR=<SADDR> <RSSI> <BATT> <SADDR> OK<LF>

SADDR	변경된 자신의 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
SADDR	변경된 자신의 SADDR

[Example format 0]

[Get]

T>AT+ADDR?<LF>

R<AT+ADDR=FFFF 0 33 FFFF OK<LF> 현재 SADDR 은 FFFF

[Set]

T>AT+ADDR=1234<LF>

R>AT+ADDR=1234 0 33 1234 OK<LF> SADDR 을 1234 로 변경

[Example format 1]

[Get]

T>AT+ADDR?<LF>	
R<+ADDR?FFFF 0 33 FFFF OK<LF>	현재 SADDR 은 FFFF
[Set]	
T>AT+ADDR=1234<LF>	
R>+ADDR=1234 0 33 OK<LF>	SADDR 을 1234 로 변경

13.9 AT+PANID

Parameter: PAN ID

쓰기/읽기(Set/Get) 지원, 다른 디바이스로 전달/문의(Send/Query) 불가

PANID 를 변경하면 자동으로 JOINSTS 가 'J'로, OPMODE 가 1(Normal)로 변경됨.

[Set/Send format]

AT+PANID=<DADDR> <PANID><CR/LF/CRLF>

PANID 8bit PANID. 0 ~ 0xFF

[Set/Send Response format]

AT+PANID=<DADDR> <RSSI> <BATT> <PANID> OK<LF>

[Get format]

AT+PANID?<CR/LF/CRLF>

[Get Response]

AT+PANID=<SADDR> <RSSI> <BATT> <PANID> OK<LF>

SADDR 자신의 SADDR

RSSI 수신신호세기[dBm]

BATT 배터리용량[100mV]

PANID PAN ID, 초기화하면 FF 반환

[Example format 0]

[Get]

T>AT+PANID?<LF>

R<AT+PANID=FFFF 0 33 FF OK<LF> Alone 상태의 SADDR 과 PANID 는 0xFFFF / 0xFF

[Set]

T>AT+PANID=FFFF 15<LF>

R<AT+PANID=FFFF 0 33 15 OK<LF>

[Example format 1]

[Get]

T>AT+PANID?<LF>

R<+PANID?FFFF 0 33 FF OK<LF> Alone 상태의 SADDR 과 PANID 는 0xFFFF / 0xFF

[Set]

T>AT+PANID=FFFF 15<LF>

R<+PANID=FFFF 0 33 0K<LF>

13.10 AT+HOPMODE

Hopping mode

쓰기/읽기/문의(Set/Get/Query) 지원

[Set/Send format]

AT+HOPMODE=<DADDR> <HOP><CR/LF/CRLF>
 HOP 1/2/3/4, Hopping mode

[Set/Send Response format]

AT+HOPMODE=<DADDR> <RSSI> <BATT> <HOP> OK<LF>

[Get/Query format]

AT+HOPMODE?[DADDR]<CR/LF/CRLF>

[Get/Query Response format]

AT+HOPMODE=<DADDR> <RSSI> <BATT> <HOP> OK<LF>
 HOP 1/2/3/4, Hopping mode

[Example format 0]

[Get]

T>AT+HOPMODE?<LF>
 R<AT+HOPMODE=0000 0 33 2 OK #Hopping mode 2(00.00)

[Set]

T>AT+HOPMODE=0000 1<LF> #Set Hopping mode to 1
 R<AT+HOPMODE=0000 0 33 1 OK<LF>

[Example format 0]

[Get]

T>AT+HOPMODE?<LF>
 R<+HOPMODE?0000 0 33 2 OK<LF> #Hopping mode 2(00.00)

[Set]

T>AT+HOPMODE=0000 1<CR> #Set Hopping mode to 1
 R<+HOPMODE=0000 0 33 OK<CR>

13.11 AT+ROLE

Device role

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+ROLE=<DADDR> <ROLE><CR/LF/CRLF>
 ROLE Device role
 NA: Nothing, 공장초기화 이후 상태
 NODE: Node
 AP: AP

[Set/Send Response format]

AT+ROLE=<DADDR> <RSSI> <BATT> <ROLE> OK<LF>

[Get/Query format]

AT+ROLE?[DADDR]<CR/LF/CRLF>

[Get/Query Response format]

AT+ROLE=<DADDR> <RSSI> <BATT> <ROLE> OK<LF>

ROLE	Device role NA/NODE/AP	
[Example format 0]		
[Get]		
T>AT+ROLE?<CR>		
R<AT+ROLE=0000 0 33 AP OK	#Role AP	
[Set]		
T>AT+ROLE=FFFF AP<CR>	#Set Role as AP	
R<AT+ROLE=0000 0 33 AP OK<CR>		#Role 을 AP 로 설정하면 자동으로 주소가 0000 으로 바뀐다.
[Example format 1]		
[Get]		
T>AT+ROLE?<CR>		
R<+ROLE?0000 0 33 AP OK	#Role AP	
[Set]		
T>AT+ROLE=FFFF AP<CR>	#Set Role as AP	
R<+ROLE=0000 0 33 OK<CR>		#Role 을 AP 로 설정하면 자동으로 주소가 0000 으로 바뀐다.

13.12 AT+OPMODE

Operation mode				
읽기/문의(Get/Query) 지원				
[Get/Query format]				
AT+OPMODE?[DADDR]<CR/LF/CRLF>				
[Get/Query Response format]				
AT+ROLE=<DADDR> <RSSI> <BATT> <OPMODE> OK<LF>				
OPMODE	Operation mode			
0:	Init mode(Alone state)			
1:	Normal mode(Join state)			
2:	Sleep mode			
4:	Joinable mode			
[Example format 0]				
T>AT+OPMODE?<CR>				
R<AT+OPMODE=0000 0 33 1 OK	#Normal mode			
[Example format 1]				
T>AT+OPMODE?<CR>				
R<+OPMODE?0000 0 33 1 OK	#Normal mode			

13.13 AT+PHY

Config PHY mode				
쓰기/읽기/문의(Set/Get/Query) 지원				
[Set/Send format]				
AT+PHY=<DADDR> <PHY><CR/LF/CRLF>				
PHY	PHY mode 0/1			
L:	Long range mode, FSK			
H:	High data rate mode, MSK 80kbps			

R<+JOINSTS?FFFF 0 33 A OK<CR>

Alone 상태의 SADDR 과 PANID 는 FFFF

13.15 AT+PARAMS

기본 Parameters 정보: RFCH, TXPWR, PHY, JOINSTS, PANID, HOPMODE

모든 parameter 정보를 save & load(save point) 활용 가능)

읽기/다른 디바이스로 문의/실행(Get/Query/Execute) 지원

[Get/Query format]

AT+PARAMS?[DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+PARAMS=<DADDR> <RSSI> <BATT> <RFCH> <TXPWR> <PHY> <JOINSTS> <PANID> <HOPMODE> OK<LF>

DADDR	목적지 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
RFCH	RF Channel number(1 ~ 21)
TXPWR	Tx Power, 출력 파워(0 ~ 210, 0.1dBm 단위, 210 = 21.0 dBm)
PHY	PHY MODE <0/1> L: Long range mode (2FSK, 10kbps) H: High data rate mode (MSK, 80kbps) U: Ultra high data rate mode (MSK, 120kbps) ELICIT-R1+ 전용
JOINSTS	Network Status, A: Alone, J: Join
PANID	PAN ID, ALONE 상태에서는 FF 반환
HOPMODE	Hopping Mode, 1:(0000), 2:(00.00), 3:(00.0.0), 4:(0.0.0.0)

[Execute format]

AT+PARAMS=[DADDR] [ACT]<CR/LF/CRLF>

ACT “SAVE” or “LOAD”

SAVE: 현재 적용중인 params 를 저장

LOAD: SAVE 로 저장했던 params 를 적용

[Example format 0]

T>AT+PARAMS?0100<CR>

R<AT+PARAMS=0100 -65 33 15 210 H J E1 2 OK<CR>

0100	SADDR
-65	RSSI, -65dBm
33	BATT, 3.3V
15	RFCH #15
210	TXPWR 21.0dBm
H	PHY mode: High data rate mode(MSK, 80kbps)
J	Join status: Join state(1)
E1	PAN ID: E1
2	Hopping mode 2(00.00)

T>AT+PARAMS=0100 SAVE

R<AT+PARAMS=0100 0 33 OK

[Example format 1]

T>AT+PARAMS?0100<CR>

R<+PARAMS?0100 -65 33 15 210 H J E1 2 OK<CR>

```
T>AT+PARAMS=0100 LOAD
R<+PARAMS=0100 0 33 OK
```

13.16 AT+JOINREQ

JOIN 요청 메시지

상위 디바이스에서 자신의 하위 Node 를 네트워크에 가입시키기 위해 송신하는 Broadcasting 메시지
Alone 상태의 디바이스가 이 메시지를 수신하면 Joinable 상태로 상태를 변경한다.

실행/전달(Execute/Send)지원

[Execute/Send format]

```
AT+JOINREQ=<DADDR> [<RFCH>]<CR/LF/CRLF>
```

RFCH	JOINREQ 메시지를 송신하려는 RF CH 생략할 경우, 모든 RF CH 에 대해 송신한다.
------	---

[Execute/Send Response format]

```
AT+JOINREQ=<DADDR> <RSSI> <BATT> OK<LF>
```

[Example format 0]

[Execute/Send]

모든 RF CH 로 메시지 송신

```
T>AT+JOINREQ=0<LF>
```

```
R<AT+JOINREQ=0000 0 33 OK<LF>
```

[Example format 1]

[Execute/Send]

모든 RF CH 로 메시지 송신

```
T>AT+JOINREQ=0<LF>
```

```
R<+JOINREQ=0000 0 33 OK<LF>
```

13.17 AT+JOINRSP

JOIN 응답 메시지

JOINREQ 를 받은 디바이스에서 네트워크 가입을 위해 JOINREQ 를 송신한 디바이스로 보내는 응답 메시지
Alone 상태에서 JOINREQ 를 받고 1분 이내에 응답해야 한다.

실행(Execute)지원

[Execute format]

```
AT+JOINRSP=<DADDR><CR/LF/CRLF>
```

DADDR	FFFF, 네트워크 Join 이전이기에 항상 FFFF 로 송신한다.
-------	---------------------------------------

[Execute Response format]

```
AT+JOINRSP=<DADDR> <RSSI> <BATT> OK<LF>
```

[Receive JOINRSP format]

```
AT+JOINRSP=<DADDR> <RSSI> <BATT> <JOINREQ ADDR> <NODE LADDR> OK<LF>
```

JOINREQ ADDR	JOINREQ 를 송신한 디바이스의 address (어떤 디바이스한테 JOINREQ 를 받았는지 표시)
--------------	--

[Example format 0]

[Execute/Send]

```

T>AT+JOINRSP=FFFF<LF>
R<AT+JOINRSP=FFFF 0 33 OK<LF>
AP_R<AT+JOINRSP=FFFF -17 33 0 012ABCD12345678 OK<LF>
[Example format 1]
[Execute/Send]
T>AT+JOINRSP=FFFF<LF>
R<+JOINRSP=FFFF 0 33 OK<LF>
AP_R<#JOINRSP=FFFF -17 33 0 012ABCD12345678 OK<LF>

```

13.18 AT+PJOIN

JOIN 허가(permit) 메시지

JOINREQ 에 대한 응답인 JOINRSP 를 송신한 디바이스를 네트워크에 가입(Join)시키는 메시지
수신한 JOINRSP 에 포함된 네트워크 가입을 원하는 디바이스의 Long Address 를 이용한다.
JOINRSP 를 받고 1 분 이내에 응답해야 한다.

실행(Execute)지원

[Execute format]

```
AT+PJOIN=<DADDR> <LADDR><CR/LF/CRLF>
```

LADDR Join 을 시키려는 디바이스의 Long Address

[Execute Response format]

```
AT+PJOIN=<DADDR> <RSSI> <BATT> OK<LF>
```

[Example format 0]

[Execute/Send]

```
R<AT+JOINRSP=FFFF -17 33 0 012ABCD12345678 OK<LF>        #수신한 JOINRSP 에 포함된 LADDR 을 확인한다.
```

```
T>AT+PJOIN=0 012ABCD12345678<LF>
```

```
R<AT+PJOIN=0 0 33 OK<LF>
```

[Example format 1]

[Execute/Send]

```
R<#JOINRSP=FFFF -17 33 0 012ABCD12345678 OK<LF>        #수신한 JOINRSP 에 포함된 LADDR 을 확인한다.
```

```
T>AT+PJOIN=0 012ABCD12345678<LF>
```

```
R<+PJOIN=0 0 33 OK<LF>
```

13.19 AT+FJOIN

JOIN 허가(forced) 메시지

JOINREQ, JOINRSP 를 사용하지 않고, FJOIN command 만 사용하여 네트워크에 가입(Join)시키는 메시지
디바이스의 Long Address 를 이용하며, 디바이스의 할당 address 를 command 로 부여한다.
User 가 할당 address 를 정의하므로 JOINREQ, JOINRSP 명령과 같이 사용하면 address 가 중복될 수 있다.
Address 가 중복 안되게 주의하며 사용해야 한다.

실행(Execute)지원

[Execute format]

```
AT+FJOIN=<DADDR> <ASSIGN ADDR> <LADDR><CR/LF/CRLF>
```

DADDR 디바이스를 JOIN 시키려는 address(상위 address)

ASSIGN ADDR 할당 address

LADDR Join 을 시키려는 디바이스의 Long Address

[Execute Response format]

```
AT+FJOIN=<DADDR> <RSSI> <BATT> OK<LF>
```

[Example format 0]

[Execute/Send]

```
T>AT+FJOIN=0 1 0012ABCD12345678<LF>
```

```
R<AT+PJOIN=0 0 100 OK<LF>
```

```
R<AT+IND=0001 -19 100 JOIN 0001 0012ABCD12345678 OK<LF>
```

[Example format 1]

[Execute/Send]

```
T>AT+FJOIN=0 1 0012ABCD12345678<LF>
```

```
R<+PJOIN=0 0 100 OK<LF>
```

```
R<#IND=0001 -19 100 JOIN 0001 0012ABCD12345678 OK<LF>
```

13.20 AT+EXIT

Elicit 네트워크에서 등록 해제

AP에서 특정 Node의 연결을 해제할 때 사용한다.

이 명령을 수신한 Node는 IND(EXIT)메시지를 AP로 전송하고 자신의 네트워크 정보를 초기화한다.

실행/전달(Execute/Send)지원

[Execute format]

```
AT+EXIT=<DADDR> <CR/LF/CRLF>
```

DADDR	네트워크 등록을 해제하려는 디바이스의 주소
-------	-------------------------

[Execute Response format]

```
AT+EXIT=<DADDR> <RSSI> <BATT> OK<LF>
```

[Example format 0]

[Execute/Send]

```
T>AT+EXIT=0100<LF>
```

```
R<AT+EXIT=0 0 33 OK<LF>
```

```
R<AT+IND=0100 0 33 EXIT OK<LF> # Node 0100이 네트워크에서 Exit
```

[Example format 1]

[Execute/Send]

```
T>AT+EXIT=0100<LF>
```

```
R<+EXIT=0 0 33 OK<LF>
```

```
R<#IND=0100 0 33 EXIT OK<LF> # Node 0100이 네트워크에서 Exit
```

13.21 AT+ASADDR

하위 Node를 등록한 카운트

신규 Node를 등록(Join)시킬 때 사용하는 카운트, 이 값으로 주소를 할당한다.

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+ASADDR=<DADDR> <ASCNT><CR/LF/CRLF>
```

ASCNT	0~ADD_MAX, Associate counter, ADD_MAX는 Hopping mode에 따라 다름
-------	--

[Set/Send Response format]

AT+ASADDR=<DADDR> <RSSI> <BATT> <ASCNT> OK<LF>

[Get/Query format]

AT+ASADDR?<DADDR><CR/LF/CRLF>

[Get/Query Response format]

AT+ASADDR=<DADDR> <RSSI> <BATT> <ASCNT> OK<LF>

[Example format 0]

[Get]

T>AT+ASADDR?<LF>

R<AT+ASADDR=0000 0 33 8 OK<LF> #지금까지 할당한 Node 의 개수는 8(첫 번째 하위 노드만 해당)

[Set Example]

T>AT+ASADDR=0000 7<LF> #Count 를 7 로 변경, 다음 신규 Node 를 할당할 때 주소는 8

R<AT+ASADDR=0000 0 33 7 OK<CR>

[Example format 1]

[Get]

T>AT+ASADDR?<LF>

R<+ASADDR?0000 0 33 8 OK<LF> #지금까지 할당한 Node 의 개수는 8(첫 번째 하위 노드만 해당)

[Set Example]

T>AT+ASADDR=0000 7<LF> #Count 를 7 로 변경, 다음 신규 Node 를 할당할 때 주소는 8

R<+ASADDR=0000 0 33 OK<CR>

13.22 AT+ECHO

UART1 local echo back mode

공장초기화(Init level2) 이후에는 Disable 된다.

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+ECHO=<DADDR> <ENABLE><CR/LF/CRLF>

ENABLE Enable/disable the local Echo mode

0: Disable

1: Enable

[Set/Send Response format]

AT+ECHO=<DADDR> <RSSI> <BATT> <EN> OK<LF>

[Get/Query format]

AT+ECHO?<DADDR><CR/LF/CRLF>

[Get/Query Response format]

AT+ECHO=<DADDR> <RSSI> <BATT> <EN> OK<LF>

[Example format 0]

[Get]

T>AT+ECHO?<CR>

R<AT+ECHO=0000 0 100 1 OK

[Set]	
T>AT+ECHO=0000 1<CR>	#Enable UART local Echo mode
R<AT+ECHO=0000 0 100 1 OK<CR>	
[Example format 0]	
[Get]	
T>AT+ECHO?<CR>	
R<+ECHO?0000 0 33 1 OK	
[Set]	
T>AT+ECHO=0000 1<CR>	#Enable UART local Echo mode
R<+ECHO=0000 0 33 OK<CR>	

13.23 AT+BAUD

Configure UART baud rate
 쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]
 AT+BAUD=<DADDR> <PORT> <BAUD><CR/LF/CRLF>

PORT	1/2, UART Port
	1: UART1
	2: UART2
BOUD	1200 ~ 115200, Baud rate (bps)

[Set/Send Response format]
 AT+BAUD=<DADDR> <RSSI> <BATT> <PORT> <BAUD> OK<LF>

[Get/Query format]
 AT+BAUD?[DADDR]<CR/LF/CRLF>

[Get/Query Response format]
 AT+BAUD=<DADDR> <RSSI> <BATT> <U1_BOUD> <U2_BOUD>OK<LF>

U1_BAUD	1200~115200, Baud rate of UART1
U2_BAUD	1200~115200, Baud rate of UART2

[Example format 0]

[Get]	
T>AT+BAUD?<CR>	
R<AT+BAUD=0000 0 33 115200 9600 OK	#UART1: 115200, UART2: 9600
[Set]	
T>AT+BAUD=0000 2 38400<CR>	
R<AT+BAUD=0000 0 33 2 38400 OK<CR>	#Set UART2 baud rate to 38400

[Example format 1]

[Get]	
T>AT+BAUD?<CR>	
R<+BAUD?0000 0 33 115200 9600 OK	#UART1: 115200, UART2: 9600
[Set]	
T>AT+BAUD=0000 2 38400<CR>	
R<+BAUD=0000 0 33 OK<CR>	#Set UART2 baud rate to 38400

13.24 AT+SEND

사용자 데이터(User data)를 지정된 주소의 지정된 port로 전송하여 지정된 형식으로 출력
목적지 주소는 16bit SADDR과 64bit LADDR 모두 가능(64bit LADDR은 SENDL을 사용)
전달(Send) 지원

[Send format]

AT+SEND=<DADDR> <PORT> <FORMAT> <"USER DATA"><CR/LF/CRLF>

DADDR	User data를 전달할 목적지 주소, SADDR/LADDR 모두 가능
PORT	목적지에서 User data를 출력할 UART port<1/2> 1: UART1 2: UART2
FORMAT	User data의 출력 형식<S/SC/SL/SCL/B/BM> S: 문자열 출력(String format), User data를 그대로 출력한다. User data에 공백이 포함될 경우 큰따옴표(")를 앞뒤로 덧붙인다. 출력 포트가 UART1인 경우: AT+SEND 응답으로 출력 출력 포트가 UART2인 경우: 문자열만 출력 SC: 출력 포트가 UART2인 경우 문자열 + CR 출력 SL: 출력 포트가 UART2인 경우 문자열 + LF 출력 SCL: 출력 포트가 UART2인 경우 문자열 + CR + LF 출력 B: Hexadecimal로 입력된 User data를 binary로 출력(Binary format) BM: Binary 출력 + CRC16(Modbus RTU용) User data를 binary로 변환한 후, 이에 대한 CRC16을 덧붙여 출력 "USER DATA": 전송할 User data, 공백이 포함된 문자열은 큰따옴표(")를 붙인다. FORMAT이 B/M일 경우, 데이터를 hexadecimal로 공백 없이 입력한다.

[Send Response]

AT+SEND=<OADDR> <RSSI> <BATT> <OK/FAIL><LF>

OADDR	응답한 발신지 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Received device]

SEND 명령을 수신한 디바이스의 출력 형식

AT+SEND=<OADDR> <RSSI> <BATT> <"STRING"> OK<LF>

OADDR	발신지, 메시지를 보낸 디바이스의 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
STRING	수신한 문자열 메시지

[Example format 0]

[Example 1] NODE(0100)의 UART1으로 문자메시지 전송

T>AT+SEND=0100 1 S "Hello world"<LF>

0100	DADDR, 목적지 Node 주소
1	DD(Destination device)의 UART 포트 번호
S	DD의 출력 형식: String

R<AT+SEND=0100 -75 33 OK<LF>

0100	DADDR, 목적지 Node의 주소
-75	RSSI, 메시지 전송 후 수신한 ACK의 신호 세기, -75[dBm]

33	BATT 용량, ACK 를 보낸 디바이스의 BATT 용량, 100mV 단위.
OK	발신지(00.00)에서 명령 전송이 성공(ACK 수신확인)했을 경우, OK 로 응답

[목적지의 UART1 출력 메시지]

R<AT+SEND=0000 -80 33 "Hello world" OK<LF>
 0000 OADDR, 메시지를 보낸 발신지의 주소(AP)
 -80 RSSI, 수신한 메시지의 신호 세기, -80[dBm]
 33 BATT 용량, 메시지를 송신한 디바이스의 BATT 용량, 100mV 단위.
 OK 정상적으로 메시지를 수신했을 경우, OK 로 응답

[Example 2] NODE(0100)의 UART2로 Modbus 명령 전송

T>AT+SEND=0100 2 BM 0104000A0001<LF>
 0100 DADDR, 목적지 Node 주소
 2 DD(Destination device)의 UART 포트 번호
 BM DD 의 출력 형식: Binary + CRC16
 R<AT+SEND=0100 -75 33 OK<LF>

[목적지의 UART2 출력 데이터]

R<(01 04 00 0A 00 01 11 C8)h
 Binary 로 출력, 수신한 데이터에 대한 CRC16 을 추가전송(C811)h

[Example format 1]

[Example 1] NODE(0100)의 UART1으로 문자메시지 전송
 T>AT+SEND=0100 1 S "Hello world"<LF>
 R<+SEND=0100 -75 33 OK<LF>

[목적지의 UART1 출력 메시지]

R<#SEND=0000 -80 33 "Hello world" OK<LF>

[Example 2] NODE(0100)의 UART2로 Modbus 명령 전송

T>AT+SEND=0100 2 BM 0104000A0001<LF>
 R<+SEND=0100 -75 33 OK<LF>

[목적지의 UART2 출력 데이터]

R<(01 04 00 0A 00 01 11 C8)h
 Binary 로 출력, 수신한 데이터에 대한 CRC16 을 추가전송(C811)h

13.25 AT+U1SND

UART1 으로 입력되는 모든 데이터를 지정한 디바이스로 자동 전송하는 모드 설정

데이터 입력이 일정 시간 이상 중단되거나, 연속으로 입력된 크기가 128Byte 이상이 될 때 전송

Baud rate 가 9600 ~ 115200 은 2ms 이상 입력이 중단되면 전송

9600 미만은 약 2 byte 전송 시간 이상 중단되면 전송

UART1 자동 전송 모드가 설정된 이후에는 AT 명령을 입력해도 수행하지 않음.

자동 전송 모드 해제는 연속으로 "AT#" 입력하거나, 다른 디바이스에서 <FORMAT> 값을 0 으로 명령 전송

자기 자신의 UART1 으로 출력 설정은 되지 않음

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+U1SND=<DADDR> <U1DADDR> <PORT> <FORMAT><CR/LF/CRLF>	
DADDR	U1SND 명령을 수행할 목적지 주소
U1DADDR	UART1로 입력되는 데이터를 자동으로 전달할 목적지 주소
PORT	데이터를 출력할 UART port<1/2>
1:	UART1, 다른 디바이스로 전송 가능(U1DADDR이 DADDR과 달라야 함)
2:	UART2
FORMAT	User data의 출력 형식<0/P/H/B/S>
0:	UART1 입력 데이터 자동 전송 중지
P:	입력 데이터를 그대로 출력(Pass-through) 데이터 변환 없이 그대로 출력(UART1에서 AT 명령 형식 적용 안함) 입력이 문자열이 아닌 Binary data이면 출력도 Binary로 출력
X:	입력 데이터를 Hexadecimal 문자열로 변환하여 출력 UART1은 AT+U1SND 형식으로 출력, UART2은 Hexadecimal만 출력 입력("Hello") UART1 출력(AT+U1SND=0000 0 33 1 X "48656C6C6F") UART2 출력("48656C6C6F") 입력(01100101b) UART1 출력: AT+U1SND=0000 0 33 1 X "65" UART2 출력: "65"
B:	입력 데이터를 Binary로 변환하여 출력 입력 데이터는 Hexadecimal 문자열로 입력해야 함 UART1, UART2 동일하게 Binary 출력 입력("0A35") UART1 출력(0000101000110101)b UART2 출력(0000101000110101)b
S:	입력 데이터를 문자열로 출력(UART1은 제어문자 ³ 출력 안함) UART1은 AT+U1SND 형식으로 출력, UART2은 문자열 출력 입력("Hello") UART1 출력: AT+U1SND=0000 0 33 1 S "Hello" UART2 출력: "Hello" 입력(01100101b) UART1 출력: AT+U1SND=0000 0 33 1 S "e" UART2 출력: "e"

[Set/Send Response format]

AT+U1SND=<DADDR> <RSSI> <BATT> <OK/FAIL><LF>	
DADDR	목적지 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[목적지의 UART1 출력 메시지 format]

AT+U1SND=<OADDR> <RSSI> <BATT> <PORT:1> <FORMAT> <DATA> OK<LF>	
OADDR	발신지 주소
RSSI	수신신호세기[dBm]
BATT	메시지를 보낸 디바이스의 배터리용량[100mV]
PORT	데이터를 출력할 UART port<1>
FORMAT	User data의 출력 형식<0/P/H/S>
DATA	수신한 User data

[Get/Query format]

AT+U1SND? [DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+U1SND=<OADDR> <RSSI> <BATT> <U1DADDR> <PORT> <FORMAT> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
U1DADDR	UART1 으로 입력되는 데이터를 자동으로 전달할 목적지 주소
PORT	데이터를 출력할 UART port<1/2>
FORMAT	User data 의 출력 형식<0/P/H/S>

[Example format 0]

NODE(0100)의 UART1에 Modbus 장비를 연결하고,

입력되는 모든 데이터를 AP(0000)로 전송하여 hexadecimal 형식으로 UART Port1 으로 출력하도록 설정

T>AT+U1SND=0100 0 1 X<LF>

0100	DADDR, 명령을 수행할 디바이스 주소(NODE 0100)
0	UART1 데이터 송신 목적지 주소(AP)
1	UART 포트 번호
X	데이터 출력 형식(Hexadecimal)

AP 에서 수신한 NODE(0100)의 UART1 데이터

R<AT+U1SND=0100 -75 33 “010402ABCD0795” OK<LF>

0100	OADDR, 데이터를 보낸 발신지 주소
-75	RSSI, 무선 메시지 수신 신호 세기, -75[dBm]
33	BATT 용량, 메시지를 보낸 디바이스의 BATT 용량, 100mV 단위.
U1DATA	NODE(0100)의 UART2 로 입력된 데이터는 (010402ABCD0795h)

[Example format 1]

NODE(0100)의 UART1에 Modbus 장비를 연결하고,

입력되는 모든 데이터를 AP(0000)로 전송하여 hexadecimal 형식으로 UART Port1 으로 출력하도록 설정

T>AT+U1SND=0100 0 1 X<LF>

AP 에서 수신한 NODE(0100)의 UART1 데이터

R<+U1SND=0100 -75 33 “010402ABCD0795” OK<LF>

13.26 AT+U2SND

UART2 로 입력되는 모든 데이터를 지정한 디바이스로 자동 전송하는 모드 설정

데이터 입력이 일정 시간 이상 중단되거나, 연속으로 입력된 크기가 128Byte 이상이 될 때 전송

Baud rate 가 9600 ~ 115200 은 2ms 이상 입력이 중단되면 전송

9600 미만은 약 2 byte 전송 시간 이상 중단되면 전송

자기 자신의 UART2 로 출력 설정은 되지 않음

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+U2SND=<DADDR> <U2DADDR> <PORT> <FORMAT><CR/LF/CRLF>

DADDR	U2SND 명령을 수행할 목적지 주소
-------	----------------------

U2DADDR	UART2로 입력되는 데이터를 자동으로 전달할 목적지 주소
PORT	데이터를 출력할 UART port<1/2>
	1: UART1
	2: UART2, 다른 디바이스로 전송 가능(U2DADDR 0이 DADDR과 달라야 함)
FORMAT	User data의 출력 형식<0/P/H/B/S>
0:	UART2 입력 데이터 자동 전송 중지
P:	입력 데이터를 그대로 출력(Pass-through) 데이터 변환 없이 그대로 출력(UART1에서 AT 명령 형식 적용 안함) 입력이 문자열이 아닌 Binary data라면 출력도 Binary로 출력
X:	입력 데이터를 Hexadecimal 문자열로 변환하여 출력 UART1은 AT+U2SND 형식으로 출력, UART2은 Hexadecimal만 출력 입력("Hello") UART1 출력(AT+U2SND=0000 0 33 1 X "48656C6C6F") UART2 출력("48656C6C6F") 입력(01100101b) UART1 출력: AT+U2SND=0000 0 33 1 X "65" UART2 출력: "65" B: 입력 데이터를 Binary로 변환하여 출력 입력 데이터는 Hexadecimal 문자열로 입력해야 함 UART1, UART2 동일하게 Binary 출력 입력("0A35") UART1 출력(0000101000110101)b UART2 출력(0000101000110101)b S: 입력 데이터를 문자열로 출력(UART1으로 제어문자는 출력 안함) UART1은 AT+U2SND 형식으로 출력, UART2은 문자열 출력 입력("Hello<CR>") UART1 출력: AT+U2SND=0000 0 33 1 S "Hello" UART2 출력: "Hello<CR>" 입력(01100101b) UART1 출력: AT+U2SND=0000 0 33 1 S "e" UART2 출력: "e"

[Set/Send Response format]

AT+U2SND=<DADDR> <RSSI> <BATT> <OK/FAIL><LF>

DADDR	목적지 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[목적지의 UART1 출력 메시지 format]

AT+U2SND=<OADDR> <RSSI> <BATT> <PORT:1> <FORMAT> < DATA> OK<LF>

OADDR	발신자 주소
RSSI	수신신호세기[dBm]
BATT	메시지를 보낸 디바이스의 배터리용량[100mV]
PORT	데이터를 출력할 UART port<1>
FORMAT	User data의 출력 형식<0/P/H/B/S>
DATA	수신한 USER DATA

[Get/Query format]

AT+U2SND? [DADDR]<CR/LF/CRLF>

[Get/Query Response format]

AT+U2SND=<OADDR> <RSSI> <BATT> <U2DADDR> <PORT> <FORMAT> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
U2DADDR	UART2로 입력되는 데이터를 자동으로 전달할 목적지 주소
PORT	데이터를 출력할 UART port<1/2>
FORMAT	User data의 출력 형식<0/P/H/S>

[Example format 0]

NODE(0100)의 UART2에 IoT 장비의 UART 콘솔을 연결하고,

입력되는 모든 데이터를 AP(0000)로 전송하여 String 형식으로 UART Port2로 출력하도록 설정

T>AT+U2SND=0100 0 2 S<LF>

0100	DADDR, 명령을 수행할 디바이스 주소
0	데이터를 보낼 목적지 주소(AP)
2	UART 포트 번호
S	데이터 출력 형식(String)

NODE(0100)의 UART2에 연결된 IoT 장비가 “Door Open!<CR>” 메시지를 출력했을 때

AP에서 수신한 NODE(0100)의 데이터 출력 – UART2로 출력

R<“Door Open!<CR>”

13.27 AT+RFCH

Parameter: RF channel

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+RFCH=<DADDR> <RFCH><CR/LF/CRLF>

RFCH RF Channel number, 1 ~ 21

[Set/Send Response]

AT+RFCH=<DADDR> <RSSI> <BATT> <RFCH> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
RFCH	RF Channel number, 1 ~ 21

[Get/Query format]

AT+RFCH? [DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+RFCH=<OADDR> <RSSI> <BATT> <RFCH> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
RFCH	RF Channel number, 1 ~ 21

[Example format 0]
 [Get]
 T>AT+RFCH?<CR>
 R<AT+RFCH=0000 0 33 15 OK<CR>
 [Set]
 T>AT+RFCH=0 3<CR>
 R<AT+RFCH=0000 0 33 3 OK<CR>

[Example format 1]
 [Get]
 T>AT+RFCH?<CR>
 R<+RFCH?0000 0 33 15 OK<CR>
 [Set]
 T>AT+RFCH=0 3<CR>
 R<+RFCH=0000 0 33 0K<CR>

13.28 AT+TXPWR

Parameter: RF Transmit Power

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+TXPWR=<DADDR> <PWR><CR/LF/CRLF>
 PWR RF Tx Power in 0.1 dBm, 200 means 20.0dBm, 0 ~ 210

[Set/Send Response]

AT+TXPWR=<DADDR> <RSSI> <BATT> <PWR> OK<LF>
 DADDR 목적지 SADDR
 RSSI 수신신호세기[dBm]
 BATT 배터리용량[100mV]
 PWR RF Tx Power in 0.1 dBm, 200 means 20.0dBm, 0 ~ 210

[Get/Query format]

AT+TXPWR?<DADDR><CR/LF/CRLF>

[Get/Query Response]

AT+TXPWR=<OADDR> <RSSI> <BATT> <PWR> OK<LF>
 OADDR 응답을 보낸 디바이스 주소
 RSSI 수신신호세기[dBm]
 BATT 배터리용량[100mV]
 PWR RF Tx Power in 0.1 dBm, 200 means 20.0dBm

[Example format 0]

[Get]
 T>AT+TXPWR?<CR>
 R<AT+TXPWR=0000 0 33 210 OK<CR>
 [Set]
 T>AT+TXPWR=0 100<CR>
 R<AT+TXPWR=0000 0 33 10 OK<CR>

```
[Example format 1]
[Get]
T>AT+TXPWR?<CR>
R<+TXPWR?0000 0 33 210 OK<CR>
[Set]
T>AT+TXPWR=0 100<CR>
R<+TXPWR=0000 0 33 OK<CR>
```

13.29 AT+TONE

Transmit test carrier(for RF test)

실행/읽기(Execute/Get)지원, 읽기는 현재 TONE test option 값을 반환

Test 목적으로 RF 신호를 끊임없이 출력한다. RF channel을 점유하기 때문에 Test 용도로만 사용해야 한다.

[Execute format]

AT+TONE=<DADDR> <OPTION><CR/LF/CRLF>

DADDR	목적지 주소는 자기 SADDR 만 허용
OPTION	0: Stop 1: Transmit un-modulated carrier 2: Transmit modulated carrier(random data)

[Execute Response]

AT+TONE=<DADDR> <RSSI> <BATT> OK<LF>

DADDR	DADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Get format]

AT+TONE?[DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+TXPWR=<DADDR> <RSSI> <BATT> <OPTION> OK<LF>

DADDR	목적지 주소는 자기 SADDR 만 허용
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
OPTION	0: Stop, 1: Un-modulated carrier, 2: Modulated carrier(random data)

[Example format 0]

[Get]

T>AT+TONE?<CR>

R<AT+TONE=0000 0 33 0 OK<CR>

[Execute]

T>AT+TONE=0 1<CR>

R<+TONE=0000 0 33 OK<CR>

[Example format 1]

[Get]

T>AT+TONE?<CR>

R<+TONE?0000 0 33 0 OK<CR>

[Execute]

```
T>AT+TONE=0 1<CR>
R<+TONE=0000 0 33 OK<CR>
```

13.30 AT+BTN

Configure Button mode

Button1, Button2 공통 적용

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+BTN=<DADDR> <MODE><CR/LF/CRLF>
```

MODE	Button Mode
11	Input with Pull-up
12	Input with Pull-down

[Set/Send Response]

```
AT+BTN=<DADDR> <RSSI> <BATT> <MODE> OK<LF>
```

[Get/Query format]

```
AT+BTN?<DADDR><CR/LF/CRLF>
```

[Get/Query Response]

```
AT+BTN=<OADDR> <RSSI> <BATT> <MODE> OK<LF>
```

[Example format 0]

[Get]

```
T>AT+BTN?<CR>
```

```
R<AT+BTN=0000 0 33 11 OK<CR> # Button mode is Input with pull-up
```

[Set Example]

```
T>AT+BTN=0 12<CR>
```

```
R<AT+BTN=0000 0 33 12 OK<CR>
```

[Example format 1]

[Get]

```
T>AT+BTN?<CR>
```

```
R<+BTN?0000 0 33 11 OK<CR> # Button mode is Input with pull-up
```

[Set Example]

```
T>AT+BTN=0 12<CR>
```

```
R<+BTN=0000 0 33 OK<CR>
```

13.31 AT+LED

Configure LED mode

LED1, LED2 공통 적용

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+LED=<DADDR> <MODE><CR/LF/CRLF>
```

MODE	LED Mode
0	LED Off
20	Push-pull output

30 Open-drain output

[Set/Send Response]

AT+LED=<DADDR> <RSSI> <BATT> <MODE> OK<LF>

[Get/Query format]

AT+LED?[DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+LED=<OADDR> <RSSI> <BATT> <MODE> OK<LF>

[Example format 0]

[Get]

T>AT+LED?<CR>

R<AT+LED=0000 0 33 20 OK<CR> # LED mode is push-pull out

[Set]

T>AT+LED=0 30<CR> #Configure LED mode to open-drain

R<AT+LED=0000 0 33 30 OK<CR>

[Example format 0]

[Get]

T>AT+LED?<CR>

R<+LED?0000 0 33 20 OK<CR> # LED mode is push-pull out

[Set]

T>AT+LED=0 30<CR> #Configure LED mode to open-drain

R<+LED=0000 0 33 0K<CR>

13.32 AT+IO

Configure IO (GPIO and AUX)

AUX 는 Input(pull-up/down)만 설정 가능, 출력 설정 불가능.

GPIO1~8 모두 입력/출력 설정 가능, 입력 이벤트 설정은 GPIO4~8, AUX 만 가능

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

GPIO4~8, AUX 는 Digital 입력에 대한 Event trigger 를 설정할 수 있다.

GPIO1,2,3 은 Event trigger 를 설정할 수 없다.

AT+IO=<DADDR> <IO> <MODE> <TRIGGER><CR/LF/CRLF>

IO IO number

1 ~ 8 GPIO1 ~ GPIO8

9 AUX

MODE IO Mode

0 Input/Output Disable

10 Input

11 Input with Pull-up

12 Input with Pull-down

13 ADC(ELICIT-R1+ 전용)

20 Push-pull output

30 Open-drain output

31 Open-drain output with Pull-up

	40	Open-source output
	41	Open-source output with Pull-down
TRIGGER	Event trigger	
	0	Disable Event
	1	Positive edge
	2	Negative edge
	3	Both edge

[Set/Send Response format]

AT+IO=<DADDR> <RSSI> <BATT> <IO> <MODE> <TRIGGER>OK<LF>

[Get/Query format]

AT+IO? [<DADDR> <IO>]<CR/LF/CRLF>

[Get/Query Response format]

<IO>를 생략하면 IO1 ~ 9 까지 모두 출력하고, IO 번호를 지정하면 지정한 IO에 대한 결과만 출력한다.

AT+IO=<DADDR> <RSSI> <BATT> <IO> <MODE> <TRIGGER> OK<LF>

[Example format 0]

[Get]

T>AT+IO?<CR>

T>AT+IO=0000 0 33 1 11 0 OK	#GPIO#1: Input with pull-up, disable event
T>AT+IO=0000 0 33 2 12 0 OK	#GPIO#2: Input with pull-down, disable event
T>AT+IO=0000 0 33 3 00 0 OK	#GPIO#3: IO disabled, disable event
T>AT+IO=0000 0 33 4 20 0 OK	#GPIO#4: Push-pull output, disable event
T>AT+IO=0000 0 33 5 30 0 OK	#GPIO#5: Open-drain output, disable event
T>AT+IO=0000 0 33 6 31 0 OK	#GPIO#6: Open-drain output with pull-up, disable event
T>AT+IO=0000 0 33 7 40 0 OK	#GPIO#7: Open-source output, disable event
T>AT+IO=0000 0 33 8 11 2 OK	#GPIO#8: Input with pull-up, Negative edge triggered event
T>AT+IO=0000 0 33 9 11 1 OK	#AUX: Input with pull-up, Positive edge triggered event
T>AT+IO=0000 0 33 1 13 0 OK	#GPIO#1: ADC (ELICIT-R1+ 전용)

[Set]

T>AT+IO=0 8 0 0<CR> #GPIO#8: IO disabled, disable event

R<AT+IO=0000 0 33 8 0 0 OK<CR>

[Example format 1]

[Get]

T>AT+IO?<CR>

T>+IO?0000 0 33 1 11 0 OK	#GPIO#1: Input with pull-up, disable event
T>+IO?0000 0 33 2 12 0 OK	#GPIO#2: Input with pull-down, disable event
T>+IO?0000 0 33 3 00 0 OK	#GPIO#3: IO disabled, disable event
T>+IO?0000 0 33 4 20 0 OK	#GPIO#4: Push-pull output, disable event
T>+IO?0000 0 33 5 30 0 OK	#GPIO#5: Open-drain output, disable event
T>+IO?0000 0 33 6 31 0 OK	#GPIO#6: Open-drain output with pull-up, disable event
T>+IO?0000 0 33 7 40 0 OK	#GPIO#7: Open-source output, disable event
T>+IO?0000 0 33 8 11 2 OK	#GPIO#8: Input with pull-up, Negative edge triggered event
T>+IO?0000 0 33 9 11 1 OK	#AUX: Input with pull-up, Positive edge triggered event

[Set]

T>AT+IO=0 8 0 0<CR> #GPIO#8: IO disabled, disable event

```
R<+IO=0000 0 33 OK<CR>
```

13.33 AT+AUX

Read DI value of AUX

읽기/문의(Get/Query)지원

[Get/Query format]

```
AT+AUX? [<DADDR>]<CR/LF/CRLF>
```

[Get/Query Response format]

```
AT+AUX=<OADDR> <RSSI> <BATT> <DIN>OK<LF>
```

DIN	DIN value of AUX
-----	------------------

0	Logic low(DIN < 0.3*VCC)
1	Logic high(DIN > 0.7*VCC)
2	This GPIO is not an input

[Example format 0]

```
T>AT+AUX?<CR>
```

```
T>AT+AUX=0000 0 33 1 OK # DI value of AUX is 1
```

[Example format 1]

```
T>AT+AUX?<CR>
```

```
T>+AUX?0000 0 33 1 OK # DI value of AUX is 1
```

13.34 AT+DI

Read DI value of GPIO

읽기/문의(Get/Query)지원

[Get/Query format]

```
AT+DI? [<DADDR> <GPIO>]<CR/LF/CRLF>
```

GPIO	GPIO Number, 1 ~ 8
------	--------------------

[Get/Query Response format]

<GPIO>를 생략하면 GPIO1 ~ 8 까지 모두 출력하고, GPIO 값을 지정하면 지정한 GPIO에 대한 결과만 출력한다.

```
AT+DI=<OADDR> <RSSI> <BATT> [<DIN1> <DIN2> <DIN3> <DIN4> <DIN5> <DIN6> <DIN7> <DIN8>] OK<LF>
```

DINx	DIN value(0/1/2) of GPIO Number x
------	-----------------------------------

0	Logic low(DIN < 0.3*VCC)
1	Logic high(DIN > 0.7*VCC)
2	This GPIO is not an input

[Example format 0]

```
T>AT+DI?<CR>
```

```
T>AT+DI=0000 0 33 0 0 1 1 2 2 2 2 OK #GPIO#1,2: 0, GPIO#3,4: 1, GPIO#5,6,7,8: Not a DI
```

```
T>AT+DI?0 3<CR>
```

```
T>AT+DI=0000 0 33 1 OK #GPIO#3: 1
```

[Example format 1]

```
T>AT+DI?<CR>
```

```
T>+DI?0000 0 33 0 0 1 1 2 2 2 2 OK #GPIO#1,2: 0, GPIO#3,4: 1, GPIO#5,6,7,8: Not a DI
```

```
T>AT+DI?0 3<CR>
T>+DI=0000 0 33 1 OK          #GPIO#3: 1
```

13.35 AT+DO

Set and Get D0 value of GPIO

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+DO=<DADDR> <GPIO> <DOUT> [<DOUT_DEFAULT> <TGPERIOD>]<CR/LF/CRLF>

GPIO	GPIO Number, 1 ~ 8
DOUT	0/1 DOUT value, Available in Output mode
DOUT DEFAULT	0/1 DOUT default value, Available in Output mode
TGPERIOD	0~6000 Toggle Period in 100ms, 1=100ms, 6000=600000ms=600sec=10min 0 Toggle disable

[Set/Send Response format]

AT+DO=<DADDR> <RSSI> <BATT> <GPIO> <DOUT> <DOUT_DEFAULT> <TGPERIOD> OK<LF>

[Get/Query format]

AT+DO?<DADDR> <GPIO>]<CR/LF/CRLF>

[Get/Query Response format]

<GPIO>를 생략하면 GPIO1 ~ 8 까지 모두 출력하고, GPIO 값을 지정하면 지정한 GPIO에 대한 결과만 출력한다.

AT+DO=<DADDR> <RSSI> <BATT> <IO> <DOUT> <DOUT_DEFAULT> <TGPERIOD> OK<LF>

DOUTx	DOUT value(0/1/2) of GPIO Number x
0	Logic low
1	Logic high
2	This IO is not an output

[Example format 0]

[Get]

T>AT+DO?0 1<LF>

R<AT+DO=0000 0 33 1 1 0 0 OK #GPIO#1 DOUT value is 1, default value is 0, Toggle disabled

T>AT+DO?<LF>

R<AT+DO=0000 0 33 1 1 0 0 OK #GPIO#1 DOUT value is 1, default value is 0, Toggle disabled

R<AT+DO=0000 0 33 2 1 1 0 OK #GPIO#2 DOUT value is 1, default value is 1, Toggle disabled

R<AT+DO=0000 0 33 3 2 2 0 OK #GPIO#3 is not a Output mode, Toggle disabled

R<AT+DO=0000 0 33 4 0 0 0 OK #GPIO#4 DOUT value is 0, default value is 0, Toggle disabled

R<AT+DO=0000 0 33 5 1 0 3 OK #GPIO#5 DOUT value is 1, default value is 0, Toggle DOUT every 300ms

R<AT+DO=0000 0 33 6 1 0 3 OK #GPIO#6 DOUT value is 1, default value is 0, Toggle DOUT every 300ms

R<AT+DO=0000 0 33 7 2 2 0 OK #GPIO#7 is not a Output mode, Toggle disabled

R<AT+DO=0000 0 33 8 2 2 0 OK #GPIO#8 is not a Output mode, Toggle disabled

[Set]

T>AT+DO=0 6 1 0 0<LF> #GPIO#6 DOUT value is 1, default value is 0, Toggle disabled

R<AT+DO=0000 0 33 6 1 0 0 OK

T>AT+DO=0 6 1<LF> #GPIO#6 DOUT value is 1, default value is not setting, Toggle disabled

R<AT+DO=0000 0 33 6 1 0 0 OK

[Example format 1]

[Get]

```
T>AT+D0?0 1<LF>
R<+D0=0000 0 33 1 1 0 0 OK      #GPIO#1 DOUT value is 1, default value is 0, Toggle disabled
T>AT+D0?<LF>
R<+D0?0000 0 33 1 1 0 0 OK      #GPIO#1 DOUT value is 1, default value is 0, Toggle disabled
R<+D0?0000 0 33 2 1 1 0 OK      #GPIO#2 DOUT value is 1, default value is 1, Toggle disabled
R<+D0?0000 0 33 3 2 2 0 OK      #GPIO#3 is not a Output mode, Toggle disabled
R<+D0?0000 0 33 4 0 0 0 OK      #GPIO#4 DOUT value is 0, default value is 0, Toggle disabled
R<+D0?0000 0 33 5 1 0 3 OK      #GPIO#5 DOUT value is 1, default value is 0, Toggle DOUT every 300ms
R<+D0?0000 0 33 6 1 0 3 OK      #GPIO#6 DOUT value is 1, default value is 0, Toggle DOUT every 300ms
R<+D0?0000 0 33 7 2 2 0 OK      #GPIO#7 is not a Output mode, Toggle disabled
R<+D0?0000 0 33 8 2 2 0 OK      #GPIO#8 is not a Output mode, Toggle disabled
```

[Set]

```
T>AT+D0=0 6 1 0 0<LF>          #GPIO#6 DOUT value is 1, default value is 0, Toggle disabled
R<+D0=0000 0 33 OK
T>AT+D0=0 6 1<LF>              #GPIO#6 DOUT value is 1, default value is not setting, Toggle disabled
R<+D0=0000 0 33 OK
```

13.36 AT+VCC

Read ADC and VCC value

읽기/문의(Get/Query) 지원

[Get/Query format]

```
AT+VCC?[DADDR]<CR/LF/CRLF>
```

[Get/Query Response format]

```
AT+VCC=<OADDR> <RSSI> <BATT> <VCC> 0K<LF>
```

VCC VCC voltage in mV

[Example format 0]

[Get/Query]

```
T>AT+VCC?0100<CR>
```

```
R<AT+VCC=0100 -65 33 6 3319 OK
```

#VCC: 3319mV

[Example format 1]

[Get/Query]

```
T>AT+VCC?0100<CR>
```

```
R<+VCC?0100 -65 33 6 3319 OK
```

#VCC: 3319mV

13.37 AT+ADC

Read ADC value

읽기/문의(Get/Query) 지원

[Get/Query format]

```
AT+ADC?[DADDR]<CR/LF/CRLF>
```

ADC 결과는 IND 형식으로 반환한다.

[Indication of Get/Query command]

```
AT+IND=<OADDR> <RSSI> <BATT> <PARAMS : ADC> <ADC> OK<LF>
```

[Example format 0]

[Get/Query]

T>AT+ADC?0100<CR>

R<AT+IND=0100 -65 33 ADC 1823 OK #ADC: 1823mV

[Example format 1]

[Get/Query]

T>AT+ADC?0100<CR>

R<#IND=0100 -65 33 ADC 1823 OK #ADC: 1823mV

13.38 AT+IOADC

Read IO pin ADC value [ELICIT-R1+ 전용]

읽기/문의(Get/Query) 지원

[Get/Query format]

AT+IOADC?[DADDR]<CR/LF/CRLF>

ADC 결과는 IND 형식으로 반환한다.

[Indication of Get/Query command]

```
AT+IND=<OADDR> <RSSI> <BATT> <PARAMS : ADC> <I01> <I02> <I03> <I04> <I05> <I06> <I07> <I08> <AUX> OK<LF>
I01~8, AUX      ADC value(unit: mV)
```

[Example format 0]

[Get/Query]

T>AT+IOADC?0100<CR>

R<AT+IND=0100 -65 33 IOADC 3333 27 0 0 0 0 0 0 0 OK #IOADC: I01:3333mV I02:27mV

[Example format 1]

[Get/Query]

T>AT+ADC?0100<CR>

R<#IND=0100 -65 33 IOADC 3333 27 0 0 0 0 0 0 0 OK #IOADC: I01:3333mV I02:27mV

13.39 AT+VDAC

Set and Get the VDAC value

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+VDAC=<DADDR> <VDAC><CR/LF/CRLF>

VDAC VDAC value in mV, Do not exceed VCC

[Set/Send Response format]

AT+VDAC=<DADDR> <RSSI> <BATT> OK<LF>

[Get/Query format]

AT+VDAC?[DADDR]<CR/LF/CRLF>

[Get/Query Response format]

AT+VDAC=<OADDR> <RSSI> <BATT> <VDAC> OK<LF>

VDAC	VDAC value in mV
[Example format 0]	
[Get]	
T>AT+VDAC?<CR>	
R<AT+VDAC=0000 0 33 1233 OK	#Current VDAC value: 1233mV
[Set]	
T>AT+VDAC=0100 1823<CR>	#Set VDAC to 1823mV
R<AT+VDAC=0100 -70 33 1822 OK<CR>	
[Example format 1]	
[Get]	
T>AT+VDAC?<CR>	
R<+VDAC?0000 0 33 1233 OK	#Current VDAC value: 1233mV
[Set]	
T>AT+VDAC=0100 1823<CR>	#Set VDAC to 1823mV
R<+VDAC=0100 -70 33 OK<CR>	

13.40 AT+I2C

Read and write the I2C device(Master only)

하나의 명령으로 Read는 최대 16byte, Write는 최대 4byte 까지 가능

I2C의 Read 결과는 IND(Indication)으로 반환한다.

I2C_ADDR를 입력하면 16 진수로 인식합니다. 따라서 “0x68”이나 “68”로 입력해도 command는 16 진수(0x68)로 인식합니다.

쓰기/전달 (Set/Send) 지원

[Set/Send format – I2C Write]

AT+I2C=<DADDR> <I2C_ADDR> <TYPE> <REG> <LEN> [<DATA1> <DATA2> <DATA3> <DATA4>] <CR/LF/CRLF>

I2C_ADDR	Address of I2C device(slave address, 7bit)
TYPE	read/write flag(write : 1 read : 2)
REG	Register address(sub address, 8bit)
LEN	1~4 Data length to write
DATAx	Data to write

[Set/Send Response format]

AT+I2C=<DADDR> <RSSI> <BATT> OK<LF>

[Get/Query format – I2C Read]

AT+I2C=<DADDR> <I2C_ADDR>> <TYPE> <REG> <LEN> <CR/LF/CRLF>

I2C_ADDR	Address of I2C device(slave address, 7bit)
TYPE	read/write flag(write : 1 read : 2)
REG	Register address(sub address, 8bit)
LEN	1~16 Data length to read

[Get/Query Response format]

AT+IND=<OADDR> <RSSI> <BATT> <I2C> [<DATA1> ... <DATA16>] OK<LF>

[Example format 0]

[Get]

T>AT+I2C=0100 0x77 2 0x18 2<CR>	#Device Address 0x77, register address 0x18, read 2bytes
R<AT+IND=0100 0 33 I2C 12 34 OK	#Indication – I2C event, data: 12, 34
[Set]	
T>AT+I2C=0100 0x77 1 0x10 1 0x3F <CR>	#Device Address 0x77, register 0x10, Write 1 byte data(0x3F)
R<AT+I2C=0100 -70 33 0x77 1 0x10 1 0x3F OK<CR>	#Success – I2C write
[Example format 1]	
[Get]	
T>AT+I2C=0100 0x77 2 0x18 2<CR>	#Device Address 0x77, register address 0x18, read 2bytes
R<#IND=0100 0 33 I2C 12 34 OK	#Indication – I2C event, data: 12, 34
[Set]	
T>AT+I2C=0100 0x77 1 0x10 1 0x3F <CR>	#Device Address 0x77, register 0x10, Write 1 byte data(0x3F)
R<+I2C=0100 -70 33 OK<CR>	#Success – I2C write

13.41 AT+TIME

RTC Time
쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+TIME=<DADDR> <EPOCH><CR/LF/CRLF>
EPOCH Unix time

[Set/Send Response format]

AT+TIME=<DADDR> <RSSI> <BATT> OK<LF>

[Get/Query format]

AT+TIME?[DADDR]<CR/LF/CRLF>

[Get/Query Response format]

AT+TIME=<OADDR> <RSSI> <BATT> <"DATE"> <EPOCH> OK<LF>
DATE YYYY-MM-DD H:M:S 형식의 문자열
EPOCH Unix time

[Example format 0]

[Get]

T>AT+TIME?<CR>

R<AT+TIME=0000 0 33 "2023-02-01 20:05:10" 1675249510 OK

[Set]

T>AT+TIME=0100 1675249510<CR> #Set Time to "2023-02-01 20:05:10"

R<AT+TIME=0100 -70 33 "2023-02-01 20:05:10" 1675249510 OK<CR>

[Example format 1]

[Get]

T>AT+TIME?<CR>

R<+TIME?0000 0 33 "2023-02-01 20:05:10" 1675249510 OK

[Set]

T>AT+TIME=0100 1675249510<CR> #Set Time to "2023-02-01 20:05:10"

R<+TIME=0100 -70 33 OK<CR>

13.42 AT+XTAL

Internal Crystal tuning – Do not change the value

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

* 내부 Crystal 의 주파수를 미세 조정할 수 있는 명령, 임의로 변경하면 성능에 상당한 영향을 줄 수 있음.

AT+XTAL=<DADDR> <TUNE> <DELTA><CR/LF/CRLF>

TUNE X-tal Tune value

DELTA Delta of TUNE

[Set/Send Response format]

AT+XTAL=<DADDR> <RSSI> <BATT> <TUNE> <DELTA> OK<LF>

[Get/Query format]

AT+XTAL?<DADDR><CR/LF/CRLF>

[Get/Query Response format]

AT+XTAL=<OADDR> <RSSI> <BATT> <TUNE> <DELTA> OK<LF>

[Example format 0]

[Get]

T>AT+XTAL?<LF>

R<AT+XTAL=0000 0 33 70 40 OK<LF>

[Set]

T>AT+XTAL=0000 70 40<LF> #Do not change

R<AT+XTAL=0000 0 33 70 40 OK<LF>

[Example format 1]

[Get]

T>AT+XTAL?<LF>

R<+XTAL?0000 0 100 70 40 OK<LF>

[Set]

T>AT+XTAL=0000 70 40<LF> #Do not change

R<+XTAL=0000 0 100 OK<LF>

13.43 AT+DIEVT

Digital Input linked event, 5 개 slot 마다 개별 IO 지정 및 명령 설정

동일한 IO 에 대해 복수의 명령 설정 가능

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+DIEVT=<DADDR> <SLOT> <IO> <ACTIVATE> <(CMD)><CR/LF/CRLF>

SLOT 1~5 Slot index, AT command 를 저장할 수 있는 공간

IO 4~8 GPIO4 ~ GPIO8

9 AUX

ACTIVATE 0/1 Slot index 에 저장된 AT command 의 실행 여부

0 Disable

1 Enable

CMD Deep sleep 에서 깨어난 후 수행할 AT command

[Set/Send Response]

```
AT+DIEVT=<DADDR> <RSSI> <BATT> <SLOT> <IO> <ACTIVATE> <(CMD)>OK<LF>
```

[Get/Query format]

```
AT+DIEVT?<DADDR> <SLOT><CR/LF/CRLF>
SLOT          1~8, Slot index
```

[Get/Query Response]

GET/QUERY 명령에서 SLOT Option 을 생략하면 Slot index 를 1 ~ 5 까지 모두 출력

```
AT+DIEVT=<DADDR> <RSSI> <BATT> <SLOT> <IO> <ACTIVATE> <(CMD)> OK<LF>
```

[Example format 0]

[Get]

```
T>AT+DIEVT?100 1<LF>
R<AT+DIEVT=0100 0 33 1 4 1 (SEND=0 1 S "Door Open") OK<LF>
T>AT+DIEVT?100<LF>
R<AT+DIEVT=0100 0 33 1 4 1 (SEND=0 1 S "Door Open") OK<LF>
R<AT+DIEVT=0100 0 33 2 4 1 (DO=100 1 1 0) OK<LF>
R<AT+DIEVT=0100 0 33 3 5 1 (SEND=0 1 S "Door Close") OK<LF>
R<AT+DIEVT=0100 0 33 4 5 1 (DO=100 1 0 0) OK<LF>
R<AT+DIEVT=0100 0 33 5 6 1 (SEND=0 1 S "Window Open") OK<LF>
```

[Set]

```
T>AT+DIEVT=100 4 3 1 (DO=100 2 0)<LF>
R<AT+DIEVT=0100 0 33 4 3 1 (DO=100 2 0) OK<LF>
```

[Example format 1]

[Get]

```
T>AT+DIEVT?100 1<LF>
R<+DIEVT?0100 0 33 1 4 1 (SEND=0 1 S "Door Open") OK<LF>
T>AT+DIEVT?100<LF>
R<+DIEVT?0100 0 33 1 4 1 (SEND=0 1 S "Door Open") OK<LF>
R<+DIEVT?0100 0 33 2 4 1 (DO=100 1 1 0) OK<LF>
R<+DIEVT?0100 0 33 3 5 1 (SEND=0 1 S "Door Close") OK<LF>
R<+DIEVT?0100 0 33 4 5 1 (DO=100 1 0 0) OK<LF>
R<+DIEVT?0100 0 33 5 6 1 (SEND=0 1 S "Window Open") OK<LF>
```

[Set]

```
T>AT+DIEVT=100 4 3 1 (DO=100 2 0)<LF>
R<+DIEVT=0100 0 33 OK<LF>
```

13.44 AT+BTEVT

Button linked event, 2 개 slot 마다 개별 명령 설정

동일한 Button 에 대해 복수의 명령 설정 가능

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+BTEVT=<DADDR> <SLOT> <BTN> <ACTIVATE> <(CMD)><CR/LF/CRLF>
```

SLOT 1~2, Slot index, AT command 를 저장할 수 있는 공간

IO 1: Button1, 2: Button2

ACTIVATE 0/1, Slot index 에 저장된 AT command 의 실행 여부


```
AT+PTEVT=<DADDR> <RSSI> <BATT> OK<LF>
```

[Get/Query format]

```
AT+PTEVT?<DADDR> <SLOT><CR/LF/CRLF>
```

SLOT 1~4, Slot index

[Get/Query Response]

GET/QUERY 명령에서 Slot index Option 을 생략하면 Slot index 를 1 ~ 4 까지 모두 출력

```
AT+PTEVT=<DADDR> <RSSI> <BATT> <SLOT> <PERIOD> <(CMD)> OK<LF>
```

[Example format 0]

[Get]

```
T>AT+PTEVT?0 1<LF>
```

```
R<AT+PTEVT=0000 0 33 1 10 0 (ADC?100) OK<LF>
```

```
T>AT+PTEVT?<LF>
```

```
R<AT+PTEVT=0000 0 33 1 10 0 (ADC?100) OK<LF>
```

```
R<AT+PTEVT=0000 0 33 2 30 2 (SEND=0100 2 BM 0104000A0001) OK<LF>
```

```
R<AT+PTEVT=0000 0 33 3 600 0 (DO=0 3 1) OK<LF>
```

```
R<AT+PTEVT=0000 0 33 4 600 60 (DO=0 3 0) OK<LF>
```

[Set]

```
T>AT+PTEVT=0 4 30 3 (SEND=100 1 S "READ MODBUS")<LF>
```

```
R<AT+PTEVT=0000 0 33 4 30 3 (SEND=100 1 S "READ MODBUS") OK<LF>
```

[Example format 1]

[Get]

```
T>AT+PTEVT?0 1<LF>
```

```
R<+PTEVT?0000 0 33 1 10 0 (ADC?100) OK<LF>
```

```
T>AT+PTEVT?<LF>
```

```
R<+PTEVT?0000 0 33 1 10 0 (ADC?100) OK<LF>
```

```
R<+PTEVT?0000 0 33 2 30 2 (SEND=0100 2 BM 0104000A0001) OK<LF>
```

```
R<+PTEVT?0000 0 33 3 600 0 (DO=0 3 1) OK<LF>
```

```
R<+PTEVT?0000 0 33 4 600 60 (DO=0 3 0) OK<LF>
```

[Set]

```
T>AT+PTEVT=0 4 30 3 (SEND=100 1 S "READ MODBUS")<LF>
```

```
R<+PTEVT=0000 0 33 OK<LF>
```

13.46 AT+STEVT

Sleep timer event, Normal/Deep sleep 공통 적용

sleep 에서 깨어나서 slot 에 저장된 모든 명령 순차 실행, 최대 2 개의 slot 지원

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+STEVT=<DADDR> <SLOT> <ACTIVATE> <(CMD)><CR/LF/CRLF>
```

SLOT 1~2, Slot index, AT command 를 저장할 수 있는 공간

ACTIVATE 0/1, Slot index 에 저장된 AT command 의 실행 여부

0: Disable, 1: Enable

CMD Sleep 에서 깨어나서 수행할 AT command

[Set/Send Response]

```
AT+STEVT=<DADDR> <RSSI> <BATT> <SLOT> <ACTIVATE> <(CMD)> OK<LF>
```

[Get/Query format]

```
AT+STEVT?<DADDR> <SLOT><CR/LF/CRLF>
      SLOT          1~2, Slot index
```

[Get/Query Response]

GET/QUERY 명령에서 SLOT Option 을 생략하면 Slot index 를 1 ~ 2 까지 모두 출력

```
AT+STEVT=<DADDR> <RSSI> <BATT> <SLOT> <ACTIVATE> <(CMD)> OK<LF>
```

[Example format 0]

[Get]

```
T>AT+STEVT?100 1<LF>
R<AT+STEVT=0100 0 33 1 1 (ADC?100) OK<LF>
T>AT+STEVT?100<LF>
R<AT+STEVT=0100 0 33 1 1 (ADC?100) OK<LF>
R<AT+STEVT=0100 0 33 2 1 (DO=100 3 1) OK<LF>
[Set]
```

```
T>AT+STEVT=100 2 1 (DO=100 3 0)<LF>
R<AT+STEVT=0100 0 33 2 1 (DO=100 3 0) OK<LF>
```

[Example format 1]

[Get]

```
T>AT+STEVT?100 1<LF>
R<+STEVT?0100 0 33 1 1 (ADC?100) OK<LF>
T>AT+STEVT?100<LF>
R<+STEVT?0100 0 33 1 1 (ADC?100) OK<LF>
R<+STEVT?0100 0 33 2 1 (DO=100 3 1) OK<LF>
[Set]
```

```
T>AT+STEVT=100 2 1 (DO=100 3 0)<LF>
R<+STEVT=0100 0 33 OK<LF>
```

13.47 AT+NSEVT

Normal sleep wake up event, trigger IO 에 사용자 명령을 지정

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+NSEVT=<DADDR> <SLOT> <IO> <ACTIVATE> <(CMD)><CR/LF/CRLF>
      SLOT          1~3, Slot index, AT command 를 저장할 수 있는 공간
      IO            7/8/9, 7: GPIO7, 8: GPIO8, 9: AUX
      ACTIVATE     0/1, Slot index 에 저장된 AT command 의 실행 여부
                    0: Disable, 1: Enable
      CMD           Normal sleep 에서 깨어난 후 수행할 AT command
```

[Set/Send Response]

```
AT+NSEVT=<DADDR> <RSSI> <BATT> <SLOT> <IO> <ACTIVATE> <(CMD)>OK<LF>
```

[Get/Query format]

```
AT+NSEVT?<DADDR> <SLOT><CR/LF/CRLF>
```

SLOT 1~3, Slot index

[Get/Query Response]

GET/QUERY 명령에서 SLOT Option 을 생략하면 Slot index 를 1 ~ 3 까지 모두 출력

AT+NSEVT=<DADDR> <RSSI> <BATT> <SLOT> <IO> <ACTIVATE> <(CMD)> OK<LF>

[Example format 0]

[Get]

T>AT+NSEVT?100 1<LF>

R<AT+NSEVT=0100 0 33 1 7 1 (SEND=0 1 S “Door Open”) OK<LF>

T>AT+NSEVT?100<LF>

R<AT+NSEVT=0100 0 33 1 7 1 (SEND=0 1 S “Door Open”) OK<LF>

R<AT+NSEVT=0100 0 33 2 8 1 (DO=100 1 0) OK<LF>

R<AT+NSEVT=0100 0 33 3 9 1 (SEND=0 1 S “Door Close”) OK<LF>

[Set Example]

T>AT+NSEVT=100 2 8 1 (DO=100 2 0)<LF>

R<AT+NSEVT=0100 0 33 2 8 1 (DO=100 2 0)OK<LF>

[Example format 1]

[Get]

T>AT+NSEVT?100 1<LF>

R<+NSEVT?0100 0 33 1 7 1 (SEND=0 1 S “Door Open”) OK<LF>

T>AT+NSEVT?100<LF>

R<+NSEVT?0100 0 33 1 7 1 (SEND=0 1 S “Door Open”) OK<LF>

R<+NSEVT?0100 0 33 2 8 1 (DO=100 1 0) OK<LF>

R<+NSEVT?0100 0 33 3 9 1 (SEND=0 1 S “Door Close”) OK<LF>

[Set Example]

T>AT+NSEVT=100 2 8 1 (DO=100 2 0)<LF>

R<+NSEVT=0100 0 33 OK<LF>

13.48 AT+DSEVT

Deep sleep wake up event, trigger IO 에 사용자 명령을 지정

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+DSEVT=<DADDR> <SLOT> <IO> <ACTIVATE> <(CMD)><CR/LF/CRLF>

SLOT 1~2, Slot index, AT command 를 저장할 수 있는 공간

IO 1/3, 1: GPIO1, 3: GPIO3

ACTIVATE 0/1, Slot index 에 저장된 AT command 의 실행 여부

0: Disable, 1: ACTIVATE

CMD Deep sleep 에서 깨어난 후 수행할 AT command

[Set/Send Response]

AT+DSEVT=<DADDR> <RSSI> <BATT> <SLOT> <IO> <ACTIVATE> <(CMD)>OK<LF>

[Get/Query format]

AT+DSEVT?<DADDR> <SLOT><CR/LF/CRLF>

SLOT 1~2, Slot index

[Get/Query Response]

GET/QUERY 명령에서 SLOT Option 을 생략하면 Slot index 를 1 ~ 2 까지 모두 출력

AT+DSEVT=<OADDR> <RSSI> <BATT> <SLOT> <IO> <ACTIVATE> <(CMD)> OK<LF>

[Example format 0]

[Get]

T>AT+DSEVT?100 1<LF>

R<AT+DSEVT=0100 0 33 1 1 1 (SEND=0 1 S “Door Open”) OK<LF>

T>AT+DSEVT?100<LF>

R<AT+DSEVT=0100 0 33 1 1 1 (SEND=0 1 S “Door Open”) OK<LF>

R<AT+DSEVT=0100 0 33 2 3 1 (SEND=0 1 S “Door Close”) OK<LF>

[Set]

T>AT+DSEVT=100 4 1 1 (D0=100 2 0)<LF>

R<AT+DSEVT=0100 0 33 4 1 1 (D0=100 2 0) OK<LF>

[Example format 1]

[Get]

T>AT+DSEVT?100 1<LF>

R<+DSEVT?0100 0 33 1 1 1 (SEND=0 1 S “Door Open”) OK<LF>

T>AT+DSEVT?100<LF>

R<+DSEVT?0100 0 33 1 1 1 (SEND=0 1 S “Door Open”) OK<LF>

R<+DSEVT?0100 0 33 2 3 1 (SEND=0 1 S “Door Close”) OK<LF>

[Set]

T>AT+DSEVT=100 4 1 1 (D0=100 2 0)<LF>

R<+DSEVT=0100 0 33 OK<LF>

13.49 AT+DSWUP

Configure IO to wake from deep sleep

Deep sleep 을 깨울 수 있는 IO(GPIO1, GPIO3) 지정, Button1 은 항상 깨울 수 있음

Positive edge 로 Wake up 기능을 설정하면 Pad 는 자동으로 Pull-down 으로 설정된다.

Negative edge 로 Wake up 기능을 설정하면 Pad 는 자동으로 Pull-up 으로 설정된다.

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+DSWUP=<DADDR> <IO1_CONF> <IO3_CONF> <CR/LF/CRLF>

XXX_CONF Set each IO to wake up from sleep or not

0: Disable wake up

1: Enable wake-up on positive edge, and set IO Pad to pull-down

2: Enable wake-up on negative edge, and set IO Pad to pull-up

[Set/Send Response]

AT+DSWUP=<DADDR> <RSSI> <BATT> <IO1_CONF> <IO3_CONF>OK<LF>

[Get/Query format]

AT+DSWUP? [<DADDR>]<CR/LF/CRLF>

[Get/Query Response]

AT+DSWUP=<OADDR> <RSSI> <BATT> <IO1_CONF> <IO3_CONF> OK<LF>

[Example format 0]

```
[Get]
T>AT+DSWUP?100<LF>
R<AT+DSWUP=0100 0 33 1 1 OK<LF>

[Set]
T>AT+DSWUP=100 1 0<LF>          #Only GPIO1 can wake up from deep sleep mode
R<AT+DSWUP=0100 0 33 1 0 OK<LF>

[Example format 1]
[Get]
T>AT+DSWUP?100<LF>
R<+DSWUP?0100 0 33 1 1 OK<LF>
[Set]
T>AT+DSWUP=100 1 0<LF>          #Only GPIO1 can wake up from deep sleep mode
R<+DSWUP=0100 0 33 0K<LF>
```

13.50 AT+SLEEP

Sleep mode 실행, AP 는 Sleep mode로 진입하지 않는다
실행/전달(Execute/Send) 지원

[Execute/Send format]

AT+SLEEP=<DADDR> <PERIOD> <MODE> <AGAIN><CR/LF/CRLF>

PERIOD	0 ~ 2,073,600 [sec], Time out of Sleep timer, 최대 24 일 Set 0 to disable the sleep timer
MODE	Sleep mode 0/3/4 0: Disable 3: Normal sleep 4: Deep sleep
AGAIN	깨어난 후 다시 잠들기(Wake up and go back to sleep again) 0: Disable 1: Enable

[Execute Response]

AT+SLEEP=<DADDR> <RSSI> <BATT> <MODE> <AGAIN> OK<LF>

[Send Response]

AT+SLEEP=<OADDR> <RSSI> <BATT> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[%]

[Example format 0]

AP(0000)에서 Node(0100)에게 명령전달: Normal sleep에 들어가고, 1시간마다 깨어난 뒤 다시 잠들도록 설정

T>AT+SLEEP=0100 3600 3 1<LF>
R<AT+SLEEP=0100 -35 33 OK<LF>

[Example format 1]

AP(0000)에서 Node(0100)에게 명령전달: Normal sleep에 들어가고, 1시간마다 깨어난 뒤 다시 잠들도록 설정

T>AT+SLEEP=0100 3600 3 1<LF>
R<+SLEEP=0100 -35 33 OK<LF>

13.51 AT+PER

PER(Packet error rate) test 실행

실행/전달(Execute/Send) 지원

[Execute/Send format]

AT+PER=<DADDR> <TADDR> <NUM> <LENGTH> <INTERVAL> <CR/LF/CRLF>

DADDR Test 를 수행할 device 의 SADDR, 0| Device 에서 Test packet 을 송신한다.

TADDR PER 을 측정할 상대 device 의 SADDR, 0| Device 로 Test packet 을 전송

NUM 1~65535, Number of test trials, Test packet 을 전송할 횟수

LENGTH 0~64, Data length of test packet

INTERVAL 1~1000ms, Test packet 을 송신할 때 시간 간격

[Execute Response]

시험 중에는 진행 상황을 다음 메시지로 알림

AT+PER.TX=<TADDR> <RSSI> <BATT> <COUNT> <TCNT> <ACKCNT> <RMRSSI> OK/FAIL(ACK/TX)<LF>

TADDR Test packet 을 보낼 디바이스 주소

PER 시험의 상대 디바이스(PER test packet 을 수신한 device)는 수신한 Test packet 을 다음과 같이 알림

AT+PER.RX=<OADDR> <RSSI> <BATT> <INDEX> <RCNT> <TNUM> OK<LF>

OADDR Test packet 을 송신하는 디바이스 주소

시험이 종료되면 PER 결과를 다음 메시지로 알림

AT+IND=<OADDR> <RSSI> <BATT> <EVENT: "PER"> <TADDR> <NUM> <TX> <RX> <AVRSSI> <AVMRSSI> OK<LF>

[Example format 0]

AP(0)에서 node(100)으로 데이터 길이 32Byte 인 PER test packet 을 10ms 간격으로 20 회 전송

T>AT+PER=0 100 20 32 10 <LF>

R<AT+PER.TX=0100 -95 33 1 1 1 -82 OK<LF>

R<AT+PER.TX=0100 0 33 2 2 1 NA FAIL(ACK)<LF> #ACK 수신 못함

R<AT+PER.TX=0100 -96 33 3 3 2 -81 OK<LF>

R<AT+PER.TX=0100 0 33 4 3 2 NA FAIL(TX)<LF> #TX fail, RF busy

R<AT+PER.TX=0100 -90 33 5 4 3 -82 OK<LF>

...

R<AT+PER.TX=0100 -94 33 20 18 16 -82 OK<LF> #20 회 중 TX 성공 18 회, ACK 수신 16 회

R<AT+IND=0000 0 33 PER 0100 20 18 16 -94 -82 OK<LF> #결과: 수신평균 -94dBm, 상대방 수신평균-82dBm

상대 디바이스(0100)에서 출력되는 PER test packet 수신 알림 메시지

R<AT+PER.RX=0000 -82 33 1 1 20 OK<LF>

R<AT+PER.RX=0000 -81 33 3 2 20 OK<LF>

R<AT+PER.RX=0000 -82 33 5 3 20 OK<LF>

...

R<AT+PER.RX=0000 -82 33 20 16 20 OK<LF> #총 20 개 중 16 개 수신

[Example format 1]

AP(0)에서 node(100)으로 데이터 길이 32Byte 인 PER test packet 을 10ms 간격으로 20 회 전송

T>AT+PER=0 100 20 32 10 <LF>

R<+PER.TX=0100 -95 33 1 1 1 -82 OK<LF>

...

R<+PER.TX=0100 -94 33 20 18 16 -82 OK<LF> #20 회 중 TX 성공 18 회, ACK 수신 16 회

R<#IND=0000 0 33 PER 0100 20 18 16 -94 -82 OK<LF> #결과: 수신평균 -94dBm, 상대방 수신평균-82dBm

상대 디바이스(0100)에서 출력되는 PER test packet 수신 알림 메시지

R<#PER.RX=0000 -82 33 1 1 20 OK<LF>

...

R<#PER.RX=0000 -82 33 20 16 20 OK<LF>

#총 20 개 중 16 개 수신

13.52 AT+PER.TX

PER(Packet error rate) test 를 수행할 때 Test packet 을 전송한 결과를 알림

Test Packet 송신을 완료하고 ACK 를 수신하거나, Time out 이 될 경우 알림

무선 채널이 혼잡하여 송신하지 못할 경우(CSMA/CA 실패)에도 알림

알림(Indication) 메시지

[Indication format]

AT+PER.TX=<TADDR> <RSSI> <BATT> <COUNT> <TXCNT> <ACKCNT> <RMRSSI> OK/FAIL(ACK/TX)<LF>

TADDR PER 을 측정할 상대 device 의 SADDR

RSSI Test packet 에 대해 수신한 ACK 의 RSSI

COUNT 1~65535, Count of test trials, Test packet 을 전송한 Counter

TXCNT Test packet 송신 성공 count

ACKCNT 송신한 Test packet 에 대해 수신한 ACK count

RMRSSI 상대방이 수신한 Test packet 의 RSSI

[Example format 0]

AP(0)에서 node(100)으로 데이터 길이 32Byte 인 PER test packet 을 10ms 간격으로 20 회 전송

T>AT+PER=0 100 20 32 10 <LF>

R<AT+PER.TX=0100 -95 33 1 1 1 -82 OK<LF> #1 번째 송신 성공, 100 이 수신한 RSSI 는 -82dBm

R<AT+PER.TX=0100 0 33 2 2 1 NA FAIL(ACK)<LF> #ACK 수신 못함, 송신은 성공,

R<AT+PER.TX=0100 -96 33 3 3 2 -81 OK<LF> #3 번째 송신 성공, 100 이 수신한 RSSI 는 -82dBm

R<AT+PER.TX=0100 0 33 4 3 2 NA FAIL(TX)<LF> #TX fail, RF busy

R<AT+PER.TX=0100 -90 33 5 4 3 -82 OK<LF>

...

R<AT+PER.TX=100 -94 33 20 18 16 -82 OK<LF> #20 회 중 TX 성공 18 회, ACK 수신 16 회

R<AT+IND=0000 0 33 PER 0100 20 18 16 -94 -82 OK<LF> #결과: 수신평균 -94dBm, 상대방 수신평균-82dBm

[Example format 1]

AP(0)에서 node(100)으로 데이터 길이 32Byte 인 PER test packet 을 10ms 간격으로 20 회 전송

T>AT+PER=0 100 20 32 10 <LF>

R<+PER.TX=0100 -95 33 1 1 1 -82 OK<LF> #1 번째 송신 성공, 100 이 수신한 RSSI 는 -82dBm

...

R<+PER.TX=100 -94 33 20 18 16 -82 OK<LF> #20 회 중 TX 성공 18 회, ACK 수신 16 회

R<#IND=0000 0 33 PER 0100 20 18 16 -94 -82 OK<LF> #결과: 수신평균 -94dBm, 상대방 수신평균-82dBm

13.53 AT+PER.RX

PER(Packet error rate) test 를 수행할 때 Test packet 을 수신한 device 에서 수신 결과를 알림

Test Packet 을 수신했을 때 알림

새로 Test 를 수행하기 전에 수신 Counter 를 초기화하기 위해 반드시 AT+PERSTOP 명령을 수행해야 함

알림(Indication) 메시지

[Indication format]

AT+PER.RX=<OADDR> <RSSI> <BATT> <COUNT> <RXCNT> <TNUM> OK<LF>

OADDR	PER Test packet 을 송신한 device 의 SADDR
RSSI	수신한 Test packet 의 RSSI
COUNT	1~65535, Count of test trials, Test packet 을 전송한 Counter
RXCNT	Test packet 수신 count
TNUM	PER Test 의 총 전송 횟수

[Example format 0]

PER test packet 수신 알림 메시지, test packet 송신한 device 는 AP(0000)

R<AT+PER.RX=0000 -82 33 1 1 20 0K<LF> #20 개 중 1 번째 송신한 packet 수신 성공, 1 개 수신

R<AT+PER.RX=0000 -81 33 3 2 20 0K<LF> #20 개 중 3 번째 송신한 packet 수신 성공, 2 개 수신

R<AT+PER.RX=0000 -82 33 5 3 20 0K<LF> #20 개 중 5 번째 송신한 packet 수신 성공, 3 개 수신

...

R<AT+PER.RX=0000 -82 33 20 16 20 0K<LF> #20 개 중 20 번째 송신한 packet 수신 성공, 16 개 수신

[Example format 1]

PER test packet 수신 알림 메시지, test packet 송신한 device 는 AP(0000)

R<#PER.RX=0000 -82 33 1 1 20 0K<LF> #20 개 중 1 번째 송신한 packet 수신 성공, 1 개 수신

...

R<#PER.RX=0000 -82 33 20 16 20 0K<LF> #20 개 중 20 번째 송신한 packet 수신 성공, 16 개 수신

13.54 AT+PERSTOP

PER(Packet error rate) test 중지

PER Test packet 을 송신 수행 중이면, 이를 중지하고, 수신 중이면 수신 Counter 를 초기화 한다.

새로 PER 시험을 진행하기 전에 Test packet 을 수신하는 device 는 반드시 이를 수행해야 한다.

실행/전달(Execute/Send) 지원

[Execute/Send format]

AT+PERSTOP=<DADDR> <CR/LF/CRLF>

DADDR Test 를 중지할 device 의 SADDR

[Execute/Send Response]

AT+PERSTOP=<OADDR> <RSSI> <BATT> 0K<LF>

[Example format 0]

T>AT+PERSTOP=0 <LF>

R>AT+PERSTOP=0 0 33 0K<LF>

PER Test 상대방 device 도 중지(Counter 초기화)

T>AT+PERSTOP=100 <LF>

R>AT+PERSTOP=0100 0 33 0K<LF>

[Example format 1]

T>AT+PERSTOP=0 <LF>

R>+PERSTOP=0 0 33 0K<LF>

13.55 AT+SCAN

RF channel 에 대해 수신단의 RSSI 를 측정

AP 의 RF CH 을 선정하기 전에 혼잡하지 않은 주파수를 찾기 위해 사용한다.

실행/전달(Execute/Send) 지원

[Execute/Send format]

AT+SCAN=<DADDR> <CH> <SCAN_T><CR/LF/CRLF>

DADDR	측정을 수행할 Device 주소
CH	0~21, 측정할 RF Channel number
0	전체 CH에 대해 측정, 측정이 모두 끝난 후에 취합 결과를 IND로 응답
1~21	지정된 CH만 측정

SCAN_T 1~60[sec], Scan time, 측정 시간

[Execute/Send Response]

각 RF CH마다 결과를 응답한다.

AT+SCAN=<OADDR> <RSSI> <BATT> <CH> <MIN_RSSI> <MAX_RSSI> <AVG_RSSI> OK<LF>

DADDR	Scan 을 수행한 Device
RSSI	수신신호세기[dBm]
BATT	배터리용량[%]
CH	Scan 한 RF channel number
MIN_RSSI	Minimum RSSI
MAX_RSSI	Maximum RSSI
AVG_RSSI	Average RSSI

전체 CH을 Scan 한 경우, 다음 IND 메시지로 Maximum RSSI 가 가장 낮은 CH을 Report 한다.

AT+IND=<OADDR> <RSSI> <BATT> <EVENT:"SCAN"> <CH> <MAX_RSSI> <OK/FAIL><LF>

[Example format 0]

T>AT+SCAN=0 0 5<LF>	#모든 CH을 CH당 5초씩 Scan
R<AT+SCAN=0000 0 33 1 -109 -100 -108 OK<LF>	#CH1: Min(-109dBm), Max(-100dBm), Avg(-108dBm)
R<AT+SCAN=0000 0 33 2 -110 -105 -105 OK<LF>	#CH2: Min(-110dBm), Max(-105dBm), Avg(-105dBm)
R<AT+SCAN=0000 0 33 3 -97 -95 -99 OK<LF>	#CH3: Min(-97dBm), Max(-95dBm), Avg(-99dBm)
...	
R<AT+SCAN=0000 0 33 21 -107 -102 -105 OK<LF>	#CH21: Min(-107dBm), Max(-102dBm), Avg(-105dBm)
R<AT+IND=0000 0 33 SCAN 2 -105 OK<LF>	#전체 채널 중 CH2을 추천, MAX RSSI가 가장 낮음

[Example format 1]

T>AT+SCAN=0 0 5<LF>	#모든 CH을 CH당 5초씩 Scan
R<+SCAN=0000 0 33 1 -109 -100 -108 OK<LF>	#CH1: Min(-109dBm), Max(-100dBm), Avg(-108dBm)
R<+SCAN=0000 0 33 2 -110 -105 -105 OK<LF>	#CH2: Min(-110dBm), Max(-105dBm), Avg(-105dBm)
R<+SCAN=0000 0 33 3 -97 -95 -99 OK<LF>	#CH3: Min(-97dBm), Max(-95dBm), Avg(-99dBm)
...	
R<+SCAN=0000 0 33 21 -107 -102 -105 OK<LF>	#CH21: Min(-107dBm), Max(-102dBm), Avg(-105dBm)
R<#IND=0000 0 33 SCAN 2 -105 OK<LF>	#전체 채널 중 CH2을 추천, MAX RSSI가 가장 낮음

13.56 AT+TIMEOUT

버튼으로 Join이나 초기화를 할 때, 남은 사용자 입력 시간을 알림

알림(Indication) 메시지

[Indication format]

AT+TIMEOUT=<OADDR> <RSSI> <BATT> <TIMEOUT> OK<LF>

OADDR Timer 작동 중인 Device

TIMEOUT [sec], 사용자 입력을 할 수 있는 남은 시간

[Example format 0]

버튼으로 초기화를 할 때, 초기화 모드 진입 후에 IO/Event 초기화를 위해 버튼 2를 누르고 있으면

R<AT+TIMEOUT=0000 0 33 5 OK<LF>	#Timer 시작, 5 초 남음
R<AT+TIMEOUT=0000 0 33 4 OK<LF>	#Timer 4 초 남음
R<AT+TIMEOUT=0000 0 33 3 OK<LF>	#Timer 3 초 남음
R<AT+TIMEOUT=0000 0 33 2 OK<LF>	#Timer 2 초 남음
R<AT+TIMEOUT=0000 0 33 1 OK<LF>	#Timer 1 초 남음
R<AT+TIMEOUT=0000 0 33 0 OK<LF>	#Timer 만료, 버튼을 떼도 됨
R<AT+IND=0000 0 33 INIT 1 OK<LF>	#초기화 완료(IO/Event 초기화)

[Example format 1]

버튼으로 초기화를 할 때, 초기화 모드 진입 후에 IO/Event 초기화를 위해 버튼 2를 누르고 있으면

R<+TIMEOUT=0000 0 33 5 OK<LF>	#Timer 시작, 5 초 남음
..	
R<+TIMEOUT=0000 0 33 1 OK<LF>	#Timer 1 초 남음
R<+TIMEOUT=0000 0 33 0 OK<LF>	#Timer 만료, 버튼을 떼도 됨
R<#IND=0000 0 33 INIT 1 OK<LF>	#초기화 완료(IO/Event 초기화)

13.57 AT+REPORT

IND 메시지를 자동으로 AP로 전송하는 기능 설정

쓰기/전달/읽기/문의(Set/Send/Get/Query) 지원

[Set/Send format]

AT+REPORT=<DADDR> <EN><CR/LF/CRLF>

EN	0/1, IND 메시지 전송 기능
0	IND 메시지 전송 기능 중지
1	IND 메시지 전송 기능 활성화

[Set/Send Response]

AT+REPORT=<DADDR> <RSSI> <BATT> <EN> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
EN	IND 메시지 전송 모드

[Get/Query format]

AT+REPORT?[DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+REPORT=<OADDR> <RSSI> <BATT> <EN> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]
EN	IND 메시지 전송 모드

[Example format 0]

[Get]

T>AT+REPORT?0100<LF>

R<AT+REPORT=0100 0 33 0 OK<LF> # IND 자동 전송 기능 중지

[Set]

```
T>AT+REPORT=0100 1<LF>
R<AT+REPORT=0100 0 33 1 OK<CR>          # IND 자동 전송 기능 활성화
[Example format 1]
[Get]
T>AT+REPORT?0100<LF>
R<+REPORT?0100 0 33 0 OK<LF>          # IND 자동 전송 기능 중지
[Set]
T>AT+REPORT=0100 1<LF>
R<+REPORT=0100 0 33 OK<CR>          # IND 자동 전송 기능 활성화
```

13.58 AT+TXRTY

Packet TX 실패 시 자동으로 재전송하는 횟수 설정 LBT mode 전용.

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

```
AT+TXRTY=<DADDR> <CNT><CR/LF/CRLF>
      CNT      0~8,    재전송 횟수
```

[Set/Send Response]

```
AT+TXRTY=<DADDR> <RSSI> <BATT> <CNT> OK<LF>
      DADDR      목적지 SADDR
      RSSI       수신신호세기[dBm]
      BATT       배터리용량[100mV]
```

[Get/Query format]

```
AT+TXRTY?[DADDR]<CR/LF/CRLF>
```

[Get/Query Response]

```
AT+TXRTY=<OADDR> <RSSI> <BATT> <CNT> OK<LF>
      OADDR      응답을 보낸 디바이스 주소
      RSSI       수신신호세기[dBm]
      BATT       배터리용량[100mV]
```

[Example format 0]

[Get]

```
T>AT+TXRTY?0100<LF>
R<AT+TXRTY=0100 0 33 8 OK<LF>
```

[Set]

```
T>AT+TXRTY=0100 3<LF>
R<AT+TXRTY=0100 0 33 3 OK<CR>
```

[Example format 1]

[Get]

```
T>AT+TXRTY?0100<LF>
R<+TXRTY?0100 0 33 8 OK<LF>
```

[Set]

```
T>AT+TXRTY=0100 3<LF>
R<+TXRTY=0100 0 33 OK<CR>
```

13.59 AT+NAME

String data 를 쓰고 읽는 명령

쓰기/전달/읽기/문의(Set/Send/Get/Query)지원

[Set/Send format]

AT+NAME=<DADDR> <STRING DATA><CR/LF/CRLF>

STRING DATA	string data(최대 10Byte)
-------------	------------------------

[Set/Send Response]

AT+NAME=<DADDR> <RSSI> <BATT> <STRING DATA> OK<LF>

DADDR	목적지 SADDR
-------	-----------

RSSI	수신신호세기[dBm]
------	-------------

BATT	배터리용량[100mV]
------	--------------

[Get/Query format]

AT+NAME?[DADDR]<CR/LF/CRLF>

[Get/Query Response]

AT+NAME=<OADDR> <RSSI> <BATT> <STRING DATA> OK<LF>

OADDR	응답을 보낸 디바이스 주소
-------	----------------

RSSI	수신신호세기[dBm]
------	-------------

BATT	배터리용량[100mV]
------	--------------

[Example format 0]

[Get]

T>AT+NAME?0100<LF>

R<AT+NAME=0100 0 33 OK<LF> #아무것도 저장되어 있지 않을 때

[Set]

T>AT+NAME=0100 ELICIT<LF>

R<AT+NAME=0100 0 33 ELICIT OK<CR>

[Example format 1]

[Get]

T>AT+NAME?0100<LF>

R<+NAME?0100 0 33 ELICIT OK<LF>

[Set]

T>AT+NAME=0100 ELICITR1<LF>

R<+NAME=0100 0 33 OK<CR>

13.60 AT+LINKMODE

LBT mode 또는 ECH mode(Elicit Channel Hopping)를 설정.

쓰기/읽기(Set/Get)지원

ELICIT-R1+ 전용. ELICIT-R1 은 LBT 고정

[Set format]

AT+LINKMODE=<DADDR> <MODE><CR/LF/CRLF>

DADDR	수행할 Device 주소
-------	---------------

MODE	0: LBT mode(Default)
------	----------------------

	1: ECH mode
--	-------------

[Set Response]

AT+LINKMODE=<DADDR> <RSSI> <BATT> <MODE> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Get format]

AT+LINKMODE?<DADDR><CR/LF/CRLF>

[Get Response]

AT+LINKMODE=<OADDR> <RSSI> <BATT> <MODE> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Example format 0]

[Get]

T>AT+LINKMODE?<LF>

R<AT+LINKMODE=0000 0 33 0 OK<LF>

[Set]

T>AT+LINKMODE=0 1<LF>

R<AT+LINKMODE=0000 0 33 1 OK<LF>

[Example format 1]

[Get]

T>AT+LINKMODE?<LF>

R<+LINKMODE?0000 0 33 0 OK<LF>

[Set]

T>AT+LINKMODE=0 1<LF>

R<+LINKMODE=0000 0 33 0K<LF>

13.61 AT+BCN

ECH 전용 command. Beacon 을 설정.

쓰기/읽기(Set/Get)지원

ELICIT-R1+ 전용

[Set format]

AT+BCN=<DADDR> <BCN_INTVL><CR/LF/CRLF>

DADDR	수행할 Device 주소
BCN_INTVL	1/2/4, beacon 의 interval mode 설정
	1: 100ms
	2: 200ms
	4: 400ms

[Set Response]

AT+BCN=<DADDR> <RSSI> <BATT> <BCN_INTVL> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]

BATT	배터리용량[100mV]
------	--------------

[Get format]

AT+BCN?<DADDR><CR/LF/CRLF>

[Get Response]

AT+BCN=<OADDR> <RSSI> <BATT> <BCN_INTVL> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Example format 0]

[Get]

T>AT+BCN?<LF>

R<AT+BCN=0000 0 33 2 OK<LF>

[Set]

T>AT+BCN=0 1 1<LF>

R<AT+BCN=0000 0 33 1 OK<LF>

[Example format 1]

[Get]

T>AT+BCN?<LF>

R<+BCN?0000 0 33 2 OK<LF>

[Set]

T>AT+BCN=0 1<LF>

R<+BCN=0000 0 33 OK<LF>

13.62 AT+ECHLIST

ECH 전용 command. Hopping channel list 를 확인/재설정

쓰기/읽기(Set/Get)지원

ELICIT-R1+ 전용

[Set format]

AT+ECHLIST=<DADDR> <RANDOM SEED><CR/LF/CRLF>

DADDR	수행할 Device 주소
RANDOM SEED	재설정 값

[Set Response]

AT+ECHLIST=<DADDR> <RSSI> <BATT> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Get format]

AT+ECHLIST?<DADDR><CR/LF/CRLF>

[Get Response]

AT+ECHLIST=<OADDR> <RSSI> <BATT> <SEED> <CH_1> <CH_2> <CH_3> ... <CH_18> OK<LF>

OADDR	응답을 보낸 디바이스 주소
-------	----------------

RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Example format 0]

[Get]

T>AT+ECHLIST?<LF>

R<AT+ECHLIST=0000 0 33 0 5 9 12 11 2 8 16 7 18 3 4 9 6 10 13 19 17 14 20 OK<LF>

[Set]

T>AT+ECHLIST=0 0<LF>

R<AT+ECHLIST=0000 0 33 OK<LF>

[Example format 1]

[Get]

T>AT+ECHLIST?<LF>

R<+ECHLIST?0000 0 33 0 5 9 12 11 2 8 16 7 18 3 4 9 6 10 13 19 17 14 20 OK<LF>

[Set]

T>AT+ECHLIST=0 0<LF>

R<+ECHLIST=0000 0 33 OK<LF>

13.63 AT+ACH

ECH 전용 command. RF scan 을 사용하여 혼잡한 channel 을 check & 제외하는 기능

쓰기/읽기(Set/Get)지원

ELICIT-R1+ 전용

[Set format]

AT+ACH=<DADDR> <EN><CR/LF/CRLF>

DADDR	수행할 Device 주소
EN	0/1, 기능 동작 여부
0	기능 중지
1	기능 활성화

[Set Response]

AT+ACH=<DADDR> <RSSI> <BATT> OK<LF>

DADDR	목적지 SADDR
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Get format]

AT+ACH?[DADDR]<CR/LF/CRLF>

[Get Response]

AT+ACH=<OADDR> <RSSI> <BATT> <EN> OK<LF>

OADDR	응답을 보낸 디바이스 주소
RSSI	수신신호세기[dBm]
BATT	배터리용량[100mV]

[Example format 0]

[Get]

T>AT+ACH?<LF>

```
R<AT+ACH=0000 0 33 0 OK<LF>
```

[Set]

```
T>AT+ACH=0 1<LF>
```

```
R<AT+ACH=0000 0 33 0K<LF>
```

[Example format 1]

[Get]

```
T>AT+ACH?<LF>
```

```
R<+ACH?0000 0 33 0 0K<LF>
```

[Set]

```
T>AT+ACH=0 1<LF>
```

```
R<+ACH=0000 0 33 0K<LF>
```

13.64 AT+PEND

ECH 전용 command. Pending address 를 입력. AP 에서만 사용.

쓰기/읽기(Set/Get)지원

ELICIT-R1+ 전용

[Set format]

```
AT+PEND=<DADDR> <NODE_ADDR> <CMD><CR/LF/CRLF>
```

DADDR 수행할 Device 주소

NODE_ADDR 0x1~0xffff, 송신하려는 node address

CMD 수행할 AT command

[Set Response]

```
AT+PEND=<DADDR> <RSSI> <BATT> <NODE_ADDR> <CMD> OK<LF>
```

DADDR 목적지 SADDR

RSSI 수신신호세기[dBm]

BATT 배터리용량[100mV]

[Get format]

```
AT+PEND?<DADDR><CR/LF/CRLF>
```

[Get Response]

```
AT+PEND=<OADDR> <RSSI> <BATT> <NODE_ADDR> <CMD> OK<LF>
```

OADDR 응답을 보낸 디바이스 주소

RSSI 수신신호세기[dBm]

BATT 배터리용량[100mV]

[Example format 0]

[Get]

```
T>AT+PEND?<LF>
```

```
R<AT+PEND=0000 0 33 0001 (ADC?) OK<LF>
```

```
R<AT+PEND=0000 0 33 0000 () OK<LF>
```

[Set]

```
T>AT+PEND=0 2 (SLEEP=2 0 0 0)<LF>
```

```
R<AT+PEND=0000 0 33 2 (SLEEP=2 0 0 0) OK<LF>
```

[Example format 1]

[Get]

T>AT+PEND?<LF>

```
R<+PEND?0000 0 33 0001 (ADC?) OK<LF>
```

```
R<+PEND?0000 0 33 0000 () OK<LF>
```

[Set]

T>AT+PEND=0 2 (SLEEP=2 0 0 0)<LF>

```
R<+PEND=0100 0 33 OK<LF>
```

13.65 AT+IND

Event 발생을 알림

알림(Indication) 지원

[Indication format]

```
AT+IND=<OADDR> <RSSI> <BATT> <EVENT> [<PARAMS> ... ] <OK/FAIL><LF>
```

OADDR IND 를 생성한 디바이스 주소

EVENT IND 를 발생시킨 Event, 이에 따라 각기 다른 파라메터가 정의되어 있다.

DIEVT	DI event 발생 알림	PARAMS: <IO> <DIN>
-------	----------------	--------------------

RESET	Reset 알림	PARAMS: <REASON>
-------	----------	------------------

I2C	I2C Read 완료 알림	PARAMS: <DATA> ... <DATA>
-----	----------------	---------------------------

ADC	ADC Read 완료 알림	PARAMS: <ADC>
-----	----------------	---------------

IOADC	IO ADC Read 완료 알림	PARAMS: <IOADC>
-------	-------------------	-----------------

JOIN	Join complete 알림	PARAMS: <Assign ADDR> <LADDR>
------	------------------	-------------------------------

EXIT	network exit 알림	PARAMS: 없음
------	-----------------	------------

INIT	Init 완료 알림	PARAMS: <LEVEL>
------	------------	-----------------

SCAN	RF CH scan 결과 알림	PARAMS: <CH> <MAX_RSSI>
------	------------------	-------------------------

PER	PER 시험 종료 알림	PARAMS: <ADDR> <NUM> <TX> <RX> <AVRSSI> <AVRMRSSI>
-----	--------------	--

RELAY_FAIL	hop Tx 실패 시 알림	PARAMS: <DST ADDR> <"CMD STRING">
------------	----------------	-----------------------------------

BCN_TIMEOUT	BEACON timeout 알림	PARAMS: 없음
-------------	-------------------	------------

[Indication Example format 0]

```
R<AT+IND=0100 -85 33 DIEVT 3 1 OK<LF>
```

#DIEVT 알림, GPIO3 의 DIN 값이 1로 변함

```
R<AT+IND=0100 -85 33 RESET 2 OK<LF>
```

#RESET 알림, Reset pin 에 의한 Reset

```
R<AT+IND=0100 -85 33 I2C 30 45 OK<LF>
```

#I2C Read 알림, I2C 읽은 값 30, 45

```
R<AT+IND=0100 -85 33 ADC 2569 OK<LF>
```

#ADC 알림, ADC 값 2569mV

```
R<AT+IND=0100 -65 33 IOADC 3333 27 0 0 0 0 0 0 0 0 OK
```

#IOADC: I01:3333mV I02:27mV

```
R<AT+IND=0300 -85 33 JOIN 300 123456789abcdef OK<LF>
```

#JOIN 완료 알림, 신규 Node 0300 네트워크 가입

```
R<AT+IND=0300 -85 33 EXIT OK<LF>
```

#EXIT 알림, Node 0300 네트워크 나감

```
R<AT+IND=0100 -85 33 INIT 1 OK<LF>
```

#INIT 알림, IO/Event 설정 초기화

```
R<AT+IND=0000 -85 33 SCAN 3 -105 OK<LF>
```

#SCAN 알림, MAX RSSI 가 -105dBm 인 CH 3 을 추천

```
R<AT+IND=0000 0 33 PER 0100 20 18 16 -94 -82 OK<LF>
```

#PER 시험 결과, Node 100 과 PER 시험

```
R<AT+IND=1100 0 33 RELAY_FAIL 1110 "SEND"<LF>
```

#Hop Tx fail 메시지

```
R<AT+IND=0001 0 33 BCN_TIMEOUT<LF>
```

#BEACON RX timeout 메시지

[Indication Example format 1]

R<#IND=0100 -85 33 DIEVT 3 1 OK<LF>	#DIEVT 알림, GPIO3 의 DIN 값이 1로 변함
R<#IND=0100 -85 33 RESET 2 OK<LF>	#RESET 알림, Reset pin에 의한 Reset
R<#IND=0100 -85 33 I2C 30 45 OK<LF>	#I2C Read 알림, I2C 읽은 값 30, 45
R<#IND=0100 -85 33 ADC 2569 OK<LF>	#ADC 알림, ADC 값 2569mV
R<#IND=0100 -65 33 IOADC 3333 27 0 0 0 0 0 0 0 OK	#IOADC: I01:3333mV I02:27mV
R<#IND=0300 -85 33 JOIN 300 123456789abcdef OK<LF>	#JOIN 완료 알림, 신규 Node 0300 네트워크 가입
R<#IND=0300 -85 33 EXIT OK<LF>	#EXIT 알림, Node 0300 네트워크 나감
R<#IND=0100 -85 33 INIT 1 OK<LF>	#INIT 알림, IO/Event 설정 초기화
R<#IND=0000 -85 33 SCAN 3 -105 OK<LF>	#SCAN 알림, MAX RSSI 가 -105dBm 인 CH 3을 추천
R<#IND=0000 0 33 PER 0100 20 18 16 -94 -82 OK<LF>	#PER 시험 결과, Node 100과 PER 시험
R<#IND=1100 0 33 RELAY_FAIL 1110 "SEND"<LF>	#Hop Tx fail 메시지
R<#IND=0001 0 33 BCN_TIMEOUT<LF>	#BEACON RX timeout 메시지