ALIEN TECHNOLOGY®

리더 인터페이스 가이드

고정형 리더기 용

2016년 3월

ALR-F800
ALR-9900+
ALR-9680
ALR-9650

법적 고지문

Alien Technology®

리더 인터페이스 가이드

ALR-F800, 9900+, ALR-9680, ALR-9650

목차

제 1 장
서론 및 리더 설정

본 리더 인터페이스 가이드는 다음의 Alien Technology® RFID readers 의 설치 및 작동 지침을 제공한다.

- ALR-F800 (ALR-F800 기능에 대해서는 아래 각주 참조)
- ALR-9900, ALR-9900+ (Enterprise)
  유의사항: ALR-9900+ 는 ALR-9900 과 동일한 명령어들을 지원하며, 단 일부 예외는 본서에 명시하였음.
- ALR-9680
- ALR-9650 (Smart Antenna)

이 가이드는 애플리케이션 소프트웨어로 데이터 입수 및 시스템 구성을 위해 호스트와 리더 간에 사용되는 프로토콜의 세부정보도 제공한다.

이 문서는 RFID 시스템 통합 책임자와 소프트웨어 개발자로서 RFID 리더의 기능들을 최대한 활용하도록 소프트웨어 제품과 확장형 시스템을 개발하려는 대상이 사용하도록 고안되었다.

RFID 테크놀로지 개요와 용어집을 보려면 Alien RFID Primer 문서를 참조하기 바란다

> 본 안내서에 열거한 ALR-F800 기능들은 초기 제작 펌웨어(2015년 12월)를 전제로 한 것 이며 ALR-9900+ 전체 기능에 부합하도록 계 속 업데이트 될 것이다.

대상

본 문서의 목적에 따라, 리더 인터페이스 안내서 독자들에게는 기본적으로 다음 사항이 전제된다:

- PC 사용이 능숙한 사용자
- IT 전문가, 네트워크 전문가, 또는 프로그래머
- RFID 테크놀로지에 관한 최소한의 사전 지식이 있는 사용자
- 소프트웨어 개발 및/또는 하드웨어 시스템 통합의 유경험자

추가적으로, 다음 각 항목을 전제로 한다:

- 직접 시리얼(Serial) 통신을 통해 리더를 설치하려면 RS-232 시리얼(Serial) 프로토콜 애플리케이션에 익숙하여야 한다.
- 네트워크 통신을 통해 리더를 설치하려면 기본적인 네트워크 구성에 익숙하여야 한다.
- 프로그래머라면 최소한 한 가지 프로그래밍 또는 스크립트 언어에 능숙하며, 해당 언어로 ASCII 기반 명령어를 사용할 능력이 있어야 한다.

글씨체 규정

- 보통 텍스트는 일반 고딕 폰트로 표시한다.
- 외부 파일과 문서를 참조할 때에는 이탤릭 텍스트로 표시한다.
- 입력하는 특정 문자나 명령어는 인용 부호 사이에 표시하거나 및/또는 고정폭 폰트로 표시한다. 예: 프롬프트에서 "`DHCP=ON`" 을 입력한다.
- 사용자가 제공하고 입력하는 값은 대문자 및 소문자로 괄호 안에 표시한다. 예: 프롬프트에서 다음을 입력한다. "`IPAddress=[IP address value]`" 또는 "`IPAddress=xxx.xxx.xxx.xxx`" 실제 입력한 명령어는 다음과 같이 표시된다. "`IPAddress=10.1.60.5`".
- 샘플 코드 블록이나 명령어는 다음과 같이 표시한다.

```
    indented, in a fixed-width serif font.
```

- 키를 누르면 괄호 안에 모두 대문자로 표시된다. 예: [ENTER] 키를 누른다.
- 명령어를 입력했으면 [ENTER]를 눌러서 명령어를 전송해야 한다.
- RFID 리더 명령어는 대소문자를 구분하지 않는다. 이 문서에서는 명확한 전달을 위해 명령어를 대문자와 소문자로 표시하기도 하였으나, 사용자가 원할 경우 전부 소문자로 입력할 수도 있다.
- "get" 이나 "set" 등의 명령어는 "`get IPAddress`"에서와 같은 특정 매개변수들 사이에는 사이 띄기를 해야 한다. 단, "IP"와 "Address" 등의 매개변수 요소들 간에는 사이띄기가 필요하지 않다.

요구사항

RFID 리더와의 인터페이스를 연결하려면 다음이 필요하다.

- PC에 Windows가 설치되어 있고 RS-232 시리얼 포트가 있을 것.
- 호스트 소프트웨어(Alien RFID Gateway Demo 소프트웨어 또는 사용자 지정 소프트웨어)
- RFID 태그 (AIDC 클래스 I 호환)

시리얼 통신 요구사항:

- 일반적으로 컴퓨터에서 실행되는 시리얼 통신 프로그램(예: HyperTerminal, TeraTerm, PuTTy)

이더넷 통신 요구사항:

- 이더넷 네트워크
- 텔넷 통신 프로그램

리더와 통신하기

리더 인터페이스 가이드의 이 섹션에서는 호스트 컴퓨터에서 리더를 연결하는 두 가지 통신방법, 즉 시리얼(RS-232) 과 텔넷(TCP/IP)을 사용해 리더와 통신하고 명령어를 실행하는 방법에 대해 설명한다.

리더와 직접 시리얼 통신을 하든 네트워크 통신을 하든, 리더의 초기 설정을 위해서는 시리얼 통신이 필요할 수 있다.

Alien은 Java, Visual Basic, .NET 프로그래밍 언어를 사용하는 다양한 코드 예제들을 제공한다. 이러한 예제들은 사용자가 새로운 소프트웨어를 개발하는데 참조할수 있으며, 리더 CD(사용자 키트에 제공된 경우), Alien Partner 포털 웹사이트(http://partners.alientechnology.com), 또는 공개 FTP 서버(ftp://ftp.alientechnology.com)에서 찾을 수 있다.

시리얼 통신

이 방법은 새 RFID 리더를 설치할 때 유용하다. 시리얼 통신은 사전 설정이 필요하지 않으며, 대부분의 컴퓨터에서 쉽게 리거이에 접근이 가능하다. 이 방법을 통해 시리얼 통신(COM) 포트를 사용해 리더를 실시간 작동시킬 수 있다. 시리얼 통신은 리더와의 인터페이스와 Alien 리더 프로토콜을 실행하는 가장 손쉬운 방법이다.

DHCP를 사용해 리더를 구성하면 라우터를 통해 네트워크 설정값을 얻을 수 있으며, 이는 손쉽고 편리한 네트워크 구성 방법이다. 한편 리더와의 통신을 위해서는 IP주소를 알고 있어야 한다는 문제점이 있다. Alien 리더는 하트비트(Heartbeat) 메커니즘이 있어서 네트워크 상에서 리더 검색을 지원하기는 하나, 이 메커니즘을 사용하려면 하트비트(Heartbeat) 메시지를 감지하여 사용자에게 보고할 수 있는 호스트 애플리케이션(Alien Gateway Demo 소프트웨어 등)이 필요하다. 이러한 상황에서는 시리얼 통신을 사용하여 리더의 네트워크 주소를 정할 수 있다.

시리얼 설정

시리얼 통신으로 리더를 직접 작동시키거나 네트워크 연결을 통해서 작동시키기 위해서는 시리얼 포트 지침에 따라 리더를 설정 하여야 한다.

유의사항: 이 섹션에 표시한 예제 화면은 HyperTerminal 이다.

1. 리더가 전원에 제대로 연결되었는지, 최소한 1개의 안테나가 있는지 확인한다. 더 자세한 정보를 원할 경우, 리더의 하드웨어 설정 안내서를 참조하기 바란다.

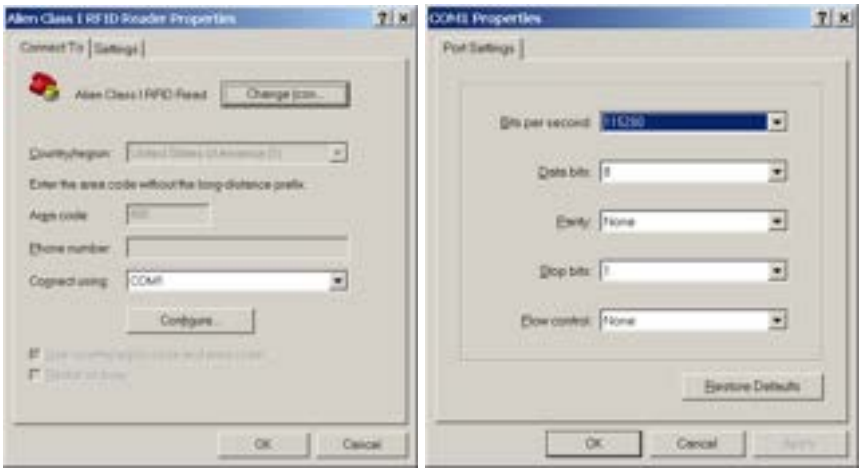
2. 시리얼 케이블의 한쪽 끝을 리더의 RS-232 포트에 연결하고, 다른 쪽 끝은 호스트 컴퓨터의 사용 가능한 COM 포트에 연결한다.

3. HyperTerminal, TeraTerm, PuTTy 등의 원하는 시리얼 통신 프로그램을 시작한다.

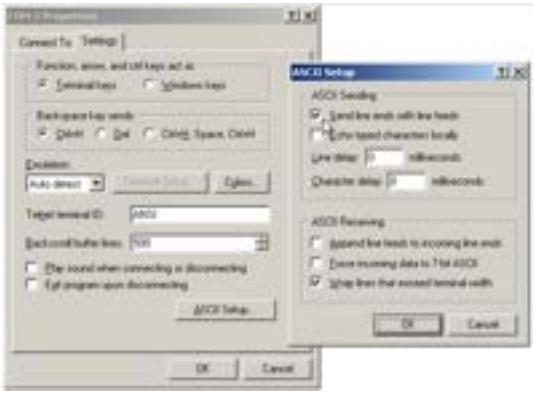
4. 다음 설정값을 입력(또는 확인)하여 직렬통신 프로그램을 구성한다:

Baud Rate	: 115200
Data Bits	: 8
Parity	: None
Stop Bits	: 1
Flow Control	: None

정상적으로 구성했다면, 소프트웨어를 통해 RFID 리더와 통신이 가능해진다. HyperTerminal 예제 구성 화면은 아래와 같다.



5. Send line ends with line feeds 를 체크 할 것을 권장한다.



6. 리더에 전원을 연결하고, 시리얼 포트에 대해 보고된 정보를 참조한다.
   네트워크 설정값이 가장 중요한 정보에 속하는데, TCP/IP를 사용해 리더와 통신하려면 이 값이 필요하기 때문이다.

리더는 부팅이 끝날 때 다음과 같은 텍스트 블록을 표시한다.

```
==========================================
------------------------------------------
Network Settings:
    MAC Address : 00:80:66:10:2D:12
    Hostname    : alien-102D12
    DHCP        : 1
    DHCPTimeout : 30
    IP Address  : 10.9.8.10
    Netmask     : 255.255.255.0
    Gateway     : 10.9.8.2
    DNS         : 10.9.8.1
    TimeServer  : time-a.timefreq.bldrdoc.gov
    TimeZone    : -7
    CommandPort : 23
------------------------------------------
==========================================
```

이 내용은 포트 23, 주소 10.9.8.10에서 TCP/IP 상에서 리더에 접촉할 수 있다는 의미이다.

7. 마지막으로 “Boot> Ready”의 표시가 보일 것이며, [ENTER] 키를 누른 뒤에 "Alien>" 명령어 프롬프트가 보일 것이다. 이제 명령어 프롬프트에서 리더 명령어를 입력하고 [ENTER] 키를 눌러서 명령어를 실행한다.

다음 기본 명령어들은 리더-호스트 인터페이스를 확인하는데 도움이 된다:

- "Help" (또는 "h") - 이용 가능한 명령어 목록 전체를 보여준다.
- "Info" (또는 "i") - 현재 리더 설정값 목록을 보여준다.
- "get TagList" (또는 "t") - 태그 필드를 스캔하여 결과를 보고한다.

유의사항: RFID 리더 명령어는 대문자/소문자를 구별하지 않으며, 원한다면 전부 소문자로 입력할 수 있다.

사용 가능한 모든 명령어에 관한 구체적인 설명을 원할 경우, 'Alien 리더 프로토콜' 내용을 참조하기 바란다.

네트워크 통신

TCP/IP 통신을 위해서는 리더의 이더넷 포트를 사용해 네트워크를 연결해야 하며 리더가 텔넷 서버처럼 작동하게 해야 한다.

이 모드에서는 직렬 인터페이스와 동일하게 RFDI 리더와의 명령어를 통해 통신할 수 있으나, 이때 RFID 리더를 네트워크 사용 환경으로 구성하고 실행해야 한다.

기본값으로, 모든 RFID 리더는 이더넷 연결 조건에서 DHCP를 사용하도록 사전 구성되어 있다. 그러나 우선 직접 직렬 연결로 설정하여 리더의 s IP 주소를 확인하거나 리더가 네트워크 상에서 송달한 "heartbeat" 메시지를 파악할 수 있어야 한다.

리더의 네트워크 설정

일단 리더를 설정 및 구성했다면, 주소를 기록해두거나, 리더의 하트비트(Heartbeat) 메시지(차후 기술할 것) 등 다른 주소 확인 수단을 사용해야 한다. 다음은 리더에서 이 정보를 검색하기 위한 지침이다.

네트워크 구성을 위해 리더를 검색할 때 사용할 수 있는 네트워크 명령어가 5개 있다. 이 명령어들은 리더의 직렬 인터페이스에서 직접 실행한다.

- DHCP?
- IPAddress?
- Netmask?
- Gateway?
- DNS?

다른 대안으로, "info network" 명령어를 실행해 네트워크 설정값을 요약할 수 있다.
일단 리더의 IPAddress와 Netmask가 있다면, TCP/IP를 사용해 연결할 수 있다.

TCP/IP 를 통해 연결하기

리더의 TCP/IP 인터페이스는 기본 텔넷 인터페이스를 모방하여 어떤 텔넷 클라이언트를 사용하든 리더에 연결할 수 있게 하였다. Windows 사용자라면 다음과 같이 텔넷 명령어를 명령어 프롬프트에서나 "Start -> Run" 창에서 사용할 수 있다.

```
telnet <IPAddress> <CommandPort>
```

예를 들어, 앞서 사용한 다음 명령어를 입력한다고 해보자.

```
telnet 10.9.8.10 23
```

프로그램을 통해 이를 실행하려면 명령어 포트(기본값은 23)에서 리더에 TCP 소켓을 열고 읽으면 된다. 명령 문자열을 적절히 종료하고, 리더의 응답이 끝났는지 파악하고, 반환된 데이터에서 "Alien>" 프롬프트 텍스트를 비활성화는 방법에 대한 세부 정보를 원한다면, 4장 Alien 리더 프로토콜을 참조하기 바란다. 기본 설정값은 다음과 같다.

```
Username = alien
Password = password
```

이제 리더와 통신할 준비가 되었다. 텔넷 작동을 위해서는 직접 직렬 통신에서와 동일한 텍스트 기반 명령어를 사용한다. 유일한 차이점은 텔넷 세션을 끝내고 세션 연결을 해제할 때 “q” 명령어를 사용하는 것뿐이다. NetworkTimeout 설정값보다 더 긴 활동이 없었다면, 리더가 유휴 상태의 TCP/IP 연결을 자동으로 해제한다.

리더 명령어와 사용 지침은 차후 Alien 리더 프로토콜 에서 다룰 것이다.

TCP/IP 구성

네트워크 작동을 위해 시스템을 구성하려면, 리더의 "Help" 디스플레이에서 "NETWORK" 소제목 하에 표시한 명령어들을 사용한다.

네트워크 구성을 위해 사용할 수 있는 네트워크 명령어가 5개 있다.

- DHCP
- IPAddress
- Netmask
- Gateway
- DNS

1. 명령어 목록을 보려면 “h” 또는 “help”를 입력한다.

***DHCP* 가 지원되는 환경이라면:**

2. 다음을 입력한다. “DHCP=ON” DHCP는 리더가 다음에 부팅할 때 나머지 네트워크 매개변수들을 자동으로 구성한다.

3. 7단계는 생략한다.

***DHCP* 가 지원되지 않는 환경이라면:**

4. 다음을 입력한다. “DHCP=OFF” 리더가 “DHCP = OFF” 메시지를 반환한다.

5. 다음 매개변수 값에 대해서는 시스템 관리자와 상의한다:
   IPAddress
   Netmask (또는 Subnet Mask)

Gateway
DNS (선택)

6. 아래 4개 명령어들을 각각 할당된 값과 함께 입력한다:

```
IPAddress = xxx.xxx.xxx.xxx
Netmask = xxx.xxx.xxx.xxx
Gateway = xxx.xxx.xxx.xxx
DNS = xxx.xxx.xxx.xxx (선택)
```

각 값이 수락되면, 리더가 수락한 값으로 응답한다.

게이트웨어와 DNS에 부정확한 값을 설정하면 리더가 부팅하는데 시간이 더 걸릴 수 있다. 그 원인은 다수의 인터넷 주소들에 대응하려는 시도가 실패했기 때문이다. 게이트웨이 주소가 유효하지 않으면 리더가 고유의 TCP 네트워크 연결을 초기화하지 못할 수 있다(태그 스트리밍, 자율모드 공지 등)

7. "Save"를 입력하여 리더가 플래시메모리에 새 값을 저장하게 한 다음, "Reboot"하여 리더를 재부팅하여 새 네트워크 설정값들을 적용한다.

제 2 장
리더 기본요소

이 장에서는 Alien RFID 리더의 주요 기능들의 개요를 제시한다. 리더 설정을 위한 구체적인 지침은 앞 장에서 제공한 바 있다. 리더 명령어와 사용법은 다음 장에서 다룰 것이다.

서론

RFID 리더의 가장 기본적인 기능은 RFID 태그를 판독하고 사용자 또는 애플리케이션에 태그 목록을 제공하는 일이다. RFID 리더는 직렬 케이블을 통해 호스트에 연결되었거나 네트워크에 연결된 상태로 이 기능을 수행하도록 설계되었다.

네트워킹 조건에서 작동하도록 하기 위해 리더에는 네트워크 관리를 단순화시키는 다음 두 가지 중요한 기능이 있다.

- 리더 "하트비트(Heartbeat)"는 네트워크 애플리케이션이 네트워크 상에서 리더를 쉽게 찾을 수 있게 해준다.
- 자율모드(AutoMode)는 연결되지 않은 리더가 태그를 찾게 해주며, 특정 조건이 발생했을 때 네트워크 상에서 통지 메시지를 발송한다.

이러한 중요한 개념을 리더와의 통신 기본사항들과 더불어 이번 장에서 논할 것이다.

리더 검색과 리더 하트비트(Heartbeat)

여러 네트워크 애플리케이션에서 공통으로 나타나는 문제점 중 한 가지는 네트워크 상에서 장치의 주소를 찾는 일이다. 네트워크 상에서 이러한 장치를 작동시키려면 사용자가 장치의 IP 주소를 반드시 알아야 한다.

IP 주소가 바뀌지 못하도록 장치에 코드화되어 있다면 이 문제는 해소된 것이며, 장치 라벨을 사용해 IP 주소를 표시하는 경우도 있다.

그러나 많은 시스템에서는 그러한 하드코드화(hard-coded)된 IP 주소를 사용하지 않으며, 따라서 장치를 부팅할 때마다 네트워크에서 주소를 장치에 할당해야 한다. 이를 DHCP라 칭하는데, 동적 호스트 구성 프로토콜(Dynamic Host Configuration Protocol)의 약자이다.

DHCP 와 자동 검색

DHCP 구성 모드에서는 사용자가 장치를 위해 네트워크 구성을 할 필요가 없다. 장치를 그저 네트워크 소켓에 연결하고 부팅하면 즉시 네트워크의 일부가 되며, DHCP 서버에서 직접 설정값을 입수한다.

단, 사용자가 장치의 IP 주소를 알아내야 한다. 현재로서는 이 장치에 IP 주소가 있고 네트워크 상에서 스스로 부팅한다는 것만 알려져 있다. 이 장치가 사용하는 실제 IP 주소는 알려져 있지 않다.

시리얼 조회

리더의 IP 주소를 찾는 가장 손쉬운 방법은 RS-232 포트를 통해 연결하고 명령어 “`IPAddress?`” 를 입력하는 것이다. 리더는 현재 사용 중인 IP 주소로 응답한다.

단, 이를 위해서는 호스트 컴퓨터와 리더 간에 물리적 연결이 되어 있어야 하며, 많은 경우 이러한 연결은 설정하기가 쉽지 않다.

네트워크 하트비트(Heartbeats)

리더의 IP 주소를 찾는 또 다른 방법은 네트워크 상에서 하트비트(Heartbeat) 메시지를 듣는 것이다.

리더가 네트워크 상에서 성공적으로 부팅했으면, 그 후로는 네트워크 상에서 하트비트(Heartbeat) 메시지를 정기적으로 전파할 것이다. 이 하트비트(Heartbeat)를 네트워크 애플리케이션이 탐지할 수 있으며, 그러면 리더가 네트워크 상에서 그 위치를 찾아 통신하기 위해 필요한 충분한 정보를 제공한다.

네트워크 용어로 표현하자면, 하트비트(Heartbeat) 메시지는 UDP(Universal Datagram Protocol) 패킷을 통해 리더 서브넷에 있는 모든 네트워크 주소로 송달된다.

리더의 명령어 입력으로 이 하트비트(Heartbeat)에 영향을 미칠 수 있는 3가지 구성 옵션이 있다.

- **HeartbeatTime:** 이 명령어는 네트워크 상에서 발송된 연속된 하트비트(Heartbeat) 메시지들을 분리할 시간 간격을 명시한다. 초 단위로 명시하며, 값이 0이면 하트비트(Heartbeat)가 꺼진 것이다. 기본값은 30초이며, 이때 30초마다 하트비트(Heartbeat) 메시지를 송달한다.

- **HeartbeatPort.** 이 명령어는 UDP 하트비트(Heartbeat) 메시지를 보낼 포트 번호를 명시한다. 네트워크 상에서 관련 당사자는 이 포트 번호에 주의를 기울여야 한다. 이 설정의 기본값은 3988이며, 이때 서브넷에서 기계마다 UDP 포트 3988로 하트비트(Heartbeat) 메시지가 발송된다.

- **HeartbeatAddress.** 이 명령어는 UDP 하트비트(Heartbeat) 메시지를 보낼 특정 IP 주소를 명시한다. 기본값은 255.255.255.255이며, 서브넷 상의 기계가 하트비트(Heartbeat)를 수신할 수 있게 해주는 특별한 "multicast"이다. 다른 하트비트 주소는 하트비트(Heartbeat)가 해당 주소를 가진 기계로만 전달되게 한다.

- **HeartbeatCount.** 이 명령어를 통해 사용자는 송달할 하트비트(Heartbeat) 메시지의 총 수를 명시한다. 이로써 리더가 온라인 상태일 때 즉시 검색할 수 있으나, 불필요한 트래픽으로 네트워크에 계속 부하를 가하지는 않는다.

하트비트(Heartbeat)의 형식은 작은 XML 텍스트 기반 메시지로서, 리더 기본정보(이름과 형태), 리더의 네트워크 연결(IP 주소와 명령어 포트), 다음 하트비트(Heartbeat)을 송달할 때까지의 시간 길이 정보를 포함한다.

```
<Alien-RFID-Reader-Heartbeat>
   <ReaderName>Alien RFID Reader</ReaderName>
   <ReaderType>
      Alien RFID Tag Reader, Model: ALR-9900
      (Four Antenna / Gen 2 / 902-928 MHz)
   </ReaderType>
   <IPAddress>10.1.60.5</IPAddress>
   <CommandPort>23</CommandPort>
   <HeartbeatTime>30</HeartbeatTime>
   <MACAddress>01:02:03:04:05:06</MACAddress>
   <ReaderVersion>07.07.18.00</ReaderVersion>
</Alien-RFID-Reader-Heartbeat>
```

하트비트(HEARTBEAT) XML TAGS

- **ReaderName** 은 리더와 관련하여 사용자가 정의한 이름이다. 이 이름은 각각의 리더를 확인할 수 있도록 사용자가 설정할 수 있다. 예를 들어, 창고 안에 여러 리더가 있을 때 이를 "적재 구획 1," "적재 구획 2" 등으로 명명함으로써 리더의 물리적인 위치를 명확히 표시하는 것이다.

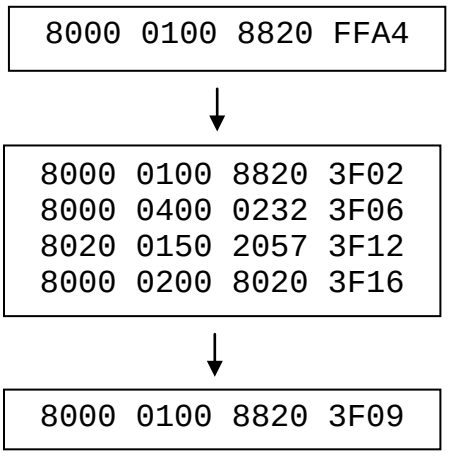
- **ReaderType** 은 하트비트(Heartbeat)를 송달하는 리더의 유형을 구체적으로 명시한다. 이 정보는 리더의 펌웨어에 하드코드화(hard-coded)되므로 사용자가 구성할 수 없다.
- **IPAddress and CommandPort** ▪ 요소는 네트워크 상에서 리더의 위치를 구체적으로 밝혀 준다. IP 주소는 리더의 네트워크 주소를 말한다. 명령어 포트는 리더가 사용자 명령어를 청취할 포트 번호이다. 일반적으로 포트 번호는 23이며, 이는 표준형 텔넷 포트로서 사용자가 "telnet [IPAddress]"를 대부분의 컴퓨터 명령어 입력란에 입력하여 네트워크 상에서 리더와 통신할 수 있게 해준다.
- **HeartbeatTime** 은 다음 하트비트(Heartbeat)가 있을 때까지의 시간이다. 이 시간(초 단위)을 근거로 애플리케이션 소프트웨어는 리더가 전원을 끄거나 네트워크 연결이 끊길 때 이를 감지할 수 있다. 예상한 경과 시간 후에도 새 하트비트(Heartbeat)를 받지 못하면, 이 정상적인 서비스 중단의 발생을 탐지할 수 있다.
- **MACAddress** 는 제조업체가 리더의 네트워크 인터페이스 하드웨어에 지정한 고유의 식별자를 제시한다.
- **ReaderVersion** 은 리더에서 현재 실행 중인 소프트웨어의 버전 정보를 제시한다.

하트비트(HEARTBEATS) 및 소프트웨어(SOFTWARE)
Alien SDK는 이러한 네트워크 하트비트(Heartbeat)를 듣기 위해 Java, .NET, Visual Basic으로 된 소스코드와 소프트웨어 라이브러리를 제공한다.

Alien RFID 게이트웨이 애플리케이션은 이러한 라이브러리의 Java 버전을 사용하여 메인 스크린에 활성화된 리더 목록을 제시한다. 본서 작성 당시 최신 게이트웨이 버전은 v2.25.00이다. Alien은 사용자가 최신 버전의 게이트웨이 소프트웨어를 다운로드하여 설치할 것을 권장한다.

태그리스트(TagList) 개념

정상 작동 중에 리더는 활성화된 태그의 내부 목록을 관리함으로써 "현재 가동 중인 대상"을 상시 파악한다. 활성화 태그는 사전 정의된 간격마다 리더가 판독한다. 리더에 새로운 태그가 제시되면 목록에 추가되고, 한 동안(PersistTime) 보이지 않은 태그는 목록에서 제거된다.



새로운 태그가 감지되면 *TagList*에 추가된다.

Reader TagList...

목록에 있는 태그가 모두 활성화 되었음.

한 동안 읽히지 않는 태그는 목록에서 삭제된다.

언제든 이 태그 목록을 검색하려면 리더에 프로그램으로 호출을 하면 된다.

지속시간(PersistTime)

"지속시간(PersistTime)"은 태그를 마지막 읽은 시간에서 TagList로부터 자동으로 퍼지한 시점까지의 지속기간을 의미한다. 이 값을 최소시간으로 설정하면(~1초) TagList는 마지막 1초 동안 리더가 본 것만 포함시킨다. 이를테면 리더가 어느 시점에 본 것을 온전히 표시해 주는 기능이다. PersistTime을 긴 기간으로 설정하면 태그 기록을 해당 기간만큼 축적할 수 있다. 예를 들어, PersistTime을 3600초로 설정하면 최근 1시간 동안 읽은 태그 일체의 세부사항을 누적하여 목록을 보여준다.

PersistTime을 -1으로 설정하면 TagList를 호스트에 전달할 때까지 태그 목록을 무제한 누적하는 특수 효과를 갖게 되며, 호스트 전달 시점에 TagList를 비운다. 이 값이 기본값이다.

태그 세부 정보(Tag Details)

TagList에 있는 기재 항목마다 하위 필드를 다수 포함하는데, 태그의 고유 ID, 카운트 수 (태그를 현재 세션에서 읽은 횟수), 발견 시간(태그를 처음 본 시간), 마지막 발견 시간, 안테나 (태그를 마지막으로 읽은 안테나 ID) 등이 이에 해당된다.

태그 리스트 크기(TagList Size)

TagList는 6000개 태그 항목까지를 보유할 수 있다. 일부 리더 모델은 정전 중에도 TagList 데이터 일부를 보존하는 기능도 있다.

네트워크 상에서 태그 읽기

Alien RFID 리더는 태그를 읽는 두 가지 방법, 즉 인터랙티브 모드(Interactive Mode)와 자율 모드(Autonomous Mode)를 제공한다.

- **인터랙티브 모드(Interactive Mode)** - 제어 애플리케이션이 리더에게 태그를 읽도록 명령을 보낸다. 이 명령으로 언제나 리더의 시야 내에 있는 태그 목록을 즉시 반환한다.

- **자율 모드(Autonomous Mode) -** 리더는 지속적으로 태그를 읽으며, 특정 이벤트가 발생하면 네트워크 상의 리스너(listener)와 통신을 시작하기도 한다.

이 두 방법 모두가 유용하나, 선택 여부는 제어 애플리케이션의 필요에 의해 결정한다.

인터랙티브 모드는 사용이 더 쉽고 필요한 코딩도 더 적긴 하나, 프로그래밍에 좀 더 투자할 경우 사용자가 AutoMode를 설정하여 여러 리더에서 규모조정이 더 용이한 시스템을 구성하는 것이 가능하다.

Interactive Mode

인터랙티브 모드에서 태그를 읽는 방식은 리더에 단일의 명령어를 전송하는 것만큼이나 단순하다. 이 명령어는 "`TagList?`", 또는 간단히 "`t`" 라고도 할 수 있다. 이로써 리더는 태그 검색을 시작하고 현재 TagList를 반환하여 보고할 수 있다. PersistTime과 최근 태그 활동에 따라서 반환된 데이터에는 이전 판독한 태그 데이터를 포함할 수도 있다. 다음은 기본 TagList 형식("텍스트")의 모양을 보여주는 예이다.

```
Tag:E200 3411 B801 0108, Disc:2007/06/29 08:30:49, Last:2007/06/29 10:38:12, Count:292, Ant:0, Proto:2
Tag:4461 7669 6445 2E4B, Disc:2007/06/29 10:38:13, Last:2007/06/29 10:38:13, Count:187, Ant:1, Proto:2
```

TagList의 형식은 "`TagListFormat`" 명령어를 사용해 지정할 수 있다. 옵션 중 하나는 XML 형식으로서, 다음과 같은 TagList를 반환한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-Tag-List>
   <Alien-RFID-Tag>
      <TagID> E200 3411 B801 0108</TagID>
      <DiscoveryTime>2007/06/29 08:30:49</DiscoveryTime>
      <LastSeenTime>2007/06/29 10:38:12</LastSeenTime>
      <ReadCount>292</ReadCount>
      <Antenna>0</Antenna>
      <Protocol>2</Protocol>
   </Alien-RFID-Tag>
   <Alien-RFID-Tag>
      <TagID>1155 4461 7669 6445 2E4B</TagID>
      <DiscoveryTime>2007/06/29 10:38:13</DiscoveryTime>
      <LastSeenTime>2007/06/29 10:38:13</LastSeenTime>
      <ReadCount>187</ReadCount>
      <Antenna>1</Antenna>
      <Protocol>2</Protocol>
   </Alien-RFID-Tag>
</Alien-RFID-Tag-List>
```

다른 형식의 예로는 "Terse", "Custom" 등이 있다. 제4장 Alien 리더 프로토콜에서 TagListFormat 섹션을 참조하면 전체 TagList 형식에 대한 세부정보를 찾을 수 있다.

오토 모드(Autonomous Mode)

AutoMode(오토모드)는 데이터의 자동화 모니터링 및 처리가 가능한 구성 및 작동 모드를 말한다. 사용자는 우선 리더에 구성 명령어들을 전송하여 태그를 읽는 방법 및 시점을 명시하고, 그에 더해 태그 데이터로 무엇을 할지를 명시하기도 한다.

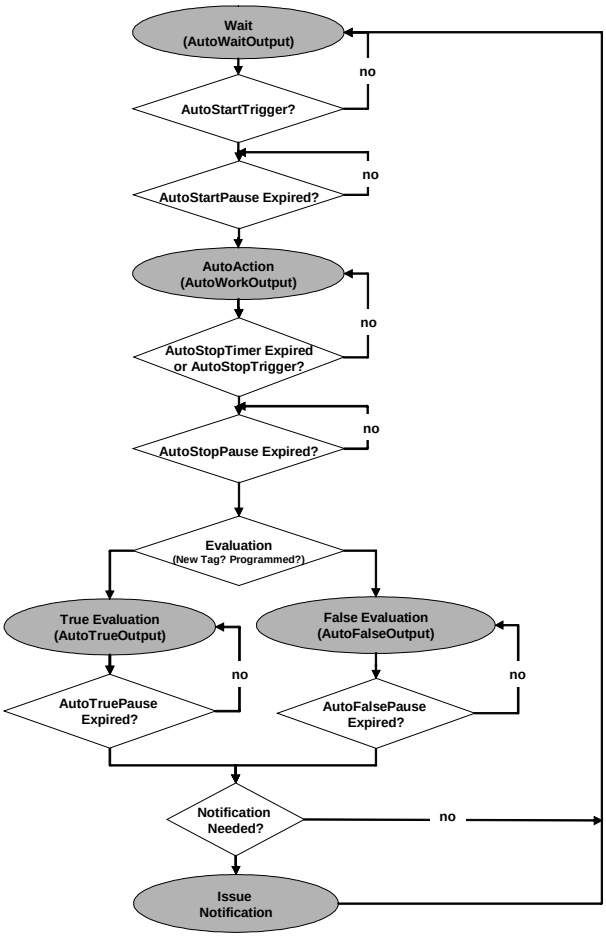
일단 이 모드로 구성했다면, 리더는 자율적으로 작동이 가능하다. 그러면 호스트 컴퓨터에 있는 애플리케이션을 설정하여 읽은 태그 데이터를 포함하여, 리더로부터 받은 공지 메시지를 청취하게 할 수 있다.

이 작동 모드의 한 가지 주요 장점은 다수의 리더를 구성하여 단일 호스트에 태그 메시지들을 전송하게 할 수 있다는 점이다. 따라서 단일 애플리케이션은 네트워크 상에서 여러 리더로부터 받은 데이터를 청취하여 처리할 수 있다.

자율적 읽기 작업(Autonomous Read Operation)의 정의

AutoMode의 기능은 아래 도표에 요약하였다. 기본적으로 AutoMode에서 작동하는 리더는 몇 가지 단계 사이를 오가는데, 바로 대기(Waiting), 작동(Working), 평가(Evaluation), 공지(Notification)이다. 대기(Waiting), 작동(Working), 평가(Evaluation) 상태는 이 단계를 진입할 때 설정되는 디지털 출력 상태로 리더를 연결한다. 상태 변경은 타이머 시간 만료, 디지털 입력 라인에서 촉발된 이벤트, TagList 상의 변경 등의 결과로 발생한다.

상태 도표의 각 요소를 아래 기술하였다. 각 요소 별로 리더 구성에 사용하는 하나 이상의 명령어들이 관련된다.



자율 모드 상태 도표

오토모드 진입 ENTER AUTOMODE

(상태 도표에는 표시되지 않았음)
사용자는 "AutoMode" 명령어를 사용하여 리더를 AutoMode로 전환할 수 있다. "AutoMode=On" 일 때 리더는 AutoMode가 된다. "AutoMode=Off" 일때는 AutoMode가 꺼진다.

대기 상태 WAITING STATE

AutoMode에 진입하면, 리더는 자동으로 대기 상태(Waiting State)가 된다. 시작 작업 트리거(Working Trigger)를 기다리는 동안(아래 참조), 리더는 디지털 출력 라인들에서 AutoWaitOutput 명령어를 사용해 설정한 값을 유지한다. 예를 들어, "AutoWaitOutput=1"이면 디지털 출력 라인 1은 높아지고, 이때 리더는 대기 상태에 있게 된다.

시작 동작 트리거 START WORKING TRIGGER

디지털 입력 라인에서 트리거 패턴을 받으면 리더는 대기 상태에서 작동 상태로 전환된다. 시작 조건이 되려면 AutoStartTrigger 명령어로 설정한다. AutoStartTrigger 명령어에서 적용하는 두 가지 매개변수는 상승에 지 패턴(rising edge pattern)과 하강 에지 패턴(falling edge pattern)이다. 각 패턴은 원하는 입력 트리거의 비트맵을 나타내는 단일 정수 형태로서, 입력값 #1은비트 0으로, 입력값 #2는 비트 1로 표시한다.

아래 표는 상이한 I/O 조합들에 따라 상응하는 비트맵 값들을 보여준다. 이 표는 입력 출력 비트맵 모두에 적용된다.

Digital I/O Value	I/O #4	I/O #3	I/O #2	I/O #1
0	-	-	-	-
1	-	-	-	✓
2	-	-	✓	
3	-	-	✓	✓
4	-	✓	-	-
5	-	✓	-	✓
6	-	✓	✓	-
7	-	✓	✓	✓
8	✓	-	-	-
9	✓	-	-	✓
10	✓	-	✓	-
11	✓	-	✓	✓
12	✓	✓	-	-
13	✓	✓	-	✓
14	✓	✓	✓	-
15	✓	✓	✓	✓

명령어 "`AutoStartTrigger=2 0`"를 실행하면 리더는 입력값#2의 상승 에지를 받아 작동 상태가 되고, "`AutoStartTrigger=0 3`"의 경우에는 리더가 입력값#1이나 #2의 하강 에지를 받은 후에 작동 상태로 진입한다. "`AutoStartTrigger=0 0`"의 경우에는 리더가 즉시 작동 상태가 된다.

AutoStartPause 속성은 리더가 시작 트리거를 받은 후 정해진 밀리세컨드가 지날 때까지 기다렸다가 작동 상태로 변경되게 한다.

작동 상태 WORKING STATE

작동 상태에서는 리더가 `AutoWorkOutput` 명령어로 정의한 값을 디지털 출력 라인에 유지한다. "`AutoWorkOutput=3`"의 경우 처음 두 출력 라인을 높게 유지하며, 이때 리더는 작동 상태가 된다. 작동 상태에서 리더가 수행하는 동작은 `AutoAction` 명령어로 정해진다. "`AutoAction=Acquire`"의 경우 리더는 `AcquireMode` 와 `PersistTime` 명령어로 설정한 매개변수를 사용해 반복적으로 TagList 데이터를 입수한다. 리더는 정지 동작 트리거(Stop Working Trigger) 조건이 충족될 때까지(아래 참조) 계속 작동한다.

정지 동작 트리거 STOP WORKING TRIGGER/TIMER

시작 동작 트리거와 마찬가지로, 정지 동작 트리거는 디지털 입력 라인을 변경할 수도 있고 타이머를 만료시킬 수도 있다. 상승/하강 에지 패턴에서 `AutoStopTrigger` 명령어를 사용해 트리거 조건을 설정한다. "`AutoStopTrigger=1 0`"의 경우 작동 상태를 나오기 전에 입력 #1 상에서 상승 에지를 찾는다. 또한 `AutoStopTimer` 명령어를 사용해 지정된 기간 동안 작동 동작을 반복할 수도 있다. 예를 들어, "`AutoStopTimer=1300`"의 경우 리더는 평가 단계로 진행하기 전에 1.3초 동안 작동 동작을 수행한다. 정지 트리거와 정지 타이머를 모두 지정한 경우라면 어떤 이벤트가 발생하든 일단 작동 상태가 정지된다.

평가 EVALUATION

평가 결정 시점이 되면 리더는 최근 평가 이래로 TagList에 추가된 새 태그가 있었는지를 확인한다. 만약 있었다면 AutoTruePause 상태가 되고, 없었다면 AutoFalsePause 상태가 된다.

유의사항: 평가(Evaluation)는 TagList와 관계되므로 PersistTime 설정값에 따라 크게 좌우된다.

참/거짓 중지 TRUE/FALSE PAUSE

평가 후에는 리더가 출력 라인을 `AutoTrueOutput` 및 `AutoFalseOutput` 명령어에서 지정된 값으로 설정한다. 이 조건은 공지(Notification) 테스트가 시작되기 수 밀리세컨드 전에 `AutoTruePause` 또는 `AutoFalsePause` 일 때 유지된다. 예를 들어, "`AutoTrueOutput=1`" 과 "`AutoTruePause=20`" 의 경우 리더는 처리 전 20밀리세컨드 동안 출력#1을 높게(및 그 외는 낮게) 유지한다.

공지 NOTIFICATION

NotifyMode를 활성화하고(`NotifyMode=On`) 지정된 NotifyTrigger가 발생하면, 리더는 공지 메시지를 전송한다. NotifyTrigger는 "`NotifyTrigger`" 명령어로 지정하며, 다음 섹션에서 기술한 것처럼 "Add", "Remove(삭제)", "Change(변경)", "True(참)", "False(거짓)", 또는 "TrueFalse(참/거짓)"으로 설정할 수 있다.

공지를 발송하면, TagList 데이터가 `NotifyAddress` 로 전달된다. 그러면 리더가 대기 상태로 돌아간다.

오토모드 예제 AutoMode Examples

EXAMPLE 1 - 백그라운드 읽기 BACKGROUND READING

이 경우 리더가 태그 필드를 계속 모니터링 하기를 원하는 것이다. 애플리케이션이 정기적으로 TagList를 요청한다. 새 태그가 보이면 출력 #1이 500 msec 동안 번쩍이고, 그렇지 않으면 출력 #2가 500 msec동안 번쩍인다.

```
AutoModeReset
AutoAction = Acquire
AutoStartTrigger = 0,0
AutoStopTimer = 0
AutoTrueOutput = 1
AutoTruePause = 500
AutoFalseOutput = 2
AutoFalsePause = 500
AutoMode = On
```

EXAMPLE 2 - 트리거로 읽기 TRIGGERED READING

포크리프트(forklift)가 일렉트릭 아이(electric eye)로 하여금 리더에 신호를 보내게 한다. 이제 사용자는 리더가 이 신호에서 상승 에지를 찾고 대기 상태로 돌아가기 전에 1.8초 동안 태그를 스캔 하기를 원하며, 출력에는 아무런 변화를 주지 않는다.

```
AutoModeReset
AutoAction = Acquire
AutoStartTrigger = 1, 0
AutoStopTimer = 1800
AutoTruePause = 0
AutoFalsePause = 0
AutoMode = On
```

EXAMPLE 3 - 트리거로 읽고 공지하기 TRIGGERED READING WITH NOTIFICATION

트리거를 사용해 읽기를 시작한다. 태그를 찾으면 이메일 메시지를 발송한다. 이메일이 발송된 뒤에 대기 상태로 돌아간다.

```
AutoModeReset
AutoAction = Acquire
AutoStartTrigger = 1, 0
AutoStopTimer = 0
AutoTruePause = 0
AutoFalsePause = 0
NotifyAddress = borg@mycompany.com
MailServer = mail.mycompany.com
NotifyTrigger = Add
NotifyMode = On
AutoMode = On
```

공지 모드 Notification Mode

AutoMode 구성의 마지막 단계는 어떤 조건 하에서 TagLists에 관하여 청취자에 공지할 것인지를 리더에 알리는 일이다. 청취자(네트워크 애플리케이션 /사람)는 특정 조건이 되었을 경우에만 공지를 받는데, 이를테면 새 태그를 읽었거나 태그가 사라졌을 때 상황 등이다.

NotifyTime

NotifyTime 명령어는 리더로 하여금 TagList 사본을 n초마다 청취자에 발송하게 하며, 이때 TagList가 바뀌었는지는 관계없다. 이는 리더가 TagList를 청취자에게 발송하게 하는 손쉬운 방법이다.

NotifyTrigger

NotifyTrigger 명령어는 TagList를 청취자에게 발송하기 전에 반드시 갖춰져야 할 조건을 명시한다. 사용할 수 있는 여러 가지 가능한 트리거가 있다.

트리거	트리거 조건	태그 데이터
Add	새 태그를 읽어 TagList에 추가함.	추가된 태그만
Remove	태그를 TagList에서 삭제함.	제거된 태그만
AddRemove	태그를 TagList에 추가했거나 삭제함	추가된 태그 목록, 다음으로 제거된 태그 목록
Change	태그를 TagList에 추가했거나 삭제함	전체 태그 목록
True	자율상태 루프의 평가 태스크가 `true`로 평가함 (일반적으로 태그를 TagList에 추가하였을 때임).	전체 태그 목록
False	자율상태 루프의 평가 태스크가 `false`로 평가함 (일반 적으로 TagList에 추가된 태그가 없을 때임)	전체 태그 목록
TrueFalse	자율상태 루프의 평가 태스크가 `true` 또는`false`로 평가함 (이를테면, 매 AutoMode 주기마다)	전체 태그 목록

NotifyAddress

NotifyAddress는 리더가 공지 메시지를 전달할 위치를 말한다. 리더는 TCP 소켓에서 네트워크 상에 특정 기기에 메시지를 전달할 수 있는데, 특정 이메일 주소로 이메일을 발송하거나 직렬포트를 이용하기도 한다. 다음 명령어를 사용해 구성한다.

```
NotifyAddress = <address>
```

<address> 형식은 다음과 같은 전달 방법을 말한다.

NotifyAddress	설명
hostname:port	네트워크에 연결된 기기의 특정 포트로 메시지를 전송. 주소의 형태는 "hostname:port" 예를 들면, "123.01.02.98:3450" 또는 "listener.alientechnology.com:10002"
user@domain.com	지정한 주소로 이메일을 통해 메시지 전송. 주소는 표준 이메일 형태로 지정한다. (예: user@domain.com) 유의사항: MailServer 매개변수를 구성해야 작동함. 현재 이메일 공인 프로토콜은 지원되지 않음.
http://domain:port/path	POST 요청 형태로 메시지를 웹 서비스에 보냄. 주소 형태는 http://로 시작해야 하며 특정 웹 서비스로 이어지는 경로 및 포트 번호(기본값 80)를 지정할 수도 있음.
SERIAL	직렬 연결로 메시지를 보냄. "SERIAL" 용어를 주소로 사용하며, 대소문자를 구별하지 않음.

NotifyFormat

사용자는 리더에게 전송할 공지 형식을 알려줄 수 있다. 공지 메시지를 보낼 때 다음 두 부분으로 구성된다.

- 첫 부분은 헤더로서 메시지를 전송하는 리더의 세부정보, 메시지를 전송하는 이유를 포함한다.
- 둘째 부분은 TagList로, 새로 추가하였거나 삭제한 태그, 또는 NotifyTrigger에 따라서 리더가 본 태그의 전에 목록.

메시지의 형식은 다음 하나의 명령어로 구성한다.

```
NotifyFormat = <format>
```

다음은 형식의 예이다:

NotifyFormat	설명
Text	일반 텍스트 메시지, 라인마다 태그 ID가 한 개씩 있음.
Terse	일반 텍스트 메시지, 라인마다 태그 ID가 한 개씩이며 컴팩트 폼.
XML	XML 텍스트 형식
Custom	텍스트 형식과 같으나, 단 각 태그 ID 라인 내용을 TaglistCustomFormat 매개변수로 정의한다.

Text 형식의 공지 형태는 다음과 같다:

```
#Alien RFID Reader Auto Notification Message
#ReaderName: Spinner Reader
#ReaderType: Alien RFID Tag Reader, Model: ALR-9900
             (Four Antenna / Gen 2 / 902-928 MHz)
#IPAddress: 10.1.70.13
#CommandPort: 23
#MACAddress: 00:80:66:10:11:6A
#Time: 2003/01/21 12:48:59
#Reason: TEST MESSAGE
#StartTriggerLines: 0
#StopTriggerLines: 0
Tag:1115 F268 81C3 C012, Disc:2003/01/21 09:00:51, Last:2003/01/21 09:00:51, Count:1,
Ant:0, Proto:1
Tag:0100 0100 0002 0709,  Disc:2003/01/21 11:00:10, Last:2003/01/21 11:00:10, Count:1,
Ant:0, Proto:1
#End of Notification Message
```

Terse 형식의 공지 형태는 다음과 같다:

```
#Alien RFID Reader Auto Notification Message
#ReaderName: Spinner Reader
#ReaderType: Alien RFID Tag Reader, Model: ALR-9900
             (Four Antenna / Gen 2 / 902-928 MHz)
#IPAddress: 10.1.70.13
#CommandPort: 23
#MACAddress: 00:80:66:10:11:6A
#Time: 2003/01/21 12:48:59
#Reason: TEST MESSAGE
#StartTriggerLines: 0
#StopTriggerLines: 0
1115 F268 81C3 C012,0,1
0100 0100 0002 0709,0,1
#End of Notification Message
```

XML 형식의 공지 형태는 다음과 같다:

```
<?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-Reader-Auto-Notification>
    <ReaderName>Spinner Reader</ReaderName>
    <ReaderType>
        Alien RFID Tag Reader, Model: ALR-9900 (Four Antenna / Gen 2 / 902-928 MHz)
    </ReaderType>
    <IPAddress>10.1.70.13</IPAddress>
    <CommandPort>23</CommandPort>
  <MACAddress>00:80:66:10:11:6A</MACAddress>
    <Time>2003/01/21 12:49:22</Time>
    <Reason>TEST MESSAGE</Reason>
    <Alien-RFID-Tag-List>
        <Alien-RFID-Tag>
          <TagID>0102 0304 0506 0709</TagID>
          <DiscoveryTime>2003/01/17 11:37:01</DiscoveryTime>
          <LastSeenTime>2003/01/17 11:37:01</LastSeenTime>
          <Antenna>0</Antenna>
          <ReadCount>1413726</ReadCount>
          <Protocol>1</Protocol>
        </Alien-RFID-Tag>
        <Alien-RFID-Tag>
        <TagID>2283 1668 ADC3 E804</TagID>
        <DiscoveryTime>2003/01/19 07:01:19</DiscoveryTime>
        <LastSeenTime>2003/01/19 07:01:19</LastSeenTime>
        <Antenna>0</Antenna>
        <ReadCount>1</ReadCount>
          <Protocol>1</Protocol>
        </Alien-RFID-Tag>
    </Alien-RFID-Tag-List>
</Alien-RFID-Reader-Auto-Notification>
```

Custom 형식의 공지는 다음 예제와 같다:

```
#Alien RFID Reader Auto Notification Message
#ReaderName: Spinner Reader
#ReaderType: Alien RFID Tag Reader, Model: ALR-9900 (Four Antenna / Gen 2 / 902-928 MHz)
#IPAddress: 10.1.70.13
#CommandPort: 23
#MACAddress: 00:80:66:10:11:6A
#Time: 2003/01/21 12:48:59
#Reason: TEST MESSAGE
The tag 1115 F268 81C3 C012 was read 3 times
The tag 0100 0100 0002 0709 was read 1 times
#End of Notification Message
```

네트워크 상에서 태그 청취하기

리더가 자율모드로 구성되었다면, 리더와의 인터랙티브 통신은 불필요하며, 자율적으로 작동하게 할 수 있다. 리더로부터 공지 메시지를 청취할지는 애플리케이션에 따라 다르다.

RFID 리더 개발자 키트에 포함된 라이브러리는 Java, .NET, Visual Basic 환경으로 이 기능을 제공한다. 어떤 경우든 청취 서비스 설정은 단순한 코딩으로 가능하며, 10행 미만의 코드가 사용된다.

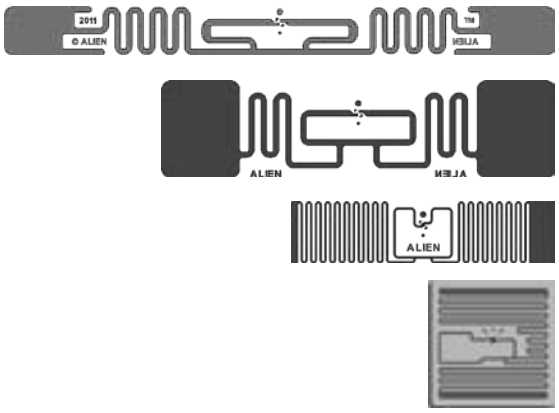
제 3 장
태그 기본요소

서론

RFID 태그 읽기는 태그의 태그 ID를 리더로 보내는 것 이상을 수반한다. 이 기본 동작을 수행하는 데 사용할 방법이 여러 가지가 있으며, 이 기본 4동작 수행을 조작하기 위해 여러 매개변수 및 설정값을 조정할 수 있다.

Alien RFID 태그

Alien 태그



클래스 1 카테고리에 속한 Alien 품목들은 보유한 IC를 사용하여 EPC 글로벌 사양에 따른 비용 및 크기 요건을 충족한다.

이 경우 통신(backscatter) 및 전력원(beam-powered) 수단과 관련하여 공통된 용어 정의에 따라 완전히 "수동적인" 태그에 해당한다. 따라서 Aliens 측에서 "수동 태그(passive tag)"라는 표현이 있으면 이런 유형의 태그를 말하는 것으로 이해하면 된다.

수집모드 (Acquisition Modes)

AcquireMode 속성은 필드에서 태그를 읽을 때 사용하는 방법을 정의한다. 태그 판독에는 두 가지 별개의 방식이 사용되며, 어떤 방식을 사용할 것인지는 해당 애플리케이션에 따라 달라진다. "`AcquireMode`" 명령어를 전송하여 수집모드를 지정한다. "글로벌 스크롤" 또는 "Inventory"의 둘 중 하나의 값을 취할 수 있다.

인벤토리(Inventory)

'inventory' 명령어는 필드에서 동시에 여러 태그의 ID를 파악하는 전 기능 보유 시스템에 해당한다. 이 상위수준 명령어는 리더-태그 간 복잡한 일련의 질의로 변환되어 결국 RFID 리더가 확인한 태그 ID 목록을 산출한다. 이러한 다중 태그 질의 및 평가 방법은 충돌방지 검색(anti-collision search)으로 알려져 있다.

클래스 1/GEN 2 인벤토리

클래스 1/Gen 2 인벤토리에서 리더는 가능한 방식으로 필드 내 태그를 계수한다. 모든 태그와 한꺼번에 통신할 수는 없으므로, 태그가 다소 임의적인 방식으로 응답하여 결국 전체 태그를 효율적으로 스캔 하게 된다. 리더는 모든 태그에 무작위 번호를 배정하게 하고, 태그 그룹은 여러 개의 "상자(bings)"로 세분한다("Q" 매개변수로 결정함). 상자 #1에서 무작위 번호를 선택하는 태그만 우선적으로 통신한다. 리더는 상자 #1에 있는 태그의 응답 모드를 디코딩 하려고 할 것이며, 태그 집단이 여전히 크므로 더 작은 상자로 나눌 것이다. 리더가 처음 태그 집단에서 받은 신호의 디코딩을 마친 후에는 나머지 태그로 하여금 상자에서 내려오게 한 후 다음 태그 집단과 통신한다.

예상한 태그집단 규모에 맞는 처음 Q 매개변수를 선택하는 것은 원하는 인벤토리 성능 달성에 있어 중요하긴 하나, Alien 리더는 Q 매개변수를 동적으로 조절할 수 있다. 이는 인벤토리가 진행되면서 더 많거나 적은 태그가 해당 시간 동안 응답하게 하여 결과적으로 초기에 Q값 선택 여부가 덜 중요해 지기 때문이다.

클래스 1/GEN 2 매개변수 수집
리더가 클래스 1/Gen 2 태그를 스캔 할 때, 인벤토리 대상인 태그를 자동으로 준비할 수 있으며 "선별(Selecting)"이라 칭함), 그 후에 이 태그에 일련의 동작을 수행한다. 태그 집단을 "상자(bins)"로 세분하여 리더가 언제든 태그 하위집단과 통신하게 한다. 이 순서는 "G2" 수집 매개변수로 정의한다. 각 값을 조정하여 리더가 사용자의 용도에 따라 최적으로 작동하도록 구성할 수 있다.

리더가 수집 동작을 할 때, `AcqG2Cycles` 매개변수로 지정한 수집 주기를 모두 수행한다. 각 주기는 안테나 별로 `AcqG2Select` SELECT 명령어를 사용해 시작하며, 이때 SELECT 명령어는 일부 태그가 "A" (인벤토리 구성 안됨) 상태가 되게 하는데 이는 "태그 깨우기(waking the tag up)"와도 유사하다. 그런 다음 안테나 별로 인벤토리 (`AcqG2Count`의 결과)를 모두 수행하여 태그가 읽힐 때 "B" 상태가 되게 한다. 태그 마스크(Mask)를 사용하여 어떤 태그가 포함될지를 제어한다.

클래스 1/Gen 2에 대한 수집 매개변수를 다음의 의사코드(pseudo-code)로 예시하였다.

```
foreach cycle in AcqG2Cycles
   foreach antenna in AntennaSequence
          SELECT (applies the current mask)
   next antenna

   foreach count in AcqG2Count
      foreach antenna in AntennaSequence
         DoAcquisition
      next antenna
   next count
next cycle
```

이에 더하여, 클래스 1/Gen 2 태그들은 다음 4개 "세션" 중 하나로 인벤토리화할 수 있으며 이때 `AcqG2Session` 매개변수로 지정한다. 각 태그는 세션 별로 별도의 "인벤토리" 상태를 유지하며, 일단 인벤토리화 했으면, 태그는 다른 SELECT 명령어를 받기 전까지는 같은 세션 중에 추가 리더 질의에는 응답하지 않는다. 태그는 사용한 세션 별로 정해진 기간 동안은 리더의 영역을 떠난 후에도 인벤토리화된 상태를 유지할 수 있다.

글로벌 스크롤 (Global Scroll)
글로벌 스크롤(Global Scroll)은 Alien RFID 리더 시스템이 지원하는 태그 ID 판독 작업 중 가장 기초적인 것이다. 글로벌 스크롤 명령어를 전송하면, RFID 리더가 전체 각 태그마다 한 개의 명령어를 전송한다. 이 명령어는 태그가 ID를 RFID 리더로 즉시 반송하도록 요청한다.

이 명령어가 이처럼 단순한 것인 장점이자 단점이기도 하다. 즉 이 명령어는 리더와 태그 간에 한 차례의 사이클만 수반하기 때문에 실행 속도가 매우 빠르다. 그러나 명령어가 단순하기 때문에 필드에 둘 이상의 태그가 있다면 문제가 발생할 수 있다. 이때 여러 태그가 모두 동일 명령어를 받고 ID를 사실상 동시에 리더에 반송하게 되는 것이다. 이런 상황은 리더가 전체 노이즈 중에서 개별 ID를 식별해 내기 어렵게 만든다. 일반적으로 가장 소음이 크거나 가장 인접한 태그 중 한 둘을 디코딩하고, 다수는 식별이 되지 않는다.

이는 사람으로 꽉 찬 암실로 걸어 들어가서 "내 목소리가 들리는 사람은 큰 소리로 이름을 말해 달라"고 명령하는 것과 같다. 방 안에 같이 있는 사람이 한 명이라면 그의 이름을 들을 수 있을 것이다. 하지만 여러 사람이 방 안에 있으면 소음이 심하기 마련이다. 한 둘의 이름은 분별해 낼 수 있을지 몰라도, 보통 그 이상은 어렵다.

그뿐 아니라, Q값은 글로벌 스크롤 인벤토리 중에는 자동으로 조정되지 않으며, 리더는 "A" 상태와 "B" 상태 사이의 클래스 1/Gen 2 태그들의 인벤토리들 간에 더 빠른 속도를 찾아 전환한다. 이러한 이유들 때문에 글로벌 스크롤 모드는 보통 RFID 상황 대다수에 대하여는 권장되지 않는다.

마스크 및 태그 메모리 구조

Alien RFID 태그를 대상으로 하는 다수의 명령어는 마스크 설정을 요구하는데, 마스크는 ID가 마스크에 부합하는 태그에서만 명령어를 전달한다. 이러한 메커니즘은 명령어가 하나의 태그, 선별한 태그 집단, 또는 전체 태그 필드로 발송되게 한다.

마스크의 용도를 이해하려면, 기본적으로 태그 메모리 구조를 이해해야 한다.

클래스 1/Gen 2 태그 메모리

클래스1 Gen2 데이터 개체 크기는 기본적으로 1바이트(8비트)가 아니라 한 단어(16비트)이다. 클래스 1/Gen 2 태그는 최대 4개의 메모리 뱅크를 포함한다.

G2 Bank	설명
Bank 0 RESERVED	'Kill' 과 'Access' 패스워드를 포함한다. 길이로는 각각 두 단어씩이다(4 바이트). 태그는 이 패스워드들은 실행할 필요가 없는데, 이 경우에 값은 0000 0000이고, 이에 해당하는 미사용 메모리 위치는 존재하지 않는다. 뱅크 0으로의 마스킹(Masking)은 허용되지 않는다.
Bank 1 EPC	하나의 CRC 단어, 하나의 PC 단어, 그에 따른 유동적인 수의 EPC 단어들을 포함한다. 구조에 관한 자세한 정보는 아래를 참조할 것.
Bank 2 TID	태그 확인 정보를 포함하며, 여기에는 할당 클래스 식별자(EPCglobal 등), 제조업체 정보, 태그 모델과 버전 정보 등이 해당된다.
Bank 3 USER	사용자별 데이터 저장을 위한, 구조화하지 않은 스크래치(scratch) 공간 태그는 USER 뱅크를 실행할 필요가 없으며, 메모리 용량은 태그 수행에 따라 달라진다.

클래스 1/GEN 2 EPC 구조

현재 클래스 1/Gen 2 태그는 보통 EPC 메모리 뱅크에 프로그래밍 가능한 메모리를 96 또는 128비트 포함하나, 사양에는 최대 496 EPC비트까지로 허용한다. EPC 코드에 더해, EPC 메모리 뱅크에도 CRC 체크섬 16비트와 프로토콜-컨트롤(PC) 16비트를 포함한다.

	CRC	PC	EPC Code (or User ID Code)						
Word	0	1	0	1	2	3	4	5	…
Bit	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127	…

Class 1/Gen 2 태그 메모리 구조

EPC 코드 어드레스 방향은 좌측에서 우측이며, 이때 좌측단 Bit는 (가장 유의미한 Bit) 32Bit이고 EPC 길이만큼 연장된다. 이 태그 구획에 있는 데이터에는 제한이 가해지지 않는다.

체크섬은 PC의 16비트와 전체 EPC 코드 상에서 태그에 의해 자동으로 계산된다. 체크섬은 리더가 자동으로 계산하여 태그로 프로그래밍 한다.

프로토콜-컨트롤(PC) 단어는 EPC 단어 수를 인코딩하고(비트 0-4), 두 개의 추가 비트(비트 5-6)와 넘버링 시스템 식별자(비트 7-15)가 뒤따른다. 넘버링 시스템 식별자(numbering system identifier)

EPCglobal™ 헤더(EPC™ 태그 데이터 표준으로 정의함) 또는 애플리케이션 제품군 식별자(ISO/IEC 15961로 정의함)을 지정한다.

프로그래밍 태그 ID와 태그 메모리에 대한 추가 정보는 '태그 프로그래밍' 섹션을 참조하기 바란다.

태그 서브셋 주소

마스크 명령어와 관련된 더 유용한 애플리케이션 중에는 필드에서 태그의 하위 집합 주소를 정의하는 것이 있다. 이때 마스크가 사용된다.

예를 들어, 다음 마스크 명령어를 전송하여 번호가 "8000"으로 시작하는 태그 ID만 주소를 정의할 수 있다.

```
AcqG2Mask = 1, 32, 16, 80 00
```

처음 값은 뱅크이다(EPC는 뱅크 #1). 두 번째 값은 사용자가 태그로 데이터 매칭을 시작하려고 의도한 뱅크 내 위치에 해당한다(실제 EPC는 비트 #32에서 시작함). 세 번째 값은 마스크 내 비트의 수이다(8비트 미만을 원할 수도 있음). 마지막으로, 16진수 바이트로 된 순서로 마스크 80 00을 지정한다.

이후 마스크를 사용하는 명령어들은 이 태그 ID로 시작하는 태그에서만 인지한다. 이 방식이 유용할 수 있는데, 예를 들어 리더는 식품 항목을 스캐닝하는 동안 특정 브랜드의 아침식사용 시리얼만 찾고 있을지 모른다. 마스크를 설정하여 아침식사용 시리얼 태그 ID만 찾게 하면, 식품 항목에 대한 수집 명령어로 해당 품목만 반환할 수 있다. 이 방법은 EPC 코드 전략과 결합했을 때 특히 더 효과가 있는데, 이때 각 제품 유형과 제조업체 코드가 잘 정의된 메모리 코드를 사용하기 때문에 마스킹이 수월한다.

또 다른 예로, 96-비트 EPC 코드의 마지막 3비트를 1로 설정해 놓은 태그를 모두 검색하는 경우가 있다. 이때 마스크 설정값은 다음과 같다.

```
AcqG2Mask = 1, 125, 3, E0
```

달리 말하면, 뱅크 #1에 마스킹 하고 96-비트 EPC가 비트 #32에서 시작하면, 마지막 3비트는 비트 #125(32+96-3)에서 시작한다. 길이는 3비트이고, $E0_{16진수}$의 데이터값에 맞추길 원한다고 하자. 마스크 값은 07_{hex} ($111_{2진수}$)가 아니라 $E0_{hex}$ ($11100000_{2진수}$)로 정의하는데, 이는 07_{hex} 는 $00000111_{2진수}$로 해석되기 때문이며, 이 경우 비트 패턴을 마스크로 적용할 때에 비트가 좌측에서 우측으로 적용된다(가장 유의미한 비트에서 덜 유의미한 비트의 방향으로).

마스킹 동작은 클래스 1/Gen 2 트랜잭션의 SELECT 구획의 일부로 수행되므로, 사용자는 AcqG2Select를 1보다 큰 값으로 설정해야 한다. AcqG2Select=0일 때, 인벤토리 중에는 SELECT 명령어를 태그에 전송하지 않으며, 마스크는 아무런 효력을 발생시키지 않는다.

제 4 장
Alien 리더 프로토콜

Alien 리더 프로토콜(ARP)은 Alien RFID 리더를 구성 및 작동하기 위한 텍스트 기반의 통신 프로토콜이다. 이 장에서는 Alien RFID 리더를 외부 세계와 연결하는 프로그래밍 인터페이스에 관해 기술한다.

리더 시스템에 관한 개요와 호스트 컴퓨터에서 리더 작동을 설정하기 위한 지침은 다음 제목의 섹션에서 참조할 수 있다. Reader 기본요소와 태그 기본요소.

일부 Alien 리더에서는 EPCglobal 저수준 리더 프로토콜(LLRP)을 지원하며, 이는 ARP의 대안으로 쓰인다. LLRP에 관한 추가 정보는 6장에서 찾을 수 있다.

리더 작동 개요

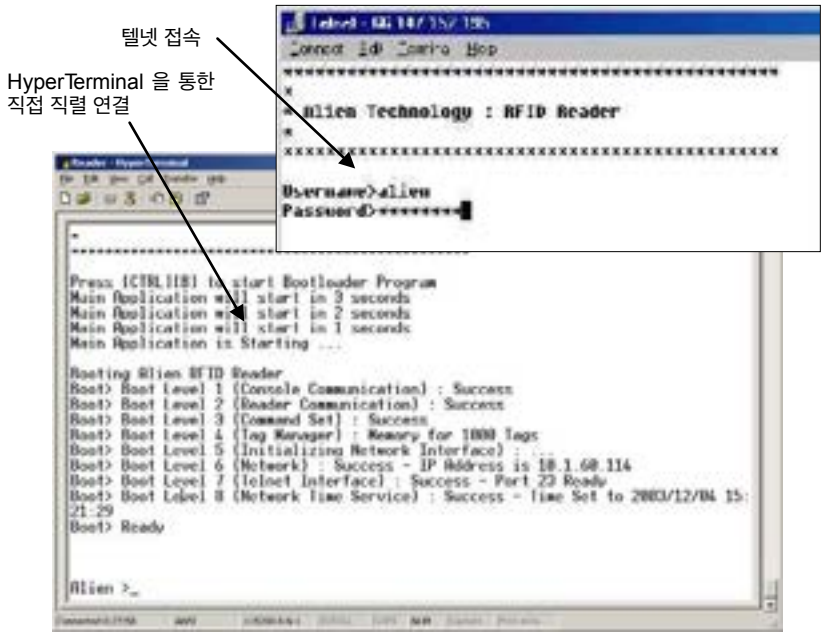
앞 장에서 자세하게 기술한 바와 같이, 사용자는 다음 두 가지 텍스트 기반 명령어 라인을 사용하여 리더와 통신하고 작업을 구성할 수 있다.

- 직접 직렬(RS-232) - 리더에서 지원하는 경우
- 텔넷 접속(TCP/IP)

이 지침들에서 직렬 및 텔넷 작업은 기본적으로 동일하다고 간주한다. 두 경우 모두 화면이 유사해 보이며, 다음 지침들의 목적에 따라서도 동일하다고 간주한다.

텔넷의 예외사항*:*

- 텔넷 작동 시 세션을 중지하려면 명령어 "q"를 전송해야 한다.
- 텔넷을 통해 리더에 접속하려면 공인된 사용자 이름과 암호가 필요하다(둘 다 일반 명령어 항목에 있는 명령어를 사용해 변경 가능하다).



명령어 개요

리더 동작을 크게 두 범주로 나눌 수 있는데, 호스트가 시작하는 것(인터랙티브 모드)과 리더 자체가 시작하는 것(자율 모드)이 있다.

INTERACTIVE MODE -인터랙티브 모드
인터랙티브 모드 동작은 별도의 호스트 컴퓨터에서 리더로 명령어를 전송하는 프로그램이나 사용자로 시작된다. 호스트는 늘 요청을 받고 트랜잭션을 시작하며, 리더는 즉시 응답한다. 호스트는 다른 명령어를 시작하기 전에 응답을 기다려야 한다.

인터랙티브 명령어를 사용하여 리더를 구성 및 작동시키기도 하고, 태그 질의와 보관되어 있는 태그리스트(TagList) 검색을 하기도 한다.

AUTONOMOUS MODE -자율 동작 모드
자율 모드(AutoMode) 작동 시 리더는 프로그래머가 설정한 조건에 따라 별도의 개입 없이 특정 태스크를 수행할 수 있다.

이러한 명령어는 일반적으로 리더가 트리거 이벤트에 기초하여 태그를 읽도록 하며(외부 디지털 입력 또는 내부 타이머), 다양한 태그리스트 트리거를 기초로 호스트 시스템에 공지 메시지를 보낸다. 예를 들어, 리더는 태그가 보일 때까지 필드를 검색하도록 지시를 받을 수 있으며, 그 다음에는 지정된 이메일 주소를 통해 전달되는 새로운 태그와 이메일을 읽기도 한다.

명령어 형식

호스트 시스템과 리더 사이에 오가는 모든 명령어는 사람이 읽을 수 있는, ASCII 텍스트 기반의 메시지이다.
특정 속성을 가진 값을 얻으려면 다음 명령어를 사용한다:

```
<attribute>?
```

특정 속성을 가진 값을 설정 하려면 다음 명령어를 사용한다.

```
<attribute> = <value>
```

예를 들어, "ReaderName" 명령어를 사용하여 리더의 이름을 설정하라는 명령어는 다음 형식을 취한다:

```
Alien>ReaderName = My Alien Reader[CR][LF]
```

리더에게 보내는 명령어는 모두 단일 행의 ASCII 문자열이며, 캐리지 리턴(carriage-return) / 줄바꿈 쌍으로 마친다. 그 작성 형태는 [CR][LF]이다. [CR] 문자는 ASCII 코드 0x0D이고, [LF] 문자는 ASCII 코드 0x0A이다. 이러한 문자는 각각 “\r”과 “\n”으로 쓰기도 한다.

리더로부터 받는 응답은 모두 단일 행 또는 다중 행의 ASCII 문자열이며, [CR][LF][0]으로 마친다, 이때 [0]은 ASCII 코드 0x00이다. 응답이 여러 텍스트 행으로 구성되는 경우에는 각 행을 단일 [CR][LF] 시퀀스로 분리한다.

명령어와 단일 행 응답의 예:

```
Alien>ReaderName?[CR][LF]
ReaderName = Alien Reader[CR][LF][0]
```

명령어와 다중 행 응답의 예:

```
Alien>t[CR][LF]
Tag:BEEF 0909 0909 0909, Disc:2008/08/12 ... Count:1, Ant:0, Proto:2[CR][LF]
Tag:DEAD BEEF CAFE 8042, Disc:2008/08/12 ... Count:1, Ant:0, Proto:2[CR][LF]
Tag:DEAD BEEF CAFE 8043, Disc:2008/08/12 ... Count:1, Ant:0, Proto:2[CR][LF][0]
```

명령어는 대소문자를 구분하지 않음:

"readername?" is equivalent to "ReADerNaME?".

명령 프롬프트 표시 안 함

기본적으로 리더는 인터랙티브 콘솔 스타일의 인터페이스를 사용해 명령어에 응답한다. 그 결과 응답 뒤에는 리더가 더 많은 사용자 입력값을 받을 준비를 하고 있음을 나타내는 명령어 프롬프트가 있다. 명령어 프롬프트가 필요하지 않을 때도 종종 있는데, 특히 클라이언트 소프트웨어를 작성하여 프로그래밍 방식으로 리더와 통신할 때가 그러하다.

명령어 프롬프트는 null로 끝나는 응답 뒤에 전송되므로, 이 프롬프트 텍스트는 다음 리더 응답의 시작으로 해석되기도 한다. 이를 위해 명령어 프롬프트는 명령어 문자열에 0x01 문자를 [1]로 써서 앞에 붙여 표시하지 않을 수도 있다. 예를 들면 다음과 같다.

INTERACTIVE COMMAND FORMAT -인터랙티브 명령어 형식

```
Alien>ReaderName?[CR][LF]
ReaderName = Alien Reader[CR][LF][0]

[CR][LF]Alien>
```

NON-INTERACTIVE COMMAND FORMAT -비 인터랙티브 명령어 형식

```
[1]ReaderName?[CR][LF]
ReaderName = Alien Reader[CR][LF][0]
```

"Get" 과 "Set" 의 호환성

Alien 리더 프로토콜의 이전 버전들에서는 값을 얻고 설정할 때 사용자가 "get <attribute>" and "set <attribute> = <value>" 을 입력할 것을 요구하였으나, 더 편리한 방법은 "<attribute>?" and "<attribute> = <value>" 이다. 이처럼 이전의 구문을 사용해 리더와 통신하는 시스템과도 호환되도록 리더는 그런 형식도 수용한다.

이전 구문	표준 구문
Get <attributeName>	<attributeName>?
Set <attributeName> = <attributeValue>	<attributeName> = <attributeValue>

예제:

```
Alien>get ReaderName
ReaderName = Alien RFID Reader

Alien>ReaderName?
ReaderName = Alien RFID Reader

Alien>set ReaderName = Alice
ReaderName = Alice

Alien>ReaderName = Alice
ReaderName = Alice
```

명령어 히스토리 버퍼와 "!"

리더는 사용자가 입력한 마지막 10개 명령어를 기억하며, 사용자는 위/아래 화살표 키를 사용하여 명령어 기록을 검색해 볼 수 있다. 상향 화살표는 비교적 오래된 명령어를 보는 방향이며, 하향 화살표는 비교적 최근의 명령어를 보는 방향이다.

명령어를 히스토리 버퍼에서 검색할 때, 커서가 행의 끝에 위치하여 명령어를 쉽게 편집할 수 있게 하였으며, 이를 전 송하려면 [ENTER]만 누르면 된다. 이 방식을 잘못 입력했을 때 확인하는데 도움이 되며, 또는 이전에 입력했던 명령 어를 찾아 다시 입력할 수도 있다.

"!" 명령어는 최근에 입력한 명령어를 반복한다. 이 명령어를 사용하면 새로운 값으로 속성을 재설정할 수 있다. 예를 들어, "ReaderName=Bob" 을 입력한 다음 "! = Larry" 를 입력하여 ReaderName을 Larry로 설정할 수 있다.

XML 메세지

텍스트 기반 응답과 메시지를 손쉬운 컴퓨터 파싱을 위해 XML 형식으로 변환하는 경우도 있다. 각 XML 문서에 전체문서형식정의(DTDs)를 이 문서 부록과 개발자 키트 CD에 제공하였다.

다음 메시지는 XML 형식으로 전송된다:

- 하트비트 (Heartbeat) 메시지
- 공지 메시지 (NotifyFormat = XML 으로 설정된 경우)
- "get TagList" 명령어 (TagListFormat = XML 으로 설정된 경우)

명령어 목록

GENERAL COMMANDS -일반 명령어

명령어	설명	F800	9900+	9680	9650
Help ("h")	이용 가능한 리더 명령어를 모두 나열함.	✓	✓	✓	✓
Info ("i")	현재 리더의 설정값을 모두 나열함.	✓	✓	✓	✓
! *(느낌표)*	최근에 사용한 명령어를 반복함.	✓	✓	✓	✓
Save	현재 설정값을 플래시메모리에 저장함.	✓	✓	✓	✓
Quit ("q")	세션 종료(텔넷의 경우에만)	✓	✓	✓	✓
Function	명령어 클래스를 활성화 및 비활성화하는 데 사용함 (이를테면 프로그래밍)	✓	✓	✓	✓
ReaderName	리더와 관계된 임의의 이름	✓	✓	✓	✓
ReaderType	리더 유형에 대한 설명	✓	✓	✓	✓
ReaderVersion	리더 소프트웨어/하드웨어 버전	✓	✓	✓	✓
DSPVersion	DSP 펌웨어 버전과 구성	✓	✓	✓	✓
ReaderNumber	리더와 관계된 임의의 이름(1-255)	✓	✓	✓	✓
BaudRate	직렬 인터페이스 전송 속도.	✓	✓	✓	✓
Uptime	리더를 최근에 부팅한 이후로 경과한 초 수.	✓	✓	✓	✓
Username	네트워크 기반 접속 제어에 사용한 사용자명(Username)	✓	✓	✓	✓
Password	네트워크 기반 접속 제어에 사용한 패스워드(Password)	✓	✓	✓	✓
MaxAntenna	주소 지정이 가능한 안테나 포트 최대수 (안테나의 총 수는 MaxAntenna+1)	✓	✓	✓	✓
AntennaSequence ("ant")	리더가 사용할 안테나 포트 시퀀스.	✓	✓	✓	✓
RFAttenuation	방출한 RF에 해당되는 디지털 감쇠의 정도.	✓	✓	✓	✓
RFLevel	출력. (RFLevel = <maxPower> - <RFAttenuation>	✓	✓	✓	✓
RFModulation	Class1/Gen2 변조 방식.	✓	✓	✓	✓
FactorySettings	리더를 공장 설정값으로 재설정함.	✓	✓	✓	✓
Reboot	리더를 리부팅함.	✓	✓	✓	✓
Service	상태 보기, 시작/정지, 보조 서비스 자동 시작 제어	✓	✓	✓	✓
ETSIMode	EN 300 220과 EN 302 208 v1.3.1 사이에 규제 모드 스위칭		✓		
MyData	사용자의 개인 데이터용 일반 스크래치 패드	✓	✓	✓	✓

NETWORK COMMANDS - 네트워크 명령어

명령어	설명	F800	9900+	9680	9650
MACAddress	리더 별로 고유한 MAC 주소가 할당됨	✓	✓	✓	✓
DHCP	DHCP가 on 일 때 리더는 전원이 커진 상태에서 IPV4 네트워크 설정값을 자동으로 구성한다.	✓	✓	✓	✓
DHCPTimeout	리더가 DHCP 서버로부터의 응답을 기다려야 하는 시간의 초 수.	✓	✓	✓	✓
IPAddress	리더의 IPV4 주소	✓	✓	✓	✓
Netmask	IPV4 서브넷 마스크	✓	✓	✓	✓
Gateway	IPV4 게이트웨이	✓	✓	✓	✓
DHCP6	DHCP6가 활성일 때 리더는 전원이 커진 상태에서 IPV6 네트워크 설정값을 자동으로 구성한다.	✓			
IPAddress6	리더의 IPV6 주소	✓			
Netmask6	IPV6 서브넷 마스크	✓			
Gateway6	IPV6 게이트웨이	✓			
DNS	네트워크 도메인 네임 서버.	✓	✓	✓	✓
Hostname	네트워크 호스트명. DHCP가 우선할 수도 있음.	✓	✓	✓	✓
UpgradeAddress	펌웨어 업데이트를 찾을 수 있는 주소	✓	✓	✓	✓
NetworkUpgrade	자동 네트워크 펌웨어 업그레이드 가능	✓	✓	✓	✓
UpgradeNow	리더 펌웨어 업그레이드 시작	✓	✓	✓	✓
NetworkTimeout	리더가 오래된 소켓을 닫기 전 시간	✓	✓	✓	✓
CommandPort	리더의 명령어 채널용 네트워크 포트 #	✓	✓	✓	✓
CommandPortLocal	리더의 로컬 명령어 채널용 네트워크 포트 #	✓	✓	✓	✓
AcceptConnections	리더가 외부 TCP 연결을 수락할 것인지를 제어함.	✓	✓	✓	✓
Ping	다른 장치와의 네트워크 연결을 테스트함.	✓	✓	✓	✓
HeartbeatAddress	하트비트(Heartbeat) 메시지를 전송할 IP 주소	✓	✓	✓	✓
HeartbeatPort	하트비트(Heartbeat) 메시지를 보낼 포트 #	✓	✓	✓	✓
HeartbeatTime	각 하트비트(Heartbeat) 사이의 시간 초 수	✓	✓	✓	✓
HeartbeatCount	부팅 후 전송할 하트비트(Heartbeat)의 총 수	✓	✓	✓	✓
HeartbeatNow	즉시 하트비트(Heartbeat) 메시지를 시작함.	✓	✓	✓	✓
WWWPort	리더의 웹 서버 포트 번호	✓	✓	✓	✓
HostLog	최근 호스트 동작의 로그 덤프 확인	✓	✓	✓	✓
DebugHost	호스트가 전송한 리더 명령어 일체의 로깅 제어	✓	✓	✓	✓

TIME COMMANDS - 시간 명령어

명령어	설명	F800	9900+	9680	9650
TimeServer	네트워크 타임 서버 주소	✓	✓	✓	✓
TimeZone	리더 시계에 적용할 UTC 시간대 오프셋	✓	✓	✓	✓
Time	리더 시계 상의 시간	✓	✓	✓	✓

MACRO COMMANDS - 매크로 명령어

명령어	설명	F800	9900+	9680	9650
MacroList	설치된 모든 매크로의 명칭 표시	✓	✓	✓	✓
MacroView <macro>	명명된 매크로의 내용 표시	✓	✓	✓	✓
MacroRun <macro>	명명된 매크로의 명령어 실행	✓	✓	✓	✓
MacroDel <macro>	명명된 매크로를 영구 제거	✓	✓	✓	✓
MacroDelAll	설치된 매크로를 모두 영구 제거	✓	✓	✓	✓
MacroStartRec <macro>	새 매크로를 만들어 기록을 시작함	✓	✓	✓	✓
MacroStopRec	매크로 기록을 정지함	✓	✓	✓	✓
MacroCopy <m1> <m2>	매크로 사본 만들기	✓			

TAGLIST COMMANDS - 태그 목록 명령어

명령어	설명	F800	9900+	9680	9650
get TagList ("t")	인벤토리를 수행하고 현재 태그리스트를 반환함	✓	✓	✓	✓
to	인벤토리를 수행하고 현재 AcqG2OpsMode 설정에 관계 없이 AcqG2Ops를 실행함	✓	✓		
PersistTime	태그리스트에서 태그에 대한 지속 시간 표시	✓	✓	✓	✓
TagListFormat	태그리스트의 출력 형식	✓	✓	✓	✓
TagListAntennaCombine	여러 안테나에서 태그 판독 내용을 모아 하나의 태그리스트 항목에 조합할 지 여부를 표시함.	✓	✓	✓	✓
TagListCustomFormat	사용자 정의 태그리스트 형식 설정 문자열	✓	✓	✓	✓
TagDataFormatGroupSize	형식화된 태그리스트 내 데이터 집단의 크기를 바이트 단위로 표시	✓	✓	✓	✓
TagListMillis	태그리스트 타임스탬프에 밀리세컨드 단위로 표시	✓	✓	✓	✓
Clear TagList	리더에서 활성 태그 목록을 지움	✓	✓	✓	✓
TagStreamMode	태그 스트리밍 On / Off	✓	✓	✓	✓
TagStreamAddress	태그 스트리밍 주소(IP:포트 또는 “Serial”)	✓	✓	✓	✓
TagStreamFormat	태그스트림(TagStream) 형식.	✓	✓	✓	✓
TagStreamCustomFormat	사용자 정의 태그스트림 형식 설정 문자열	✓	✓	✓	✓
TagStreamKeepAliveTime	유휴 상태의 태그스트림 소켓을 열어 두는 시간(초)	✓	✓	✓	✓
TagStreamCountFilter	스트리밍한 대그 데이터의 양의 제한함	✓	✓	✓	✓

TagStreamServer	리더의 TagStreamServer 포트에 연결한 TCP 소켓으로 태그 데이터를 스트리밍 함	✓			
StreamHeader	태그스트림이나 IO스트림(IOStream) 소켓을 열 때 리더가 헤더를 확인할 수 있게 함	✓	✓	✓	✓

ACQUIRE COMMANDS - 수집 명령어

명령어	설명	F800	9900+	9680	9650
AcquireMode	태그 판독에 사용할 인벤토리 모드	✓	✓	✓	✓
AcqG2Cycles	각각의 Gen2 태그 인벤토리 중 수행할 수집 주기 수	✓	✓	✓	✓
AcqG2Count	각각의 Gen2 인벤토리 주기 중 수행할 판독 수	✓	✓	✓	✓
AcqG2Q	각각의 Gen2 인벤토리 중 사용할 첫 "Q" 값	✓	✓	✓	✓
AcqG2QMax	Gen 2 판독 중 Q 매개변수의 최대값	✓	✓	✓	✓
AcqG2Select	Gen2 인벤토리 시작 시 B->A를 실행하는 횟수	✓	✓	✓	✓
AcqG2Session	태그 인벤토리 수행 중 사용할 G2 세션	✓	✓	✓	✓
TagType	찾고 있는 태그 유형을 명시	✓	✓	✓	✓
G2Wake	태그를 인벤토리화하지 않은 상태가 되도록 Class1/Gen2 Select 명령어를 전송함	✓	✓		
AcqG2Mask	Class1/Gen2 마스크 메모리 뱅크	✓	✓	✓	✓
AcqG2MaskAction	AcqG2Mask가 일치하는 태그를 포함시킬지 배제할지 여부	✓	✓	✓	✓
AcqG2MaskAntenna	G2 마스크를 적용할 안테나의 비트맵	✓	✓		
AcqG2SL	G2 SL 플래그를 인벤토리 중 사용할지 여부를 제어	✓	✓		
AcqG2AccessPwd	패스워드로 보호한 Gen2 태그 작업에 사용할 접속(Access) 패스워드	✓	✓	✓	✓
AcqG2TagData	태그리스트 검색을 위해 추가로 Class1/Gen2 태그 데이터 블록을 지정함	✓	✓		
AcqG2Target	Class1/Gen2 인벤토리 대상을 설정함(A, B, A<->B)	✓	✓	✓	✓
AcqG2AntennaCombine	Gen2 SELECT 명령어를 인벤토리 주기 중 전송할 방법을 제어	✓	✓		
AcqG2Ops	G2Ops 태그 인벤토리 중 수행할 추가적인 C1G2 태그 동작을 설정함	✓	✓		
AcqG2OpsMode	전체 인벤토리 중에서 AcqG2Ops 작동을 활성화함	✓	✓		
AcqTime	인벤토리 지속 기간의 한계	✓	✓		
SpeedFilter	속도 및 방향을 근거로 태그를 필터링함	✓	✓		
RSSIFilter	RF 신호 강도를 근거로 태그를 필터링함	✓	✓	✓	✓
TagAuth	Alien 동적 태그 인증 필터	✓	✓		

+ = ALR-9900+ only

I/O COMMANDS - 입/출력 명령어

명령어	설명	F800	9900+	9680	9650
ExternalInput	외부 입력값을 받음	✓	✓	✓	✓
ExternalOutput	외부 출력값을 받아 설정함	✓	✓	✓	✓
InvertExternalInput	외부 입력값 전환 On / Off 전환했을 경우, 입력 전압이 올라가면 현재 낮다는 의미임	✓	✓	✓	✓
InvertExternalOutput	외부 출력값 전환 On / Off 전환했을 때 출력을 높게 설정하면 전압이 낮아진다.	✓	✓	✓	✓
InitExternalOutput	리더 부팅 중과 후에 외부출력 상태	✓	✓	✓	✓
IOList ("ios")	I/O 이벤트의 형식화한 목록을 받음	✓	✓	✓	✓
Clear IOList	IOList에서 모든 I/O 이벤트를 지움	✓	✓	✓	✓
IOType	추적할 I/O 이벤트 유형	✓	✓	✓	✓
IOListFormat	IOList 형식	✓	✓	✓	✓
IOListCustomFormat	사용자 정의 IOList 형식 설정 문자열	✓	✓	✓	✓
IOPersistTime	IOList에서 IO 이벤트에 대한 지속 시간 표시	✓	✓	✓	✓
IOStreamMode	I/O 스트리밍 On / Off	✓	✓	✓	✓
IOStreamAddress	I/O 이벤트 스트리밍 주소(IP:포트 또는 “Serial”)	✓	✓	✓	✓
IOStreamFormat	I/O 스트림 형식	✓	✓	✓	✓
IOStreamCustomFormat	사용자 정의 I/O 스트림 형식 설정 문자열	✓	✓	✓	✓
IOStreamKeepAliveTime	유휴 상태의 I/OStream 소켓을 열어 두는 시간(초)	✓	✓	✓	✓
BlinkLED	사용자가 정의한 방식으로 리더의 상태 LED를 점멸함	✓	✓	✓	✓

AUTOMODE COMMANDS - 자율모드 명령어

명령어	설명	F800	9900+	9680	9650
AutoMode	자율모드 On / Off	✓	✓	✓	✓
AutoWaitOutput	대기 상태에 있을 때 출력값	✓	✓	✓	✓
AutoStartTrigger	AutoAction을 시작하는 입력 트리거	✓	✓	✓	✓
AutoStartPause	시작 트리거를 받은 후부터 AutoAction을 시작하기 전까지 시간 (밀리세컨드)	✓	✓	✓	✓
AutoWorkOutput	작동 상태에 있을 때 출력값	✓	✓	✓	✓
AutoAction	AutoMode의 작동 상태 중 수행하는 동작	✓	✓	✓	✓
AutoStopTrigger	AutoAction을 중단하는 입력 트리거	✓	✓	✓	✓
AutoStopTimer	AutoAction을 수행하는 시간(밀리세컨드)	✓	✓	✓	✓
AutoStopPause	중지 트리거를 받은 후부터 AutoAction을 중지하기 전까지 시간 (밀리세컨드)	✓	✓	✓	✓
AutoTrueOutput	AutoMode가 '참'으로 평가할 때 출력값	✓	✓	✓	✓
AutoTruePause	AutoMode가 '참'으로 평가한 후 정지하는 시간(밀리세컨드)	✓	✓	✓	✓
AutoFalseOutput	AutoMode가 '거짓'으로 평가할 때 출력값	✓	✓	✓	✓
AutoFalsePause	AutoMode가 '거짓'으로 평가한 후 정지하는 시간(밀리세컨드)	✓	✓	✓	✓
AutoErrorOutput	AutoMode가 '거짓'으로 평가할 때 오류종속(error-dependent) 출력값	✓	✓		
AutoProgError	AutoMode 프로그래밍 동작이 실패할 때 보고할 오류 코드 설정	✓	✓		
AutoModeTriggerNow	AutoStartTrigger 이벤트를 발생시킴	✓	✓	✓	✓
AutoModeReset	AutoMode 값을 모두 기본상태로 재설정함	✓	✓	✓	✓

NOTIFY COMMANDS - 공지 명령어

명령어	설명	F800	9900+	9680	9650
NotifyMode	공지 모드 On / Off	✓	✓	✓	✓
NotifyFormat	TagList 공지 메시지 형식	✓	✓	✓	✓
NotifyHeader	공지 메시지 중 추가 헤더(및 바닥글) On / Off 전환	✓	✓	✓	✓
NotifyAddress	TagList 공지를 전송할 주소	✓	✓	✓	✓
NotifyTime	타이머 트리거 공지 메시지들 간의 간격(초)	✓	✓	✓	✓
NotifyTrigger	TagLists 푸싱을 위해 AutoMode로 발생시킨 트리거	✓	✓	✓	✓
NotifyKeepAliveTime	유휴 상태의 공지 소켓을 열어 두는 시간(초)	✓	✓	✓	✓
MailServer	이메일 공지를 위한 SMTP 메일 서버	✓	✓	✓	✓
MailFrom	RFID 리더의 이메일 주소 식별자	✓	✓	✓	✓
NotifyRetryCount	소켓 공지가 실패했을 때 재시도하는 횟수	✓	✓	✓	✓
NotifyRetryPause	공지 메시지가 실패한 뒤 재시도하는 간격(초)	✓	✓	✓	✓
NotifyQueueLimit	차후에 전송하기 위해 늘어놓은 실패한 공지 수	✓	✓	✓	✓
NotifyInclude	공지에 태그 데이터 또는 I/O 이벤트 데이터 또는 둘 다를 포함시킬 것인지 여부	✓	✓	✓	✓
NotifyNow	공지 시스템을 통해 메시지를 보냄	✓	✓	✓	✓

프로그래밍 태그 메모리와 관계된 명령어는 나중 섹션에서 전체적으로 다룰 것임.

다음 섹션들은 각 명령어 카테고리에 대해 기술하여, 각 명령어, 용법, 및 응답 형식 등을 자세히 설명한다.

유의사항: RFID 리더 명령어는 대소문자 구별을 **하지 않으며**, 대문자나 소문자 또는 그 조합을 사용하더라도 리더가 이를 인지한다. 본 문서에서 명령어를 대문자화하여 사용하고 실제 명령어 응답 형태로 표시한 것은 가독성을 높이기 위한 목적일 뿐이다.

일반 명령어

일반 명령어(General commands)에는 기본적인 리더/안테나 기능들과 도움말 및 기본정보를 포함한다.

도움말 Help (h)

F800 | 9900 | 9680 | 9650

이 명령어는 이용 가능한 모든 리더 명령어를 열거하며 기능 별로 범주화하였다. Help(도움말) 명령어에 명령어 이름을 입력하면 해당 명령어에 대한 도움말을 볼 수 있다.

- 명령어 “h”는 “Help” 명령어의 약어이다.

Help 예제	
Command Response	>help ********** GENERAL COMMANDS ********** Help: Display descriptive help text. Info: Display current settings. !: Repeat last command. Save: Save current settings to permanent storage. ...
Command Response	// Help for single command >h NotifyNow NotifyNow: Manually trigger NotifyMode.

정보 Info (i)

F800 | 9900 | 9680 | 9650

이 명령어는 현재 리더 설정값 목록을 보여준다. 리더 설정값은 기능 별로 범주화하였다.

- 명령어 "i"는 "Info" 명령어의 약어이다.

Info 명령어에 명령어 그룹명을 입력하면 해당 그룹에 따른 현재 설정값을 보여준다. 그룹명은 Help와 Info 덤프에 별표 사이에 표시된다.

Info 예제	
Command Response	>Info ********** GENERAL COMMANDS ********** Function = Reader ReaderName = Alien RFID Reader ReaderType = Alien RFID Tag Reader, Model: ALR-9800... ReaderVersion = 07.01.10.00 DSPVersion = DSP:02.01.05 reader:05 country:01 radio:32 board:01 ReaderNumber = 255 ...
Command Response	// Info for a single command group >i network ********** NETWORK COMMANDS ********** MACAddress = 00:80:66:10:11:6A DHCP = ON IPAddress = 10.10.82.72 Hostname = fubar UpgradeAddress = 0.0.0.0 NetworkUpgrade = OFF ...

!

F800 | 9900 | 9680 | 9650

이 명령어는(느낌표) 리더에게 최근 전송한 명령어를 반복하도록 요청한다.

! 예제	
Command Response	>Program Tag = DE AD BE EF CA FE B0 B0 Error 134: No Tag Found. >! Program Tag = Success!

이전 “set” 명령어를 반복할 때 “! =” 단축키를 사용해 새 값을 입력할 수도 있다. 이전 명령어가 “set” 명령어가 아니었으면 리더는 ‘잘못된 메시지’ 오류라고 응답한다.

"! =" 예제	
Command Response Command Response	>ExternalOutput = 0 ExternalOutput = 0 >! = 1 ExternalOutput = 1
Command Response Command Response	>ReaderName? ReaderName = myAlienReader >! = yourAlienReader Error: Malformed Message

저장 Save

F800 | 9900 | 9680 | 9650

저장(Save) 명령어는 현재 설정값을 플래시 메모리에 기록한다. 이로써 리더가 정전이 되었을 때 동일 상태로 재시작하게 할 수 있다.

ALR-F800 리더는 사용자가 현재 설정값을 파일로 저장할 수 있게 해준다. 그 결과로 생성된 프로파일은 리더 매크로 형태이며, 리더의 명령어 전체가 리더를 저장한 상태로 복구하게 한다. "Save <macroName>"를 사용하면 이제 해당 이름을 가진 새 리더 매크로를 이용할 수 있다.

Save 예제	
Command Response	>Save All settings have been saved to flash!
 Command Response Command Response Command Response	// ALR-F800 Only >Save mySettings Settings saved to profile "mySettings". >MacroList example1 example2 example3 mySettings >MacroView mySettings # # GENERAL SETTINGS # ReaderName = Alien RFID Reader ReaderNumber = 255 BaudRate = 115200 ... etc...

종료 Quit (q)

F800 | 9900 | 9680 | 9650

(텔넷 작동에만 해당됨) 종료(Quit) 명령어를 사용하면 현재 텔넷 세션에서 나갈 수 있다.

- 명령어 "q"는 "Quit" 명령어의 약어이다.

기능 Function

F800 | 9900 | 9680 | 9650

기능(Function) 명령어는 이전 리더 모델과의 후향 호환성을 지원한다. 이전 버전의 리더에서 프로그래밍 기능들은 (program, lock, erase, kill 등) 우선 "리더"에서 "프로그래머"로 기능 속성을 변경해야 활성화할 수 있었다. 현재 리더는 프로그래밍 기능이 항상 활성화되어 있으므로, 이 명령어는 더 이상 필요하지 않다.

- 허용된 값: "Reader" | "Programmer"
- 기본값: "Reader"

Function 예제	
Command Response	> Function? Function = Reader
Command Response	>Function = Programmer Function = Programmer

리더 이름 ReaderName

F800 | 9900 | 9680 | 9650

리더에는 임의의 텍스트명이 지정되어 다중 리더 환경에서도 식별이 가능하다. 이 이름은 리더 작동 중 언제든 검색 및 변경할 수 있다.

- 허용된 값: String[1-254]
- 기본값: "Alien RFID Reader"

ReaderName 예제	
Command Response	>ReaderName? ReaderName = My First Alien Reader
Command Response	>ReaderName = My Second Alien Reader ReaderName = My Second Alien Reader

리더 유형 ReaderType

F800 | 9900 | 9680 | 9650

리더 유형은 이 명령어를 사용해 검색할 수 있다. 그 결과 생성된 텍스트는 리더의 모델 번호와 관련 정보를 기술한 단일 행 응답이다.

ReaderType 예제	
Command Response	>ReaderType? ReaderType = Alien RFID Tag Reader, Model: ALR-F800
Command Response	>ReaderType? ReaderType = Alien RFID Tag Reader, Model: ALR-9900 (Four Antenna / Gen 2 / 902-928 MHz)
Command Response	>ReaderType? ReaderType = Alien RFID Tag Reader, Model: ALR-9650 (One Antenna / Gen 2 / 902-928 MHz)

리더 버전 ReaderVersion

F800 | 9900 | 9680 | 9650

리더 버전은 이 명령어를 사용해 검색할 수 있다. 그 결과 생성된 텍스트는 다중 행 응답이다. 응답의 각 행은 리더의 주요 구성요소의 버전 번호와 부수적인 지역 정보를 기술한다.

ReaderVersion 예제	
Command Response	>ReaderVersion? ReaderVersion = 06.06.18.00

DSP 버전 DSPVersion

F800 | 9900 | 9680 | 9650

DSP 버전은 이 명령어를 사용해 검색할 수 있다. 문제해결 시 Alien 지원팀에 유용할 만한 구성 및 지역 정보를 표시한다.

DSPVersion 예제	
Command Response	// ALR-9900 >DSPVersion? DSPVersion = DSP:04.06.01 reader:08 country:01 radio:35 board:04 FPGA:09.03.17.00
Command Response	// ALR-9680 & 9650 >DSPVersion? DSPVersion = DSP:02.003.032-01.003.000 reader:09 country:01 radio:12 board:20

리더 번호 ReaderNumber

F800 | 9900 | 9680 | 9650

리더에는 임의의 텍스트명이 지정되어 다중 리더 환경에서 식별을 용이하게 한다. 이 번호는 리더 작동 중 언제든 검색 및 변경할 수 있다.

- 허용된 값: Integer(1..255)
- 기본값: 255

ReaderNumber 예제	
Command Response	>ReaderNumber? ReaderNumber = 255
Command Response	>ReaderNumber = 15 ReaderNumber = 15

전송 속도 비율 BaudRate

F800 | 9900 | 9680 | 9650

리더의 직렬 인터페이스는 보통 115,200 보드(baud)에서 작동한다. 이는 대부분의 직렬 포트 수행 표준에 해당하는 가장 빠른 데이터 전송률이다. 리더의 직렬 보드 전송률은 이보다 느린 네 단계의 전송률로 조정하여 레거시 장비와 호환되도록 하였다.

- 허용된 값: 9600 | 19200 | 38400 | 57600 | 115200
- 기본값: 115200
- 0..4 값도 허용되며, 이로써 이전 리더 모델들과도 호환된다.

0 = 115,200 baud
1 = 57,600 baud
2 = 38,400 baud
3 = 19,200 baud
4 = 9,600 baud

- 변경은 즉시 효력을 나타내지 않고, 리더를 재부팅하여 BaudRate 변경이 적용되도록 해야 한다.
- 보드 전송률은 리더 명령어 인터페이스에만 영향을 미친다. 직렬 서비스를 중단해야 할 경우, 리눅스 쉘 로그인 조건은 115,200 baud이다. 직렬 포트를 완전히 해제할 경우(사용자의 애플리케이션 사용을 위해), 사용자가 스스로 보드 전송률을 제어한다.

BaudRate Examples	
Command Response	>BaudRate = 19200 BaudRate = 19200

가동시간 Uptime

F800 | 9900 | 9680 | 9650

Uptime 명령어는 최근 리더 재부팅 이후로 경과 시간을 초 단위로 반환한다.

Uptime 예제	
Command Response	>Uptime? Uptime = 702048

사용자명 Username

F800 | 9900 | 9680 | 9650

리더는 네트워크 상에서 작동할 수 있다. 이 모드에서 작동하는 중에, 원하지 않는 방문자가 접속해 있을 때 이를 중단시키려면 간단한 사용자명/암호 인증 수단을 사용할 수 있다. 이 명령어로써 현재 사용자명을 확인할 수 있다.

- 사용자명/패스워드 쌍은 직렬 연결로 리더를 작동할 때에는 필요하지 않다.
- 사용자명은 대소문자를 구별한다.
- ALR-F800은 다음 두 명의 사용자를 지원한다: “alien”과 “guest”. 게스트(guest) 사용자로 로그인했을 때, 민감한 리더 명령어들 다수를(네트워크 및 펌웨어 업그레이드 명령어 등) 사용할 수 없게 된다.
- ALR-9900/9650/9680 리더는 한 명의 사용자만 가능하나, 사용자의 이름은 변경할 수 있다.

Username 예제	
Command Response	>Username? Username = alien
Command Response	>Username = hal Username = hal

비밀번호 Password

F800 | 9900 | 9680 | 9650

리더는 네트워크 상에서 작동할 수 있다. 이 모드에서 작동하는 중에, 원하지 않는 방문자가 접속해 있을 때 이를 중단시키려면 간단한 사용자명/암호 인증 수단을 사용할 수 있다. 이 명령어를 사용하면 패스워드를 정의 및 획득할 수 있다.

- 허용된 값: String[1-80]
- 기본값: “password”
- 사용자명/패스워드 쌍은 직렬 연결 시에는 필요하지 않다.
- 패스워드는 대소문자를 구분하며, 보이는 ASCII 문자를 포함해야 한다.
- ALR-F800 리더는 보안상의 이유로 사용자가 현재 패스워드를 보도록 허용하지 않는다.

Password 예제	
Command Response	>Password? Password = password
Command Response	>Password = 1234fab Password = 1234fab

최대 안테나 MaxAntenna

F800 | 9900 | 9680 | 9650

리더의 주소지정이 가능한 안테나 포트 최대수를 결정하려면, MaxAntenna 명령어를 사용한다. 안테나 포트는 0부터 시작하여 번호를 지정하며, 따라서 실제 포트 수는 MaxAntenna 값보다 1이 많은 수이다. 4개의 안테나 포트가 있는 리더는 MaxAntenna값으로 3을 반환한다. 마찬가지로, 2개의 안테나 포트만 있는 리더는 MaxAntenna값으로 1을 반환한다.

MaxAntenna 예제	
Command Response	>MaxAntenna? MaxAntenna = 3

안테나 순서 AntennaSequence (ant)

F800 | 9900 | 9680 | 9650

리더는 여러 안테나 사용을 지원할 수 있다. 이 명령어를 사용하면 어떤 안테나 포트를 사용할지, 어떤 순서로 할지를 선택할 수 있다. AntennaSequence에서 하나의 안테나를 지정하기로 선택할 수도 있다.

리더는 전체 안테나에 걸쳐 주기가 반복되며, 각 공중핵(atomic air) 프로토콜 별로 AntennaSequence에서 지정한 순서를 따른다. 여기에는 일반적인 인벤토리 명령어에 더하여 Sleeps, Wakes, Selects도 포함된다.

- 허용된 값: 1-8 정수(각각, 0..3)
- 기본값: "0 1"
- 명령어 "ant"는 "AntennaSequence" 명령어의 약어이다.
- 최대 8개 안테나 포트값을 순서대로 지정할 수 있다. 안테나 포트는 여러 차례 나열할 수 있으며, 이로써 리더는 이 지정된 안테나들에서 더 많은 시간을 할당할 수 있다.

Mono-Static AntennaSequence 예제	
Command Response	>AntennaSequence? AntennaSequence = 0
Command Response	// To always use antenna 1: >AntennaSequence = 1 AntennaSequence = 1
Command Response	// To cycle between antenna 0 and antenna 1: >AntennaSequence = 0 1 AntennaSequence = 0 1
Command Response	// To weight antenna 0 more than antenna 1: >AntennaSequence = 0 0 0 1 AntennaSequence = 0 0 0 1

RF 감쇠 RFAttenuation

F800 | 9900 | 9680 | 9650

Alien RFID 리더 출력은 각 안테나 별로 최대 1 와트 RF 전력 조건이다. 이 전력이 투과 및 범위 조건에는 충분히 적합하나, 언제나 이상적인 것은 아니다. 여러 리더가 동일 영역에 있을 경우, 신호들에 간섭이 발생할 수 있다. 또한 태그가 있는 제품들이 인접해 있는 상황에서도 한 번에 하나의 제품씩만 판독이 가능하기 때문에(건베이어 벨트 등), 투과력과 범위 조건에 제약이 따른다.

RF 감쇠로 인해 역량이 줄어드는 문제를 해결하는 두 가지 방법이 있다. 첫 번째 방법은 안테나 케이블에 일렬로 감쇠기를 배치하는 것이다. 이 방법은 즉시 효과가 나타나나 유동적이지는 못하며, 더 중요한 것으로 방출한 RF 전력을 감소시키고 더 나아가 이미 약해진 상태로 태그 신호를 반환한다. 그 결과 리더의 태그 식별 능력이 손상된다.

두 번째 방법은 소프트웨어로 제어하는 디지털 감쇠 장치를 Alien 리더 내에 설치하는 것이다. 소프트웨어로 제어하는 디지털 감쇠장치를 사용하면 방출된 전력은 감소시킬 수 있으나 신호를 반환하지는 않는다. RFAttenuation 값의 범위는 0(감쇠 없음. 최대 전력)에서 maxAttenuation (최대 감쇠, 최소 전력)까지이며, RF 감쇠 1dB를 나타내는 10씩 증분된다.

이 명령어는 다음 조건들에서 사용이 가능하다.

1. `RFAttenuation = <attenuation>`

   이 명령어는 여러 안테나에 같은 감쇠를 설정한다.

2. `RFAttenuation = <antenna> <attenuation>`

   이 명령어는 각 안테나에 개별적으로 감쇠를 설정한다.

3. `RFAttenuation = <antenna> <readattenuation> <writeattenuation>`

   `<antenna>` 는 안테나 번호이다.
   `<readattenuation>` 는 태그를 읽을 때와 프로그래밍 중 태그를 읽고 식별할 때 사용하는 RF 감쇠 수단이다.
   `<writeattenuation>` 은 태그에 작성할 때 사용하는 RF 감쇠 수단이다.

   이 명령어는 프로그래밍 환경을 더 잘 제어할 수 있도록 '프로그래밍 중 전력 읽기'와 '프로그래밍 중 전력 쓰기'를 위해 RF 감쇠값을 설정한다. RF 감쇠값은 각 안테나 별로 지정한다.

예를 들어, 저전력에서 태그를 읽고 식별할 프로그램 존(program zone)을 작게 유지해야 할 때(RF 감쇠값이 큼), 일단 태그를 식별했으면 전출력을 사용해(RF 감쇠 없음) 태그에 기록한다. 이로써 쓰기 동작을 완수할 전력은 항상 충분해진다.

유의사항: RFAttenuation 명령어의 이 옵션은 ALR-9680과 ALR-9650에서는 사용할 수 없다.

`<attenuation>` 매개변수 값은 다음과 같다:

- 허용된 값: 정수(0..150) (지방에 따라 달라질 수 있음)
- 기본값: 0 (ALR-9680 & 9650의 경우는 10)
- maxAttenuation은 리더의 무선 수신기를 보정한 방식에 따라 달라지는데, 보통은 150(15 dB에 해당)이다.
- RFAttenuation이 10 오르면 RF 전력은 1 dBm 감소한다.
- RFAttenuation 값은 선택적 안테나 사용 명령어를 전송하지 않는 한, 모든 안테나에 적용된다. 다음은 몇 가지 예이다.

RFAttenuation 예제

Command Response	>RFAttenuation? RFAttenuation = 0
Command Response	// To reduce power by 3 dB: >RFAttenuation = 30 RFAttenuation = 30
Command Response	// Set attenuation for only antenna 0 >RFAttenuation = 0 30 RFAttenuation = 30
Command Response	// Compare attenuation of ant 0 with ant 1 >RFAttenuation 0? RFAttenuation = 30
Command Response	>RFAttenuation 1? RFAttenuation = 0
Command Response	// get RFAttenuation (without antenna) always gives ant 0 value >RFAttenuation? RFAttenuation = 30

Command
Response
	// read/singulate tags at low power (attenuation of 100 or 10 dB)
	// and program at high power (0 dB attenuation)
	>RFAttenuation = 0 100 0
	RFAttenuation = 0 100 0

RF 감도 RFLevel

F800 | 9900 | 9680 | 9650

RFAttenuation을 설정하는 대신(전출력에서 전력을 감소시키는 것), 리더의 실제 전력 출력량을 지정하고 검색할 수도 있다. 이 방법은 두 가지 명령어로 동일한 무선수신 하드웨어를 제어해야 할 때에만 편리하다.

RFLevel과 RFAttenuation은 언제나 다음 공식이 관계된다. MaxPower = RFLevel + RFAttenuation 이 명령어는 다음 조건들에서 사용이 가능하다.

1. `RFlevel = <power>` 이 명령어는 모든 안테나에 같은 전력 수준을 적용한다.

2. `RFlevel = <antenna> <power>`
   이 명령어는 각 안테나에 개별적으로 전력 수준을 설정한다.

3. `RFlevel = <antenna> <readpower> <writepower>`

   `<antenna>` 는 안테나 번호이다.
   `<readpower>` 는 태그를 읽을 때와 프로그래밍 중 태그를 읽고 식별할 때 사용하는 RF 전력 수준을 가리킨다.
   `<writepower>` 는 태그에 작성할 때 사용하는 RF 전력 수준을 가리킨다.

   이 명령어는 프로그래밍 환경을 더 잘 제어할 수 있도록 '프로그래밍 중 전력 읽기'와 '프로그래밍 중 전력 쓰기'를 위해 전력 수준을 설정한다. 전력 설정값은 각 안테나 별로 지정한다.

   예를 들어, 저전력에서 태그를 읽고 식별할 프로그램 존(program zone)을 작게 유지해야 할 때, 일단 태그를 식별했으면 전출력을 사용해 태그에 기록한다. 이로써 쓰기 동작을 완수할 전력은 항상 충분해진다.

   유의사항: RFLevel 명령어의 이 옵션은 ALR-9680과 ALR-9650에서는 사용할 수 없다.

`<power>` 매개변수 값은 다음과 같다:

- 허용된 값: Integer (<minPower>..<maxPower>) (지방에 따라 다를 수 있음)
- 기본값: <maxPower>
- RFLevel의 최소값은 MaxPower-MaxAttenuation이다.
- RFLevel이 10씩 증가할 때 RF 전력은 1 dB씩 증가하며, RFAttenuation에서도 10씩 그만큼 감소하는 것을 볼 수 있다.
- RFLevel 값은 선택적 안테나 사용 명령어를 전송하지 않는 한, 모든 안테나에 적용된다. 다음은 몇 가지 예이다.

RFLevel 예제

Command Response	>RFLevel? RFLevel = 316
 Command Response	// To reduce power by 3 dB: >RFLevel = 286 RFLevel = 286

Command
Response
 	
Command
Response
 	
Command
Response
 	
Command
Response
 	
Command
Response
	// Set RFLevel of only antenna 0
	>RFLevel = 0 250
	RFLevel = 250
	
	// Compare RFLevel of ant 0 with ant 1
	>RFLevel 0?
	RFLevel = 250
	
	>RFLevel 1?
	RFLevel = 316
	
	// Get RFLevel (without arguments) always gives ant 0 value
	>RFLevel?
	RFLevel = 250
	
	// read/singulate tags at low power (200) and program at high power (300)
	>RFLlevel = 0 200 300
	RFLlevel = 0 200 300

RF 변조 RFModulation

F800 | 9900 | 9680 | 9650

RFModulation 명령어를 사용하면 Gen2 프로토콜로 사용 가능한 변조 모드 중에서 선택할 수 있다. 각 모드 별로 사전 정의된 Gen2 공중 프로토콜들을 선택할 수 있으며, 특정 환경에 가장 적합한 내부 인벤토리 알고리즘도 선택할 수 있다.

- 허용된 값: "STD" | "HS" | "DRM" | "25FM0" | "06FM0" | "25M4" | "12M4" | "06M4" (지역에 따라 다름)
- 기본값: "DRM" (지역에 따라 다름)

이 모드 중 세 가지는 다른 정의된 모드들에서 사용하기에 편리한 이름이다.

- DRM – Dense Reader Mode (25M4와 같음)
- STD – Standard (25FM0와 같음)
- HS – High Speed (06FM0과 같음)
- 25FM0 – 25 µsec Tari, FM0
- 06FM0 – 06 µsec Tari, FM0
- 25M4 – 25 µsec Tari, Miller4
- 12M4 – 12 µsec Tari, Miller4
- 06M4 – 06 µsec Tari, Miller4

어떤 리더 모델이나 이 모델들의 국가별 버전은 규제 상의 이유로 모든 변조 모드를 지원하지 않을 수도 있다. 또한 ALR-9650/9680 리더는 DRM(Dense Reader Mode)만 지원한다.

값	설명
STD	표준 작동 모드 이 모드는 일반적인 용도에 적합하다. 속도와 소음/간섭에서 보호받도록 적절히 조합되어 있다. 이 모드는 "EPCglobal Dense Reader Mode" 스펙트럼 마스크 요건을 충족한다.
HS	고속 작동 모드 태그 수가 적고 리더 수도 적은 환경에서 더 나은 성능을 확보하기 위해 리더 태그 데이터 전송률을 높이는 조건
DRM	고밀도 리더 배치 모드(기본값). "소음이 많은" 환경에서 더 나은 성능 확보를 위해 필터링을 개선함 다수의 리더를 같은 지역에서 작동시킬 때 권장함 이 모드는 "EPCglobal Dense Reader Mode" 스펙트럼 마스크 요건을 충족한다.
[Tari][Mod]	나머지 모드는 모두 다음의 표준 명명 규칙을 따른다:Tari (in µsec), 변조 방식은 FM0, M4 (Miller 4) 등.

RFModulation 예제	
Command Response	>RFModulation? RFModulation = DRM
Command Response	>RFModulation = HS RFModulation = HS

공장초기화 FactorySettings

F800 | 9900 | 9680 | 9650

FactorySettings 명령어는 모든 리더 설정값을 공장 기본값으로 재설정한다. 이후 ALR-x780 리더는 스스로 재부팅한다. 새 설정값(특히 네트워크 값)을 적용하려면 재부팅해야 한다.

FactorySettings 예제	
Command	>FactorySettings
Response	All settings reset to factory defaults.

재시동 Reboot

F800 | 9900 | 9680 | 9650

리부트(Reboot) 명령어는 리더가 즉시 모든 네트워크 접속을 중단하고 재부팅하게 한다.

- 네트워크 구성 매개변수가 변경되어 리더를 재부팅해야 할 때 권장한다.
- 리더는 웹 인터페이스를 통해서도 재부팅할 수 있다. 웹 브라우저에서 리더의 IP 주소를 검색하고, 'Manage Reader(리더 관리)' 링크를 클릭한 다음 제공한 지침을 따른다.

Reboot 예제	
Command	>Reboot
Response	Rebooting System...

서비스 Service

F800 | 9900 | 9680 | 9650

Service 명령어를 사용하면 직렬 인터페이스, SNMP 등의 보조 리더 서비스를 제어할 수 있다. 'Service' 뒤에 제어하기 원하는 서비스 명칭을 쓰고 그 다음에 명령어를 쓴다. "all" 키워드를 서비스 명칭 대신 입력하면 설치된 모든 서비스를 제어한다.

```
Service <servicename> <command>
```

현재 서비스 목록은 다음과 같다:

서비스	설명
Serial	리더의 직렬 인터페이스를 제어한다. 직렬 인터페이스를 끄면, 직렬 포트가 Alien 용으로만 사용할 리더의 리눅스 운영체제 프롬프트에 연결된다. ALR-F800 리더에는 직렬 포트가 2 개씩 있으며, 각각 다음과 같은 서비스 명칭을 갖는다. “serial0” (RS-232)와 “serial1” (USB).
Snmp	리더가 간이 망 관리 프로토콜(Simple Network Management Protocol) v1, MIB2를 실행한다. 이 서비스를 사용하지 않는다면 꺼서 시스템 자원을 확보한다.
Heartbeat	HeartbeatCount=0으로 설정하면 리더의 하트비트 서비스를 완전히 끄거나 비활성화할 수 있으며, 이때 다른 하위 단계의 비활성화 방법보다 우선한다. 하트비트(Heartbeat) 서비스를 중단시키고 비활성화하면 펌웨어 업그레이드 후에도 다시 시작하지 않는다.
Ifmon	이더넷 인터페이스를 모니터링하는(5분 간격으로) 리더 서비스는 다음과 같다. DHCP가 ON이고 이더넷이 DOWN이거나 DHCP 임대기간이 만료되었다면, DHCP 임대를 갱신한다(또는 DHCP 갱신이 안 될 경우 정적 IP를 설정한다). DHCP가 OFF이고 이더넷이 DOWN이면, 정적 IP로 설정한다. 정적 IP를 설정하면 서비스가 192.168.1.100/255.255.255.0 또는 전에 마지막으로 사용했던 정적 IP/네트워크값을 사용한다.

<u>유의사항:</u> snmp, heartbeat, ifmon 서비스는 이런 기능이 리더 OS 시스템 서비스에 포함되어 있는 ALR-F800에서는 표시되지 않는다.

이용 가능한 서비스 명령어는 다음과 같다:

명령어	설명
Status	선택한 서비스의 현재 상태를 보고하거나 서비스 명칭 대신 "all"을 선택했다면 설치된 모든 서비스의 상태를 나열한다. 각 행마다 서비스 명칭을 열거하며, 현재 실행 중인지(R = 실행 중, s = 중지됨), 자동시작이 가능한지(A = 가능, d = 불가능), 서비스 시동 순서를 표시하는 번호 등이 제시된다.
Start **stop**	서비스를 시작 또는 정지한다. 이로써 서비스의 상태를 즉각적으로 제어할 수 있다. 자동시작이 가능하면, 다음에 리더를 재부팅할 때 서비스를 다시 시작한다. 리더는 해당 서비스에 대하여 새 상태 표시 행으로 응답한다.
Enable **disable**	서비스의 자동시작 기능을 활성화 또는 비활성화한다. 자동시작이 활성화되면, 서비스는 리더가 부팅할 때 자동으로 시동한다. 자동시작 설정을 변경해도 서비스를 즉시 시작하거나 중지하는 것은 아니다. 이를 위해서는 시작/정지 명령어를 사용해야 한다. 리더는 해당 서비스에 대하여 새 상태 표시 행으로 응답한다.

“시작” 및 “정지” 서비스 명령어는 즉시 작동하나, 리더가 재부팅할 때 어떤 서비스를 자동으로 시작하게 할 것인지는 변경하지 못한다. “활성화” 및 “비활성화” 서비스 명령어는 차후 리부팅할 때 리더가 시작하는 방식에 영향을 미치나, 어떤 서비스를 즉시 실행할지는 변경하지 못한다.

개별 서비스의 시작/정지와 활성화/비활성화 설정값은 다른 리더 속성들처럼 "저장"이 필요하지 않다.

ALR-F800 리더에서는 두 개의 직렬 포트 중 어떤 것이든 “해제” 할 수 있으며, 리더는 그 사용을 완전히 중지시킨다. 이로써 직렬 포트를 해제하고 다른 사용을 위해 확보해 둘 수 있다(예를 들면, 온보드 Ruby 애플리케이션). “해제(release)” 하위 명령어를 사용해 다음과 같이 작업할 수 있다. “service serial0 release”는 RS-232 포트를 해제한다. 이 포트를 비활성화하여 리더가 다음 부팅 시에 다시 사용하려고 시도하지 않게 해야 한다(예:

"service serial0 disable"). 이는 해제된 직렬포트가 다음에 직렬 인터페이스를 시작할 때 자동으로 "해제 취소"되지 않게 하기 위함이다.

그에 더하여, 어떤 직렬포트를 ALR-F800 리더가 이 부팅/콘솔 출력에 사용할 것인지를 지정하기 위해 "service serial bootlog serial0" (RS-232 부트 로그), "service serial bootlog serial1" (USB 상의 부트 로그), 또는 조용한 부팅을 위해서는 "service serial bootlog off" 로 설정할 수 있다.

Service 예제	
Command Response	>service serial status R A 05 serial // currently Running, Autostart enabled, startup priority 5
Command Response	>service all status R A 05 serial R A 80 snmp
Command Response	>service serial stop s A 05 serial // the serial service is stopped, but it starts up again on the next boot
Command Response	>service serial disable s d 05 serial // service is disabled, it will not be started on the next boot

Command
Response
 	
Command
Response
 	
Command
Response
Command
Response
	// ALR-F800 examples...
	
	// report the status of both serial ports
	>service serial status
	R A 30 serial0
	R A 30 serial1
	
	// report the status of just serial0 (RS-232)
	>service serial0 status
	R A 30 serial0
	
	// release RS-232 for personal use
	>service serial0 release
	service serial0 RELEASED
	>service serial0 disable
	s d 30 serial0
	
	// turn off the serial bootlog
	>service serial bootlog off

내 데이터 MyData

F800 | 9900 | 9680 | 9650

MyData 명령어를 사용하면 254 문자 스크래치패드를 사용해 원하는 데이터를 저장할 수 있다. 일례로, GPS 좌표 상에 리더의 위치를 가리키거나 리더의 권한설정 상태 또는 히스토리를 표시한다.

- 'Save' 명령어를 사용해 사용자의 MyData 변경을 영구히 적용할 수도 있다.
- MyData 내용은 다른 리더 속성들과 마찬가지로 펌웨어 업그레이드 후에도 계속 유지된다.

MyData 예제	
Command	>MyData?
Response	MyData = (Not Set)
Command	>MyData = It's warm and sunny here: Lat: 37.14276, Long: -121.65615
Response	MyData = It's warm and sunny here: Lat: 37.14276, Long: -121.65615

ETSIMode

F800 | **9900+** | 9900 | 9680 | 9650

EMA 리더가 작동하는 기본 ETSI 모드는 EN 302 208 v1.3.1이다. 유럽/중동/아시아에 맞춰 구성된 리더는 ETSIMode 명령어를 사용하여 EN 300 220 표준 요건을 충족할 수 있다.

- 허용된 값: “302.208v1.3.1” 또는 “300.220”
- 기본값: “302.208v1.3.1”

리더는 ETSIMode 변경사항이 온전히 적용될 수 있도록 리부팅해야 한다. ETSIMode가 변경된 후에는 ReaderType이 새로운 규제 표준을 반영한다.

ETSIMode 예제	
Command	>ETSIMode?
Response	ETSIMode = 302.208
Command	>ReaderType?
Response	ReaderType = Alien RFID Tag Reader, Model ALR-9900+ (Four Antenna / Gen 2 / EN 302 208)
Command	>ETSIMode = 300.220
Response	ETSIMode = 300.220
Command	>ReaderType?
Response	ReaderType = Alien RFID Tag Reader, Model ALR-9900+ (Four Antenna / Gen 2 / EN 300 220)

네트워크 구성 명령어

이러한 명령어는 사용자가 리더와 네트워크 간 통신에 관계된 설정값을 구성하고 검색할 수 있게 해준다.

MACAddress

F800 | 9900 | 9680 | 9650

MAC(Media Access Control) 주소는 네트워크 인터페이스가 있는 각 장치를 식별하기 위한 고유의 하드코딩된 값이다. 리더의 MAC 주소는 MACAddress 명령어로 검색할 수 있다.

반환된 값은 16진수 6파이트의 시퀀스로서, 각각 콜론으로 분리한다.

MACAddress 예제	
Command	>MACAddress?
Response	MACAddress = 00:90:c2:c3:14:38

DHCP

F800 | 9900 | 9680 | 9650

DHCP6

F800 | 9900 | 9680 | 9650

리더는 널리 사용되는 DHCP 프로토콜을 사용하여 자동 네트워크 구성을 지원한다. DHCP가 리더 설치 장소에서 이용 가능할 경우, 이 프로토콜을 켤 수 있다. DHCP가 이용 가능하지 않거나 원하지 않을 경우에는 이 프로토콜 사용을 꺼 놓아야 한다.

- 허용된 값: "ON" | "OFF"
- 기본값: "ON"
- 이 명령어로 변경한 후에는 `Save` 명령어를 전송하고 리더를 재부팅하여 변경사항을 적용시켜야 한다.
- DHCP가 켜진 상태에서 리더가 부팅될 때 이용 가능한 서버가 없는 경우에는 기본 네트워크 주소(192.169.1.100)를 사용할 것이며, 정기적으로 DHCP 서버와 통신을 시도할 것이다.
- ALR-F800 리더 또한 IPV6 네트워킹을 지원하며, DHCP6 명령어로 DHCP의 IPV6 버전을 제어한다.

DHCP 예제	
Command	>DHCP?
Response	DHCP = ON
Command	>DHCP = OFF
Response	DHCP = OFF

DHCPTimeout

F800 | 9900 | 9680 | 9650

리더를 설정하여 DHCP 서버로부터 자동으로 네트워크 구성 설정을 가져올 경우, 일정 시간이 지난 후에 DHCP 서버로부터 응답을 받기 위해 대기하기를 중단하고 대신 기본 (정적) 네트워크 구성값을 사용한다

(IP: 192.168.1.100, 넷마스크: 255.255.255.0 등). DHCP 서버 응답을 대기하는 시간은 DHCPTimeout 속성에서 명시된다.

일반적으로, 기본값인 90초가 대부분의 상황에서 적합하지만, 대기시간이 긴 네트워크에서나 리더가 LAN 드롭에 직접 연결되어 있을 경우(라우터나 네트워크 스위치가 아님), 시간이 더 긴 것이 좋다. 리더가 부팅할 때 DHCP 서버에서 네트워크 구성 설정값을 얻지 못할 경우, 사용자는 DHCPTimeout을 120초 이상으로 늘릴 수 있다.

- 허용된 값: 정수 (0..1000)
- 기본값: 90
- 이 명령어로 변경한 후에는 `Save` 명령어를 전송하고 리더를 재부팅하여 변경사항을 적용시켜야 한다.

DHCPTimeout 예제	
Command Response	>DHCPTimeout? DHCPTimeout = 90
Command Response	>DHCPTimeout = 120 DHCPTimeout = 120

IPAddress
F800 | 9900 | 9680 | 9650
IPAddress6
F800 | 9900 | 9680 | 9650
DHCP를 자동 구성에 사용하지 않는 경우, 리더는 네트워크 상에서 이용할 수 있도록 수동으로 구성해야 한다. IPAddress 명령어를 사용하면 호스트의 IP 주소를 할당 및 검색할 수 있다.

- DHCP는 IPAddress를 변경할 때에는 꺼두어야 한다.
- 이 명령어로 변경한 후에는 Save 명령어를 전송하고 리더를 재부팅하여 변경사항을 적용시켜야 한다.
- ALR-F800 리더 또한 IPV6 네트워킹을 지원하며, IPAddress6 명령어로 IPAddress의 IPV6 버전을 제어한다.

IPAddress 예제	
Command Response	>IPAddress? IPAddress = 12.34.56.78
Command Response	>IPAddress =34.55.33.12 IPAddress = 34.55.33.12

Gateway
F800 | 9900 | 9680 | 9650
Gateway6
F800 | 9900 | 9680 | 9650
DHCP를 자동 구성에 사용하지 않는 경우, 리더는 네트워크 상에서 이용할 수 있도록 수동으로 구성해야 한다. 게이트웨이 명령어는 네트워크 게이트웨이를 할당 및 검색할 때 사용한다.

- 게이트웨이는 수치로 된 IP 주소로 지정해야 한다.

- DHCP는 게이트웨이를 변경할 때에는 꺼두어야 한다.
- 이 명령어로 변경한 후에는 Save 명령어를 전송하고 리더를 재부팅하여 변경사항을 적용시켜야 한다.
- 게이트웨이 주소가 네트워크 상에서 유효한 주소를 가리키지 않을 경우, 리더는 아웃바운드 연결을 하지 못할 수도 있다(예: TagStream, Heartbeats, Notification 메시지).
- ALR-F800 리더는 또한 IPV6 네트워킹을 지원하며, Gateway6 명령어로 게이트웨이 IPV6 버전을 제어한다.

Gateway 예제	
Command Response	>Gateway? Gateway = 34.56.78.90
Command Response	>Gateway=12.56.23.01 Gateway = 12.56.23.01

Netmask

F800 | 9900 | 9680 | 9650

Netmask6

F800 | 9900 | 9680 | 9650

DHCP를 자동 구성에 사용하지 않는 경우, 리더는 네트워크 상에서 이용할 수 있도록 수동으로 구성해야 한다. 서브넷 마스크 명령어는 서브넷 마스크를 할당 및 검색할 때 사용한다.

- 서브넷 마스크는 수치로 된 IP 주소로 지정해야 한다.
- DHCP는 넷마스크를 변경할 때에는 꺼두어야 한다.
- 이 명령어로 변경한 후에는 Save 명령어를 전송하고 리더를 재부팅하여 변경사항을 적용시켜야 한다.
- ALR-F800 리더는 또한 IPV6 네트워킹을 지원하며, Netmask6 명령어로 넷마스크 IPV6 버전을 제어한다.

Netmask 예제	
Command Response	>Netmask? Netmask = 255.255.255.128
Command Response	>Netmask=255.255.255.0 Netmask = 255.255.255.0

DNS

F800 | 9900 | 9680 | 9650

DHCP를 자동 구성에 사용하지 않는 경우, 리더는 네트워크 상에서 이용할 수 있도록 수동으로 구성해야 한다. DNS 명령어는 DNS 서버 위치를 할당 및 검색할 때 사용한다.

- DNS 서버는 수치로 된 IP 주소로 지정해야 한다.
- DHCP는 DNS를 변경할 때에는 꺼두어야 한다.
- 이 명령어로 변경한 후에는 Save 명령어를 전송하고 리더를 재부팅하여 변경사항을 적용시켜야 한다.

DNS 예제	
Command Response	>DNS? DNS = 12.34.56.78
Command Response	>DNS=45.224.124.34 DNS = 45.224.124.34

Hostname

F800 | 9900 | 9680 | 9650

호스트명(Hostname) 명령어로 네트워크 상에서 리더의 ID를 구성할 수 있다. 리더의 호스트명이 로컬 DNS에 알려져 있을 경우, 리더는 IP 주소를 알지 못해도 호스트명으로 네트워크에 접속할 수가 있다.

일반적으로 DHCP 서버는 네트워크 장치에 호스트명을 할당할 수 있으며, 이 경우 리더는 DHCP가 제공한 호스트명은 무엇이든 사용한다. 사용자가 Hostname 명령어로 다른 호스트명을 지정했더라도 마찬가지이다.

- 허용된 값: String[1-16]
- 기본값: "alien-xxyyzz"에서 xx, yy, zz는 리더의 고유한 MAC 주소에서 마지막 3개 바이트에 해당한다.
- 호스트명을 변경하면 즉각 효력이 나타난다.

Hostname 예제	
Command Response	>Hostname? Hostname = alien-10116A
Command Response	>Hostname = reader1 Hostname = reader1

NetworkUpgrade

F800 | 9900 | 9680 | 9650

Alien 리더는 부팅 시 스스로 자동 업그레이드 되도록 구성할 수 있다. UpgradeAddress를 구성한 다음 NetworkUpgrade 플래그를 ON으로 설정하여 이 기능을 활성화시키고 그 다음 리더를 재부팅하면 된다. 또는 UpgradeNow 명령어를 전송하여 즉시 업그레이드 체크를 할 수도 있다.

리더가 부팅되면, NetworkUpgrade 플래그를 확인하여 업그레이드를 찾아야 하는지 결정한다. 찾아야 할 경우, FTP, HTTP, 또는 HTTPS 서버로부터 업그레이드 파일을 다운 받고자 할 것이다. 그런 다음 업그레이드를 설치하고, 부팅 과정을 계속 진행한다.

여러 업그레이드 수단에 대한 세부정보는 이 문서 끝에 있는 부록 C에서 찾을 수 있다.

- 허용된 값: “ON” | “OFF”
- 기본값: “OFF”
- NetworkUpgrade를 “On” 으로 설정하고, “Save” 명령어를 전송한 다음 리더를 재부팅하여 네트워크 업그레이드를 시작한다.

NetworkUpgrade 예제	
Command Response	>NetworkUpgrade? NetworkUpgrade = Off
Command Response	>NetworkUpgrade = On NetworkUpgrade = On

UpgradeAddress

F800 | 9900 | 9680 | 9650

UpgradeAddress 명령어를 사용하여 리더가 업그레이드를 찾을 네트워크 호스트를 지정할 수 있다. 지정된 값은 HTTP, HTTPS, 또는 FTP에 대한 전체 URL이며, 업그레이드 파일이 위치할 디렉터리 경로도 포함한다. 리더는 지정한 경로에서 "제어" 파일을 다운로드하려 할 것이며, 이 경로에는 다운로드할 업그레이드 파일 이름이 포함되어 있다. 이 과정의 세부정보 일체를 부록 C에서 찾을 수 있다.

- UpgradeAddress의 기본값은 0.0.0.0이다.
- 리더는 이전 리더 버전과의 호환성을 위해 “UpgradeIPAddress” 명령어를 승인한다.
- UpgradeAddress 명령어에 변경이 있으면 다음 재부팅 때 즉각 효력이 발생한다("Save" 명령어를 전송한 다음).

UpgradeAddress 예제	
Command Response	>UpgradeAddress? UpgradeAddress = 0.0.0.0
Command Response	>UpgradeAddress = http://192.168.1.200/alienUpgrades/ UpgradeAddress = http://192.168.1.200/alienUpgrades/

UpgradeNow

F800 | 9900 | 9680 | 9650

UpgradeNow 명령어를 사용하면 리더를 재부팅할 필요 없이 펌웨어 업그레이드를 시작한다. 네트워크 업그레이드 기능에 관한 더 많은 정보는 부록 C의 NetworkUpgrade, UpgradeAddress (위), “Pulling Upgrades” 섹션을 참조하기 바란다.

UpgradeNow를 사용하는 가장 손쉬운 방법은 펌웨어와 제어 파일이 있는 URL로 UpgradeAddress를 미리 설정해 둔 다음 UpgradeNow 명령어를 전송하면 된다. 또는 UpgradeNow 명령어에 대한 인수로 URL을 포함시킬 수도 있다.

```
UpgradeNow
```

또는

```
UpgradeNow http://192.168.100.150/firmware/
```

또는 “목록(list)” 인수를 명령어에 포함시켜서 리더가 UpgradeAddress(또는 제공한 URL)에 무엇이 있는지 확인하고 찾은 파일을 나열하게 할 수도 있다.

```
UpgradeNow list
```

또는

```
UpgradeNow list http://192.168.100.150/firmware/
```

마지막으로, “force” 인수는 리더가 어떤 펌웨어이든 버전에 관계 없이 다운로드하고 설치하게 한다.

```
UpgradeNow force
```

또는

```
UpgradeNow force http://192.168.100.150/firmware/
```

- "`list`" 와 "`force`" 인수는 함께 사용할 수 없다.
- 일단 펌웨어 업그레이드가 시작되었으면, 리더 명령어를 전송하지 말아야 하며, 리더가 처리 과정 중에 정전이 되지 않게 해야 한다.
- 펌웨어 업그레이드 중에 모든 리더 서비스와 태그 판독 작업이 중단된다. 네트워크 연결이 중단되고 리더의 하트비트도 업그레이드 완료 시점까지 중단된다. **"경고: 전원을 뽑지 마십시오!" 메시지를 보면, 리더가 다시 반응하여 TCP 연결이 수락되고, 직렬 라인에 Alien> 프롬프트가 표시되고, 리더 하트비트(Heartbeat)가 재개될 때까지 기다려야 한다. 전원 플러그를 절대로 뽑아서는 안 된다! 이 시점에서 전력이 차단되면 펌웨어가 손상되어 리더를 공장으로 반품해야 할 수도 있기 때문이다.**

UpgradeNow 예제	
Command Response Command Response	// UpgradeNow using a preset UpgradeAddress (same firmware version) >UpgradeAddress = http://192.168.1.123/fw/alien UpgradeAddress = http://192.168.1.123/fw/alien >UpgradeNow ##### Running Network Upgrade 04/14/08 13:29:45 GMT+8A Upgrade URL : http://192.168.1.123/fw/alien Reader Version : 08041100 Package Downgrading : enabled List of Packages : ===== ALRx800_fw_080411.tar.aef ===== control line : ALRx800_fw_080411.tar.aef package version : 08041100 ...skipping ALRx800_fw_080411.tar.aef (same version) ##### No upgrades performed

Command
Response
	// UpgradeNow, with explicit address (new firmware version)
	>UpgradeNow http://192.168.1.123/fw/alien
	##### Running Network Upgrade 08/13/08 10:44:55 GMT+7A
	Upgrade URL : http://192.168.1.123/fw/alien
	Reader Version : 08081200
	Package Downgrading : enabled
	List of Packages :
	=====
	ALRx800_fw_080813_BETA.tar.aef
	=====
	control line : ALRx800_fw_080813_BETA.tar.aef
	package version : 08081300
	...downloading ALRx800_fw_080813_BETA.tar.aef
	...decrypting ALRx800_fw_080813_BETA.tar.aef
	...unpacking ALRx800_fw_080813_BETA.tar
	...installing ALRx800_fw_080813_BETA
	WARNING: DO NOT UNPLUG THE POWER!
	
	// The following output is visible only on the serial port:
	Alien>
	Alien Technology Corporation RFID Reader
	[ALIEN DRV] ALIEN READER driver ver 1.54.080111 UNLOADED
	[ALIEN DRV] ALIEN READER driver ver 1.54.080111 LOADED
	
	HOST activated (build: Aug 13 2008 02:10:19)
	
	Boot> Ready

UpgradeNow List & Force 예제	
Command Response	// UpgradeNow, listing the files >UpgradeNow list ALRx800_fw_080813_BETA.tar.aef

Command
Response
	// UpgradeNow with the force option (reload same firmware version)
	>UpgradeNow force
	##### Running Network Upgrade 08/13/08 10:52:52 GMT+7A
	Upgrade URL : http://10.10.82.10/dailybuild/
	Reader Version : 08081300
	Package Downgrading : enabled
	List of Packages :
	==========
	ALRx800_fw_080813_BETA.tar.aef
	==========
	control line : ALRx800_fw_080813_BETA.tar.aef
	package version : 08081300
	...downloading ALRx800_fw_080813_BETA.tar.aef
	...decrypting ALRx800_fw_080813_BETA.tar.aef
	...unpacking ALRx800_fw_080813_BETA.tar
	...installing ALRx800_fw_080813_BETA
	WARNING: DO NOT UNPLUG THE POWER!
	
	// The following output is visible only on the serial port:
	Alien>
	Alien Technology Corporation RFID Reader
	[ALIEN DRV] ALIEN READER driver ver 1.54.080111 UNLOADED
	[ALIEN DRV] ALIEN READER driver ver 1.54.080111 LOADED
	
	HOST activated (build: Aug 13 2008 02:10:19)
	
	Boot> Ready

NetworkTimeout

F800 | 9900 | 9680 | 9650

리더가 명령어 포트로 명령어를 받으면 TCP 소켓을 열고 데이터가 도착할 때까지 기다린다. 인바운드 통신이 중단되면, 리더는 소켓을 무한정 열어두는 대신 일정 시간을 대기한 뒤에 연결을 자동으로 닫는다. 또한 이미 수신되었을 수 있는 부분적인 명령어는 무시한다. 이 시간을 NetworkTimeout라 한다.

- 허용된 값: Integer (10..65535)
- 기본값: 90초

NetworkTimeout Examples	
Command Response	>NetworkTimeout? NetworkTimeout = 90
Command Response	>NetworkTimeout = 120 NetworkTimeout = 120

	// Example Behavior (using Telnet)
	>NetworkTimeout = 15
	NetworkTimeout = 15
	
	// Wait more than 15 seconds
	Connection Timeout.
	Closing Connection…Bye!
	
	// Telnet session ends

CommandPort

F800 | 9900 | 9680 | 9650

리더는 표준형 네트워크 소켓을 사용하여 네트워크 상에서 구성 및 작동시킬 수 있다. CommandPort 설정값을 사용하여 이 네트워크 연결을 위해 리더가 사용하는 정확한 포트 번호를 할당하고 검색한다.

- 허용된 값: Integer(1..65535)
- 기본값: 23 (표준형 텔넷 포트)
- 이 설정을 변경해도 직렬 통신 및/또는 리더와의 웹 통신에는 영향을 미치지 않는다.
- 이 설정을 변경해도 리더가 재부팅할 때까지는 영향을 미치지 않는다.

CommandPort 예제	
Command Response	>CommandPort? CommandPort = 23
Command Response	>CommandPort = 10004 CommandPort = 10004

CommandPortLocal

F800 | 9900 | 9680 | 9650

`CommandPortLocal` 명령어를 사용하여 리더 자체에서(localhost) 시작된 TCP 연결의 포트 번호를 할당할 수 있다. 이 포트의 기본값은 2300이다. 이 기능은 온-리더 애플리케이션을 지원하기 위함이다(Ruby). 이는 포트 23에서 표준 명령어 채널에 더하여 독립적인 명령어 채널이다(`CommandPort` 명령어로 할당함). 포트 2300을 사용하면 표준 포트 23은 리더와의 외부 연결을 위해 확보해 둘 수 있으며, 이로써 리더는 온-리더 애플리케이션과 포트 23에서의 명령어 채널을 통해 동시에 제어할 수 있다. 이 방식은 권장할 만하며 사용자 애플리케이션의 로직의 영향을 받지 않으므로, 어떤 포트가 각자의 경우에 맞는지 직접 선택할 수 있다. 포트 23에서 외부 명령어 채널을 비활성화하려면 `AcceptConnections=local` 로 설정한다.

- 허용된 값: Integer(1..65535)
- 기본값: 2300
- 이 설정을 변경해도 직렬 통신 및/또는 리더와의 웹 통신에는 영향을 미치지 않는다.
- 이 설정을 변경해도 리더가 재부팅할 때까지는 영향을 미치지 않는다.

CommandPortLocal 예제	
Command Response	>CommandPortLocal? CommandPort = 2300
Command Response	>CommandPortLocal = 2345 CommandPortLocal = 2345

AcceptConnections

F800 | 9900 | 9680 | 9650

(localhost)에서 시작된 TCP 연결을 수락할 수 있다. 명령어 `AcceptConnections = {ANY | LOCAL}` 로 이 동작을 제어한다.

`AcceptConnections = LOCAL` 로 설정하면 리더 밖에서 시작된 TCP 연결이 수락되지 않게 차단할 수 있다. 리더 내부에서 시작된 연결만(이를테면, 리더에서 실행되는 사용자 지정 Ruby 애플리케이션에서 시작된 연결) 수락될 것이다.

변경은 저장해야 하며, 리더는 새 동작이 발효되기 전에 재부팅해야 한다.

- 허용된 값: ANY | LOCAL
- 기본값: ANY
- 이 설정을 변경해도 직렬 통신 및/또는 리더와의 웹 통신에는 영향을 미치지 않는다.
- 이 설정을 변경해도 리더가 재부팅할 때까지는 영향을 미치지 않는다.

AcceptConnections 예제	
Command Response	>AcceptConnections? AcceptConnections = ANY
Command Response	> AcceptConnections = local AcceptConnections = local

Ping

F800 | 9900 | 9680 | 9650

ping 명령어는 리더와 지정된 네트워크 주소 간에 네트워크 연결성을 테스트할 때 사용한다. ping 명령은 일반적으로 대상 장치의 IP 주소만(또는 호스트명) 알면 된다. 리더가 ICMP "ECHO_REQUEST" 명령어를 대상으로 보내면, 대상은 요청을 성공적으로 받는 즉시 리더에 응답한다. ping 명령은 연결이 성공적인지를 테스트하는데 더해서 트랜잭션의 왕복 시간도 표시해 주는데, 이는 네트워크 상에서 대기 시간을 확인하는데 유용하다.

Alien 리더는 또한 ping 명령어의 수정본도 제공하는데, 이로써 대상 주소에 더해 연결할 포트 번호도 지정한다. 이 경우 리더는 해당 포트 번호로 장치에 연결할 소켓을 열려고 할 것이다. 그런 다음 대상과 연결할 네트워크의 경로를 보고할 뿐 아니라 지정한 포트로 대상이 연결 대상을 청취할지도 보고한다. 따라서 사용자는 자신의 MailServer 또는 NotifyAddress 속성이 올바로 설정되었는지 확인하기 위해 테스트해 볼 수 있다.

Ping (IP Address) 예제	
Command Response	// Successful Ping: >ping 192.167.1.200 PING 192.167.1.200 (192.167.1.200): 56 data bytes 64 bytes from 192.167.1.200: icmp_seq=0 ttl=128 time=1.6 ms 64 bytes from 192.167.1.200: icmp_seq=1 ttl=128 time=1.2 ms 64 bytes from 192.167.1.200: icmp_seq=2 ttl=128 time=1.2 ms --- 192.167.1.200 ping statistics --- 3 packets transmitted, 3 packets received, 0% packet loss round-trip min/avg/max = 1.2/1.3/1.6 ms
Command Response	// Unsuccessful Ping: >ping 192.168.1.199 PING 192.168.1.199 (192.168.1.199): 56 data bytes --- 192.168.1.199 ping statistics --- 3 packets transmitted, 0 packets received, 100% packet loss

Ping (IP Address : Port) 예제	
Command Response	// Successful Ping: >ping 192.167.1.200:3600 Opening Socket at 192.168.1.200:3600 [192.168.1.200:3600] Remote Host Located Establishing Connection Established Connection Closing Connection Closed Connection Socket Successfully Located, Opened and Closed
Command Response	// Unsuccessful Ping: >ping 192.168.1.199:3600 Opening Socket at 192.168.1.199:3600 [192.168.1.199:3600] Remote Host Located Establishing Connection Error: Unable to Open Socket on Remote Host (Timeout)

HeartbeatPort

F800 | 9900 | 9680 | 9650

리더는 하트비트(Heartbeat) 메시지를 네트워크로 정기적으로 발송하도록 구성할 수 있다. 이 하트비트(Heartbeat)는 전체 서브넷 또는 특정 주소로 방송할 단일의 UDP(Universal Datagram Packet) 패킷 형태를 취한다.

HeartbeatPort 명령어를 사용하면 사용자는 이 패킷을 발송할 실제 포트 번호를 구성할 수 있다.

이 하트비트(Heartbeat)를 청취하면 네트워크 상에 있는 리더의 초기 위치를 파악할 수 있고 이후에는 해당 리더가 아직 연결되어 있는지도 확인 가능하다.

- 허용된 값: Integer(0..65535)
- 기본값: 3988

리더 하트비트(Heartbeat)에 대한 더 자세한 정보는 2장에서 참조하기 바란다.

HeartbeatPort 예제	
Command Response	>HeartbeatPort? HeartbeatPort = 3004
Command Response	>HeartbeatPort=10002 HeartbeatPort = 10002

HeartbeatTime

F800 | 9900 | 9680 | 9650

리더는 하트비트(Heartbeat) 메시지를 네트워크로 정기적으로 발송하도록 구성할 수 있다. 이 하트비트(Heartbeat)는 전체 서브넷 또는 특정 주소로 방송할 단일의 UDP(Universal Datagram Packet) 패킷 형태를 취한다.

하트비트(Heartbeat) 간 시간 간격은 이 명령어를 사용하여 지정 및 검색할 수 있다.

- 허용된 값: Integer(0..86400)
- 기본값: 30
- 모든 간격은 초 단위로 지정한다.
- 0(초)로 설정하면 추가 하트비트(Heartbeat) 출력이 정지된다.

HeartbeatTime 예제	
Command Response	>HeartbeatTime? HeartbeatTime = 30
Command Response	>HeartbeatTime=60 HeartbeatTime = 60

HeartbeatAddress

F800 | 9900 | 9680 | 9650

리더는 하트비트(Heartbeat) 메시지를 네트워크로 정기적으로 발송하도록 구성할 수 있다. 이 하트비트(Heartbeat)는 전체 서브넷 또는 특정 주소로 방송할 단일의 UDP(Universal Datagram Packet) 패킷 형태를 취한다.

이 패킷을 받을 호스트의 주소는 HeartbeatAddress 명령어로 정의한다.

- 기본값 255.255.255.255는 특별한 "멀티캐스트(multicast)" 주소로서, 서브넷 상의 모든 장치가 패킷을 받을 수 있게 한다.

HeartbeatAddress 예제	
Command Response	>HeartbeatAddress? HeartbeatAddress = 255.255.255.255
Command Response	>HeartbeatAddress =10.1.70.17 HeartbeatAddress = 10.1.70.17

HeartbeatCount

F800 | 9900 | 9680 | 9650

HeartbeatCount 속성은 부팅 후에 리더가 방송할 하트비트(Heartbeat) 메시지의 수를 지정한다. 이 시점 이후로는 리더가 HeartbeatCount 변경 전까지 또는 리더가 재부팅할 때까지 하트비트(Heartbeat) 메시지 발송을 중단한다.

- 허용된 값: Integer(-1..65535)
- 기본값: -1 (리더는 하트비트(Heartbeat) 메시지를 무제한 발송한다)

HeartbeatAddress 예제	
Command Response	>HeartbeatCount? HeartbeatCount = -1
Command Response	>HeartbeatCount = 5 HeartbeatCount = 5

HeartbeatNow

F800 | 9900 | 9680 | 9650

HeartbeatNow 명령어는 리더가 UDP 하트비트(Heartbeat)를 즉시 발송하게 하며, 이때 HeartbeatTime 또는 HeartbeatCount 값은 무시된다. 이 명령어는 방송된 전체 XML 형식 하트비트(Heartbeat) 패킷을 반환한다.

HeartbeatNow 예제	
Command Response	>HeartbeatNow <?xml version="1.0" encoding="UTF-8"?> <Alien-RFID-Reader-Heartbeat> <ReaderName>Bob's 9800</ReaderName> <ReaderType>Alien RFID Tag Reader, Model: ALR-9800 (Four Antenna / Multi-Protocol / 902-928 MHz)</ReaderType> <IPAddress>192.168.100.150</IPAddress> <CommandPort>23</CommandPort> <HeartbeatTime>3</HeartbeatTime> <MACAddress>00:80:66:10:11:6A</MACAddress> <ReaderVersion>08.06.02.00</ReaderVersion> </Alien-RFID-Reader-Heartbeat>

ReaderList

F800 | 9900 | 9680 | 9650

Alien 리더는 스스로를 알리기 위해 네트워크 상에서 정기적으로 USP "heartbeat" 패킷을 전송한다. 사용자는 소프트웨어 API를 사용하여 이 하트비트(Heartbeat) 패킷을 탐지할 자신의 애플리케이션을 작성할 수 있으며, ALR-F800 리더는 또한 다른 리더로부터의 하트비트(Heartbeat)를 청취하기도 한다. 사용자는 ReaderList 명령어로 네트워크 상에서 탐지한 다른 리더 목록을 언제든 요청할 수 있다.

- "get ReaderList" (또는 "readerlist?")는 검색한 리더들의 목록을 반환한다.
- "ReaderList=Off"를 사용하여 이 기능을 끌 수 있다(ReaderList=On은 이 기능을 다시 켠다).
- "Clear ReaderList"를 사용하여 리더 목록을 지운다.

ReaderList 출력물의 각 행은 탐지한 리더의 세부 정보를 제시하는데, 다음은 그 예이다.

- MAC 주소(e.g. 00:1B:5F:00:00:65)
- IP 주소 및 명령어 포트(e.g. 10.10.82.141:23)
- 펌웨어 버전(e.g. 15.09.03.00)
- 모델 번호(e.g. ALR-F800)
- 첫 번째와 마지막 하트비트(Heartbeat) 타임스탬프(e.g. 2015/09/11 19:51:41)
- 수신한 하트비트(Heartbeat)의 수와 빈도 (e.g. 408@30)
- ReaderName (e.g. Alien RFID Reader)

ReaderList 예제	
Command Response	>ReaderList? MAC:00:1B:5F:00:00:63, IP:10.10.82.141:23, FW:15.09.03.00, ↵ Model:ALR-F800, First:2015/09/11 19:51:41, Last:2015/09/11 ↵ 23:14:40, HB:408@30, Name:Alien RFID Reader MAC:00:1B:5F:00:01:60, IP:10.10.82.142:23, FW:14.07.01.00, ↵ Model:ALR-9900, First:2015/09/11 19:52:04, Last:2015/09/11 ↵ 23:14:34, HB:406@30, Name:RegTest-9900 MAC:00:1B:5F:00:00:04, IP:10.10.82.132:23, FW:15.09.10.00, ↵ Model:ALR-F800, First:2015/09/11 21:08:47, Last:2015/09/11 ↵ 23:14:20, HB:253@30, Name:Alien RFID Reader
Command Response	>Clear ReaderList Reader List has been cleared!
Command Response	>ReaderList = Off ReaderList = OFF

WWWPort

F800 | 9900 | 9680 | 9650

리더의 웹 서버는 보통 수신되는 접속 신호를 포트 80으로 청취한다(기본 www 포트). WWWPort는 허용된 포트 번호에 맞춰 변경이 가능한데, 이때 텔넷, NTP, SNMP 등의 다른 서비스에 통용되는 포트 번호를 사용하지 않도록 주의해야 한다.

- 허용된 값: 정수(0..65535)
- 기본값: 80
- WWWPort를 0으로 설정하면 리더의 웹 서버과 비활성화 되는데, 이는 보안 목적으로 선호되기도 한다. 유의할 점은 리더의 펌웨어 업그레이드를 위한 주요 수단 중 한 가지로 웹 서버가 사용된다는 것이다.
- WWWPort를 변경하면 즉각 효력이 나타난다.

WWWPort 예제

Command Response	>WWWPort? WWWPort = 80
Command Response	>WWWPort = 8080 WWWPort = 8080

HostLog

F800 | 9900 | 9680 | 9650

HostLog는 전체 호스트 동작의 로그를 덤프한다(접속, 접속해제, 시간종료 등). 이를 사용하면 리더와 언제 접속이 있었는지를 알 수 있으며, 직렬인지 TCP인지, 접속 호스트의 IP 주소는 무엇인지도 알 수 있다(TCP인 경우).

- HostLog는 "get" 명령어를 사용해 "Get HostLog" 또는 "HostLog?" 형태가 된다.
- HostLog 데이터는 리더를 부팅할 때마다 재설정되며, 작동 과정 내내 연관된 로그 파일들이 특정 크기가 달했을 때 정기적으로 회전한다.

다음은 HostLog의 예에 주석을 첨가한 것이다.

HostLog 예제

Command
Response
	>HostLog?
	// A user at 10.10.82.75 is connected
	Oct 4 16:18:36 TCP: REQUEST connection from 10.10.82.75 port 1366
	Oct 4 16:18:36 TCP: ACCEPTED connection from 10.10.82.75 port 1366
	Oct 4 16:18:36 TCP: There are 1 clients active
	
	// User at 10.10.82.75 allowed his connection to timeout:
	Oct 4 16:20:45 TCP: No data within 90 seconds! Closing host socket...
	Oct 4 16:20:45 TCP: CLOSING connection from 10.10.82.75.
	
	// A user at 10.10.82.50 connects, but fails to enter the correct password
	Oct 4 16:20:59 TCP: REQUEST connection from 10.10.82.50 port 4358
	Oct 4 16:20:59 TCP: ACCEPTED connection from 10.10.82.50 port 4358
	Oct 4 16:20:59 TCP: There are 1 clients active
	Oct 4 16:21:00 TCP: Error: Login failed password
	Oct 4 16:21:00 TCP: Error: Invalid Username and/or Password
	Oct 4 16:21:01 TCP: CLOSING connection from 10.10.82.50.
	
	// The user at 10.10.82.75 connects again:
	Oct 4 17:02:49 TCP: REQUEST connection from 10.10.82.75 port 4506
	Oct 4 17:02:49 TCP: ACCEPTED connection from 10.10.82.75 port 4506
	Oct 4 17:02:49 TCP: There are 1 clients active
	
	// 10.10.82.50 tries to reconnect but refused while 10.10.82.75 is active:

	Oct 4 17:02:58 TCP: REQUEST connection from 10.10.82.50 port 1127 Oct 4 17:02:58 TCP: ConnectionCount=1, Child PID=26530. Oct 4 17:02:58 TCP: Connection count limit exceeded. Refusing new connection // Finally, the user at 10.10.82.75 gracefully closes his connection Oct 4 17:05:15 TCP: CLOSING connection from 10.10.82.75.

DebugHost

F800 | 9900 | 9680 | 9650

DebugHost 명령어를 사용하면 직렬과 TCP 인터페이스 모두에서 리더에 전송된 모든 명령어의 로그를 만들 수 있다. 명령어 목록은 HostLog 명령어로 볼 수 있다(위 참조).

"내 PersisTime은 왜 늘 재설정되는가?"와 같이 특정 문제를 진달하려는 경우가 아니라면 DebugHost를 켜 두는 것은 좋지 않다. 각 명령어를 로딩하면 처리 오버헤드가 유입되는데, 로그 파일 공간이 제한적이므로 외부 데이터로 채웠을 때 오래된 항목은 로그에서 밀려 나간다.

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"

다음은 DebugHost 명령어를 사용한 예로서, DebugHost가 일정 기간 켜진 이후에 HostLog 처리 과정을 보여주는 예에 주석을 첨가하였다.

DebugHost 예제

Command	>DebugHost = On
Response	DebugHost = ON
Command	>get HostLog

Response	// Some commands were issued on the serial interface:
	Oct 5 08:41:51 SER: [readername = This is my reader now!]
	Oct 5 08:42:31 SER: [tagtype=16]
	Oct 5 08:42:32 SER: [t]
	Oct 5 08:42:33 SER: [t]
	Oct 5 08:42:34 SER: [t]
	Oct 5 08:42:34 SER: [t]
	
	// A TCP connection is made, and they issue some commands:
	Oct 5 08:42:42 TCP: REQUEST connection from 10.10.82.50 port 1179
	Oct 5 08:42:42 TCP: ACCEPTED connection from 10.10.82.50 port 1179
	Oct 5 08:42:42 TCP: There are 1 clients active
	Oct 5 08:42:54 TCP: [get readername]
	Oct 5 08:42:56 TCP: [q]
	Oct 5 08:42:56 TCP: CLOSING connection from 10.10.82.50.
	
	// Finally, the command that initially generated this dump:
	Oct 5 08:43:00 SER: [hostlog?]

시간 명령어

The time at which tags are read by a reader is particularly important for many applications. For this reason, the reader has three time commands to ensure that the onboard real-time clock is always set accurately.

TimeServer

F800 | 9900 | 9680 | 9650

리더는 네트워크 시간 서비스와 동기화하여 재부팅할 때마다 내부 시계를 정확히 설정하며 작동 시간 중에도 필요하면 조정하여 맞춘다. 이때 사용하는 프로토콜을 NTP(Network Time Protocol)라 칭하며, 이는 일반적으로 UTC 형식으로 시간 값을 반환한다.

이 기능을 사용하려면 TimeServer를 지정해야 한다. 이는 NTP 서비스를 계속 실행하는 기계의 네트워크 주소이다. US에서는 정부가 소유하고 작동하는 기계가 상당수 있으며, 인터넷 사용자를 위해 시간과 날짜 정보를 자세히 제공한다.

- 기본적으로 리더는 부팅 시 이러한 기기에 접속하여 현재 시각을 알도록 구성한다. 부팅 후에는 리더가 NTP 서비스에서 보고한 현재 시각에 맞도록 시계를 계속 조정한다.
- 이 서버에 대한 더 자세한 설명과 공개적으로 접속 가능한 DPS의 목록을 원한다면, 다음을 참조할 것. http:// www.boulder.nist.gov/timefreq/service/its.htm
- 이 명령어의 기본 설정값은 132.163.4.101-103 즉 주요 NIST 네트워크 시간 서버에 맞춰져 있다. 그외 다른 시간 서버는 다음과 같다.

  time-a.nist.gov : 129.6.15.28
  time-b.nist.gov : 129.6.15.29
  time.nist.gov : 192.43.244.18

- 이 명령어로 변경한 후에는 저장하고 재부팅하여 변경사항을 적용시켜야 한다.
- 리더가 네트워크 구성 시 DHCP를 사용하도록 설정되면, DHCP 서버가 사용할 TimeServer를 제공할 수 있는데, 이 경우에는 사용자가 이 명령어로 설정할 특정 TimeServer를 무효화하게 된다.
- 리더는 DHCP 서버가 제공하거나 TimeServer 명령어로 별도 공간에서 서버 목록이 제공될 때, 사용할 TimeServer의 목록을 수락할 수 있다. 이로써 시간 정확성이 개선된다.

TimeServer 예제	
Command Response	>TimeServer? TimeServer = 129.6.15.28
Command Response	>TimeServer = 129.6.15.28 TimeServer = 129.6.15.28
Command Response	// Setting & Getting multiple NTP servers >TimeServer = 132.163.4.101 132.163.4.102 TimeServer = 132.163.4.101 132.163.4.102

TimeZone

F800 | 9900 | 9680 | 9650

이 명령어들은 현재 시간대를 지정하거나 리더에서 검색할 수 있게 해준다. 시간대는 현지 시간을 결정하기 위해 UTC(Coordinated Universal Time; 또는 GMT나 Zulu로도 알려져 있음)에 추가하거나 빼야 할 시간 수를 지정한다.

예를 들어, UTC에서 PST로 전환하려면 TimeZone을 -8로 설정한다. UTC에서 PDT로 전환하려면 TimeZone을 -7로 설정한다.

- 허용된 값: 정수(-12..12)
- 기본값: -7 (Pacific Daylight Time)
- PDT가 -7인 이유는 PDT가 UTC 시간 - 7시간이기 때문이다.
- 시간대, 서버와 UTC에 대한 더 자세한 정보를 원한다면 Get/Set TimeServer 명령어 아래 나열한 웹사이트를 참조할 수 있다.
- ALR-F800 리더는 사용자가 "TZ database" 이름을 사용하여 시간대를 지정할 수 있게 한다. (e.g. "US/Pacific"). TZ 데이터베이스 영역 명칭의 목록을 여기서 찾을 수 있다: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

리더는 일광 절약 시간에 맞춰 자동으로 시간을 조정하지는 않는다. DST에 맞춰 조정하는 손쉬운 방법은 TimeZone 값에서 수동으로 1시간을 더하거나 빼는 것이다.

TimeZone 예제	
Command Response	>TimeZone? TimeZone = -8
Command Response	>TimeZone = 3 TimeZone = 3

Time

F800 | 9900 | 9680 | 9650

이 명령어들은 현재 시간을 지정하거나 리더에서 검색할 수 있게 해준다.

- 이 명령어로 사용하는 시간은 항상 지역 시간으로 지정되며, TimeZone 명령어로 정의한다.
- 시간은 항상 YYYY/MM/DD hh:mm:ss 형식으로 지정한다.
- 이 명령어로 수행한 변경은 즉시 효력을 발휘한다.

Time 예제	
Command Response	>Time? Time = 2002/6/3 9:23:01
Command Response	>Time = 2002/6/3 19:23:01 Time = 2002/6/3 19:23:01

외부 입/출력 명령어

이러한 명령어는 사용자가 리더의 외부 입력/출력값에서 얻은 데이터를 구성하고 검색할 수 있게 해준다. 태그 읽기 버퍼링 TagList의 개념은 이제 새로운 IOList 구성 및 관련 명령어의 영역까지 확대되었는데, 외부 입출력값에 대한 변경사항을 버퍼링 및 보고하고 그 값을 후에 검색할 수 있다. TCP/IP 상에서 원격 청취자에게 I/O 이벤트를 스트리밍하는 IOStream 수단도 있어서 호스트 애플리케이션이 I/O 변경 시점에 대해 리더에 계속 알릴 필요가 없다.

ExternalInput

F800 | 9900 | 9680 | 9650

리더는 외부 입력값을 모니터링하는데, 이는 외부 근접 정도 식별기기와 그밖에 "electric eyes"나 자기 스위치 등 입력 장치로 제어가 가능하다. 이 명령어를 사용하면 이러한 외부 입력값을 입수할 수 있다. 핀 배치 도표를 보려면 하드웨어 설정 안내서(Hardware Setup Guide)를 참조해 보기 바란다.

- 이 명령어는 외부 입력 상태의 비트맵을 표시하는 단일 바이트 결과를 반환한다. 비트 0(LSB)은 입력값 #0의 상태를, 비트 1은 입력값 #1의 상태를 나타낸다.

ExternalInput 예제	
Command	>ExternalInput?
Response	ExternalInput = 2 (i.e., binary "10")

ExternalOutput

F800 | 9900 | 9680 | 9650

리더는 외부 출력값을 제어하는데, 차후에 이를 사용하여 도어/게이트, 보안등과 같은 외부 장치를 제어할 수 있다. 핀 배치 도표를 보려면 하드웨어 설정 안내서(Hardware Setup Guide)를 참조해 보기 바란다.

이 명령어를 사용하면 외부 출력 상태 정보를 얻거나 설정할 수 있다. 단일 매개변수/반환값은 외부 출력값 상태를 나타내는 정수 비트맵이다.

- 비트 0은 출력값 #1의 상태를, 비트 1은 출력값 #2의 상태를 나타낸다.
- 예를 들어, 출력값 #2가 높게, 출력값 #1이 낮게 설정되면, 10진수 2의 비트맵을 사용하면 되는데, 이를 달리 표현하면 2진수 10에 해당된다.

ExternalOutput 예제	
Command	>ExternalOutput = 2
Response	ExternalOutput = 2
Command	>ExternalOutput?
Response	ExternalOutput = 2

InvertExternalInput

F800 | 9900 | 9680 | 9650

기본 구성에서는 외부 입력값에 고전압을 적용하면 외부 입력값이 "1" 또는 "on"이 된다. 연결된 장치의 전자공학적 속성에 따라서 핀에 고전압이 가해진 것은 장치가 "활성"이 아닌 "비활성" 상태이기 때문인 것일 수 있다. 이러한 경우를 대비하고 외부 입력 상태의 비트맵을 조사할 때 혼동을 피하려면, InvertExternalInput을 켜서 모든 외부 입력값의 "의미(sense)"를 역전시키는 방법이 있는데, 이렇게 하면 핀 전압이 낮아지거나 그 반대일 때 외부 입력값이 "1"이 된다.

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"

InvertExternalInput 예제	
Command Response	>ExternalInput? ExternalInput = 0
Command Response	>InvertExternalInput = ON InvertExternalInput = ON
Command Response	>InvertExternalInput? InvertExternalInput = ON
Command Response	>ExternalInput? ExternalInput = 15 (inverted value - inputs haven't changed)

InvertExternalOutput

F800 | 9900 | 9680 | 9650

기본 구성에서 외부 입력값을 "1"이나 "on"으로 설정하면 해당 출력 핀 상의 전압이 상승한다. 연결된 장치의 전자공학적 속성에 따라서, 핀 상에 고전압이 흐르면 해당 장치가 "활성화"되거나 그 반대가 될 수 있다. 이러한 경우를 대비하고 원하는 외부 출력 상태의 비트맵을 명시할 때 혼동을 피하려면, InvertExternalOutput을 켜서 모든 외부 출력값의 "의미(sense)"를 역전시키는 방법이 있는데, 이렇게 하면 외부 출력값이 "0"으로 설정되어 있거나 그 반대일 때 핀 전압이 상승한다.

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"
- ON 상태이면, 출력 상태의 역전이 출력 상태 정보를 얻거나 설정할 때 모두에 적용된다. 따라서 출력 핀에서의 전압을 테스트하는 경우가 아니라면 변경 내역이 있는 그대로 확인된다.

- InvertExternalOutput 상태는 부팅 중에는 InitExternalOutput에 영향을 미치지 않는다. InitExternalOutput에서 정의한 상태는 InvertExternalOutput 설정에 관계 없이 출력값에 적용된다.

InvertExternalOutput 예제	
Command Response	>ExternalOutput? ExternalOutput = 0
Command Response	>InvertExternalOutput = ON InvertExternalOutput = ON
Command Response	>InvertExternalOutput? InvertExternalOutput = ON
Command Response	>ExternalOutput? ExternalOutput = 15 (inverted value - outputs haven't changed)

InitExternalOutput

F800 | 9900 | 9680 | 9650

리더의 전원을 켜면, 외부 출력을 InitExternalOutput 속성으로 정의한 상태로 설정한다. 이로써 사용자는 어떤 출력의 기본값을 높게, 어떤 출력은 낮게 선택할지에서 선택 폭이 생기며, 이는 리더 I/O 포트가 제어하는 기계 장치가 많을 때 특히 매우 중요한 고려사항이다.

이 명령어를 사용하면 초기에 외부 출력 상태 정보를 얻거나 설정할 수 있다. 단일 매개변수/반환값은 시동 중이나 후에 외부 출력값 상태를 나타내는 정수 비트맵이다.

- 비트 0은 출력값 #1의 원하는 상태를, 비트 1은 출력값 #2의 원하는 상태를 나타낸다.
- 예를 들어, 출력값 #2가 높게, 출력값 #1이 낮게 설정되면, 10진수 2의 비트맵을 사용하면 되는데, 이를 달리 표현하면 2진수 0010에 해당된다.
- 이 명령어로 수행한 변경은 다음 재부팅 중에 효력을 발휘한다.
- 사용자가 지정하는 비트맵은 시동 시 출력값에 직접 적용되며, 이전 같으면 InvertExternalOutput으로 설정했었을 역전은 필요하지 않다.

InitExternalOutput 예제	
Command	>InitExternalOutput = 7 *(0111$_{binary}$ = #1, 2, 3 high, #4 low)*
Response	InitExternalOutput = 7
Command	>InitExternalOutput?
Response	InitExternalOutput = 7

Get IOList (ios)

F800 | 9900 | 9680 | 9650

"Get IOList" 명령어를 사용하면 리더가 저장한 IOList를 검색할 수 있다. 리더는 디지털 입력/출력에 적용되는 변경을 모두 모니터링하며, 이를 내부 IOList에 기록한다. I/O 이벤트는 발생 순서대로 나열된다. 목록을 검색하면 저장된 이벤트들은 지워진다.

- IOList에 보관할 수 있는 I/O 이벤트의 최대수는 100,000이다.
- "Get IOList"와 "ios"는 서로 바꿔 사용할 수 있다.

이 명령어로 반환되는 데이터의 형식은 IOListFormat 명령어를 사용하여 지정하며, 사용자는 IOType 명령어를 사용하여 입력 또는 출력 또는 둘 모두(이벤트 시간에 따라 상호 배치하거나 별도로 열거하여) 추적할 것인지 지정할 수 있다. 이 명령어는 둘 다 차후에 자세히 설명할 것이다. 일반적으로 IOList 상의 각 항목은 해당 이벤트가 디지털 입력(DI)나 디지털 출력(DO)에 대한 변경인지 여부, 이벤트가 발생한 때의 타임스탬프, 입력 또는 출력의 새로운 상태(비트 마스크) 등을 표시한다.

IOList에 표시한 값은 어떤 I/O가 변경되었는지 또는 어떤 방향으로 변경되었는지 명확하게 알려주지 않는다는 것에 유의해야 한다. 단순히 모든 I/O의 새로운 상태를 알려줄 뿐이다. 어떤 I/O 상태가 어떤 방향으로 바뀌었는지 알려면, 이전 값과 새 값을 비교해야 한다.

IOList 예제	
Command	>get IOList
Response	IO:DI, Time:2007/01/18 10:15:57.018, Data:0 IO:DO, Time:2007/01/18 10:16:07.486, Data:0
Command	>ExternalOutput = 15
Response	ExternalOutput = 15
Command	>ios
Response	IO:DO, Time:2007/01/18 11:43:42.582, Data:15

IOPersistTime

F800 | 9900 | 9680 | 9650

IOPersistTime은 IO 이벤트가 리더의 내부 IOList 내에 남아 있는 시간 길이를 명시한다. 사용자는 IOList를 완전히 꺼 둘 수도 있으며, 다음에 요청할 때까지 IO 이벤트를 계속 유지할 수 있다. 이때 "지속 시간(persist time)"을 선택할 수 있어서, 이 후에는 중요하지 않은 이벤트를 목록에서 제거한다.

- 허용된 값: Integer(-1..86400)
- 기본값: 0 (어떤 IO 이벤트도 보관하지 않음)
- 지속 시간은 초 단위로 지정한다.
- IOPersistTime을 -1로 설정하면 `get IOList` 명령어를 전송할 때까지 히스토리를 무한정 형성한다. 이 명령어 전송 시 IOList가 반환되며, 즉시 목록이 지워진다.
- IOPersistTime > 0은 IO 이벤트가 자동으로 삭제되기 전까지 얼마 동안은 IOList에 남아있다는 의미이다. 그 전에 IOList를 요청하지 않으면, 이 이벤트들을 잃게 된다.
- IO 이벤트는 IOList 상에 보관되기 전에 우선 스트리밍 된다(IOStreamMode가 구성된 경우에 해당됨). IOPersistTime은 IO 스트리밍에 아무런 영향을 미치지 않는다.

IOPersistTime 예제	
Command Response	>IOPersistTime? IOPersistTime = 0
Command Response	>IOPersistTime=300 IOPersistTime = 300

IOType

F800 | 9900 | 9680 | 9650

사용자는 IOType 명령어로 추적 및 보고하고자 하는 I/O 이벤트를 지정한다. IOType 값은 다음과 같이 논리적 열거형, DI, DO, DIO 중 하나로 지정한다.

DI - 입력값만
DO - 출력값만
DIO - 입력 및 출력을 이벤트 시간 별(기본값)로 상호 배치

IOType 열거 항목들은 "|" (수직 막대), 스페이스, 콤마, 세미콜론(;), 또는 앰퍼샌드(&) 문자로 조합할 수 있다. 리더는 콤마로 분리할 수 있도록 이 값들의 형식을 바꾼다.
`IOType` 의 기본값은 `DI`이다.

현재 조합이 가능한 항목은 DI와 DO 뿐이나, 향후 조합이 가능한 I/O 이벤트가 늘어날 것이다. 예를 들면 다음과 같다.
`IOType = DI, DO`
'`DI,DO`'는 '`DIO`'와는 다른데, 입력값을 출력값과는 별도로 열거하기 때문이다(상호배치 하지 않음)

IOType 예제	
Command Response	>IOType? IOType = DIO
Command Response	>get IOList IO:DI, Time:2007/01/19 09:48:16.471, Data:0 IO:DO, Time:2007/01/19 09:48:26.918, Data:0 IO:DI, Time:2007/01/19 10:08:17.627, Data:13 IO:DI, Time:2007/01/19 10:08:32.718, Data:9 IO:DO, Time:2007/01/19 10:08:35.266, Data:11 IO:DO, Time:2007/01/19 10:08:37.856, Data:3 IO:DI, Time:2007/01/19 10:08:39.766, Data:13 IO:DI, Time:2007/01/19 10:08:40.538, Data:15 IO:DO, Time:2007/01/19 10:08:42.867, Data:0 IO:DO, Time:2007/01/19 10:08:45.477, Data:211
Command Response	*// Assuming the same I/O events as above…* >IOType = DI IOType = DI
Command Response	>ios IO:DI, Time:2007/01/19 09:48:16.471, Data:0 IO:DI, Time:2007/01/19 10:08:17.627, Data:13 IO:DI, Time:2007/01/19 10:08:32.718, Data:9 IO:DI, Time:2007/01/19 10:08:39.766, Data:13 IO:DI, Time:2007/01/19 10:08:40.538, Data:15
Command Response	*// Assuming the same I/O events…* >IOType = DO IOType = DO
Command Response	>ios IO:DO, Time:2007/01/19 09:48:26.918, Data:0 IO:DO, Time:2007/01/19 10:08:35.266, Data:11 IO:DO, Time:2007/01/19 10:08:37.856, Data:3 IO:DO, Time:2007/01/19 10:08:42.867, Data:0 IO:DO, Time:2007/01/19 10:08:45.477, Data:211
Command Response Command Response	*// Assuming the same I/O events…* >IOType = DI, DO IOType = DI,DO IO:DI, Time:2007/01/19 09:48:16.471, Data:0 IO:DI, Time:2007/01/19 10:08:17.627, Data:13 IO:DI, Time:2007/01/19 10:08:32.718, Data:9 IO:DI, Time:2007/01/19 10:08:39.766, Data:13 IO:DI, Time:2007/01/19 10:08:40.538, Data:15 IO:DO, Time:2007/01/19 09:48:26.918, Data:0 IO:DO, Time:2007/01/19 10:08:35.266, Data:11 IO:DO, Time:2007/01/19 10:08:37.856, Data:3 IO:DO, Time:2007/01/19 10:08:45.477, Data:211

IOListFormat

F800 | 9900 | 9680 | 9650

IOListFormat 명령어를 사용하면 사용자가 IOLists의 형식을 지정할 수 있다. 이 명령어는 텍스트 문자열을 인수로 취하며, 다음 예들과 같다.

형식	설명
Text	일반 텍스트 메시지로 표시되는 IOList, 행마다 I/O 이벤트가 하나씩.
Terse	일반 텍스트 메시지로 표시하는 IOList, 행마다 I/O 이벤트가 하나씩. 단 단순화된 출력값으로, 이벤트 유형, 타임스탬프, 값만 표시하며 레이블은 없음.
XML	XML 텍스트 형식으로 표시하는 IOList
Custom	IOListCustomFormat로 제시된 형식으로 IOList를 표시함.

- IOList 상의 모든 타임스탬프는 밀리세컨드 단위로 표시되며 사용자가 읽을 수 있는 형태의 초 단위 뒤에 .xxx가 붙거나 유닉스 시간(1970년 1월 1일) 이후 밀리세컨드 수로 표시된다.

- 기본값은 "Text"이다.
- 텍스트 형식의 IOLists는 일반적으로 다음 형태를 취한다.

```
IO:{DI|DO}, Time:YYYY/MM/DD hh:mm:ss.xxx, Data:<integer value>
```

예:

```
IO:DI, Time:2007/01/19 09:48:16.471, Data:0
IO:DO, Time:2007/01/19 09:48:26.918, Data:0
```

- 개요 형식의 IOLists는 일반적으로 다음 형태를 취한다(1 = DI, 2 = DO).

```
{1|2},<millisecond timestamp>,<integer value>
```

예:

```
2,1169231710479,1  // DOs went to state 1
1,1169231712696,13 // DIs went to state 1
```

- XML 형식의 IOLists는 일반적으로 다음 형태를 취한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-IO-List>
   <Alien-RFID-IO>
      <Type>{DI | DO}</Type>
      <Time>YYYY/MM/DD hh:mm:ss.xxx</Time>
      <Data><integer value></Data>
   </Alien-RFID-IO>
   …
</Alien-RFID-IO-List>
```

예:

```
<?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-IO-List>
   <Alien-RFID-IO>
      <Type>DO</Type>
      <Time>2007/01/19 10:39:16.349</Time>
      <Data>5</Data>
   </Alien-RFID-IO>
</Alien-RFID-IO-List>
```

- 사용자 정의 형식의 IOLists는 IOListCustomFormat으로 명시한 형식지정 문자열을 사용하며, 정의된 "사용자 지정 표시(custom tokens)" 자리에 실제 값을 대체한다. 더 자세한 정보와 예를 보려면 IOListCustomFormat 섹션(아래)을 참조하기 바란다.

IOListFormat 예제	
Command Response	>IOListFormat? IOListFormat = Text
Command Response	>IOListFormat = XML IOListFormat = XML

IOListCustomFormat

F800 | 9900 | 9680 | 9650

IOListCustomFormat 명령어를 사용하면 사용자 정의 IOList를 정의할 수 있다. 일단 형식을 정의했으면, 명령어 "`IOListFormat = Custom`"를 전송하여 적용하면 된다.

IOListCustomFormat 명령어는 각 I/O 이벤트를 나타낼 방법을 정의하는 단순 텍스트 행 인수를 취한다. 이 인수는 아래 표에 정의된 대로 텍스트와 사용자 정의 표시를 조합한 형태이다. IOListCustomFormat 정의의 최대 길이는 255 문자이다.

리더가 사용자정의 IOList를 출력하면, 사용자 정의 형식의 표시를 각 I/O 이벤트 별로 실제 값으로 대체한다. 사용자는 짧고 긴 표시를 상호 간에 사용할 수 있다.

짧은 표시	긴 표시	설명
%E	**${EVENT}**	I/O 이벤트 유형 (텍스트): "DI" or "DO"
%e	**${ID}**	I/O 이벤트 유형 (수치): 1=DI, 2=DO
%T	**${TIME}**	I/O 이벤트 시간 (텍스트): hh:mm:ss.xxx
%t	**${MSEC}**	I/O 이벤트 시간 (수치): 01/01/1970 이후로 msec
%d	**${DATE}**	I/O 이벤트 일자: YY/MM/DD
%D	**${DATELONG}**	I/O 시간/일자: YY/MM/DD hh:mm:ss.xxx
%v	**${DATA}**	DI 또는 DO의 새로운 (비트 마스크) 상태
%N	**${NAME}**	리더의 ReaderName
%H	**${HOST}**	리더의 Hostname
%I	**${IP}**	리더의 IP 주소
%M	**${MAC}**	리더의 MAC 주소

IOListCustomFormat 예제	
Command Response Command Response	>IOListCustomFormat = The new %E value is %v. IOListCustomFormat = The new %E value is %v. >ios The new DO value is 6. The new DI value is 3.
Command Response Command Response	>IOListCustomFormat = E=%E e=%e T=%T t=%t d=%d D=%D v=%v IOListCustomFormat = E=%E e=%e T=%T t=%t d=%d D=%D v=%v >IOList? E=DO e=2 T=14:04:48.286 t=1169244288286 d=2007/01/19 D=2007/01/19 14:04:48.286 v=2 E=DI e=1 T=14:04:51.732 t=1169244291732 d=2007/01/19 D=2007/01/19 14:04:51.732 v=13

Clear IOList

F800 | 9900 | 9680 | 9650

"Clear IOList" 명령어는 리더가 내부 IOList를 즉시 지우도록 지시한다.

Clear IOList 예제	
Command Response	>Clear IOList IO List has been cleared!

IOStreamMode

F800 | 9900 | 9680 | 9650

IOStreamMode 명령어는 IOStream 기능을 켜거나 끌 수 있다. 디지털 입력값이 변경되었는지 결정하려고 리더를 계속 확인할 필요 없이, 리더가 디지털 I/O 이벤트를 TCP 소켓이나 리더의 직렬포트로 스트리밍되도록 구성할 수 있다.

IOStreamMode가 켜져 있으면, 디지털 입력 및 출력 상태의 변화가 있을 때마다 리더는 이벤트 정보를 IOStreamAddress로 스트리밍 한다(아래 참조). IOType 명령어(DI, DO, DIO 등)으로 인한 이벤트만 스트리밍이 가능하다.

스트리밍한 데이터의 형식은 IOStreamFormat와 IOStreamCustomFormat 명령어로 지정한다(아래).

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"
- 리더가 데이터 스트리밍을 제대로 하지 못하면(TCP 소켓 오류 등), 데이터를 잃게 된다.

IOStreamMode 예제	
Command Response	>IOStreamMode? IOStreamMode = OFF
Command Response	>IOStreamMode = on IOStreamMode = ON

IOStreamAddress

F800 | 9900 | 9680 | 9650

IOStreamAddress 명령어는 IOStream 이벤트 데이터를 어디로 전송할 것인지를 지정한다. 지원되는 전송 방법이 두 가지 있는데, 아래 표에 제시하였다.

IOStreamAddress	설명
hostname:port	메시지를 네트워크에 있는 기기 상의 지정된 포트로 전송한다. 주소는 "hostname:port"의 형식을 취한다. 예를 들어, "123.01.02.98:3450" 또는 "listener.alientechnology.com:10002"
SERIAL	메시지를 직렬포트로 전송한다. "serial"이라는 단어를 주소에 사용한다. 단어의 대소문자 구별은 없다. ALR-F800: 메시지를 SERIAL0 (RS-232)과 SERIAL1 (USB) 포트 모두에 전송한다.
(ALR-F800 only) **SERIAL0** **SERIAL1**	ALR-F800의 직렬포트 중 하나로 메시지를 전송한다. SERIAL0는 RS-232 포트로 메시지를 보낸다. SERIAL1은 USB 포트로 보낸다.

- 유의사항: 직렬포트로 데이터 스트리밍할 때 주의를 요한다. I/O 이벤트는 매우 빠르게 발생할 수 있으며, 동시에 여러 명령어를 전송하면 직렬 포트의 반응성에 좋지 않은 영향을 미칠 수 있다.

IOStreamAddress 예제	
Command Response	>IOStreamAddress? IOStreamAddress = 10.1.0.12:4000
Command Response	>IOStreamAddress = serial IOStreamAddress = SERIAL

IOStreamFormat

F800 | 9900 | 9680 | 9650

IOStreamFormat 명령어는 사용자가 IOStream 데이터(IOList와는 구분됨)의 형식을 지정할 수 있게 한다. 이 명령어는 텍스트 문자열을 인수로 취하며, 다음 예들과 같다.

형식	설명
Text	일반 텍스트 메시지로 표시되는 IOStream 데이터, 행마다 I/O 이벤트가 하나씩.
Terse	일반 텍스트 메시지로 표시하는 IOStream 데이터, 행마다 I/O 이벤트가 하나씩. 단 단순화된 출력값으로, 이벤트 유형, 타임스탬프, 값만 표시하며 레이블은 없음.
XML	XML 텍스트 형식으로 표시하는 IOStream 데이터
Custom	IOStream 데이터는 IOStreamCustomFormat가 제시한 형식으로 표시된다.

- IOStream 상의 모든 타임스탬프는 밀리세컨드 단위로 표시되며 사용자가 읽을 수 있는 형태의 초 단위 뒤에 .xxx가 붙거나 유닉스 시간(1970년 1월 1일) 이후 밀리세컨드 수로 표시된다.

- 기본값은 "Terse"이다.
- IOStream 데이터의 형식은 IOList 데이터와 동일하다. 출력 형식과 각 유형별 예는 IOListFormat 설명 섹션을 참조할 것.

IOStreamFormat 예제	
Command Response	>IOStreamFormat? IOStreamFormat = Terse
Command Response	>IOStreamFormat = Text IOStreamFormat = Text

IOStreamCustomFormat

F800 | 9900 | 9680 | 9650

IOStreamCustomFormat 명령어를 사용하면 사용자 정의 IOStream 형식을 정의할 수 있다. 일단 형식을 정의했으면, 명령어 "`IOStreamFormat = Custom`"를 전송하여 적용하면 된다.

IOStreamCustomFormat 명령어는 각 I/O 이벤트를 나타낼 방법을 정의하는 단순 텍스트 행 인수를 취한다. 이 인수는 아래 표에 정의된 대로 텍스트와 사용자 정의 표시를 조합한 형태이다. IOStreamCustomFormat 정의의 최대 길이는 255 문자이다.

리더가 사용자정의 IOStream 이벤트를 출력하면, 사용자 정의 형식의 표시를 I/O 이벤트 별로 실제 값으로 대체한다.

- IOStream에서 사용 가능한 사용자 정의 표시는 IOList와 동일하다. 허용되는 표시 목록에 대해서는 IOListCustomFormat 설명을 참조할 것.
- IOStreamCustomFormat 기본값은 "%E,%v "이다.

IOStreamCustomFormat 예제	
Command Response	>IOStreamCustomFormat? IOStreamCustomFormat = %E,%v
Command Response	>IOStreamCustomFormat = %E changed to %v at %T. IOStreamCustomFormat = %E changed to %v at %T.

IOStreamKeepAliveTime

F800 | 9900 | 9680 | 9650

리더가 IOStream 이벤트 메시지를 네트워크 상에서 전송하려면 TCP 소켓을 열어야 한다. 소켓을 열고 닫으면 일부 프로세서와 네트워크 오버헤드가 필요하며, 메시지가 리더를 빠르게 빠져 나오는 경우에는 오버헤드가 짐이 되어 리더의 반응성을 늦춘다.

IOStreamKeepAliveTime 명령어는 리더가 IOStream 소켓을 열어두는 시간을 설정한다(초 단위). IOStream 이벤트의 신속성을 위해서는 리더가 소켓을 계속 열어두는 것이 유리할 것이다. 이를 위해서는 IOStreamKeepAliveTime을 이벤트 간에 예상되는 시간 보다 더 큰 수로 설정하면 된다.

한편 소켓을 열어두면 네트워크에 부담이 된다. IOStream 이벤트가 자주 있다면, IOStreamKeepAliveTime을 적게 하여 소켓이 하나의 메시지를 발송하는데 충분한 길이만큼만 열어두고 메시지 전송 후에 자동으로 닫게 하는 것이 좋다.

- 허용된 값: 정수(0..32767)
- 기본값: 30초

- 리더는 소켓을 자동으로 다시 열고, 데이터를 다음 번에 전송해야 할 때까지 일시 중단 시간이 있다.

IOStreamKeepAliveTime 예제	
Command Response	>IOStreamKeepAliveTime? IOStreamKeepAliveTime = 30
Command Response	>IOStreamKeepAliveTime = 2 IOStreamKeepAliveTime = 2

BlinkLED

F800 | 9900 | 9680 | 9650

특히 유용한 BlinkLED 명령어가 있는데, 이로써 두 패턴 사이에 리더의 여러 LED 상태 표시등이 순환하면서 각 상태 별 지정된 지속기간과 순환하는 시간을 표시한다. 이 기능을 사용하면 오류 상태를 표시하거나 근처 운용기사에게 지침을 전달할 수 있다. 또는 상태 LED가 모두 작동하는지 확인할 수도 있다.

```
BlinkLED = <LEDState1> <LEDState2> <duration> <count>
```

<LEDState1>과 <LEDState2>는 두 개의 LED 상태로서, 10진수 비트마스크로 표시한다. 맵핑 도표를 아래 제시하였다. 비트1 은 최하위 비트이다.

ALR-F800	ALR-9900	ALR-9680	ALR-9650
bit 1 – CPU (red)	bit 1 – Ant 0	bit 1 – RF On	bit 1 – RF On
bit 2 – CPU (green)	bit 2 – Ant 1	bit 2 – Read	bit 2 – Read
bit 3 – Read	bit 3 – Ant 2	bit 3 – Fault (red)	bit 3 – Fault (red)
bit 4 – Sniff	bit 4 – Ant 3	bit 4 – N/A	bit 4 – N/A
bit 5 – Ant 0	bit 5 – CPU	bit 5 – N/A	bit 5 – N/A
bit 6 – Ant 1	bit 6 – Read	bit 6 – N/A	bit 6 – N/A
bit 7 – Ant 2	bit 7 – Sniff	bit 7 – N/A	bit 7 – N/A
bit 8 – Ant 3	bit 8 – Fault (red)	bit 8 – N/A	bit 8 – N/A

<duration>은 밀리세컨드 단위의 시간이며, 각 출력 상태를 유지하는 기간을 가리킨다. <count>는 두 가지 상태를 순환하는 횟수이다.

- <LEDState> 허용값: Integer(0..255) (최대값은 리더의 LED 수에 따라 다름)
- <duration> 허용값: Integer(0..2500)
- <count> 허용값: Integer(0..255)
- BlinkLED 명령어의 총 지속기간(2 * <duration> * <count>)은 25초 이하여야 한다.
- BlinkLED 명령어 실행 중에는 다른 명령어를 전송할 수 없다.
  AutoMode는 BlinkLED 명령어가 끝날 때가지 정지한다. 리더는 BlinkLED 실행 중에는 태그를 읽을 수 없다.

BlinkLED 예제	
Command Response	// Flash all LEDs On and Off 3 times, pausing for 1 sec each time >BlinkLED = 0 255 1000 3 BlinkLED = 0 <--> 255 (1000 msec, 3 times)
Command Response	// Flash the fault light (ALR-9900) rapidly for 2 sec total. >BlinkLED = 128 0 100 20 BlinkLED = 128 <--> 0 (100 msec, 20 times)

TagList Commands

TagList 명령어는 리더가 읽고 저장한 태그 목록을 바로 검색할 수 있게 해주며, TagList 기능 관련 매개변수들을 지정하고 검색할 수 있다.

Get TagList (t)

F800 | 9900 | 9680 | 9650

"Get TagList" 명령어를 사용하면 리더가 저장한 TagList를 검색할 수 있다.

- TagList에 보관할 수 있는 태그의 최대수는 6,000이다. 일부 리더 모델은 정전 중에도 TagList 데이터 일부를 보유할 역량이 있다.
- "Get TagList"와 "t"는 서로 바꿔 사용할 수 있다.

Get TagList를 사용하여 저장된 리스트를 검색하려면 다음 조건이 갖춰졌을 때 가능하다.

- **리더가 자율 모드가 아닐 때**, 리더는 전체 태그 검색(읽고 보고하기)을 즉시 수행하고 현재 내부 TagList를 표시한다. 응답은 다중행 응답이며, 각 행마다 활성 태그를 열거한다. TagList가 비어 있으면, “(No Tags)” 메시지를 반환한다.
- **리더가 자율 모드일 경우**, 리더는 현재 내부 TagList만 반환한다.

이 명령어로 반환되는 데이터의 형식은 아래 기술한 TagListFormat 명령어를 사용하여 지정한다.

TagList 예제	
Command Response	>get TagList Tag:8000 8004 0000 003B, Disc:2003/12/04 12:35:11, Last:2003/12/04 12:35:11, Count:3, Ant:0 Tag:8000 8004 9999 0004, Disc:2003/12/04 12:35:11, Last:2003/12/04 12:35:11, Count:3, Ant:0
Command Response	>get TagList (No Tags)

PersistTime

F800 | 9900 | 9680 | 9650

PersistTime은 태그 데이터가 리더의 내부 활성 태그 목록에 남아 있는 시간 길이를 명시한다.

- 허용된 값: Integer(-1..86400)
- 기본값: -1
- 지속 시간은 초 단위로 지정한다.
- 지속 기간이 없으면(0) 태그는 TagList에 보관하지 않는다. 단, 인터랙티브 모드에서 `get TagList` 명령어를 전송하면 TagList에 보관하지 않을지라도 찾아낸 태그를 즉시 반환한다.
- 지속 시간을 -1로 설정하면 get TagList 명령어를 전송할 때까지 히스토리를 무한정 형성한다. 이 명령어 전송 시 TagList가 반환되며, 즉시 목록이 지워진다. 이는 기본 설정값이다.

내부 TagList에 보관할 수 있는 태그의 최대수는 6,000이다. 일단 이 태그 한계에 달하면, 이전 태그 항목은 새 것으로 대체된다. ALR-9900+에는 입수한 태그 판독값을 비휘발성 메모리에 정기적으로 보관하는 추가 기능이 있어서 정전 시에도 리더가 재부팅하면 동일한 태그 목록으로 돌아갈 수 있다.

PersistTime 예제	
Command Response	>PersistTime? PersistTime = -1
Command Response	>PersistTime=300 PersistTime = 300

TagListFormat

F800 | 9900 | 9680 | 9650

Get 과 Set TagListFormat 명령어는 TagLists 형식을 지정한다. 이 명령어는 텍스트 문자열을 인수로 취하며, 다음 예들과 같다.

Format	Description
Text	일반 텍스트 메시지로 표시되는 TagList, 행마다 태그 ID가 하나씩.
Terse	일반 텍스트 메시지로 표시되는 TagLists, 행마다 태그 ID가 하나씩. 단, TagID, Antenna, ReadCount만 표시되고 레이블은 없음.
XML	XML 텍스트 형식으로 표시하는 TagLists
Custom	TagLists는 TagListCustomFormat로 설명한 형식으로 표시된다.

- Text 형식의 TagLists는 일반적으로 다음 형태를 취한다:

```
Tag:E200 3411 B801 0108, Disc:2007/06/29 08:30:49, Last:2007/06/29 10:38:12,
Count:292, Ant:0, Proto:2
Tag:4461 7669 6445 2E4B, Disc:2007/06/29 10:38:13, Last:2007/06/29 10:38:13,
Count:187, Ant:1, Proto:2
```

- Terse 형식의 TagLists는 일반적으로 다음 형태를 취한다:

```
1115 F268 81C3 C012,0,4
0100 0100 0002 0709,0,6
1054 A334 54E1 7409,0,2
```

  개요 형식으로 제시되는 필드는 다음과 같다. TagID, ReadCount, Antenna.

- XML 형식의 TagLists는 일반적으로 다음 형태를 취한다:

```
<?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-Tag-List>
    <Alien-RFID-Tag>
        <TagID>0000 0000 0000 0000 0000 0000</TagID>
        <DiscoveryTime>2005/05/31 17:39:13</DiscoveryTime>
        <LastSeenTime>2005/05/31 17:39:13</LastSeenTime>
        <Antenna>1</Antenna>
        <ReadCount>22</ReadCount>
        <Protocol>0</Protocol>
    </Alien-RFID-Tag>
    <Alien-RFID-Tag>
        <TagID>A5A5 FFFF 8000 8004 6546 6091</TagID>
        <DiscoveryTime>2005/05/31 17:39:13</DiscoveryTime>
        <LastSeenTime>2005/05/31 17:39:13</LastSeenTime>
        <Antenna>0</Antenna>
        <ReadCount>3</ReadCount>
        <Protocol>1</Protocol>
    </Alien-RFID-Tag>
    <Alien-RFID-Tag>
        <TagID>3000 2141 60C0 0400 0000 6013</TagID>
        <DiscoveryTime>2005/05/31 17:39:13</DiscoveryTime>
        <LastSeenTime>2005/05/31 17:39:14</LastSeenTime>
        <Antenna>1</Antenna>
        <ReadCount>19</ReadCount>
        <Protocol>2</Protocol>
    </Alien-RFID-Tag>
</Alien-RFID-Tag-List>
```

어떤 경우든 다음 정보를 태그 별로 보고한다.

- TagID: 태그의 개별 ID (EPC 코드).
- Disc: 현재 세션에서 태그를 리더가 처음 읽은 시간
- Last: 현재 세션에서 태그를 리더가 마지막으로 읽은 시간
- Count/ReadCount: 현재 세션에서 태그를 읽은 횟수
- Ant: 태그를 마지막으로 본 안테나 포트의 번호 다중 정적 리더의 경우, 보고된 안테나 포트는 송신 안테나이다.

TagListFormat Examples	
Command Response	>TagListFormat? TagListFormat = Text
Command Response	>TagListFormat = XML TagListFormat = XML

TagListCustomFormat

F800 | 9900 | 9680 | 9650

TagListCustomFormat 명령어를 사용하면 사용자 정의 TagList를 정의할 수 있다. 일단 형식을 정의했으면, 명령어 "`TagListFormat = Custom`"를 전송하여 적용하면 된다.

TagListCustomFormat 명령어는 각 태그를 화면 상에 나타낼 방법을 정의하는 단순 텍스트 행 인수를 취한다. 이 인수는 아래 표에 정의된 대로 텍스트와 표시를 조합한 형태이다. 사용자 정의 형식의 최대 길이는 255 문자이다.

사용자 정의 표시는 길고 짧은 형식으로 가능하다. 일반적인 (짧은) 표시는 %X의 형태이며, 여기서 "X"는 표시를 식별하기 위한 단일 문자이다. 긴 표시는 ${var_name}의 형태를 취한다. 짧고 긴 표시를 상호 간에 사용할 수 있다.

RFID 리더가 Taglist를 생성해야 할 경우, 사용자 정의 형식의 표시와 변수들을 각 태그 별로 실제 값으로 대체한다.

다음 표시는 모든 Alien 리더에 적용된다:

짧은표현	긴표현	설명
%i	**${TAGIDW}**	바이트 쌍 별로 공간이 있는 태그 ID: 8000 00FE 8010 2AB7
-	**${TAGIDB}**	바이트 간에 공간이 있는 태그 ID: 80 00 00 FE 80 10 2A B7
%k	**${TAGID}**	공간이 전혀 없는 태그 ID: 800000FE80102AB7
%d	**${DATE1}**	태그 발견 일자, 형식은 YY/MM/DD
%t	**${TIME1}**	태그 발견 시간, 형식은 hh:mm:ss
-	**${MSEC1}**	태그 발견 일자와 시간, 밀리세컨드 단위이며 1/1/1970 이후로 계산.
%D	**${DATE2}**	태그 최근 확인 일자, 형식은 YY/MM/DD
%T	**${TIME2}**	태그 최근 확인 시간, 형식은 hh:mm:ss
-	**${MSEC2}**	태그 최근 확인 일자와 시간, 밀리세컨드 단위이며 1/1/1970 이후로 계산.
%r	**${COUNT}**	태그 계수, 즉 태그를 읽은 횟수
%a	**${TX}**	(송신) 태그를 마지막 본 안테나
%A	**${RX}**	(수신) 태그를 마지막 본 안테나
%p	**${PROTO#}**	태그 프로토콜을 표시하는 정수값 (0 = Class0, 1 = Class1/Gen1, 2 = Class2/Gen2)
%P	**${PROTO}**	태그 프로토콜을 표시하는 문자열
%l	**${PCWORD}**	PC Word (Class1/Gen2)
%m	**${RSSI}**	태그 RSSI 측정값
-	**${RSSI_MAX}**	이 태그의 RSSI 최대값
%N	**${NAME}**	리더의 ReaderName
%H	**${HOST}**	리더의 HostName
%I	**${IP}**	리더의 IP 주소
%M	**${MAC}**	리더의 MAC 주소

다음 표시는 ALR-9000+ 리더 모델부터 지원된다.

짧은표현	긴표현	설명
%s	**${SPEED}**	태그 속도(m/s) - 읽기 성능에 영향을 미침
-	**${SPEED_MIN}**	태그 별로 측정한 최소(most negative) 속도값
-	**${SPEED_MAX}**	태그 별로 측정한 최대(most positive) 속도값
-	**${SPEED_TOP}**	태그 별 최고속도(방향에 관계 없음)
-	**${DIR}**	방향: "-" (접근), "+" (후향), 또는 "0" (정주)
-	**${G2DATA1...4}**	1차, 2차, 3차, 4차 AcqG2TagData 항목에 해당되는 태그 데이터
-	**${G2OPS}**	실행된 모든 AcqG2Ops의 결과값, 항목은 수직 막대(|)로 분리 함.
-	**${G2OPS1...8}**	번호가 부여된 8개 AcqG2Ops 중 하나에 해당하는 결과값
-	**${AUTH}**	태그의 Alien 동적 인증 데이터(태그 제조자의 id 코드 포함). 더 자세한 정보는 `TagAuth` 명령어 설명 섹션 참조.
-	**${XPC}**	XPC 단어(Class1/Gen2)
-	**${FREQ}**	태그를 취하는 빈도(MHz)

다음 표시는 ALR-F800 리더 모델부터 지원된다.

짧은표현	긴표현	설명
-	**${G2DATA1...4}**	1차, 2차, 3차, 4차 AcqG2TagData 항목에 해당되는 태그 데이터
-	**${G2OPS}**	실행된 모든 AcqG2Ops의 결과값, 항목은 수직 막대(|)로 분리 함.
-	**${G2OPS1...8}**	번호가 부여된 8개 AcqG2Ops 중 하나에 해당하는 결과값
-	**${XPC}**	XPC 단어(Class1/Gen2)
-	**${FREQ}**	태그를 취하는 빈도(MHz)

TagListCustomFormat 예제

```
Command     >TagListCustomFormat = Here is a tag %i
Response    TagListFormat = Here is a tag %i

Command     >get TagList
Response    Here is a tag 8000 0000 0000 0808
            Here is a tag 102F ED3D 0303 0001
```

```
Command     >TagListCustomFormat = Tag %k, read %r times from antenna %a
Response    TagListFormat = Tag %k, read %r times from antenna %a

Command     >get TagList
Response    Tag 8000000000000808, read 3 times from antenna 0
            Tag 102FED3D03030001, read 120 times from antenna 1
```

```
Command     >TagListCustomFormat = Tag ${TAGIDW} was going ${SPEED} m/s.
Response    TagListFormat = Tag ${TAGIDW} was going ${SPEED} m/s.

Command     >t
Response    Tag 8000 0000 0000 0808 was going 1.064 m/s.
            Tag 102F ED3D 0303 0001 was going -0.003 m/s.
```

XPC 기능 지원
F800 | 9900 | 9680 | 9650
EPC Gen2 태그와 ISO18000-6c 태그에서 반환된 정보에는 프로토콜 제어 정보 단어(16비트)가 포함되어 있다(PC 단어). 이 정보에는 패킷 길이, 태그의 추가 정보가 포함되어 있는데, 이를테면 '유저 뱅크'에 작성된 정보가 있는지, ISO 번호 부여 시스템 식별자, 유해 자료 태깅 비트 등의 EPC 정보 비트가 포함된다. 또한 일부 Gen2 태그에는 XPC(eXtended Protocol Control)라 통칭하는 정보가 한 두 단어로 포함되어 있기도 하다. 이를 정의하는 것은 배터리 태그를 식별하고, 주의를 요하는 태그의 예외적 조건을 표시하려는 목적이다(예를 들면, 온도 태그에 온도가 너무 높은 경우 등).

PC와 XPC 비트에서 이용 가능한 정보는 EPC와 ISO 표준 위원회가 표준화한다. 이러한 비트의 예상 용도와 의미는 향후 변경될 수 있으며, 다른 XPC 단어들이 정의될 수도 있다.

Alien 리더의 소프트웨어는 ${XPC}와 ${PCWORD} 사용자정의 태그 목록 형식의 표시를 사용하여 사용자 정의 태그 목록에서 PC 및 XPC 단어 비트를 디코딩하고 제시한다.

예제:

```
Alien>TagListCustomFormat = id:%i pc:${PCWORD} xpc:${XPC}
TagListCustomFormat = id:%i pc:${PCWORD} xpc:${XPC}

Alien>TagListFormat = custom
TagListFormat = custom

Alien>t
id:E9C1 E9C2 E9C3 E9C4 E9C5 E9C6 pc:3A00 xpc:0002
```

보고된 PC 단어가 **0x3A00** 값을 가지며 이는 7단어로 된 EPC 길이에 해당된다는 점에 유의해야 한다. 이 태그가 여분의 XPC 단어를 포함하므로, PC 단어 값을 보고할 때 6단어로 된 EPC 길이에 1을 더한다.
EPC 뱅크를 G2Read 명령어로 직접 읽을 경우, PC 단어 위치에 있는 비트는(EPC 뱅크에서 두 번째 단어) **0x3200**값을 갖는다. ('**2**'는 XPC 단어가 제시되었음을 의미한다):

```
Alien>g2read=1 0 0
G2Read = 14 1D 32 00 E9 C1 E9 C2 E9 C3 E9 C4 E9 C5 E9 C6
```

TagDataFormatGroupSize

F800 | 9900 | 9680 | 9650
새로운 TagDataFormatGroupSize 명령어를 사용하면 태그 데이터 형식을 정할 때 바이트 그룹을 지정할 수 있다. 이 매개변수는 태그의 EPC 형식과 텍스트, XML, 개요 형식에 해당하는 AcqG2TagData 값에 영향을 미친다. ProgramEPC, G2Read 등의 프로그래밍 관련 명령어 형식에는 영향을 주지 않는다.

형식은 다음과 같다:

```
TagDataFormatGroupSize = <group size>
```

여기서 <group size>는 그룹 내 바이트 수를 의미한다(0 ~ 2). 기본값은 2이다.

이 기능은 사용자가 자신의 고유 데이터 데이터 파서(parser)를 설계할 때 유용하다.

TagDataFormatGroupSize 예제	
Command Response	>TagDataFormatGroupSize? TagDataFormatGroupSize = 2
Command Response	>AcqG2TagData = 1 2 2 AcqG2TagData = 1 2 2
Command Response	>t Tag:E200 9002 5214 0283 1740 60EB, Disc:2010/02/22 16:47:31.272, Last:2010/02/22 16:47:31.272, Count:1, Ant:0, Proto:2, D1:**E200 9002**
Command Response	>TagDataFormatGroupSize = 1 TagDataFormatGroupSize = 1
Command Response	Alien>t Tag:E2 00 90 02 52 14 02 83 17 40 60 EB, Disc:2010/02/22 16:48:17.257, Last:2010/02/22 16:48:17.257, Count:1, Ant:0, Proto:2, D1:**E2 00 90 02**
Command Response	>TagDataFormatGroupSize = 0 TagDataFormatGroupSize = 0
Command Response	Alien>t Tag:E20090025214028317406OEB, Disc:2010/02/22 16:44:52.878, Last:2010/02/22 16:44:52.878, Count:1, Ant:0, Proto:2, D1:**E2009002**

TagListAntennaCombine

F800 | 9900 | 9680 | 9650

TagListAntennaCombine 명령어는 안테나 조합 모드를 켜거나 끌 수 있다. TagListAntennaCombine이 켜져 있으면, 리더는 ID들을 다른 안테나들에서 읽더라도 단일의 TagList에 태그 ID들을 조합한다. 이 값을 OFF로 설정하면 TagList는 각 안테나에서 태그 ID의 여러 사본을 유지하도록 지시를 받는다.

예를 들어, 안테나0과 1 모두에서 보이는 태그를 읽을 때 AntennaSequence가 "0, 1"이라면 다음의 TagList가 생성된다.

TagListAntennaCombine = ON
```
Tag:8000 8004 2665 8426, Count:2, Ant:1
```

TagListAntennaCombine = OFF
```
Tag:8000 8004 2665 8426, Count:1, Ant:0
Tag:8000 8004 2665 8426, Count:1, Ant:1
```

- 허용된 값: "ON" | "OFF"
- 기본값: "ON"

TagListAntennaCombine 예제	
Command Response	>TagListAntennaCombine? TagListAntennaCombine = ON
Command Response	>TagListAntennaCombine = off TagListAntennaCombine = OFF

TagListMillis

F800 | 9900 | 9680 | 9650

표준형 TagList 데이터에는 초 단위 감도의 타임스탬프를 포함하나, 리더는 사실 내부적으로 밀리세컨드 단위 감도의 타임스탬프를 기록한다. TagListMillis 명령어를 사용하면 TagList 타임스탬프 내에 밀리세컨드 데이터를 표시할 수 있다.

- 허용된 값: "ON" | "OFF"
- 기본값: "ON"
- TagListMillis 를 "On"으로 설정하면, TagList 타임스탬프의 형식이 다음과 같다.
  YYYY/MM/DD HH:MM:SS.mmm
- TagListMillis 를 "Off"로 설정하면, TagList 타임스탬프의 형식이 다음과 같다.
  YYYY/MM/DD HH:MM:SS

TagListMillis 예제	
Command Response	>TagListMillis? TagListMillis = On
Command Response	>TagList? Tag:DEAD BEEF CAFE 8042, Disc:2006/05/30 12:11:57.388, Last:2006/05/30 12:11:57.388, Count:1, Ant:0, Proto:1
Command Response	>TagListMillis = Off TagListMillis = Off
Command Response	>TagList? Tag:DEAD BEEF CAFE 8042, Disc:2006/05/30 12:12:02, Last:2006/05/30 12:12:02, Count:1, Ant:0, Proto:1

Clear TagList

F800 | 9900 | 9680 | 9650

"Clear TagList" 명령어는 리더가 내부 TagList를 즉시 지우도록 지시한다.

Clear TagList 예제	
Command Response	>Clear TagList TagList has been cleared!

TagStreamMode

F800 | 9900 | 9680 | 9650

TagStreamMode 명령어는 TagStream 기능을 켜거나 끌 수 있다. 태그 데이터를 계속 불러오게 하거나 시간 지연을 야기하는 NotifyMode 기능들에 의존할 필요 없이, 리더가 태그 읽기 이벤트를 TCP 소켓이나 리더의 직렬포트로 스트리밍되도록 구성할 수 있다.

TagStreamMode가 켜져 있으면, 태그를 읽을 때마다 리더는 태그 정보를 TagStreamAddress로 스트리밍한다(아래 참조). TagList에서처럼 데이터 버퍼링이나 여러 안테나로부터 데이터를 조합하지는 않는다. 단일 인벤토리가 동일한 태그를 세 번 발견할 경우, 세 개의 태그 읽기 이벤트를 스트리밍한다.

스트리밍한 데이터의 형식은 TagStreamFormat와 TagStreamCustomFormat 명령어로 지정한다(아래).

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"
- 리더가 데이터 스트리밍을 제대로 하지 못하면(TCP 소켓 오류 등), **스트리밍 데이터를 잃게 된다**.

TagStreamMode 예제	
Command Response	>TagStreamMode? TagStreamMode = OFF
Command Response	>TagStreamMode = on TagStreamMode = ON

TagStreamAddress

F800 | 9900 | 9680 | 9650

TagStreamAddress 명령어는 TagStream 이벤트 데이터를 어디로 전송할 것인지를 지정한다. 지원되는 전송 방법이 두 가지 있는데, 아래 표에 제시하였다.

TagStreamAddress	설명
hostname:port	메시지를 네트워크에 있는 기기 상의 지정된 포트로 전송한다. 주소는 "hostname:port"의 형식을 취한다. 예를 들어, "123.01.02.98:3450" 또는 "listener.alientechnology.com:10002"
SERIAL	메시지를 직렬포트로 전송한다. "serial"이라는 단어를 주소에 사용한다. 단어의 대소문자 구별은 없다. ALR-F800: 이 경우 메시지를 SERIAL0 (RS-232)와 SERIAL1 (USB) 포트 모두에 전송한다.
(ALR-F800 only) **SERIAL0** **SERIAL1**	ALR-F800의 직렬포트 중 하나로 메시지를 전송한다. SERIAL0는 RS-232 포트로 메시지를 보낸다. SERIAL1은 USB 포트로 보낸다.

- 유의사항: 직렬포트로 데이터 스트리밍할 때 주의를 요한다. Tag 읽기 이벤트는 매우 빠르게 발생할 수 있으며, 동시에 여러 명령어를 전송하면 직렬 포트의 반응성에 좋지 않은 영향을 미칠 수 있다.

TagStreamAddress 예제	
Command Response	>TagStreamAddress? TagStreamAddress = 10.1.0.12:4001
Command Response	>TagStreamAddress = serial TagStreamAddress = SERIAL

TagStreamFormat

F800 | 9900 | 9680 | 9650

TagStreamFormat 명령어는 사용자가 TagStream 데이터(TagList와는 구분됨)의 형식을 지정할 수 있게 한다. 이 명령어는 텍스트 문자열을 인수로 취하며, 다음 예들과 같다.

형식	설명
Text	일반 텍스트 메시지로 표시되는 TagStream 데이터, 행마다 태그 이벤트가 하나씩.
Terse	일반 텍스트 메시지로 표시하는 TagStream 데이터, 행마다 태그 이벤트가 하나씩. 단 단순화된 출력값으로, TagID, 안테나, ReadCount만 표시하며 레이블은 없음.
XML	XML 텍스트 형식으로 표시하는 TagStream 데이터
Custom	TagStream 데이터는 TagStreamCustomFormat가 제시한 형식으로 표시된다.

- 기본값은 "Terse"이다.
- TagStream 데이터의 형식은 TagList 데이터와 동일하다. 출력 형식과 각 유형별 예는 TagListFormat 설명 섹션을 참조할 것.

TagStreamFormat 예제	
Command Response	`>TagStreamFormat?` `IOStreamFormat = Terse`
Command Response	`>TagStreamFormat = Text` `TagStreamFormat = Text`

TagStreamCustomFormat

F800 | 9900 | 9680 | 9650

TagStreamCustomFormat 명령어를 사용하면 사용자 정의 TagStream 형식을 정의할 수 있다. 일단 형식을 정의했으면, 명령어 "`TagStreamFormat = Custom`"를 전송하여 적용하면 된다.

TagStreamCustomFormat 명령어는 각 태그 이벤트를 나타낼 방법을 정의하는 단순 텍스트 행 인수를 취한다. 이 인수는 아래 표에 정의된 대로 텍스트와 사용자 정의 표시를 조합한 형태이다. TagListCustomFormat 정의의 최대 길이는 255 문자이다.

리더가 사용자정의 TagStream 이벤트를 출력하면, 사용자 정의 형식의 표시와 변수들을 태그 이벤트 별로 실제 값으로 대체한다.

- TagStream에서 사용 가능한 사용자 정의 표시는 TagList와 동일하다. 허용되는 표시 목록에 대해서는 TagListCustomFormat 설명을 참조할 것.
- TagStreamCustomFormat 기본값은 "%k"이다.

TagStreamCustomFormat 예제	
Command Response	`>TagStreamCustomFormat?` `TagStreamCustomFormat = %k`
Command Response	`>TagStreamCustomFormat = Tag:%i Ant:%a` `TagStreamCustomFormat = Tag:%i Ant:%a`

TagStreamKeepAliveTime

F800 | 9900 | 9680 | 9650

리더가 TagStream 이벤트 메시지를 네트워크 상에서 전송하려면 TCP 소켓을 열어야 한다. 소켓을 열고 닫으면 일부 프로세서와 네트워크 오버헤드가 필요하며, 메시지가 리더를 빠르게 빠져 나오는 경우에는 오버헤드가 짐이 되어 리더의 반응성을 늦춘다.

TagStreamKeepAliveTime 명령어는 리더가 TagStream 소켓을 열어두는 시간을 설정한다(초 단위). TagStream 이벤트의 신속성을 위해서는 리더가 소켓을 계속 열어두는 것이 유리할 것이다. 이를 위해서는 TagStreamKeepAliveTime을 이벤트 간에 예상되는 시간 보다 더 큰 수로 설정하면 된다.

한편 소켓을 열어두면 네트워크에 부담이 된다. TagStream 이벤트가 자주 있다면, TagStreamKeepAliveTime을 적게 하여 소켓이 하나의 메시지를 발송하는데 충분한 길이만큼만 열어두고 메시지 전송 후에 자동으로 닫게 하는 것이 좋다.

- 허용된 값: 정수(0..32767), 기본값은 30
- 리더는 소켓을 자동으로 다시 열고, 데이터를 다음 번에 전송해야 할 때까지 일시 중단 시간이 있다.

TagStreamKeepAliveTime 예제	
Command Response	>TagStreamKeepAliveTime? TagStreamKeepAliveTime = 30
Command Response	>TagStreamKeepAliveTime = 2 TagStreamKeepAliveTime = 2

StreamHeader

F800 | 9900 | 9680 | 9650

리더가 IOStream 또는 TagStream의 TCP 소켓을 열면, 일부 헤더 정보를 출력하기도 하는데, 이로써 수신하는 애플리케이션에 대하여 리더를 식별해 주고 스트림이 TagStream인지 IOStream인지도 확인해 준다.

- 허용값: “ON” | “OFF”. 기본값은 “ON”
- StreamHeader 명령어는 IOStream과 TagStream 모두에 적용된다.
- TagStream 헤더는 첫 번째 행만 제외하고 IOStream 헤더와 동일하다.
- StreamHeader는 소켓 연결 초기에만 전송된다. 소켓이 열려 있는 한, 추가 헤더는 전송되지 않는다. 소켓을 닫고 다시 열면, 새로운 StreamHeader가 전송된다.

- StreamHeader는 직렬포트로 스트리밍할 때 포함된다.

다음은 IOStream 초기부터의 헤더 예제이다. 이 형식은 텍스트, 개요, 사용자 정의 IOStreamFormats에서 사용된다. XML 스트림 헤더에는 동일한 데이터가 XML 태그 속에 담겨 포함된다.

```
#Alien RFID Reader I/O Stream\r\n
#ReaderName: Alien RFID Reader\r\n
#Hostname: alien-10116A\r\n
#IPAddress: 192.168.1.100\r\n
#CommandPort: 23\r\n
#MACAddress: 00:80:66:10:11:6A\r\n
#Time: 2006/12/18 11:19:26\r\n\0
```

StreamHeader 예제	
Command Response	>StreamHeader? StreamHeader = ON
Command Response	>StreamHeader = off StreamHeader = OFF

TagStreamServer

F800 | 9900 | 9680 | 9650

TagStreamServer는 태그 데이터를 사용자에게 스트리밍하는 기능이나, 리더가 TCP를 사용자에 연결해 주는 대신 (TagStreamMode 참조), 사용자가 리더를 연결하여 스트리밍한 데이터를 수신하는 방식이다. 사용자가 TagStreamServer에 연결되어 있는 한 리더가 보는 태그를 모두 볼 수 있다.

TagStreamServer에는 여러 동시적 접속들을 허용하는 추가적인 이점이 있으며, 때문에 사용자의 여러 프로세스가 동시에 리더의 태그 판독을 모니터링할 수 있다.

TagStreamServer 명령어의 형식은 다음과 같다.

```
TagStreamServer = <maxConnections> [<port> [<scope>]]
```

여기서:

`<maxConnections>` 는 허용되는 동시 접속 수를 의미한다 (0 ~16)
`<port>` (선택) 사용자가 TagStreamServer 에 연결되어 있는 TCP 포트(기본값은 3333)
`<scope>` (선택) TagStreamServer가 "아무" 출처로부터든(기본값) 접속을 승인할 것인지, 아니면 리더 상에서 직접 실행되는 사용자 정의 Ruby 애플리케이션 등의 "로컬" 출처와 연결할 것인지를 지정한다.

- TagStreamServer는 TagStreamFormat에 기술한 형식을 사용한다. StreamHeader는 전송되지 않는다.
- TagStreamServer를 설정할 때, `<scope>` 또는 `<port>`와 `<scope>` 둘 다 생략 가능하다. 리더는 사용자가 디 둘을 생략할 때 현재값을 사용할 것이다.
- `<maxConnections>` 는 상시 필요하며, 사용자는 `<maxConnections>` 를 0 으로 설정하여 TagStreamServer를 끌 수 있다.
- 명령어 “TagStreamServer #?”는 현재 TagStreamServer에 접속된 수를 보고한다.

TagStreamServer 예제	
Command Response	>TagStreamServer? TagStreamServer = 0 3333 any
Command Response	// Allow 1 connection, on port 4444, from only local sources >TagStreamServer = 1 4444 local TagStreamServer = 1 4444 local
Command Response	// Update current setting to allow 3 connections >TagStreamServer = 3 TagStreamServer = 3 4444 local
Command Response	// Turn off TagStreamServer >TagStreamServer = 0 TagStreamServer = 0 4444 local
Command Response	// Ask for the number of current connections >TagStreamServer #? TagStreamServer = 1

매크로(Macro) 명령어

리더는 보관된 "매크로" 사용을 지원하는데, 매크로는 하나의 리더 명령어로 실행되는 명령어 그룹 별로 명명한다. 매크로는 a .arm (Alien Reader Macro) 확장자의 텍스트 파일로, 리더 명령어, 비어있는 행들, 주석 행들("#" 또는 "//"로 시작)을 포함할 수 있다.

- 매크로 이름은 매크로의 파일명을 바탕으로 하며(- .arm 확장자), 따라서 구체적인 제한조건을 준수해야 한다. 문자 숫자식(a-z, A-Z, 0-9)만 허용된다.
- 매크로 이름은 대소문자를 구별하며, "myMacro"는 "mymacro"와 구분한다.
- 리더를 시동한 후에는 "default"라는 이름의 보관된 매크로들은 무엇이든 자동으로 실행한다. 리더는 "default" 매크로 내에 있는 "재부팅" 명령어는 무시한다.

예를 들어, "myMacro"라는 이름의 매크로 정의는 다음과 같다.

```
# myMacro.arm
# I made it sometime in January

# Start out at the beginning
FactorySettings

# Set the readername
ReaderName = MyTestReader

# Set up AutoMode
AutoStartTrigger = 1 2
AutoMode = on
```

매크로는 외부에서 생성 가능하며, 그 후 웹 인터페이스를 통해 리더로 업로드할 수 있다. 또는 유용한 매크로 레코딩 기능을 사용해 리더의 명령어 라인 인터페이스에서 곧바로 생성할 수도 있다. ALR-F800은 웹 인터페이스에서 강력한 매크로 관리 기능들을 제공하는데, 그 중에는 사용자의 매크로를 보고 편집할 수 있는 기능도 있다.

일단 매크로를 리더에 로딩하면, 재부팅하거나 리더를 업그레이드해도 지속된다. 매크로는 새 매크로 명령어들을 사용해서 리더에서 나열, 보기, 실행, 제거할 수도 있다. 매크로가 실행되면, 리더는 사용자가 명령어 라인에서 입력하는 것처럼 매크로 파일의 각 행을 실행한다.

MacroList

F800 | 9900 | 9680 | 9650

사용자는 MacroList 명령어를 사용하여 현재 리더 내 있는 모든 매크로의 목록을 볼 수 있으며, 행 별로 매크로 이름이 하나씩 제시된다.

- 설치한 매크로가 없는 경우에는 리더가 "(No Macros Installed)" 메시지로 응답한다.
- 일례로, "example1"과 "example2"라는 두 개의 매크로가 이미 사용자의 리더에서 검색되어 차후에 업데이트할 수 있다고 하자. 이처럼 동일한 이름으로 자신의 매크로를 만든 다음 이후에 업그레이드 중에 교체할 수 있는 것이다.

MacroList Examples	
Command Response	>MacroList example1 example2
Command Response	>MacroList (No Macros Installed)

MacroView

F800 | 9900 | 9680 | 9650

MacroView 명령어를 사용하고, 그 다음에 설치한 매크로의 이름을 입력하여 매크로의 내용을 볼 수 있다.

- 지정한 매크로 이름이 설치한 매크로와 일치하지 않을 경우, 리더는 다음과 같이 응답한다. "오류 28: 알 수 없는 오류. 해당 이름을 가진 매크로를 읽을 수 없음."

MacroView 예제

Command
Response
	>MacroView example1
	// Example 1 Macro
	
	// Look for all types, small number of tags expected
	RFModulation = STD
	TagType = 31
	AcquireMode = Global Scroll
	
	// We'll want to see a notification when new tags are seen
	NotifyAddress = 10.10.82.50:3600
	NotifyTrigger = Add
	
	// Set up AutoMode to run very quickly
	AutoModeReset
	PersistTime = 10
	AutoStopTimer = 100
	
	// Go!
	NotifyMode = On
	AutoMode = On

Command Response	>MacroView foo Error 28: Unknown error. There is no readable macro with that name.

MacroDel

F800 | 9900 | 9680 | 9650

MacroDel 뒤에 설치한 매크로의 이름을 입력하면 해당 매크로를 영구 삭제한다.

- 경고나 추가 확인 메시지는 없다.
- 지정한 매크로 이름이 설치한 매크로와 일치하지 않을 경우, 리더는 다음과 같이 응답한다. "오류 28: 알 수 없는 오류. 삭제할 수 없음. 해당 이름을 가진 매크로가 없음."

MacroDel 예제

Command Response	>MacroDel example1 Macro deleted!
Command Response	>MacroDel foo Error 28: Unknown error. Unable to delete - there is no macro with that name.

MacroDelAll

F800 | 9900 | 9680 | 9650

MacroDelAll은 설치한 모든 매크로를 삭제한다.

- 경고나 추가 확인 메시지는 없다.

MacroDelAll 예제	
Command Response	>MacroDelAll All macros deleted!

MacroRun

F800 | 9900 | 9680 | 9650

MacroRun 뒤에 설치한 매크로의 이름을 입력하면 해당 매크로에 속한 명령어들을 실행한다. 매크로가 실행되면, 리더는 관련 .arm 파일을 스캔하고 포함된 항목들을 실행한다(비어있는 행과 "#" or "//"로 시작하는 행은 생략).

사용자는 매크로 내 명령어 목록이 임의의 구성에서 원하는 상태로 리더를 설정하기에 충분한지 확인해야 한다. 예를 들어, 다른 AutoMode 속성을 우선 설정하지 않고 "AutoMode = On" 명령어만 포함시키는 것만으로는 충분하지 않을 것이다.

- 리더가 매크로의 각 행을 실행할 때, 전달 중인 명령어가 표시되나, 각 명령어에 대한 응답은 표시되지 않는다.
- 매크로를 성공적으로 실행했으면, 리더는 곧바로 "Macro Run Successfully"라는 메시지로 응답한다.
- 지정한 매크로 이름이 설치한 매크로와 일치하지 않을 경우, 리더는 다음과 같이 응답한다. "오류 28: 알 수 없는 오류. 해당 이름을 가진 매크로를 읽을 수 없음."
- 매크로 내 명령어 중 오류를 발생시키는 것이 있으면, 리더는 설명하는 메시지를 내보내고 매크로의 추가 실행을 중단한다. 오류 응답에는 문제가 되는 행의 내용도 포함된다.
- 리더를 시동한 후에는 "default"라는 이름의 보관된 매크로들은 무엇이든 자동으로 실행한다. 리더는 "default" 매크로 내에 있는 "재부팅" 명령어는 무시한다.

MacroRun 예제	
Command Response	>MacroRun example1 RFModulation = STD TagType = 31 AcquireMode = Global Scroll NotifyAddress = 10.10.82.50:3600 NotifyTrigger = Add AutoModeReset PersistTime = 10 AutoStopTimer = 100 NotifyMode = On AutoMode = On Macro Run Successfully
Command Response	>MacroRun macro1 ExternalOutput = foo Error 28: Unknown error. Macro error handling line: ExternalOutput = foo
Command Response	>MacroRun foo Error 28: Unknown error. There is no readable macro with that name.

ALR-F800 리더군의 경우, MacroRun 명령어에 추가 인수를 첨가할 수 있으며, $1, $2 등이 포함되어 있을 때마다 이 문자열 인수가 매크로 내에서 대체될 것이다. 이 기능을 사용하면 매크로가 실행될 때 조율 및 사용자 정의가 가능한 일반 매크로를 작성할 수 있다.

예를 들어, TagStreamAddress와 NotifyAddress를 구성하는 매크로를 가지고 있더라도, 나중에는 IP 주소나 TCP 포트를 변경하고자 할 수 있다. 사용자가 매크로를 변경하고 사용자의 리더에 새로 업로드 할 필요 없이, 자신의 리더에 일반 매크로를 생성할 수 있는 것이다. 이때 자신의 PC의 실제 IP 주소와 별도의 스트림에 대하여 $1, $2, $3 플레이스 홀더를 지정하고, 포트에 알린 다음 매크로 명령어로 실제 값을 제시한다.

예를 들어 다음의 리더 명령어를 실행한다고 해 보자.

```
MacroRun config 10.10.82.50 2222 3333
```

이 “config” 매크로는 이미 로딩한 상태이다.

```
NotifyAddress    = $1:$2
TagStreamAddress = $1:$3
```

이 결과 변수를 대체한 후에 다음 명령어가 전송된다.

```
NotifyAddress    = 10.10.82.50:2222
TagStreamAddress = 10.10.82.50:3333
```

MacroStartRec
MacroStopRec

F800 | 9900 | 9680 | 9650

MacroStartRec에 이어 매크로 이름을 붙이면 제시한 이름에 근거하여 새로운 매크로 파일이 열리고, 사용자가 입력한 대체 명령어들 전부를 이 매크로 파일에 작성한다. 명령어들은 사용자가 MacroStopRec 또는 다른 MacroStartRec 명령어를 전송할 때까지 기록된다. 새로 생성한 매크로는 이제 사용자가 외부에서 생성하였거나 업로드한 것처럼 사용 가능한 상태가 된다.

MacroStartRec 명령어에 대한 수락 응답이 진행 방법에 대한 간략한 지침과 함께 제시된다.

- 매크로를 기록(recording)하면 동일한 이름의 다른 매크로를 대체한다.
- 기록하는 동안, "Alien>" 프롬프트가 "Alien>R>"로 바뀌어 사용자가 기록 모드에 있음을 알린다.
- 매크로 이름은 a-z, A-Z, 0-9 등의 문자 숫자 조합식으로 제한된다.
- 다른 매크로 내에서는 매크로를 실행할 수 없다.

MacroStartRec & MacroStopRec 예제

Command Response Command Response	>MacroStartRec recordedMacro1 MacroStartRec = Recording a new macro. Enter commands, then use MacroStopRec to stop. Alien>R>ReaderName = Recorded Macro 1 ReaderName = Recorded Macro 1
Command Response Command Response Command Response	Alien>R>MacroStopRec MacroStopRec = Success! Alien>MacroList example1 example2 recordedMacro1 Alien>MacroView recordedMacro1 ReaderName = Recorded Macro 1

MacroCopy

F800 | 9900 | 9680 | 9650

MacroCopy는 기존 매크로의 사본을 만든다. 원래 매크로의 이름을 명기하고 그 다음에 새로운 사본의 이름을 입력한다. 이렇게 하면 새로운 이름으로 기존의 매크로를 덮어쓰기 한다.

MacroCopy 예제

```
Command     >MacroList
Response    example1
            example2
            example3

Command     >MacroCopy example1 myNewMacro
Response    Macro copied!

Command     >MacroList
Response    example1
            example2
            example3
            myNewMacro
```

수집(Acquire) 명령어

수집(Acquire) 명령어를 사용하면 특정 태그군을 가장 잘 판독하기 위해 다양한 프로토콜을 활용하는 방법과 관계된 리더 매개변수들을 구성할 수 있다.

AcquireMode

F800 | 9900 | 9680 | 9650

리더가 태그를 읽도록 호출되면 현재 AcquireMode를 사용하여 이를 수행한다. 현재 허용되는 모드는 다음과 같다.

AcquireMode	설명
Inventory	여러 태그의 전체 인벤토리를 수행한다(권장)
Global Scroll	한 개의 태그를 빠르게 검색한다(권장하지 않음)

기본설정은 '인벤토리(Inventory)'이다. 여러 모드에 대한 자세한 설명을 원한다면, 앞 섹션의 "태그 기본요소(Tag Fundamentals)"를 참조하기 바란다. 글로벌 스크롤은 일반적으로 권장하지 않는다.

INVENTORY

인벤토리 수집 모드는 리더의 시야 내에서 태그의 충돌방지 검색 기능을 수행한다. 이 방법은 동시에 리더 앞에 있는 여러 태그들의 위치를 파악하고 구별한다.

GLOBAL SCROLL

글로벌 스크롤 수집 모드는 리더가 한 개의 태그를 반복해서 읽도록 지시한다. 이는 매우 빠른 태그 판독 방법으로서, 어느 시점에 태그가 리더 범위 내에 있을 것으로 예상될 때, 이를테면 컨베이어 벨트 애플리케이션 내에 있을 것으로 보일 때 가장 유용한 방법이다. 이러한 상황에서는 단일 태그 읽기 성능이 인벤토리 모드에서 전체 태그 검색을 반복해서 할 때보다 현저하게 빨라진다.

유의사항: 이 모드를 사용할 때 여러 태그가 리더 범위 내에 있을 경우, 리더는 읽고 보고할 태그 하나만 선택하거나(보통은 “가장 강하거나’” “가장 소리가 큰” 것) 아무 태그도 읽지 않는다.

AcquireMode 예제	
Command Response	>AcquireMode = Global Scroll AcquireMode = Global Scroll
Command Response	>AcquireMode = Inventory AcquireMode = Inventory

TagType

F800 | 9900 | 9680 | 9650

현재 Alien RFID 리더 장치는 모두 EPCglobal Class 1 / Gen 2 프로토콜을 지원하며, 허용된 유일한 TagType 값은 16.

TagType Examples	
Command Response	>TagType = 16 TagType = 16
Command Response	>TagType? TagType = 16

AcqG2Cycles

F800 | 9900 | 9680 | 9650

AcqG2Cycles는 1에서 255 사이에서 하나의 정수 매개변수만 취한다. Class1 / Gen2 태그를 리더가 스캔할 때마다 수행하는 수집 주기의 횟수를 의미한다.

유의사항: 이 속성의 최대값은 255(다른 수집 설정값과 마찬가지임)이며, 높은 값으로 설정할 수록 수집 시간이 길어져서 리더가 응답하지 않는 것처럼 보일 수 있다. 예를 들어, AcqG2Cycles와 AcqG2Count를 둘 다 255로 설정하면 리더는 Class1 / Gen2 태그를 찾도록 지시받았을 때 65,000 회 이상 수집을 수행할 것이다.

AcqG2Cycles 매개변수는 Class1 / Gen2 수집 주기의 "외부 루프"를 제어하는데 이는 "태그 읽기 기본요소" 섹션에서 자세히 다룬다. 이 값은 리더가 태그에 대하여 스캔을 수행할 때마다 걸리는 시간에 큰 영향을 끼친다.

- 허용된 값: Integer(0..255)
- 기본값: 1
- 이 매개변수가 어떻게 태그 인벤토리에 영향을 미치는지에 관하여는 3장에서 더 많은 정보를 찾을 수 있다.

AcqG2Cycles 예제	
Command Response	>AcqG2Cycles? AcqG2Cycles = 1
Command Response	>AcqG2Cycles = 2 AcqG2Cycles = 2

AcqG2Count

F800 | 9900 | 9680 | 9650

AcqG2Count는 1에서 255 사이에서 하나의 정수 매개변수만 취한다. Class1 / Gen2 수집 주기마다 수행하는 읽기 횟수를 의미한다.

예를 들어, AcqG2Count가 10으로 설정되어 있으면, 각 수집 주기 마다 수집 명령어가 10회 전달된다.

- 허용된 값: Integer (0..255)
- 기본값: 1
- 이 매개변수가 어떻게 태그 인벤토리에 영향을 미치는지에 관하여는 3장에서 더 많은 정보를 찾을 수 있다.

AcqG2Count 예제	
Command Response	>AcqG2Count = 10 AcqG2Count = 10
Command Response	>AcqG2Count? AcqG2Count = 10

AcqG2Q

F800 | 9900 | 9680 | 9650

AcqG2Q는 0에서 7 사이에서 하나의 정수 매개변수만 취한다. 처음 "Q" 값은 Class1/Gen2 프로토콜 수행을 조율하는데 사용된다. 예를 들어, AcqG2Q를 3으로 설정하면, 리더는 Q = 3인 태그를 찾기 시작한다. 리더는 인벤토리 중에 활성화된 Q 값을 상하 조정할 수 있으나(AcquireMode=글로벌 스크롤을 사용할 때는 아님), 처음에는 항상 이 값으로 시작한다.

Gen2 태그군이 작으면 Q값이 작을 때 이점이 있으며(0-1), Gen2 태그군이 크면 Q값이 클 때(2-5) 이점이 있다.

- 허용된 값: Integer (0..7)
- 기본값: 3
- 이 매개변수가 어떻게 태그 인벤토리에 영향을 미치는지에 관하여는 3장에서 더 많은 정보를 찾을 수 있다.

AcqG2Q 예제	
Command Response	>AcqG2Q = 1 AcqG2Q = 1
Command Response	>AcqG2Q? AcqG2Q = 1

AcqG2QMax

F800 | 9900 | 9680 | 9650

Gen2 인벤토리 중에는 리더가 AcqG2Q와 동일한 Q값으로 시작하나, 그 다음에는 필드에서 찾는 태그의 수에 따라 Q값을 늘리거나 줄인다. AcqG2QMax를 사용하면 Q값의 상한값을 정할 수 있다.

- 허용된 값: Integer(0..15)
- 기본값: 7

AcqG2QMax 예제	
Command Response	>AcqG2QMax? AcqG2QMax = 7
Command Response	>AcqG2QMax = 3 AcqG2QMax = 3

AcqG2Select

F800 | 9900 | 9680 | 9650

AcqG2Select는 0에서 255 사이에서 하나의 정수 매개변수만 취한다. Class1/Gen2 인벤토리 주기를 시작할 때마다 전송되는 SELECT 명령어의 횟수로서, Class1/Gen2 프로토콜 수행을 조율할 때 사용한다.

매번 G2 인벤토리 주기가 시작할 때, 리더는 SELECT 명령어를 전송하는데, 이로써 현재 마스크에 부합하는 태그들을 Class1/Gen1에 WAKE 명령어와 비슷한 "인벤토리 없음" 상태가 되게 한다. AcqG2Select 명령어를 사용하면 SELECT 명령어를 전송할지, 몇 번이나 전송할지를 지정할 수 있다. Selects 명령어를 전송하지 않는다면, 사용자의 AcqG2Mask가 아무런 효력이 없을 것이다.

크고 정적인 태그군의 인벤토리를 만들려고 할 때, SELECT를 비활성화하는 것이 도움이 되는데, 이렇게 하면 읽기 쉬운 태그가 인벤토리 상태를 유지하여 리더가 읽기 힘든 태그에 주력할 수 있기 때문이다.

- 허용된 값: Integer(0..255)
- 기본값: 1
- AcqG2Select = 0으로 설정하면 리더는 SELECT 명령어를 전송하지 않는다.

- 이 매개변수가 어떻게 태그 인벤토리에 영향을 미치는지에 관하여는 3장에서 더 많은 정보를 찾을 수 있다.

AcqG2Select 예제	
Command Response	>AcqG2Select? AcqG2Select = 1
Command Response	>AcqG2Select = 0 AcqG2Select = 0

AcqG2Session

F800 | 9900 | 9680 | 9650

AcqG2Session은 0에서 3 사이에서 하나의 정수 매개변수만 취한다. Class1/Gen2 태그를 수집할 때 리더가 사용하는 인벤토리 세션을 말한다.

Class1/Gen2 프로토콜을 사용하면 태그는 0부터 3까지 네 개의 세션 중에 "인벤토리가 있는 상태"를 유지한다. 이론적으로는 각기 다른 세션에서 동작하는 여러 리더가 동일한 태그군을 동시에 질의하는 것이 가능하다(실제로는 태그가 한 번에 하나의 리더씩만 전송할 때 가장 잘 작동하는 경향을 보인다). AcqG2Session 명령어는 리더가 작동할 세션을 지정한다. 각가 다른 세션을 사용해야 하는 또 다른 이유로, 태그는 여러 다양한 "지속" 시간 중에 "인벤토리가 있는" 상태를 유지하는 것으로, 이 경우 용례가 다를 때 특히 이점이 있다.

G2 Session	일반 지속시간
0	Tag energized: Indefinite Tag not energized: None
1	Tag energized: 500 ms – 5 sec Tag not energized: 500 ms – 5 sec
2	Tag energized: Indefinite Tag not energized: > 2 sec
3	Tag energized: Indefinite Tag not energized: > 2 sec

- AcqG2Session의 기본값은 1이다.
- 이 매개변수가 어떻게 태그 인벤토리에 영향을 미치는지에 관하여는 3장에서 더 많은 정보를 찾을 수 있다.

AcqG2Session 예제	
Command Response	>AcqG2Session? AcqG2Session = 1
Command Response	>AcqG2Session = 2 AcqG2Session = 2

G2Wake

F800 | 9900 | 9680 | 9650

G2Wake 명령어는 태그 필드에 Gen2 "Select" 명령어를 전송한다. Select 명령어 전송 횟수를 지정하기로 선택할 수도 있다. 이 명령어는 인벤토리 중 Select 없이 긴 세션에(2 또는 3) 태그 인벤토리를 만들 때 특히 유용한데, 이 경우 태그는 장기간 인벤토리에 응답하지 않을 것이다. G2Wake 명령어를 전송하면 이 태그들이 다시 다음 인벤토리부터 참여하기 시작한다.

사용자가 정수 1~255 사이에서 G2Wake 명령을 따를 경우, 리더는 여러 차례 Select 명령어를 전송한다.

G2Wake 예제	
Command	>G2Wake
Response	OK
Command	>G2Wake 10
Response	OK

AcqG2Mask

F800 | 9900 | 9680 | 9650

AcqG2Mask 명령어를 사용하면 Class1/Gen2 태그를 인벤토리화할 때 사용하는 마스크를 지정할 수 있다. 마스크는 태그 주소를 정하고 질의할 때 중요하다. 마스크에 대한 자세한 설명을 원한다면, 앞 섹션의 "태그 읽기 기본요소(Tag Reading Fundamentals)"를 참조하기 바란다.

기본 형태의 AcqG2Mask 명령어는 다음 4개의 매개변수를 갖는다.

- (1-3) 마스크에 대한 메모리 뱅크
- 10진수인(0-2097151) 메모리 뱅크 bitPtr
- 10진수인(0-255) 마스크 bitLen
- 백색 공간으로 분리된 Hex Bytes 배열

AcqG2Mask 명령어는 뱅크 필드가 필요한데, 그 수는 뱅크 1~3개로 제한된다(뱅크 0에서의 마스킹은 G2 프로토콜에서는 허용되지 않음).

G2 bank 0 - RESERVED (마스킹이 가능하지 않음)
G2 bank 1 - EPC (CRC + PC + EPC)
G2 bank 2 - TID (태그 식별자)
G2 bank 3 - USER (사용자 지정 데이터로 모든 태그가 지원하지는 않음)

사용자는 이 명령을 사용하여 별도의 최대 4개까지 Class1/Gen2 마스크를 지정할 수있다. 요청한 마스크 전부와 부합하는 태그만 인벤토리에 포함된다. AcqG2MaskAction이 "제외"일 경우에는 요청한 마스크 전혀 부합하지 않는 태그만 인벤토리에 포함된다. 마스크 항목은 각기 수직막대(|)로 이전 항목과 분리된다.

호환성 유의사항:

오래된 AcqMask(또는 "Mask") 명령어는 호환성 문제로 계속 남아 있다. 예전의 AcqMask 또는 Mask 명령어(EPC 뱅크에서만 작동하며, 항상 EPC 데이터 초반에 시작함)를 사용해도 작동하며, AcqG2Mask 매개변수는 이때 상응하는 Gen2-style 마스크를 보고한다. 마찬가지로, AcqG2Mask를 설정하면 AcqMask와 Mask 매개변수를 상응하는 마스크 값으로 설정하나, 이는 AcqG2Mask가 유효한 EPC 데이터 구획(CRC나 PC 단어가 아님)을 참조할 때에만 해당된다. 따라서 유효한 AcqG2Mask를 가질 수는 있으나, Mask/AcqMask의 값은 0이다(예를 들어, AcqG2Mask가 EPC 뱅크가 아닌 경우). 리더는 항상 AcqG2Mask 설정값을 사용한다.

- AcqG2Mask = 0 (또는 “All”)으로 설정하면 현재 RF 필드에 있는 모든 Gen2 태그의 주소를 지정한다.
- 이때 제시된 데이터 바이트는 최소한 bitLen가 제시한 비트 수를 지원할 수 있어야 한다.
- 마스크 데이터는 MSB에서 LSB로 전달된 바이트에서 읽으므로, bitLen이 8비트 영역에 있지 않으면 나머지 비트는(데이터의 마지막 바이트) 가장 1순위의(most-significant) 비트여야 한다.
- bitLen=0(사용자가 길이가 0인 마스크를 전달한 경우)일때 태그는 bitPtr을 가리키는 메모리 위치가 유효한 메모리 위치일 때만 반응한다.

AcqG2Mask 예제	
Command Response	// Clearing the AcqG2Mask >AcqG2Mask = 0 AcqG2Mask = 00
Command Response	// Mask for any tag with EPC starting with "03" >AcqG2Mask = 1, 32, 8, 03 AcqG2Mask = 1 32 8 03
Command Response	// Mask for only Alien tags (TID bank, byte #3=0x34 >AcqG2Mask = 2, 16, 8, 34 AcqG2Mask = 2 16 8 34
Command Response	// Mask only tags with User memory (zero-length mask, no data!) >AcqG2Mask = 3, 0, 0 AcqG2Mask = 3 0 0

Command
Response
	// Mask only tags starting with "DE", with User
	// memory (multiple masks)
	>AcqG2Mask = 1, 32, 8, DE | 3, 0, 0
	AcqG2Mask = 1 32 8 DE|3 0 0

AcqG2MaskAction

F800 | 9900 | 9680 | 9650

Class1/Gen2 프로토콜은 마스크를 포함시키거나(마스크 응답에 부합하는 태그만) 제외시킬(마스크 응답에 부합하지 않는 태그만) 수도 있다. 다수의 마스크를 사용하는 경우, 리더는 사용자가 원하는 마스크를 얻기 위해 적절한 Gen2 동작을 자동으로 수행할 것이다. 또는 사용자가 AcqG2MaskAction 명령어의 확장판 버전을 사용하여 Gen2 프로토콜의 세부사항 전체를 직접 전달할 수도 있다.

사용자가 마스크 동작을 직접 지정하기로 결정할 경우, 이러한 동작은 목록에 있는 첫 번째 동작이 마스크 목록에 있는 첫 번째 마스크와 부합하는 방식으로 나열될 것이다(수직막대'|'로 분리됨). 이 "상태 인식" 마스킹을 사용하면, 사용자가 지정하는 동작은 수치일 수도 있고(0-7) 텍스트일 수도(A-, AB, -B, 등) 있다. 부합하는 마스크가 없는 여분의 동작 항목은 무시되며, 부합하는 동작이 없는 여분의 마스크 항목은 목록의 마지막 동작을 계속 사용할 것이다.

이 마스크 동작은 사용자가 태그에 있는 "인벤토리가 있는" 플래그로 무엇을 하기 원하는지를 보여줄 것이며(A 상태로 이동시킴, B 상태로 이동시킴, 혼자 내버려 두거나, 상태를 바꿈), 마스크에 부합하는 태그뿐 아니라 부합하지 않는 태그도 해당된다. 마스크는 현재 사용 중인 것 외에 완전히 다른 세션에 대해서만 "인벤토리가 있는" 플래그를 변경시킬 수 있으며, 사용자는 특정 안테나에서만 작동하도록 각 마스크에 제한을 가할 수 있다.

AcqG2MaskAction의 허용값은 다음과 같다:

최대 4 |-다음과 같이 구성된 별도의 값:

`0 | 1 | 2 | 3 | 4 | 5 | 6 | 7` - C1G2로 정의한 동작들 - 아래 참조

`AB | A- | -B | s- | BA | B- | -A | -s` - 수치값에 상응하는 텍스트 또는 다음 중 하나:

또는:

"Include" – same as "0 |1 |1 |1" or "AB | A- | A- | A-"

"Exclude" – same as "4 | 5 | 5 | 5" or "BA | B- | B- | B-"

기억할 점은, "move to A"라 함은 일반적으로 태그가 인벤토리에 포함될 준비가 되었음을 말하며, "move to B"는 태그가 인벤토리에 포함되지 않을 것임을 의미한다. 사용자가 AcqG2Target을 "A"이외에 다른 것으로 변경하였다면, 인벤토리는 다르게 동작할 것이다.

C1G2 Action	Text Version	동작 설명	
		Matching Tags	Non-Matching Tags
0	AB	assert SL or Inventoried→A	deassert SL or Inventoried→B
1	A-	assert SL or Inventoried→A	do nothing
2	-B	do nothing	deassert SL or Inventoried→B
3	S-	negate SL or Inventoried A←→B	do nothing
4	BA	deassert SL or Inventoried→B	assert SL or Inventoried→A
5	B-	deassert SL or Inventoried→B	do nothing
6	-A	do nothing	assert SL or Inventoried→A
7	-S	do nothing	negate SL or Inventoried A ←→ B

동작의 수치 또는 텍스트 버전들을 조합하여 사용 가능하다. 특별값인 “Include”와 “Exclude”는 다음의 수치 및 텍스트 값과 상응한다.

Special Action	Numeric Version	Textual Version	설명
Include	0|1|1|1	AB|A-|A-|A-	첫 번째 마스크에서 부합하는 것 모두를 선택하거나 부합하지 않는 것 모두를 선택 해제한다. 모든 이후 마스크들에서는 부합하는 것만 선택하거나 부합하지 않는 것만 남겨 둔다.
Exclude	4|5|5|5	BA|B-|B-|B-	첫 번째 마스크에서 부합하는 태그 모두를 선택 해제하거나 부합하지 않는 것 모두를 선택한다. 모든 이후 마스크들에서는 부합하는 것만 선택 해 제하거나 부합하지 않는 것만 남겨 둔다.

마지막 동작은 추가 마스크에 사용할 것이기 때문에 'Include' 유형은 “0|1” 로만 지정되었을 수 있으며, 'Exclude' 유형은 “4|5”로만 지정되었을 수 있다. 리더의 용도는 그 밖에도 더 있다. 0의 단일 마스크 동작만 제시하면, 다른 마스크들도 자동으로 1의 동작을 갖게 할 것이며(0이 아님) - "Include"가 참인 경우처럼 동작하게 한다. 마찬가지로, 4의 단일 마스크 동작만 제시하면, 리더는 다른 마스크들도 자동으로 5의 동작을 갖게 할 것이며(4가 아님) - "Exclude"가 참인 경우처럼 동작하게 한다.

AcqG2MaskAction 예제	
Command	>AcqG2MaskAction?
Response	AcqG2MaskAction = Include
Command	>AcqG2MaskAction = Exclude
Response	AcqG2MaskAction = Exclude
Command	>AcqG2MaskAction = AB | A-
Response	AcqG2MaskAction = AB | A-
Command	>AcqG2MaskAction = 0 | 1
Response	AcqG2MaskAction = 0 | 1

AcqG2MaskAntenna

F800 | 9900 | 9680 **|** 9650

AcqG2MaskAntenna 명령어는 최대 4개까지 값을 받아들이며, AcqG2MaskAntenna 목록의 항목들과 AcqG2Mask 목록의 항목들 간에는 1대1로 대응해야 한다. 부합하는 마스크가 없는 여분의 안테나 항목은 무시되며, 부합하는 동작이 없는 여분의 마스크 항목은 목록의 마지막 안테나 항목을 계속 사용할 것이다.

AcqG2MaskAntenna의 값은 각각 비트맵으로서, 해당 안테나 별로 상응하는 마스크가 작동할 것이다. 네 개의 안테나는 각각 처음 4개의 저순위 비트로 선택한다(bit #1은 안테나 0에 부합하며, bit #2는 안테나 1에 부합함). 물론 이론적으로는 이 중 어떠한 조합이든 선택할 수 있다(또는 여러 개도 가능). 예를 들어, 마스크가 네 개의 안테나 모두에서 동작하게 하려면, 안테나값이 0x0F(00001111)일 것이다.

Antenna Value	Binary Equiv.	Selected Antenna
0x01	00000001	Antenna 0
0x02	00000010	Antenna 1
0x04	00000100	Antenna 2
0x08	00001000	Antenna 3

- 제공한 값은 01-0F이어야 한다.
- 기본 AcqG2MaskAntenna는 0F이다. (모든 안테나)

AcqG2MaskAntenna 예제	
Command Response	>AcqG2MaskAntenna? AcqG2MaskAntenna = 0F
Command Response	>AcqG2AccessPwd? AcqG2AccessPwd = 01 02 03 04
Command Response	// 1st mask on Ant #0, 2nd mask on Ant #1, 3rd mask on Ant #2, etc. >acqg2maskantenna = 1 | 2 | 4 | 8 AcqG2MaskAntenna = 01 | 02 | 04 | 08

AcqG2SL

F800 | 9900 | 9680 **|** 9650

SL 플래그는 "인벤토리가 있는" 플래그와 유사한 방식으로 작동하나, 단 하나의 SL 플래그만 있으며 어떤 세션에서든 접속이 가능하다. 마스크 동작으로 SL 플래그 상태를 변경할 수 있으며, SL 플래그 또한 인벤토리 중 변경될 수 있다. SL 플래그는 인벤토리가 있는 플래그처럼 지속성이 있으며, 대부분의 Alien 태그는 분 단위로 비교적 긴 SL 지속 시간이 특징이다.

AcqG2SL의 기본값은 “all"이며, 이는 리더가 태그를 인벤토리화할 때 SL 플래그를 무시하도록 지시한다. AcqG2SL = SL일 때, 'asserted' SL 플래그가 있는 태그만 인벤토리화가 되며, AcqG2SL = nSL일 때는 'deasserted' SL 플래그가 있는 태그만 인벤토리화가 된다.

- 제공되는 값은 “SL”, “nSL”, 또는 “ALL”이어야 한다.
- 기본 AcqG2SL은 'ALL' 이다.

AcqG2SL	
Command Response	>AcqG2SL? AcqG2SL = All
Command Response	// Only tags with SL asserted will be inventoried >AcqG2SL = SL AcqG2SL = SL

AcqG2AccessPwd

F800 | 9900 | 9680 | 9650

AcqG2AccessPwd 값은 패스워드로 보호하는 Class 1/Gen2 태그에 프로그래밍을 할 때 사용한다. 이러한 태그에 0이 아닌 접속 패스워드가 있는 경우에는, 리더가 같은 패스워드를 제시하여 태그를 "보안이 된" 상태 즉 잠금 및 잠금 해제가 가능한 상태가 되게 하고 동작 내역을 기록해야 한다.

제공한 AcqG2AccessPwd가 태그에 보관된 접속 암호(Access Pwd)에 맞지 않을 경우, 해당 태그에 수행한 프로그래밍이 실패하여 "접속 실패" 오류 메시지가 나타날 것이다.

이 명령어는 유사한 듯 보이는 ProgG2AccessPwd 명령어와는 다르다. ProgG2AccessPwd는 Access Pwd 값을 태그에 작성하기 위한 플레이스홀더인 반면, AcqG2AccessPwd는 이미 패스워드로 보호하는 Class 1/Gen2 태그에서 작업할 때 사용하는 태그의 현재 접속암호를 제공한다.

- 제공한 값은 4-바이트 값이어야 한다.
- 기본 AcqG2AccessPwd는 00 00 00 00이다.

AcqG2AccessPwd 예제	
Command Response	>AcqG2AccessPwd = 1 2 3 4 AcqG2AccessPwd = 01 02 03 04
Command Response	>AcqG2AccessPwd? AcqG2AccessPwd = 01 02 03 04

AcqG2Target

F800 | 9900 | 9680 | 9650

리더가 Gen2 태그의 인벤토리를 만들 때, 태그가 인벤토리화 되었는지를 표시하는 태그 내 A/B 플래그를 보여준다. 기본 동작은 리더가 모든 태그를 A 상태가 되도록 "선택"한 다음 태그들의 인벤토리를 만들 때 B 상태로 이동시키는 것이다.

사용자는 이 동작을 변경하여 B 상태에서 태그를 시작한 다음 A 상태로 이동하도록 하거나 하나의 인벤토리에서 다음 인벤토리로 A와 B 상태를 전환할 수도 있다(보통 태그 읽기를 더 많이 수행해야 함).

AcqG2Target 모드를 변경할 때 태그 인벤토리에 미치는 영향은 사용 중인 AcqG2Session, 읽기 영역 제어, 필드에 태그가 머무는 시간에 따라 크게 달라진다. 사용자는 이 설정을 변경하기 전에 Gen2 프로토콜을 제대로 파악해야 한다.

- 허용된 값: "A" | "B" | "AB".
- 기본값: "A".

AcqG2Target 예제	
Command Response	>AcqG2Target? AcqG2Target = A
Command Response	>AcqG2Target = AB AcqG2Target = AB

AcqG2TagData

F800 | 9900 | 9680 | 9650

AcqG2TagData 명령어를 사용하면 Gen2 태그 메모리의 추가 구획을 지정하여 인벤토리 중에 불러올 수 있다(패스워드, TID, 사용자 메모리 등). 리더는 읽고 있는 태그마다 추가 데이터를 불러와서 TagList에 포함시킨다. 이 경우 읽기 성능에 큰 영향을 미칠 수 있으나, 인벤토리 작업을 한 다음 원하는 데이터에 대하여 각 태그를 개별적으로 쿼리하는 것보다는 훨씬 빠르다.

추가로 불러올 태그 데이터 블록을 뱅크, 워드 포인터, 워드 길이값 별로 지정한다. 최대 4개까지의 태그 메모리 데이터 블록을 지정할 수 있으며, 이는 수직막대(|)로 구분한다. 단어 길이를 0으로 지정하면, 리더는 '길이-0으로 읽기'를 수행하여 워드 포인터 이후 전체 메모리를 반환한다(본 문서 후반에 있는 "G2Read" 참조).

요청된 태그 메모리 데이터 블록이 존재하지 않거나 리더가 읽지 못하면(예를 들어 소음 때문에), 데이터 블록이 TagList에 보고되지 않는다.

`AcqG2AccessPwd` 를 설정하면 패스워드로 보호한 메모리 뱅크에서 데이터를 읽어올 수 있다. 예를 들어, `HideAlienUserBlocks` 명령어를 사용해 사용자 메모리 뱅크에 있는 민감한 데이터를 패스워드로 보호하고, 정확한 `AcqG2AccessPwd` 패스워드를 설정하여 인벤토리 중에 데이터를 읽게 할 수 있다.

AcqG2TagData를 수집하였다면, 해당 데이터는 text- 및 XML- 형식의 TagLists에 자동으로 포함되며, "D1", "D2" 등의 접두사가 붙는다. 태그 메모리 데이터 블록은 다음의 표시를 사용해 사용자 정의 형식 TagLists에 포함시킬 수도 있다: ${G2DATA1}, ${G2DATA2}, ${G2DATA3}, ${G2DATA4} (TagListCustomFormat 설명 참조).

- AcqG2TagData를 설정하면 리더는 태그를 읽는 동안 추가 작업을 할 수 있으며, 이는 읽기 성능에 영향을 미칠 수 있다.
- 각 데이터 블록의 형식은 G2Read 명령어를 사용할 때와 동일하므로, 어떤 G2Read 명령어가 원하는 데이터를 줄지를 결정하고 해당 명령어를 사용해 AcqG2TagData를 구성해야 한다.

- AcqG2TagData를 비활성화시키려면, 0으로 설정하면 된다. 이는 기본 설정값이다.

AcqG2TagData 예제

Command Response	>AcqG2TagData? AcqG2TagData = 0
Command Response Command Response	// Show the PC word and the 1st word of the EPC (bank 1, words 1-2) >AcqG2TagData = 1 1 2 AcqG2TagData = 1 1 2 >t Tag:0000 0000 0000 0000 0000 0001, …, D1:3000 0000 Tag:DEAD BEEF CAFE 8042 55AA 0006, …, D1:3000 DEAD Tag:AABB 3344 DDEE 7788 9900 1132, …, D1:3101 AABB
Command Response Command Response	// Show all of User memory too >AcqG2TagData = 1 1 2 | 3 0 0 AcqG2TagData = 1 1 2 | 3 0 0 >t Tag:0000 0000 0000 0000 0000 0001, …, D1:3000 0000, D2:1FF1 1FF2 Tag:DEAD BEEF CAFE 8042 55AA 0006, …, D1:3000 DEAD Tag:AABB 3344 DDEE 7788 9900 1132, …, D1:3101 AABB // (some tags don't have User memory)

AcqG2AntennaCombine

F800 | 9900 | 9680 **|** 9650

AcqG2AntennaCombine 명령어는 인벤토리 주기 중에 Gen2 SELECT 명령어를 어떻게 전송할지를 제어한다.

ON(기본값)으로 설정되어 있으면, 리더가 AntennaSequence 내에 있는 전체 안테나에 Gen2 SELECT 명령어를 전송하고, 그런 다음 인벤토리 루프로 작업을 진행한다. 이 설정은 필드에 태그가 많을 때 효과적일 수 있으나, 먼저 "깨우기(wake them up)"를 한 다음 인벤토리를 만들어서 (긴 세션으로 - 2 또는 3) 여러 안테나들에서 잉여 읽기 결과가 없도록 할 것이다.

OFF로 설정되어 있으면, Gen2 SELECT 명령어를 사용 중인 각 안테나에 전송한다. 이 설정은 필요할 때 읽기 횟수를 늘릴 수 있는데, 예를 들어 컨베이어/속도 조절이 필요한 경우에 유용하다. 또한 별도의 읽기 영역을 사용할 때 도움이 되는데, 이 경우 사용자는 태그가 하나의 안테나 지대에서 다음으로 지나가게 해야 한다. 각 안테나는 태그군 전체를 읽을 기회를 갖는다. 또한 사용자가

태그의 위치 또는 동작의 방향을 측정하기 위해 여러 안테나에서 속도와 방향 데이터를 사용할 때 이 방식이 필수적이다.

- 허용된 값: "ON" | "OFF"
- 기본값: "ON"

AcqG2AntennaCombine 예제	
Command Response	>AcqG2AntennaCombine? AcqG2AntennaCombine = ON

Command
Response
 	
Command
Response
	// set reader parameters
	>AntennaSequence = 0 1
	AntennaSequence = 0 1
	>AcqG2Session = 1
	AcqG2Session = 1
	>TagListAntennaCombine = OFF
	TagListAntennaCombine = OFF
	
	// first, issue selects on all antennas,
	// then perform an inventory on all antennas
	>AcqG2AntennaCombine = ON
	AcqG2AntennaCombine = ON
	>t
	Tag:8000 8004 2665 8426, Count:1, Ant:0
	// the tag was only inventoried on the first antenna
	
	// perform 'select-inventory' sequence on each antenna
	>AcqG2AntennaCombine = OFF
	AcqG2AntennaCombine = OFF
	>t
	Tag:8000 8004 2665 8426, Count:1, Ant:0
	Tag:8000 8004 2665 8426, Count:1, Ant:1
	// tag was selected and inventoried on both antennas

AcqG2Ops

F800 | 9900 | 9680 **|** 9650

태그군에서 인벤토리 작업을 수행하는 주된 목적은 각 태그의 고유 EPC 코드를 식별하기 위함이다. 사용자는 또한 AcqG2Data 명령어를 사용하여 리더가 인벤토리 중 추가 작업을 수행하게 할 수도 있는데, 예를 들면 태그 메모리의 다른 구획을 읽거나(Access & Kill 패스워드, TID와 USER 메모리 뱅크), 태그리스트에 있는 추가 데이터 필드로 반환하게 할 수도 있다.

이 새로운 AcqG2Ops 명령어를 사용하면 'write, lock, kill'과 같은 작업을 수행할 수 있으며, 특히 여러 안테나에서 인벤토리 중간에 바로 수행할 수 있다. 추가 작업을 통해 Alien Higgs4 태그의 새로운 BlastWrite 기능을 지원할 수 있으며, 이로써 사용자는 동시에 전체 태그군에서 이러한 작업을 수행할 수 있다.

사용자는 최대 8가지 별도의 작업을 정의할 수 있으며, 리더가 인벤토리 중에 각 태그를 식별하면서 각 태그에 이러한 작업들을 수행할 수 있다. 일단 작업이 완료되면(또는 그 중 하나가 실패하면), 리더는 인벤토리에 있는 다음 태그로 통신을 시작하고 해당 태그에 같은 작업을 수행한다.

AcqG2Ops는 1부터 8까지 번호가 부여되고 그 순서대로 실행된다. 순번은 생략할 수 있으며, 상시 개별 작업을 삭제하거나 정의할 수도 있다. 일단 AcqG2Ops를 정의했다면 다음 중 한 가지 방식으로 활성화할 수 있다.

1. AcqG2OpsMode = On로 설정한다.
    a. 기본값은 "Off"이다.
    b. 일단 켜지면, 리더가 수행한 모든 태그 인벤토리 작업이 다음과 같이 AcqG2Ops에 대하여 수행된다. "t", "acquire", AutoMode 실행 등.
2. 이 새로운 "to" (taglist ops) 명령어를 "t" 대신 전송한다.
    a. 이때 AcqG2OpsMode=Off일지라도 AcqG2Ops를 실행한다.
    b. 사용자는 "to"와 "t"를 앞뒤로 전환하면서 실행 중인 AcqG2Ops가 있는지 여부에 관계없이 인벤토리를 수행할 수 있다.

ACQG2OPS COMMAND SYNTAX

일반구문으로 다음과 같이 특정 AcqG2Op를 설정한다.

```
AcqG2Ops = <opsNum> <opsAction> <opsArgs>
```

<opsNum> 은 이 op의 번호를 가리킨다(1-8).
<opsAction> 은 작업의 명칭을 가리킨다(또는 관련된 1바이트 코드).
<opsArgs> 는 제시된 <opsAction>에서 필요한 인수들을 사이띄기로 구분한 목록이다.

다음은 현재 지원되는 작업 목록과 각각의 인수들이다. 16진수값의 인수들에는 인수명 앞에 바이트 수가 표시되며(e.g. <2:blockMask1> 는 2바이트 16진수값임), 또는 십진수값으로 표시된다. 작업, 인수, 각 G2Ops가 반환하는 데이터/결과에 관한 세부 정보를 원할 경우, 다음 설명을 참조할 것.

Gen2 Action Names	Code	Arguments
G2Read	01	<bank> <wordPtr> <wordCount>
G2Write	02	<bank> <wordPtr> <2n:data…> [+, -, ++]
G2Kill	03	<4:killPwd>
(reserved)	04	-
G2Lock	05	<field> <lockType>
G2BlockErase	06	<bank> <wordPtr> <wordCount>
(reserved)	07	-
G2BlockPermalock	08	<blockPtr> <2:blockMask1> [<2:blockMask2>…]
G2GetPermalock	09	<blockPtr>
Alien Gen2 Action Names	**Code**	**Arguments**
AlienG2UserReadlock	0A	<1:blockMask>
AlienG2BlastLock	85	<field> <lockType>
AlienG2BlastErase	86	<bank> <wordPtr> <wordCount>
AlienG2BlastWrite	87	<bank> <wordPtr> <2n:data…>
AlienG2BlastBlockPermalock	88	<blockPtr> <2:blockMask1> [<2:blockMask2>…]
(reserved)	89	-
AlienG2BlastUserReadlock	8A	<1:blockMask>
(reserved)	8B	-
AlienG2TagStatus	8C	<1:statusMask>

여기서:

<bank> : 0 – 3
<wordPtr> : 0 – 2097151 (base 10)
<wordCount> : 0 – 32 (base 10)
[+, -, ++] : 이 중 하나를 G2write에 붙여 데이터를 증분시킨다.
+ = 성공적으로 쓰기한 후
- = 쓰기 실패 후
++ = 항상
<field> : 0 (killPwd), 10 (accessPwd), 1, 2, 3 (EPC, TID, USER banks)
<lockType> : 원하는 잠금 동작:
L 또는 Lock
U 또는 Unlock
PL 또는 PermaLock
PU 또는 PermaUnlock
<blockPtr> : 0 - 4064를 사용하여 사용자 메모리에 있는 데이터 블록을 가리킴
<blockMask> : 어떤 사용자 블록을 활성화해야 할지 표시할 때 사용함.
최우선 비트 = Block #1
차선 비트 = Block #2의 순으로
G2BlockPermaLock은 16비트 <2:blockMask>를 사용한다.
AlienG2UserReadLock은 8비트 <1:blockMask>를 사용한다.
<statusMask>: 비트로 태그에서 가져올 상태값을 표시한다.
1 = 뱅크 writeLocks 요청(bit #1)
2 = 사용자 블록 readLocks 요청(bit #2)
3 = writeLocks와 readLocks 요청(bits #1 and #2)

인벤토리 중에는 사용자가 훨씬 더 많은 일을 할 수 있으므로, 이전보다 더 많은 병목현상/제약을 경험하게 될 수 있다. 예를 들어 미국에서는 리더가 400ms마다 새로운 주파수로 호핑(hop)해야 한다. 이로 인해 태그와의 모든 통신이 방해를 받기 때문에 BlastWrite 작업 중에는 다음 홉(hop)을 위해 끝내야 하는 작업도 생긴다. 또한 인벤토리가 있는 태그는 A 상태에서 B 상태로 내부 플래그가 전환되며, 확장된 인벤토리가 수행 시간이 더 길어지기 때문에 B 상태에 있던 초기 태그들 중 일부는 A 상태로 "되돌아갈" 수도 있다. 이는 프로토콜의 자연스런 상태로서, 사용 중인 AcqG2Session에 따라 달라진다.
AcqG2Ops로 태그 작업을 수행할 때에는 AcqG2Session = 2 (매우 긴 B 지속 시간임)를 사용할 것을 권한다.

- 인벤토리에서 AcqG2Ops를 수행할 때에는 다음의 순서대로 실행된다: <opsNum>.
- 단, 항상 마지막에 실행되는 AlienG2BlastXXXX 작업은 예외이다. “블라스트(blast)” 작업에서는 리더가 EPC 목록을 항상 알아야 하기 때문에, 인벤토리 말엽에 이러한 작업을 수행하는 것이 훨씬 더 효율적이다.
- 작업이 실패할 경우, 해당 태그의 이후 AcqG2Ops는 실행되지 않는다. 리더는 다음 태그와 처음 구성된 AcqG2Ops로 인벤토리를 계속 진행할 것이다.
- 이러한 작업의 결과물은 새로운 ${G2Ops}와 ${G2Ops1..8} 사용자 정의 태그리스트 신호를 사용하여 태그리스트에 수집된다.

AcqG2Ops 예제	
Command Response	// Set AcqG2Ops #1 (write “DEAD BEEF” to the start of USER memory) > AcqG2Ops = 1 g2write 3 0 DE AD BE EF AcqG2Ops = 1 g2write 3 0 DE AD BE EF
Command Response	// Set AcqG2Ops #3 (lock USER bank) > AcqG2Ops = 3 g2lock 3 lock AcqG2Ops = 3 g2lock 3 lock
Command Response	// Get all AcqG2Ops: > AcqG2Ops? 1 g2write 3 0 DE AD BE EF 3 g2lock 3 lock
Command Response	// See it all in the Info dump: > i acquire ********** ACQUIRE COMMANDS ********** …snip… AcqG2Ops = 1 g2write 3 0 DE AD BE EF AcqG2Ops = 3 g2lock 3 lock …snip…
Command Response	// Get just AcqG2Ops #1: > AcqG2Ops 1? 1 g2write 3 0 DE AD BE EF
Command Response	// Clear AcqG2Ops #3: > AcqG2Ops = 3 AcqG2Ops = 3
Command Response	// Clear all AcqG2Ops: > AcqG2Ops = 0 AcqG2Ops = 0

ACQG2OPS RESULTS SYNTAX

AcqG2Ops 작업의 결과물은 리더의 TagList에서 각 태그 별로 관리되며, 사용자에게 전달되는 TagLists에 포함된다. 이때 새로운 사용자 정의 태그리스트 표시의 예는 다음과 같다.
${G2Ops} 또는 ${G2Ops1}…${G2Ops8} AcqG2Ops의 각 결과 블록에는 다음의 일반 구문이 포함된다.

<1:opsActionCode> <1:opsResultCode> [<data>]

달리 말하면, 실행한 작업에는 <opsActionCode> 가 단일 바이트에 포함되며, 뒤이어 결과 코드가 단일 바이트에 포함된다. 그 다음에는 작업에서 요청된 데이터가 뒤따른다(g2read, g2getpermalock, alieng2tagstatus 등).
<opsActionCode> 는 상기 AcqG2Ops 작업 목록에서 제시된 것과 동일한 코드이다.

작업이 성공했을 때 <opsResultCode> 는 00이며, 그 밖의 경우에는 이 리더 인터페이스 안내서의 부록 F에 나열한 (16진수) DSP 오류 코드에 해당하는 오류 코드가 표시된다.

<data> 반환은 실행한 특정 AcqG2Ops 동작에 따라 달라진다.

G2READ (0X01) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

G2Read 작업은 태그 메모리의 읽기 가능한 구획에서 저수준 읽기를 수행한다. 사용자는 읽을 단어로 시작하여 읽을 단어 수 등의 내용으로 뱅크를 지정해야 한다. 리더에게 0 단어를 읽도록 요청하면, 리더는 뱅크에 남은 데이터를 모두 읽는다(태그가 이 기능을 지원하는 경우에).

```
AcqG2Ops = <opsNum> G2Read <bank> <wordPtr> <wordCount>
AcqG2Ops = <opsNum>   01   <bank> <wordPtr> <wordCount>

// Read the first three words of USER memory
AcqG2Ops = 1 G2Read 3 0 3
```

잠긴 AccessPwd 또는 KillPwd, 또는 읽기 잠금 상태의 USER 메모리는 사용자가 정확한 AcqG2AccessPwd를 제시하지 않는 한 읽을 수 없다.

AcqG2Ops 결과 내 <data>는 16진수 데이터 중 하나 이상의 블록으로서, 그 형식은 TagDataFormatGroupSize 설정값에 따라 결정된다(0=compressed hex, 1=bytes, 2=words).

```
// Successful G2Read result
01 00 FEED FACE BEEF
```

G2WRITE (0X02) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

G2Write 작업은 태그 메모리의 쓰기 가능한 구획에서 저수준 읽기를 수행한다. 쓰기 잠금 상태의 데이터는 사용자가 정확한 AcqG2AccessPwd를 제시할 때에만 쓰기 가능하며, PermaLocked 상태의 데이터는 일절 쓸 수 없다. 사용자는 작성할 워드 포인터와 그 뒤이어 쓸 데이터, 선택적 증분 기호로 뱅크를 지정해야 한다.

```
AcqG2Ops = <opsNum> G2Write <bank> <wordPtr> <2n:data…> [+, -, ++]
AcqG2Ops = <opsNum>   02    <bank> <wordPtr> <2n:data…> [+, -, ++]

// Write a kill password
AcqG2Ops = 1 G2Write 0 0 DE AD DE AD

// Write an incrementing EPC
AcqG2Ops = 1 G2Write 1 2 00 00 00 00 00 00 00 00 00 00 +
```

데이터를 자동 증분하도록 선택한 경우, AcqG2Ops는 계속하여 다음 쓰기 값을 추적할 것이며, 사용자가 중지하고 리더의 인벤토리를 재시작할 경우에는 마지막 중단된 곳에서부터 계속된다. 완전히 동일한 EPC 코드로 여러 태그를 프로그래밍하지 않도록 조심해야 하는데, 이런 태그에 추가 작업을 할 경우 리더가 이들을 따로 구분할 수 없어 어려워진다.

G2Write에 대한 AcqG2Ops 결과에 포함된 <data>는 증가값이 없을 때의 결과 코드로서, 증분이 활성화되었을 때 작성된 실제 데이터만 포함한다. 작성된 16진수 데이터는 TagDataFormatGroupSize 설정에 따라 형식이 결정된다(0=compressed hex, 1=bytes, 2=words). G2Write에 증분이 있는 경우, 명령어 코드는 0x02에서 0x82로 바뀐다.

```
// Successful G2Write result (non-incrementing)
02 00

// Successful G2Write result (incrementing)
82 00 0000 0000 0000 0000 0000 01F3
```

G2KILL (0X03) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

G2Kill 작업은 태그를 완전히 비활성화하여 향후 인벤토리에서 작동하지 못하게 한다. 태그를 없애려면(kill), 이미 그 안에 0 이외 'Kill Password'를 포함해야 하며, 사용자는 'Kill' 명령어에 그에 상응하는 Kill Password를 제시해야 한다.

이 명령어는 매우 위험하므로 주의하여 사용해야 하는데, 인접한 모든 태그들에 영향을 미칠 것이기 때문이다!

```
AcqG2Ops = <opsNum> G2Kill <4:killPwd>
AcqG2Ops = <opsNum>   03   <4:killPwd>

// Kill a Tag
AcqG2Ops = 1 G2Kill DE AD DE AD
```

G2Kill 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful G2Kill result
03 00
```

G2LOCK (0X05) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

G2Lock 명령어는 태그 메모리의 필드/뱅크 잠금 상태를 변경한다. "field"라는 용어를 "bank" 대신 사용하는 이유는 예약된(Reserved) 뱅크인 뱅크 0에는 Kill Password와 Access Password가 포함되어 있고, 각기 독립적으로 잠글 수 있기 때문이다. 사용자는 원하는 대로 필드를 잠그거나 잠금 해제할 수 있으나, 필드를 한번만 PermaLock 또는 PermaUnlock할 수 있다.

잠긴 필드는 태그 별로 정확한 AcqG2AccessPwd를 제시해야만 다시쓰기 할 수 있다. Kill Password의 잠긴 접속 패스워드는 읽을 수 없다. TID 뱅크는 늘 PermaLocked 상태이다.

사용자는 인수로 잠글 <field>와 원하는 <lockType> 항목을 통과한다.

```
AcqG2Ops = <opsNum> G2Lock <field> <lockType>
AcqG2Ops = <opsNum>   05   <field> <lockType>

// Lock the User Bank
AcqG2Ops = 1 G2Lock 3 Lock

// PermaLock the Kill Password
AcqG2Ops = 1 G2Lock 0 PermaLock
```

<field>는 다음 정수 중 하나에 해당한다.

```
 0 = Kill password
10 = Access password
 1 = EPC bank
 2 = TID bank
 3 = USER bank
```

<lockType>는 다음 문자열 또는 그 단축키 중 하나에 해당한다.

```
 L or Lock
 U or Unlock
PL or PermaLock
PU or PermaUnlock
```

G2Lock 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful G2Lock result
05 00
```

G2BLOCKERASE (0X06) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

G2BlockErase 작업으로 태그 메모리의 특정 범위를 지울 수 있다(00으로 설정). 사용자는 지울 단어로 시작하여 지울 단어 수 등의 내용으로 뱅크를 지정해야 한다. 이 명령어는 유사한 G2Write보다 약간 더 빠른데, 0으로 채운 바이트를 모두 전송할 필요는 없기 때문이다.

```
AcqG2Ops = <opsNum> G2BlockErase <bank> <wordPtr> <wordCount>
AcqG2Ops = <opsNum>      06      <bank> <wordPtr> <wordCount>

// Erase the first four words of USER memory
AcqG2Ops = 1 G2BlockErase 3 0 4
```

G2BlockErase 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful G2BlockErase result
06 00
```

G2BLOCKPERMALOCK (0X08) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

G2BlockPermalock 작업은 사용자 메모리에서 특정 블록을 영구적으로 쓰기 잠금 한다. 여러 태그와 태그 제조자 별로 사용자 메모리의 '블록'이 얼마나 큰지 정의가 다르다. 사용자는 처음 블록 번호를 지정하고, 하나 이상의 16비트 마스크로 메모리의 각 해당 블록에서 원하는 PermaLock 상태를 표시해야 한다. 대부분의 현재 태그에는 16개 이하의 사용자 블록이 포함되며, 따라서 일반적으로 `<blockPtr>` 는 0이고 단 하나의 `<2:blockMask>` 만 있다.

```
AcqG2Ops = <opsNum> G2BlockPermalock <blockPtr> <2:blockMask1>…
AcqG2Ops = <opsNum>        08        <blockPtr> <2:blockMask1>…

// Permanently lock the first two User blocks
// “C0 00” = 11000000 00000000 → first two blocks locked
AcqG2Ops = 1 G2BlockPermalock 0 C0 00
```

G2BlockPermalock 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful G2BlockPermalock result
08 00
```

G2GETPERMALOCK (0X09) COMMAND & RESULTS

F800 **| 9900 |** 9680 **|** 9650

G2GetPermalock 작업은 사용자 메모리 블록의 현재 BlockPermalock 상태를 반환한다. 처음 블록의 번호를 지정하고 (16까지 증분), 이어지는 16개 블록 중 2바이트 상태 정보를 회수한다.

```
AcqG2Ops = <opsNum> G2GetPermalock <blockPtr>
AcqG2Ops = <opsNum>       09       <blockPtr>

// Fetch the Permalock status of the first blocks of USER memory
AcqG2Ops = 1 G2GetPermalock 0
```

G2GetPermalock 작업 결과는 성공/실패 코드이며, 뒤이어 <2:blockMask> 데이터에는 각 블록의 상태 정보가 포함된다(해당 단어 내 비트에 상응함. 최상위(가장 좌측)에서 최하위(가장 우측) 순으로).

```
// Successful G2GetPermalock result
// “C0 00” = 11000000 00000000 → first two blocks locked
09 00 C0 00
```

ALIENG2USERREADLOCK (0X0A) COMMAND & RESULTS

F800 | 9900 | 9680 **|** 9650

AlienG2UserReadLock 작업은 Alien Higgs 기반의 태그 기능을 활용하여 사용자가 블록에 "읽기 잠금" 기능을 적용하여 사용자 메모리의 개별 블록을 감출 수 있게 해준다. 읽기 잠금 상태의 블록은 사용자가 정확한 AcqG2AccessPwd를 제시하지 않고 읽으려 하면 0이나 오류를 반환한다. 사용자 메모리의 개별 블록은 G2BlockPermalock 명령어와 유사하게 주소가 지정되며, 이때 다음 인수를 사용한다. `<1:blockMask>` 단, 이 마스크는

G2BlockPermalock에 사용할 16비트 단어가 아닌 단일 바이트일 때는 예외로 한다. <1:blockMask>에 있는 비트는 사용자 메모리에 있는 블록과 상응하며, 최상위 비트에서 최하위 비트의 순이다.

```
AcqG2Ops = <opsNum> AlienG2UserReadLock <1:blockMask>
AcqG2Ops = <opsNum>                  0A <1:blockMask>

// Hide the first 3rd and 4th blocks of USER memory
// "30" = 00110000 → 3rd and 4th blocks
AcqG2Ops = 1 AlienG2UserReadLock 30
```

AlienG2UserReadLock 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful AlienG2UserReadLock result
0A 00
```

ALIENG2TAGSTATUS (0X8C) COMMAND & RESULTS

F800 | **9900** | 9680 | 9650

AlienG2TagStatus 작업은 태그의 각 필드/뱅크의 쓰기 잠금 상태를 사용자에게 모두 반환할 수 있으며, 사용자 뱅크 내 AlienG2UserReadLocks(예: "hidden" 블록)의 상태도 반환한다. 이 명령어의 인수는 하나의 16진수 비트마스크인 <1:statusMask>로서, 이 값은 어떤 상태 요소를 반환하기 원하는지를 표시한다. AlienG2TagStatus는 현재 Alien Higgs4 기반 태그에서만 지원된다.

```
AcqG2Ops = <opsNum> AlienG2TagStatus <1:statusMask>
AcqG2Ops = <opsNum>        8C        <1:statusMask>

<1:statusMask>:   1 = bank/field write-locks (bit 1)
                  2 = User read-locks (bit 2)
                  3 = both write-locks and read-locks (bits 1 & 2)

// Ask for both the bank write-locks and the user read-locks
AcqG2Ops = 1 AlienG2TagStatus 03
```

AlienG2TagStatus 작업의 결과는 성공/실패 코드이며, 뒤에 이어지는 헤더 단어에는 보고할 상태 작업의 수와 관련 statusMask가 포함된다. 그 다음에는 각각의 상태 작업에 대한 16비트 단어가 이어지는데, 이는 필드/뱅크의 비트마스크 또는 사용자 블록 잠금 상태 중 요청된 것을 가리킨다.

```
<1:numStatusBits><1:statusMask> [<2:writeLockMask>] [<2:readLockMask>]

Where,
   <writeLockMask> : Lock/Unlock and Permalock bits are given for each tag field,
                     spanning the two bytes:
```

Byte 1:	KPW_l	KPW_p	APW_l	APW_p	EPC_l	EPC_p	TID_l	TID_p
Byte 2:	USR_l	USR_p	RFU	RFU	RFU	RFU	RFU	RFU

```
                  _l: 1=Locked, 0=Unlocked
                  _p: 1=permanent lock, 0=regular lock

Note: An unlocked tag will report 0300, since the TID bank is PermaLocked at the
factory.

   <readLockMask>  : Each bit, starting with the most-significant bit, corresponding
                     to each User block's read-lock status:
```

Byte 1:	block1	block2	block3	block4	block5	block6	block7	block8
Byte 2:	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU

AlienG2TagStatus 작업의 결과 데이터 예:

```
// AlienG2TagStatus result – just the write-locks
// "A340" = 10100011 01000000
//    → KillPwd and AccessPwd are locked, EPC unlocked, TID PermaLocked
//    → User PermaUnlocked
8C 00 0101 A340

// AlienG2TagStatus result – write- and read-locks
// write-locks = "A340" (same as above)
// read-locks  = "3000" = 00110000 00000000 → 3rd and 4th blocks read-locked
8C 00 0203 A340 3000
```

Alien *Blast*Write™ – Special Higgs4 Tag Capability

F800 | 9900 | 9680 | 9650

나머지 AcqG2Ops 동작들은 모두 Alien의 새 BlastWrite™ 특성을 수반하며, 이는 Higgs4 기반 태그에 도입되었다. BlastWrite는 리더가 동일한 작업을 인벤토리화된 태그들에서 동시에 수행되게 해준다. 리더가 여러 태그에 동시에 주소를 지정하므로, 이 작업은 여러 태그에 매우 신속하게 적용 가능하다. 사용자는 여러 태그 집합 전체를 제공하는데, 이 중에는 EPC의 공통 구획 쓰기, USER 메모리, Access 및 Kill 패스워드 쓰기, 여러 필드 잠그기 등이 해당되며, 모두 단일의 매우 빠른 리더 작업을 수반한다.

BlastWrite는 태그 인벤토리와 관련하여 수행되는 기능으로, 인벤토리화되었을 때 한 번에 하나의 태그에 대하여 작업을 수행하는 것이 다르다. 리더는 인벤토리 중에 확인된 태그들의 목록을 수집한 다음 마지막에 전체 태그 그룹에 대하여 Blast 작업을 수행한다. 이 때문에 사용자가 자신의 AcqG2Ops 작업을 구성할 때 블라스트(blast) 관련 작업들은 작업 순서 끝에 배치해야 한다.

블라스트 작업들 중에는 태그가 안전한 상태여야 할 필요는 없을 것이다(이를테면, 태그가 0 이외의 접속암호를 가져야 하는 것은 아니다). 이는 태그를 특수한 "pingable(ping이 가능한)" 상태에 배치함으로써 블라스트 작업을 할 준비를 할 때 "임의" 상태로 되돌아가기 때문이며, 이전에 "안전한(secure)" 상태에 있었더라도 마찬가지이다(즉 정확한, 0 이외의 AcqG2AccessPwd를 제시함). 이 때문에 접속 암호 작성은(태그 보안을 활성화하게 됨) 가능하다면 지양해야 한다.

이 작업이 "대량" 작업인 특성 상, BlastWrite 작업 중에 태그로부터의 데이터를 반환할 수 있는 경우는 없다.

ALIENG2BLASTLOCK (0X85) COMMAND & RESULTS

F800 | 9900 | 9680 | 9650

AlienG2BlastLock 명령어는 한번에 필드 내 모든 태그에 대하여 메모리 뱅크/필드의 잠김 상태를 변경한다. "field"라는 용어를 "bank" 대신 사용하는 이유는 예약된(Reserved) 뱅크인 뱅크 0에는 Kill Password와 Access Password가 포함되어 있고, 각기 독립적으로 잠글 수 있기 때문이다. 사용자는 원하는 대로 필드를 잠그거나 잠금 해제할 수 있으나, 필드를 한번만 PermaLock 또는 PermaUnlock할 수 있다.

사용자는 인수로 잠글 `<field>`와 원하는 `<lockType>` 항목을 통과한다.

```
AcqG2Ops = <opsNum> AlienG2BlastLock <field> <lockType>
AcqG2Ops = <opsNum>        85        <field> <lockType>

// Lock the User Banks of all tags
AcqG2Ops = 1 AlienG2BlastLock 3 Lock

// PermaLock the Kill Passwords of all tags
AcqG2Ops = 1 AlienG2BlastLock 0 PermaLock
```

<field>는 다음 정수 중 하나에 해당한다:

```
 0 = Kill password
10 = Access password
 1 = EPC bank
 2 = TID bank
 3 = USER bank
```

<lockType>는 다음 문자열 또는 그 단축키 중 하나에 해당한다.

```
 L or Lock
 U or Unlock
PL or PermaLock
PU or PermaUnlock
```

AlienG2BlastLock 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful AlienG2BlastLock result
85 00
```

ALIENG2BLASTERASE (0X86) COMMAND & RESULTS

F800 | 9900 | 9680 | 9650

AlienG2BlastErase 작업은 한번에 필드 내 모든 태그의 태그 메모리 중 지정된 범위를 삭제한다(00으로 설정함). 사용자는 지울 단어로 시작하여 지울 단어 수 등의 내용으로 뱅크를 지정해야 한다. 이 명령어는 유사한 G2Write보다 약간 더 빠른데, 0으로 채운 바이트를 모두 전송할 필요는 없기 때문이다.

```
AcqG2Ops = <opsNum> AlienG2BlastErase <bank> <wordPtr> <wordCount>
AcqG2Ops = <opsNum>        86         <bank> <wordPtr> <wordCount>

// Erase the first four words of USER memory of all tags
AcqG2Ops = 1 AlienG2BlastErase 3 0 4
```

AlienG2BlastErase 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful AlienG2BlastErase result
86 00
```

ALIENG2BLASTWRITE (0X87) COMMAND & RESULTS

F800 | 9900 | 9680 | 9650

AlienG2BlastWrite 작업은 한번에 필드 내 모든 태그에 대하여 태그의 쓰기 가능한 구획에 저수준 쓰기를 수행한다. 사용자는 뱅크와 처음 작성할 워드 포인터, 뒤이어 쓰기 할 데이터를 지정해야 한다. AlienG2BlastWrite에는 증분 옵션이 없는데, 동일한 명령어를 한번에 모든 태그에 전송해야 하기 때문이다.

```
AcqG2Ops = <opsNum> AlienG2BlastWrite <bank> <wordPtr> <2n:data…>
AcqG2Ops = <opsNum>        87         <bank> <wordPtr> <2n:data…>

// Write a kill password
AcqG2Ops = 1 AlienG2BlastWrite 0 0 DE AD DE AD
```

AlienG2BlastWrite 에 대한 AcqG2Ops 결과에 포함된 <data>는 곧 작업의 결과 코드이다.

```
// Successful AlienG2BlastWrite result
87 00
```

ALIENG2BLASTBLOCKPERMALOCK (0X88) COMMAND & RESULTS

F800 | 9900 | 9680 | 9650

AlienG2BlastBlockPermalock 작업은 한번에 필드 내 모든 태그의 사용자 메모리에서 특정 블록들을 영구적으로 쓰기 잠금 한다. 여러 태그와 태그 제조자 별로 사용자 메모리의 '블록'이 얼마나 큰지 정의가 다르다. 사용자는 처음 블록 번호를 지정하고, 하나 이상의 16비트 마스크로 메모리의 각 해당 블록에서 원하는 PermaLock 상태를 표시해야 한다. 대부분의 현재 태그에는 <= 16 사용자 블록이 있어서, 일반적으로 <blockPtr> 는 0이며, 단 하나의 <2:blockMask> 만 있다.

```
AcqG2Ops = <opsNum> AlienG2BlastBlockPermalock <blockPtr> <2:blockMask1>…
AcqG2Ops = <opsNum>             88             <blockPtr> <2:blockMask1>…

// Permanently lock the first two User blocks of all tags
// “C0 00” = 11000000 00000000 → first two blocks locked
AcqG2Ops = 1 AlienG2BlastBlockPermalock 0 C0 00
```

AlienG2BlastBlockPermalock 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful AlienG2BlastBlockPermalock result
88 00
```

ALIENG2BLASTUSERREADLOCK (0X8A) COMMAND & RESULTS

F800 | 9900 | 9680 | 9650

AlienG2BlastUserReadLock 작업은 Alien Higgs 기반의 태그 기능을 활용하여 사용자가 블록에 "읽기 잠금" 기능을 적용하여 사용자 메모리의 개별 블록을 감출 수 있게 해준다. 이 명령어의 Blast 형은 한번에 필드 내 모든 눈에 보이는 태그에서 작동한다.

읽기 잠금 상태의 블록은 사용자가 정확한 AcqG2AccessPwd를 제시하지 않고 읽으려 하면 0이나 오류를 반환한다. 사용자 메모리의 개별 블록들은 <1:blockMask> 인수를 사용해 G2BlockPermalock 명령어와 유사하게 주소를 지정한다. 단, 이 마스크는 G2BlockPermalock에 사용하는 16비트 단어가 아닌, 단일 바이트라는 점에서 차이가 있다. <1:blockMask>에 있는 비트는 사용자 메모리에 있는 블록과 상응하며, 최상위 비트에서 최하위 비트의 순이다.

```
AcqG2Ops = <opsNum> AlienG2BlastUserReadLock <1:blockMask>
AcqG2Ops = <opsNum>           8A             <1:blockMask>

// Hide the first 3rd and 4th blocks of USER memory of all tags
// “30” = 00110000 → 3rd and 4th blocks
AcqG2Ops = 1 AlienG2BlastUserReadLock 30
```

AlienG2BlastUserReadLock 작업의 결과는 성공/오류 코드뿐이다.

```
// Successful AlienG2UserReadLock result
8A 00
```

AcqG2OpsMode

F800 | 9900 | 9680 | 9650

AcqG2OpsMode 명령어는 구성한 AcqG2Ops 작업들을 리더가 수행하는 태그 인벤토리로 실행할 것인지 여부를 제어한다. 이 기능이 꺼지면(기본값), AcqG2Ops는 사용자가 "to" 명령어를 전송할 때에만 실행된다. 이 기능이 켜지면, AcqG2Ops는 리더가 태그를 수집할 때마다 실행되며, “t”나 “get taglist” 명령어를 실행하는지 여부는 관계 없이 실행된다. 또는 Automode가 백그라운드로 수행되는 인벤토리에서 실행될 때에도 실행된다.

- 허용된 값: "ON" | "OFF"
- 기본값: "ON"

AcqG2OpsMode 예제	
Command Response	>AcqG2OpsMode? AcqG2OpsMode = Off
Command Response	>AcqG2OpsMode = On AcqG2OpsMode = On

AcqTime

F800 | 9900 | 9680 **|** 9650

AcqTime 명령어는 인벤토리 지속시간을 제한하고 지정된 시간을 넘으면 인벤토리를 취소한다. 인벤토리를 취소해도 이미 수집된 태그를 버리지는 않는다. 리더나 RFID 프로토콜에 시간 상의 제약이 있기 때문에 인벤토리 지속시간이 명령어에 지정한 값과 정확히 일치할 것으로 기대해서는 안 된다. 이 명령어의 형식은 다음과 같다.

```
AcqTime = <time>
```

여기서, <time>은 밀리세컨드 단위의 인벤토리 지속시간이다(0 ~ 30000 msec). AcqTime=0이면 인벤토리 시간 제약 조건을 지운다.

- Allowed Values: Integer (0..30000), in milliseconds
- Default Value: 0 (no inventory timing restrictions)

AcqTime 예제	
Command Response	>AcqTime? AcqTime = 0
Command Response	// abort the inventory if it takes longer than 2 seconds >AcqTime = 2000 AcqTime = 2000

SpeedFilter

F800 | 9900 | 9680 **|** 9650

SpeedFilter 명령어를 사용하면 속도를 근거로 어떤 태그를 인지해서 TagList에 배치할 것인지를 지정할 수 있다. 필터의 해당 범위 내에서 계산된 속도로 태그를 읽은 것만 리더가 인지한다. 포함시킬 범위(예: -1m/s ~ +1m/s) 또는 제외시킬 범위(예: -5m/s 미만 또는 +5m/s 초과)를 지정할 수 있다.

속도 범위는 두 속도 조건인S1과 S2로 정의하며, 사용자가 최대 네 가지 별도의 범위를 지정할 수 있으며, 각각 수직막대(|)로 구분한다. 각각의 속도 범위가 특정 안테나에서만 작동하도록 구성할 수도 있다. 속도가 이 범위들 중 최소한 한 개에 부합하면 태그를 보고한다. 속도 단위는 미터/초이다.

```
SpeedFilter = S1, S2 [ | S3, S4]…
```

콤마는 필요하지 않다.

```
SpeedFilter = S1 S2 [ | S3 S4]…
```

S1 < S2일 때, 속도가 S1-S2 범위 내에 있는 태그는 모두 보고한다.
S1 > S2일 때, 속도가 S2-S1 범위 밖인 태그는 모두 보고한다.

사용자는 각각의 범위 내에 제3의 매개변수를 포함시켜서 별도의 안테나에 다른 SpeedFilter 범위를 지정할 수도 있는데, 즉 특정 범위가 활성화된 안테나의 정수 비트맵을 지정하는 것이다. 안테나 비트맵의 저순위 비트는 해당 리더 상에서 번호가 부여된 각 안테나 포트에 상응한다. 예를 들어, 속도 범위가 1순위의 두 안테나에서만 활성화되게 하려면, 안테나 비트맵 “3”을 사용하면 된다(3 = 00000011 2진수이므로) SpeedFilter = 0 999 3일 때에는 멀어지는 1순위 2개의 안테나 상 에 있는 태그만 필터링 한다(속도는 0 ~ 999). 안테나 비트맵을 지정하지 않으면, 리더는 사용자가 "모든 안테나"를 사용 할 것으로 가정하고 15(4 포트 리더의 경우) 또는 3(2 포트 리더의 경우) 값을 사용한다.

사용자의 필터 범위 내에서 안테나 비트맵을 사용할 경우, 사용자의 비트맵에 부합하지 않는 안테나로 들어오는 읽기 값은 무시될 것이다. 태그 읽기 값은 보고를 위해 최소한 한 개의 필터를 통과해야 하며,

안테나 비트맵 중에 해당 태그 읽기 값의 안테나에 부합하지 않으면, 데이터는 버려진다.

```
SpeedFilter = S1 S2 [A1] [ | S3 S4 [A2]]…
```

기본적으로 속도가 정확히 계산되지 않는 태그는(태그 또는 소음 정보를 얻는 중 통신이 취소된 이유로) 버려지고 태그리스트에 보고되지 않는다. SpeedFilter의 "nospeed" 옵션은 리더가 이 태그들을 태그리스트에 계속 남겨두게 하지만, 'non valid(유효하지 않은)' 속도값을 지정한다. 이런 방식으로 태그를 계속 읽을 수 있으며, 속도값이 유효하지 않다는 것도 쉽게 확인할 수 있다. 유효하지 않은 속도값을 가진 태그의 방향은 '0'으로 보고된다.

- SpeedFilter를 켜면 리더는 태그를 읽는 동안 추가 작업을 할 수 있으며, 이는 읽기 성능에 영향을 미칠 수 있다.
- 사용자는 TagList에 있는 각 태그의 실제 속도를 파악하고 %s 또는 ${SPEED} 표시를 포함한 사용자 정의 TagList 형식을 지정할 수 있다.
- SpeedFilter를 비활성화시키려면, 0으로 설정하면 된다. 이는 기본 설정값이다.
- 속도를 계산할 수 없는 태그를 포함시키려면 'nospeed' 옵션을 추가한다.

SpeedFilter 예제	
Command Response	>SpeedFilter? SpeedFilter = 0
Command Response	// read only tags essentially stationary (inclusive range) >SpeedFilter = -0.5 0.5 SpeedFilter = -0.5 0.5
Command Response	// read only tags that are moving (exclusive range) >SpeedFilter = 1 -1 SpeedFilter = 1 -1
Command Response	// read only tags going between -3 and -5 m/s or 0.5 and 1 m/s >SpeedFilter = -5 -3 | 0.5 1 SpeedFilter = -5 -3 | 0.5 1

Command
Response
	// read ALL tags (do not filter based on speed values and
	// also report tags for which speed could not be calculated)
	>SpeedFilter = nospeed
	SpeedFilter = nospeed

Command
Response
	// read tags with speeds between -1 and 1 m/sec as well as
	// tags for which speed could not be calculated
	>SpeedFilter = -1, 1 | nospeed
	SpeedFilter = -1, 1 | nospeed

Command Response	// read only stationary tags on ant #0 and moving tags on ant #1 >SpeedFilter = -0.5 0.5 1 | 1 -1 2 SpeedFilter = -0.5 0.5 1 | 1 -1 2

RSSIFilter

F800 | 9900 | 9680 | 9650

RSSIFilter 명령어를 사용하면 특정한 신호 강도의 태그만 볼 수 있다. 신호가 강하거나(일반적으로 더 근접) 약한(일반적으로 더 먼 거리) 태그만 읽기 원할 수도 있다. SpeedFilter와는 달리, RSSIFilter를 활성화시켜도 리더가 태그를 읽을 때 추가 작업을 수행해야 하는 것은 아니며, 따라서 읽기 성능이 영향을 받지 않는다.

RSSI 범위는 두 값인 R1과 R2로 정의하며, 사용자가 최대 네 가지 별도의 범위를 지정할 수 있으며, 각각 수직막대(|)로 구분한다. 각각의 RSSI 범위가 특정 안테나에서만 작동하도록 구성할 수도 있다.

RSSI가 이 범위들 중 최소한 한 개에 부합하면 태그를 보고한다. 리더가 보고한 RSSI 값은 단위가 정해져 있지 않고, 상이한 리더 모델들이 동일한 조건이라도 각기 다른 RSSI값을 보고할 수 있는데, 이는 무선신호 아키텍처가 다르기 때문이다. RSSIFilter 값을 리더 별로 조율하여 여러 기업체들 간에 결과의 일관성을 확보할 필요가 있을 수 있다.

```
RSSIFilter = R1, R2 [ | R3, R4]…
```

콤마는 필요하지 않다.

```
RSSIFilter = R1 R2 [ | R3 R4]…
```

R1 < R2일 때, RSSI가 R1-R2 범위 내에 있는 태그는 모두 보고한다. 이는 포함 범위이다.
R1 > R2일 때, RSSI가 R1-R2 범위 밖에 있는 태그는 모두 보고한다. 이는 제외 범위이다. 이 기능은 큰 RSSI, 이를테면 10,000를 초과하는 RSSI를 가진 태그를 필터링하는 데 유용하다. “10000 0” 범위는 RSSI가 10000 초과이거나 0 미만(불가능함)인 태그를 필터링한다.

사용자는 각각의 범위 내에 제3의 매개변수를 포함시켜서 별도의 안테나에 다른 RSSIFilter 범위를 지정할 수도 있는데, 즉 특정 범위가 활성화된 안테나의 정수 비트맵을 지정하는 것이다. 안테나 비트맵의 저순위 비트는 해당 리더 상에서 번호가 부여된 각 안테나 포트에 상응한다. 예를 들어, RSSI 범위가 1순위의 두 안테나에서만 활성화되게 하려면, 안테나 비트맵 “3”을 사용하면 된다(3 = 00000011 2진수): RSSIFilter = 5000 0 3은 1순위의 강한 안테나 2개 있는 태그만 필터링 한다. (RSSI가 5000을 초과함). 안테나 비트맵을 지정하지 않으면, 리더는 사용자가 "모든 안테나"를 사용할 것으로 가정하고 15(4 포트 리더의 경우) 또는 3(2 포트 리더의 경우) 값을 사용한다.

사용자의 필터 범위 내에서 안테나 비트맵을 사용할 경우, 사용자의 비트맵에 부합하지 않는 안테나로 들어오는 읽기 값은 무시될 것이다. 태그는 보고를 목적으로 필터 중 최소 하나는 통과해야 하며, 안테나 비트맵 중에 해당 태그 읽기 값의 안테나에 부합하지 않으면, 데이터는 버려진다.

```
RSSIFilter = R1 R2 [A1] [ | R3 R4 [A2]]…
```

- RSSIFilter는 리더가 추가 작업을 수행할 것을 요구하지 않으므로 읽기 성능에 부정적 영향을 미치지 않는다.
- 사용자는 TagList에 있는 각 태그의 실제 RSSI값을 파악하고 %s 또는 ${RSSI} 표시를 포함한 사용자 정의 TagList 형식을 지정할 수 있다.
- RSSIFilter를 비활성화시키려면, 0으로 설정하면 된다. 이는 기본 설정값이다.

RSSIFilter 예제	
Command Response	>RSSIFilter? RSSIFilter = 0
Command Response	// Show only tags with RSSI between 0 and 35000 > RSSIFilter = 0 35000 RSSIFilter = 0 35000
Command Response	// Show only tags with RSSI lower than 5000 or higher than 35000 > RSSIFilter = 35000 5000 RSSIFilter = 35000 5000

Command
Response
	// Show only weak tags on ant #0,1 and string tags on ant #2,3
	// antennas 0 & 1 : 00000011_{binary} = 3 for antenna bitmap
	// antennas 2 & 3 : 00001100_{binary} = 12 for antenna bitmap
	> RSSIFilter = 0 1000 3 | 1000 0 12
	RSSIFilter = 0 1000 3 | 1000 0 12

TagStreamCountFilter

F800 | 9900 | 9680 | 9650

TagStreamCountFilter를 사용하면 사용자는 일단 주어진 태그를 최소한의 횟수만큼 읽었거나 및/또는 최대 계수에 도달할 때까지는 리더가 태그 데이터만 스트리밍하도록 지시할 수 있다.

```
TagStreamCountFilter = min [max]
```

`min` (최소값)만 지정할 경우에는, 태그는 리더 태그리스트 내 해당 태그의 현재 계수값이 `min`이상일 경우에만 스트리밍된다. `max` 값을 지정할 경우, 태그는 이 `max`값을 초과하면 더 이상 스트리밍 되지 않는다. `min` 과 `max` 의 허용 범위는 0...65535이며, `max` (지정한 경우)는 `min`보다 반드시 커야 한다.

기본값은 "`0`"이며, 이 조건에서는 TagStreamCountFilter가 꺼지고 리더는 모든 태그를 스트리밍한다. 또한 "`TagStreamCountFilter = 1`"일 때에도 같은 효과를 낸다.
태그를 한 번만 스트리밍하려면, `TagStreamCountFilter = 0 1`를 사용한다.

태그를 스트리밍할 것인지를 결정하기 위해 사용하는 현재 태그 계수는 태그리스트의 PersistTime 개념에 따라 달라진다는 점에 유의해야 한다. 또한 TagStreamMode를 전적으로 사용자의 태그 데이터를 얻기 위해서만 사용할 경우, -1이 아닌 수로 PersistTime을 설정해야 한다. 그렇지 않으면 리더의 내부 태그리스트는 데이터 퍼지가 되지 않을 것이다.

TagStreamCountFilter 예제	
Command Response	>TagStreamCountFilter? TagStreamCountFilter = 0
Command Response	*// stream tags only once* >TagStreamCountFilter = 0 1 TagStreamCountFilter = 0 1

TagAuth

F800 | 9900+ | 9900 | 9680 | 9650

TagAuth 명령어는 Alien Higgs3과 Higgs4에 기초한 태그에서 이용 가능한 Alien 동적 태그 인증 기능을 지원한다. TagAuth를 활성화하면, 리더는 태그를 읽는 동안 추가적인 확인 절차를 수행하여 인증된 Alien 태그가 맞는지 확증한다.

TagAuth를 활성화하면 리더는 태그를 읽는 동안 추가 작업을 할 수 있으며, 이는 읽기 성능에 영향을 미칠 수 있다. 이 명령어의 형식은 다음과 같다.

```
TagAuth = off | on | h3 | -h3 | h4 | -h4 | * | not
```

여러 옵션을 사용할 경우에는 '|' 분리자를 넣어 조합한다. 다음과 같은 옵션이 있다.

옵션	리더 인증	리더 보고
off	인증하지 않음	모든 태그
on	인증 가능한 모든 태그 유형 (Alien Higgs3 & Higgs4)	모든 태그(인증하지 않은 태그와 인증 프로토콜을 지원하지 않는 태그 포함)
h3	Alien Higgs3 태그	인증된 Higgs3 태그만
-h3	Alien Higgs3 태그	인증 실패한 태그. 이를테면, 제조자의 ID를 Higgs3로 설정하였으나 확증 테스트에서 실패한 태그
h4	Alien Higgs4 태그	인증된 Higgs4 태그만
-h4	Alien Higgs4 태그	인증 실패한 태그. 이를테면, 제조자의 ID를 Higgs4로 설정하였으나 확증 테스트에서 실패한 태그
*	h3/-h3/h4/-h4 옵션과 관련해 사용하여 리더가 발견한 모든 태그를 보고하게 함. 여기에는 인증되지 않은 태그와 인증 프로토콜을 지원하지 않는 태그도 포함됨. 이 옵션은 TaglistFormat을 '사용자 정의'로 설정한 경우에 유용하며, 태그 인증 표시 ${AUTH} 를 포함한다(아래 예 참조).	
not	h3/-h3/h4/-h4 옵션과 관련해 사용하여 옵션의 의미를 역전시킴. 둘 이상의 옵션을 '|' 분리자를 사용해 지정하면, 'not'이 전체 옵션들에 적용된다. 'not' 옵션은 'on' 또는 'off'와 함께 사용할 수는 없다.	

${AUTH} 사용자 정의 태그 목록 형식 표시를 사용하면 태그 인증에 관한 정보를 보고한다. 보고된 값에는 태그 제조자의 id 코드는 물론 다음과 같이 Alien 동적 태그 인증 결과를 보여주는 기호도 포함한다.

+ 인증 성공함
- 인증 실패함

제조자의 id 다음에 기호가 없으면 태그가 Alien 동적 인증을 지원하지 않거나 인증 절차가 실패하였음을 의미한다.

리더가 태그에서 제조자의 id 코드를 검색할 수 없는 경우에는(예를 들어, 소음이나 트랜잭션 실패로 인해), ${AUTH} 필 드는 '?' 기호로 표시된다.

TagAuth 예제	
Command Response	>TagAuth? TagAuth = OFF
Command Response	*// Dynamically Authenticate Alien Higgs-3 tags and only report Higgs-3 tags* *>TagAuth = h3* TagAuth = h3
Command Response	>TagAuth = h3 | * TagAuth = h3 | * *// Dynamically authenticate Higgs-3 tags but also report ALL other tags* *// found no matter if they passed or failed authentication or whether* *// they support authentication protocol or not.* *// This setting is useful when tag list format is set to 'custom' and* *// includes the Tag Authentication token ${AUTH} as it allows one to see* *// authentication properties as well as tag's manufacturer's codes for* *// all tags in the field.* *// Currently there is only one type of authentication (Higgs-3) supported,* *// so the 'h3 | *' setting happens to have the same effect as the* *// option 'on' (authenticate all known tag types and report all tags)*
Command Response	>TagAuth = -h3 TagAuth = -h3 *// Authenticate Higgs-3 tags and only report tags that **failed** dynamic* *// authentication. This could be an indication that this is a fake Higgs-3 tag.* *// Note that the reader will not report tags not supporting Alien Dynamic* *// Authentication, for example Alien Higgs-2 tags.*
Command Response	>TagAuth = h3 | -h3 TagAuth = h3 | -h3 *// Authenticate Higgs-3 tags and only report Alien Higgs-3 tags* *// that **passed** verification and/or tags that have Alien Higgs-3* *// signature (manufacturer's Id) but **failed** the verification procedure.* *// Other types of tags (for example, Higgs-2) will not be reported.*
Command Response	>TagAuth = not h3 TagAuth = not h3 *// Authenticate Higgs-3 tags and **exclude** tags that pass verification* *// process, i.e. the reader will not report tags that are true Alien* *// Higgs-3 tags but other tags like Alien Higgs-2 or Higgs-3 tags that* *// failed authentication, or tags from other manufacturers will be reported.*
Command Response	// authenticate Alien Higgs-3 or Higgs-4 tags, also report all other tags Alien>TagAuth = h3|h4|* TagAuth = h3|h4|* Alien>TagListCustomFormat = %k auth:${AUTH} TagListCustomFormat = %k auth:${AUTH} Alien>TagListFormat = custom TagListFormat = Custom Alien>t A5A5A5A5A5A5A5A5A5000A auth:E200 3411 // Higgs-2 A5A4A5A5A5ABA5A5A5A50019 auth:E200 3412+ // Higgs-3 authenticated 5068696C69707300AAAAAAAA auth:E200 4001 // not Alien tag A5A4A5A5A5ABA5A5A5A5001E auth:? // unable to retrieve tag information A5A4A5A5A5ABA5A5A5A5002C auth:E200 3414+ // Higgs-4 authenticated A5A4A5A5A5ABA5A5A5A5000C auth:E200 3412- // Higgs-3 authentication FAILED 535410012002300340045005 auth:E200 7240 // not Alien tag 4E585000AC1F3681EC880468 auth:E200 6003 // not Alien tag E2003412DC03011706095651 auth:E200 3412+ // Higgs-3 authenticated 4D32494D3333777755556666 auth:E200 1071 // not Alien tag 48322D3015081509150A150B auth:E200 3411 // Higgs-2 A5A4A5A5A5ABA5A5A5A5002F auth:E200 3412+ // Higgs-3 authenticated

AutoMode Commands

AutoMode는 태그의 '핸드-프리' 모니터링을 가능하게 하는 작동 모드이다. 설정하려면 일련의 구성 명령어들을 리더에 전송해야 한다. 이 명령어들은 태그를 읽는 방법과 시점을 자세히 설정하며 태그를 발견한 때, 어느 대상에 지시할지도 명시한다. 일단 구성했으면, 리더는 스스로 작동할 수 있다.

AutoMode 시스템에 대한 자세한 설명을 원한다면, 이 안내서 2장을 참조하기 바란다.

AutoMode

F800 | 9900 | 9680 | 9650

AutoMode 명령어는 자율 모드를 켜거나 끌 수 있다.

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"
- AutoMode를 켜면, 리더는 외부 출력 상태를 기록한다.
  AutoMode를 끄면, 리더는 AutoWaitOutput = -1가 아닌 한 저장되어 있는 외부 출력 상태를 재적용한다.

AutoMode 예제	
Command Response	>AutoMode? AutoMode = ON
Command Response	>AutoMode=on AutoMode = ON

AutoWaitOutput

F800 | 9900 | 9680 | 9650

AutoWaitOutput은 AutoMode 대기 상태 중 적용할 외부 출력 설정값을 지정한다. 매개변수는 4개의 외부 출력값에 대한 비트맵으로서, "1"은 출력을 높게, "0"은 낮게 설정한다. 출력 #1은 마스크에 있는 비트로 지정하고, 출력 #2는 마스크에 있는 비트 1로 지정하는 방식이다.

이것이 AutoMode 초기 상태이고, AutoWaitOutput=0으로 설정하면 AutoMode 주기가 시작할 때마다 모든 출력값이 지워진다.

- 허용된 값: 정수(-1..255) (리더 모델에 따라 달라질 수 있음)
- 기본값: 0
- AutoWaitOutput = -1로 설정하면 리더는 대기 상태로 진입할 때 외부 출력 설정을 변경하지 않는다. 일반적으로 AutoMode를 끄면 외부 출력을 AutoMode를 켰을 때의 상태로 재설정한다. AutoWaitOutput = -1일 때는 이렇게 되지 않는다.

AutoWaitOutput 예제	
Command Response	>AutoWaitOutput? AutoWaitOutput = 0
Command Response	>AutoWaitOutput = 3 (sets outputs #1 and #2 high) AutoWaitOutput = 3

AutoStartTrigger

F800 | 9900 | 9680 | 9650

AutoStartTrigger는 모니터링할 외부 출력을 지정하여 자율모드가 대기 상태에서 작동 상태로 넘어가게 할 수 있다. 트리거는 저-고(상승) 또는 고-저(하강) 형태의 입력 조건을 만들 수 있다. 각 변경 유형 별로, 정수 비트맵을 제공하여 변경 내용을 청취하도록 입력값을 지정해야 한다.

이 명령어는 두 개의 공간 분리형 매개변수, 즉 상승형 비트맵과 하강형 비트맵을 취한다. 이 비트맵들은 “OR” 비트맵으로서, 입력값 변동이 비트맵에 있는 설정 비트에 상응할 경우, AutoMode가 시작한다. 리더는 한 번에 상승 혹은 하강 입력만 청취할 수 있으며 한 번에 둘 모두는 불가능하다.

AutoStartTrigger를 "-1 -1"으로 설정할 수도 있다. 이렇게 하면 리더는 외부 입력값에 관계 없이 대기 상태를 유지한다. 사용자는 AutoModeTriggerNow 명령어로 AutoMode 주기를 한번 시작할 수 있다. 이로써 Automode 상태의 기기에서 어느 정도 수동으로 제어가 가능하다(소프트웨어를 통해).

AutoStartTrigger 예제	
Command Response	>AutoStartTrigger? AutoStartTrigger = 0 0 (rising, falling)
Command Response	>AutoStartTrigger =3 0 AutoStartTrigger = 3 0 (rising, falling)

AutoStartPause

F800 | 9900 | 9680 | 9650

AutoStartPause 기능을 사용하면 리더는 시작 트리거를 받은 후 실제 시작하기 전에 지정된 밀리세컨드 시간 동안 대기한다.

이로써 사용자는 컨베이어 등의 상황을 감안하여 대처할 수 있는데, 이런 경우 사용자는 'photo-eye'로 리더를 트리거하고 싶지만 'photo-eye'를 리더에서 어느 정도 떨어진 거리에 배치해야 할 수 있다. AutoStartPause를 패키지가 photo-eye 빔을 지나치고 나서 읽기 영역에 도달할 때까지의 시간에 가깝게 값을 설정함으로써, 사용자는 불필요한 RF 작업을 줄이고 다른 태그에서 잘못 읽을 가능성을 줄일 수 있다. 이런 일들은 아직도 필드에서 실제로 발생하곤 한다.

- 허용된 값: 정수(0..86400000)
- 기본값: 0 msec

AutoStartTrigger 예제	
Command Response	>AutoStartPause? AutoStartPause = 0
Command Response	>AutoStartPause = 150 AutoStartPause = 150

AutoWorkOutput

F800 | 9900 | 9680 | 9650

AutoWorkOutput은 AutoMode 작업 상태 중 적용할 출력 설정값을 지정한다. 매개변수는 4개의 외부 출력값에 대한 비트맵으로서, "1"은 출력을 높게, "0"은 낮게 설정한다. 출력 #1은 마스크에 있는 비트로 지정하고, 출력 #2는 마스크에 있는 비트 1로 지정하는 방식이다.

- 허용된 값: 정수(-1..255) (리더 모델에 따라 달라질 수 있음)
- 기본값: 0
- AutoWorkOutput = -1로 설정하면 리더는 작업 상태로 진입할 때 외부 출력 설정을 변경하지 않는다.

AutoWorkOutput 예제	
Command Response	>AutoWorkOutput? AutoWorkOutput =0
Command Response	>AutoWorkOutput =3 (sets outputs #1 and #2 high) AutoWorkOutput =3

AutoAction

F800 | 9900 | 9680 | 9650

AutoAction 명령어는 AutoMode 작업 상태에서 실행할 때 수행할 동작을 지정한다. 리더는 작업 상태 중 계속해서 AutoAction을 수행할 수 있다. AutoAction이 프로그래밍 명령어라면, 리더는 이 작업이 성공하거나 작업 상태가 완료될 때까지(타이머나 외부 트리거로 인해) 이 작업을 계속 시도할 것이다.

AutoAction	설명
None	작업 수행하지 않음
Acquire	태그 인벤토리 수행 이는 기본 설정값이다.
ProgramEPC ("Program")	태그의 EPC 프로그래밍. 더 자세한 정보는 '태그 프로그래밍' 섹션 참조.
Erase	태그의 EPC 지우기 더 자세한 정보는 '태그 프로그래밍' 섹션 참조.
ProgramAndLockEPC ("Program and Lock")	태그의 EPC 프로그래밍 및 잠금 더 자세한 정보는 '태그 프로그래밍' 섹션 참조.
ProgramUser	태그의 사용자 메모리 프로그래밍 더 자세한 정보는 '태그 프로그래밍' 섹션 참조.
ProgramAndLockUser	태그의 사용자 메모리 프로그래밍 및 잠금 더 자세한 정보는 '태그 프로그래밍' 섹션 참조.
ProgramAlienImage	전체 태그 이미지를 Alien Higgs2 또는 Higgs3 태그에 작성.
Macro <macroname>	명명한 리더 매크로를 실행.

AutoAction 예제	
Command Response	>AutoAction? AutoAction = Acquire
Command Response	>AutoAction = Acquire AutoAction = Acquire

AutoStopTrigger

F800 | 9900 | 9680 | 9650

AutoStopTrigger는 모니터링할 외부 출력을 지정하여 자율모드가 작동 상태에서 평가 상태로 넘어가게 할 수 있다. 트리거는 저-고(상승) 또는 고-저(하강) 형태의 입력 조건을 만들 수 있다. 각 변경 유형 별로, 정수 비트맵을 제공하여 변경 내용을 청취하도록 입력값을 지정해야 한다.

이 명령어는 두 개의 공간 분리형 매개변수, 즉 상승형 비트맵과 하강형 비트맵을 취한다. 이 비트맵들은 "OR" 비트맵으로서, 입력값 변동이 비트맵에 있는 설정 비트에 상응할 경우, AutoMode가 정지한다. 리더는 한 번에 상승 혹은 하강 입력만 청취할 수 있으며 한 번에 둘 모두는 불가능하다.

AutoStopTrigger 예제	
Command Response	>AutoStopTrigger? AutoStopTrigger = 0 0 (rising, falling)
Command Response	>AutoStopTrigger = 3 0 AutoStopTrigger = 3 0 (rising, falling)

AutoStopTimer

F800 | 9900 | 9680 | 9650

AutoStopTimer는 작동 상태에서 평가 상태로 넘어갈 대안을 제공한다. 이 기능은 시간에 기반한 정지 트리거로서, 각 주기 중 AutoAction을 실행할 길이의 상한값을 설정한다. 이 매개변수는 밀리세컨드 단위로 지정하는 단일의 시간 길이이다.

- 허용된 값: Integer(-1..86400000)
- 기본값: 1000 msec
- AutoStopTimer = -1인 경우 리더는 작동 상태에서 또는 구성한 AutoStopTrigger를 수신할 때까지 무한정 지속될 것이다. AutoStopTimer나 AutoStopTrigger가 없다면, 리더는 평가/공지 상태로 결코 넘어가지 않으므로, 출력값이 변경되거나 공지 메시지가 전송되는 일이 없을 것이다.
- AutoStopTimer = 0인 경우 리더는 작동 상태에서 정확히 하나의 작업만 수행해야 하며(읽기, 프로그래밍 등), 그런 다음 평가 상태로 넘어간다.
- AutoStopTimer가 86,400,000 msec인 경우 24시간에 해당한다.

AutoStopTimer가 만료될 때 리더가 인벤토리 중이라면, 인벤토리는 완료된 후에 평가 상태로 넘어간다.

AutoStopTimer 예제	
Command Response	>AutoStopTimer? AutoStopTimer = 1000
Command Response	>AutoStopTimer = 5000 AutoStopTimer = 5000

AutoStopPause

F800 | 9900 | 9680 | 9650

AutoStoptPause 기능을 사용하면 리더는 정지 트리거를 받은 후 실제 정지하기 전에 지정된 밀리세컨드 시간 동안 대기한다. 이 지연 기능은 AutoAction이 AutoStopTimer 만료 때문이 아니라 외부 I/O 트리거 때문에 중단되었을 때에만 적용된다.

AutoStopPause를 사용하면 사용자는 컨베이어 등의 상황을 감안하여 대처할 수 있는데, 이런 경우 사용자는 'photo-eye'로 리더를 트리거하여 시작 및 정지하게 하고 싶지만 'photo-eye'를 리더에서 너무 가깝게 배치해야 할 수 있다. AutoStopPause를 패키지가 photo-eye 빔을 지나치고 나서 유효한 읽기 영역에 도달할 때까지의 시간에 가깝게 값을 설정함으로써, 사용자는 정지 트리거를 발한 이후에도 리더가 계속 작업을 하도록 할 수 있다.

- 허용된 값: Integer(0..86400000)
- 기본값: 0 msec

AutoStopPause 예제	
Command Response	>AutoStopPause? AutoStartPause = 0
Command Response	>AutoStartPause = 150 AutoStartPause = 150

AutoTrueOutput

F800 | 9900 | 9680 | 9650

AutoTrueOutput은 AutoMode 모드가 '참'으로 평가할 경우 적용할 출력 설정값을 지정한다. 매개변수는 4개의 외부 출력값에 대한 비트맵으로서, "1"은 출력을 높게, "0"은 낮게 설정한다. 출력 #1은 마스크에 있는 비트로 지정하고, 출력 #2는 마스크에 있는 비트 1로 지정하는 방식이다.

- 허용된 값: 정수(-1..255) (리더 모델에 따라 달라질 수 있음)
- 기본값: 0
- AutoTrueOutput = -1 로 설정하면 리더는 '참' 상태 평가로 진입할 때 외부 출력 설정을 변경하지 않는다.

AutoTrueOutput 예제	
Command Response	>AutoTrueOutput? AutoTrueOutput = 0
Command Response	>AutoTrueOutput = 3 (sets outputs #1 and #2 high) AutoTrueOutput = 3

AutoTruePause

F800 | 9900 | 9680 | 9650

AutoTruePause는 자율 평가 모드가 '참'으로 평가할 때 적용할 밀리세컨드 단위의 정지 시간을 지정한다. 이 정지는 AutoTrueOutput 명령어를 처리한 후에 발생한다. 리더가 정지 중일 때에는 태그를 읽지 않기 때문에 성능에 영향을 미칠 수 있다.

- 허용된 값: 정수(0..86400000)
- 기본값: 0 msec

AutoTruePause 예제	
Command Response	>AutoTruePause? AutoTruePause(ms) = 0
Command Response	>AutoTruePause = 500 AutoTruePause = 500

AutoFalseOutput

F800 | 9900 | 9680 | 9650

AutoFalseOutput은 AutoMode 모드가 '거짓'으로 평가할 경우 적용할 출력 설정값을 지정한다. 매개변수는 4개의 외부 출력값에 대한 비트맵으로서, "1"은 출력을 높게, "0"은 낮게 설정한다. 출력 #1은 마스크에 있는 비트로 지정하고, 출력 #2는 마스크에 있는 비트 1로 지정하는 방식이다.

- 허용된 값: 정수(-1..255) (리더 모델에 따라 달라질 수 있음)
- 기본값: 0
- AutoFalseOutput = -1로 설정하면 리더는 '거짓' 상태 평가로 진입할 때 외부 출력 설정을 변경하지 않는다.

AutoFalseOutput 예제	
Command Response	>AutoFalseOutput? AutoFalseOutput = 0
Command Response	>AutoFalseOutput = 3 AutoFalseOutput = 3

AutoFalsePause

F800 | 9900 | 9680 | 9650

AutoFalsePause는 자율 평가 모드가 '거짓'으로 평가할 때 적용할 밀리세컨드 단위의 정지 시간을 지정한다. 이 정지는 AutoFalseOutput 명령어를 처리한 후에 발생한다. 리더가 정지 중일 때에는 태그를 읽지 않기 때문에 성능에 영향을 미칠 수 있다.

- 허용된 값: 정수(0..86400000)
- 기본값: 0 msec

AutoFalsePause 예제	
Command Response	>AutoFalsePause? AutoFalsePause (ms) = 0
Command Response	>AutoFalsePause = 500 AutoFalsePause = 500

AutoErrorOutput

F800 | 9900 | 9680 | 9650

AutoErrorOutput은 AutoMode 동작이 열거한 오류 중 하나를 '거짓'으로 평가할 때 적용할 출력 설정값을 지정한다. 부합하는 대상이 없으면, 출력은 AutoFalseOutput 값으로 설정된다. 첫 매개변수는 외부 출력값의 비트맵이며, 뒤이어 선택적으로 추가 오류 코드를 추가한다. 비트값이 "1"이면 출력을 높게, "0"이면 낮게 설정한다. 출력 #1은 비트맵의 첫(저순위) 비트로 지정하고, 출력 #2는 두 번째 비트로 지정하는 방식이다.

```
AutoErrorOutput = <output>[,err1 [,err2]...]
```

하나의 <output> 매개변수만 지정하면, '-1' (비활성화)가 된다. 이는 기본 설정값이다.

- 허용된 값:
  <output bitmap>: 정수 (-1..255) (리더 유형에 따라 달라질 수 있음)
  <err>: 정수 (1..255)

- 기본값: -1 (비활성화)
- AutoFalseOutput = -1로 설정하면 리더는 '거짓' 상태 평가로 진입할 때 외부 출력 설정을 변경하지 않는다.

AutoErrorOutput 예제	
Command Response	>AutoErrorOutput? AutoErrorOutput = -1
Command Response	// set output pin 2 if AutoMode action results in errors 134 or 149 >AutoErrorOutput = 4 134 149 AutoErrorOutput = 4 134 149

AutoProgError

F800 | 9900 | 9680 | 9650

AutoMode 프로그래밍 동작이 실패하면, 새 태그를 태그리스트에 추가하고 안테나 값은 255로 설정한다. AutoProgError 명령어를 사용하면 사용자정의 오류값을 선택하거나 실제 오류코드를 보고할 수 있다. AutoProgError를 '-1'으로 설정하면 실제 오류 코드를 보고한다.

```
AutoProgError = <error>
```

- 허용된 값: Integer(-1..255)
- 기본값: 255
- AutoProgError를 '-1'로 설정하면 실제 오류 코드를 보고한다.

AutoProgError 예제	
Command Response	>AutoProgError? AutoProgError = 255

Command
Response
	>AutoProgError = -1 // set to -1 to report actual error code
	AutoProgError = -1

AutoModeReset

F800 | 9900 | 9680 | 9650

AutoModeReset 명령어는 모든 자율모드 매개변수를 기본값으로 재설정하며, 여기에는 자율모드를 'off'로 설정하는 것도 포함된다.

AutoModeReset 예제	
Command Response	>AutoModeReset All AutoMode settings have been reset!

AutoModeTriggerNow

F800 | 9900 | 9680 | 9650

AutoModeTriggerNow 명령어는 외부 IO 트리거 이벤트를 에뮬레이트하여 AutoMode가 시작되게 한다. 이 명령어는 RFID 리더가 이미 AutoMode에 있고 시작 트리거 조건을 기다릴 때에만 작동한다. 이 시점에서, AutoModeTriggerNow 명령어를 전송하면 실제 외부 트리거 이벤트와 동일한 상태가 되어 AutoMode가 작동 주기를 시작한다.

AutoModeTriggerNow 예제	
Command Response	>AutoModeTriggerNow Auto Mode Triggering Now

Notify Mode Commands

공지모드(Notify Mode) 명령어를 사용하면 타이머 만료 시에나 리더가 AutoMode로 실행될 때 발생하는 이벤트를 트리거 해제했을 때 자율화된 이벤트 공지를 설정할 수 있다.

NotifyMode

F800 | 9900 | 9680 | 9650

NotifyMode 명령어는 공지 모드를 켜거나 끌 수 있다.

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"

NotifyMode 예제	
Command Response	>NotifyMode? NotifyMode = ON
Command Response	>NotifyMode = on NotifyMode = ON

NotifyAddress

F800 | 9900 | 9680 | 9650

NotifyAddress 명령어는 공지 메시지가 트리거 되었을 때 전송할 대상을 지정한다. 주소 형식에 따라 배송 방법이 결정된다.

지원되는 전송 방법이 네 가지 있는데, 아래 표에 제시하였다.

NotifyAddress	설명
hostname:port	메시지를 네트워크에 있는 기기 상의 지정된 포트로 전송한다. 주소는 “hostname:port”의 형식을 취한다. 예를 들어, “123.01.02.98:3450” 또는 “listener.alientechnology.com:10002”
user@domain.com	이메일 메시지를 지정된 주소로 전송한다. 주소는 표준형 이메일 형식으로 지정한다. 예: user@domain.com 유의사항: 이 경우 MailServer 매개변수를 반드시 설정해야 한다. MailFrom 매개변수를 사용하도록 선택할 수도 있다. 현 시점에는 이메일 인증 프로토콜은 지원되지 않는다.
http://domain:port/path	웹 서비스에 POST 요청으로 메시지를 전송한다. 주소는 “http://”로 시작해야 하며, 특정 웹 서비스 경로와 포트 번호(기본값은 80)를 지정할 수도 있다. 공지 메시지로 POST 메시지 내용을 채우고, 'key=value form-style' 형식은 사용하지 않는다. 대신 사용자가 입력 스트림을 직접 읽는다.
SERIAL	메시지를 직렬 연결 대상으로 전송한다. “SERIAL”이라는 단어를 주소에 사용한다. 단어의 대소문자 구별은 없다. ALR-F800: 이 경우 메시지를 SERIAL0 (RS-232)와 SERIAL1 (USB) 포트 모두에 전송한다.
(ALR-F800 only) **SERIAL0** **SERIAL1**	ALR-F800의 직렬포트 중 하나로 메시지를 전송한다. SERIAL0는 RS-232 포트로 메시지를 보낸다. SERIAL1은 USB 포트로 보낸다.

예비용 NotifyAddress를 지정하여 주요 NotifyAddress 공지가 실패할 때 사용할 수도 있다. 리더는 예비 주소를 사용하기 전에 늘 1차 주소를 먼저 시도해야 한다.

이 두 주소는 서로 다른 배송 프로토콜을 가리킨다. 예를 들어, 1차 주소는 TCP 소켓 주소이고, 예비용 주소는 이메일 주소일 수 있다.

사용자는 1차 주소에 이어 예비용 주소를 지정하고 다음과 같이 수직막대(|)를 둘을 분리한다.

```
NotifyAddress = <address1> | <address2>
```

NotifyAddress 예제	
Command Response	>NotifyAddress? NotifyAddress = 10.1.0.12:4000
Command Response	>NotifyAddress = user@msn.com NotifyAddress = user@msn.com
Command Response	// Specify a fail-over address >NotifyAddress = 10.1.0.12:4000 | user@msn.com NotifyAddress = 10.1.0.12:4000|user@msn.com

NotifyTime

F800 | 9900 | 9680 | 9650

NotifyTime 명령어는 청취(listening) 기기에 자동으로 TagList를 전송하는 시간 간격을 지정하고 검색한다.

- 허용된 값: Integer(0..65535)
- 기본값: 0 sec
- 이 시간은 초 단위로 지정한다.
- 이 값이 0이면, 시간 기반 자동 공지는 비활성된다.
- 양의 값이면 표준 공지 메시지를 해당 값을 간격으로 발송한다.

NotifyTime 예제	
Command Response	>NotifyTime? NotifyTime = 30
Command Response	>NotifyTime = 30 NotifyTime = 30

NotifyTrigger

F800 | 9900 | 9680 | 9650

NotifyTrigger 명령어는 이벤트 조건을 지정 및 검색하며(시간 기반이 아님), 이 조건을 기초로 공지 메시지를 전송한다. Notify 메시지는 다음 조건 하에서 발족할 수 있다.

Trigger	의미
Add	새로운 태그를 읽고 TagList에 추가할 때 메시지를 전송 추가된 태그만 전송한다.
Remove	TagList에서 태그를 제거할 때 메시지를 전송. 제거된 태그만 전송한다.
AddRemove	TagList에서 태그를 추가 또는 제거할 때 메시지를 전송. 추가한 태그 목록과 제거한 태그 목록을 전송.
Change	TagList에서 태그를 추가 또는 제거할 때 메시지를 전송. 현재 TagList를 전송.
True	자율 상태 루프의 평가가 '참'으로 평가될 때 메시지를 보낸다. 보통은 태그를 TagList에 추가할 때가 이에 해당된다. 현재 TagList를 전송.
False	자율 상태 루프의 평가가 '거짓'으로 평가될 때 메시지를 보낸다. 보통은 태그를 TagList에 추가하지 않을 때가 이에 해당된다. 현재 TagList를 전송.
TrueFalse	자율 상태 루프의 평가가 '참' 또는 '거짓'으로 평가될 때 메시지를 보낸다(예: 매 시간). 현재 TagList를 전송.

NotifyTrigger 예제	
Command Response	>NotifyTrigger? NotifyTrigger = Remove
Command Response	>NotifyTrigger = add NotifyTrigger = Add

NotifyFormat

F800 | 9900 | 9680 | 9650

NotifyFormat 매개변수는 공지 메시지 형식을 지정한다. 형식은 다음 중 하나이다.

Format	Description
Text	TagList는 일반 텍스트 메시지로 전송된다. 행마다 태그 ID가 하나씩.
Terse	텍스트 형식과 유사하며, 단 TagList 데이터가 개요 형식인 것은 다르다.
XML	TagLists는 XML 텍스트 형식으로 전송한다.
Custom	텍스트 형식과 유사하며, 단 TagListCustomFormat 명령어로 정의된 대로 형식이 정해지는 것이 다르다.

- 텍스트 형식의 TagLists는 다음 형태를 취한다:

```
#Alien RFID Reader Auto Notification Message
#ReaderName: Spinner Reader
#ReaderType: Alien RFID Tag Reader (Class 1 / 915MHz)
#IPAddress: 10.1.70.13
#CommandPort: 23
#MACAddress: 00:80:66:10:11:6A
#Time: 2003/01/21 12:48:59
#Reason: TEST MESSAGE
Tag:8000 8004 0000 003B, Disc:2003/12/04 15:08:59, Last:2003/12/04 15:08:59, Count:4,
Ant:0
Tag:8000 8004 9999 0004, Disc:2003/12/04 15:08:59, Last:2003/12/04 15:08:59, Count:3,
Ant:0
#End of Notification Message
```

- Terse 형식의 TagLists는 다음 형태를 취한다.

```
#Alien RFID Reader Auto Notification Message
#ReaderName: Spinner Reader
#ReaderType: Alien RFID Tag Reader (Class 1 / 915MHz)
#IPAddress: 10.1.70.13
#CommandPort: 23
#MACAddress: 00:80:66:10:11:6A
#Time: 2003/01/21 12:48:59
#Reason: TEST MESSAGE
8000 8004 0000 003B,0,4
8000 8004 9999 0004,0,3
#End of Notification Message
```

- XML 형식의 TagLists는 다음 형태를 취한다.

```
<Alien-RFID-Reader-Auto-Notification>
  <ReaderName>Spinner Reader</ReaderName>
  <ReaderType><Alien RFID Tag Reader (Class 1 / 915MHz)</ReaderType>
  <IPAddress>10.1.70.13</IPAddress>
  <CommandPort>23</CommandPort>
  <MACAddress>00:80:66:10:11:6A</MACAddress>
  <Time>2003/01/21 12:49:22</Time>
  <Reason>TEST MESSAGE</Reason>
    <Alien-RFID-Tag-List>
        <Alien-RFID-Tag>
        <TagID>8000 8004 0000 003B </TagID>
        <DiscoveryTime>2003/12/04 15:08:59</DiscoveryTime>
        <LastSeenTime>2003/12/04 15:08:59</LastSeenTime>
          <Antenna>0</Antenna>
        <ReadCount>4</ReadCount>
          <Protocol>1</Protocol>
        </Alien-RFID-Tag>
        <Alien-RFID-Tag>
        <TagID>8000 8004 9999 0004</TagID>
        <DiscoveryTime>2003/12/04 15:08:59</DiscoveryTime>
        <LastSeenTime>2003/12/04 15:08:59</LastSeenTime>
        <Antenna>0</Antenna>
        <ReadCount>3</ReadCount>
          <Protocol>1</Protocol>
        </Alien-RFID-Tag>
    </Alien-RFID-Tag-List>
</Alien-RFID-Reader-Auto-Notification>
```

NotifyHeader

F800 | 9900 | 9680 | 9650

NotifyHeader 명령어는 각각의 공지 메시지의 헤더 부분을 켜거나 끈다. NotifyHeader를 끄면 공지 메시지에는 메시지의 TagList 구획만 포함된다.

- 허용된 값: "ON" | "OFF"
- 기본값: "ON"

NotifyMode 예제	
Command Response	>NotifyHeader? NotifyHeader = On
Command Response	>NotifyHeader = off NotifyHeader = Off

NotifyKeepAliveTime

F800 | 9900 | 9680 | 9650

리더가 공지 메시지를 네트워크 상에서 전송하려면 TCP 소켓을 열어야 한다. 소켓을 열고 닫으면 일부 프로세서와 네트워크 오버헤드가 필요하며, 공지가 리더를 빠르게 빠져 나오는 경우에는 오버헤드가 짐이 되어 리더의 반응성을 늦춘다.

NotifyKeepAliveTime 속성은 리더가 TagStream 소켓을 열어두는 시간을 설정한다(초 단위). 신속한 공지를 위해서는 리더가 소켓을 계속 열어두는 것이 유리할 것이다. 이를 위해서는 NotifyKeepAliveTime을 공지 간에 예상되는 시간 보다 더 큰 수로 설정하면 된다.

한편 소켓을 열어두면 네트워크에 부담이 된다. 공지가 자주 있지 않다면, NotifyKeepAliveTime을 적게 하여 소켓이 하나의 메시지를 발송하는데 충분한 길이만큼만 열어두고 메시지 전송 후에 자동으로 닫게 하는 것이 좋다.

- 허용된 값: Integer(0..65535)
- 기본값: 30

NotifyKeepAliveTime 예제	
Command Response	>NotifyKeepAliveTime? NotifyKeepAliveTime = 30
Command Response	>NotifyKeepAliveTime = 90 NotifyKeepAliveTime = 90

MailServer

F800 | 9900 | 9680 | 9650

MailServer 명령어를 사용하면 SMTP 메일 서버를 정의할 수 있다. SMTP 서버는 반드시 인증되지 않은 접근을 지원해야 한다. 이 메일 서버는 자동 공지를 구성하여 메일을 전송 방법을 설정했을 때에만 사용한다(Notify 명령어 참조).

MailServer 예제	
Command Response	>MailServer? MailServer = 12.34.56.78
Command Response	>MailServer = 45.224.124.34 MailServer = 45.224.124.34

MailFrom

F800 | 9900 | 9680 | 9650

MailFrom 명령어를 사용하면 RFID 리더와 관련하여 이메일 주소를 정의할 수 있다. RFID 리더가 전송하는 이메일은 이메일 헤더 필드의 형식으로 매개변수를 설정한다.

MailFrom 예제	
Command Response	>MailFrom? MailFrom = AlienRFIDReader
Command Response	>MailFrom = reader@mycompany.com MailFrom = reader@mycompany.com

NotifyRetryCount

F800 | 9900 | 9680 | 9650

네트워크 공지가 지정한 공지 주소(NotifyAddress)로 호스트에 연결하지 못하면, 리더는 공지 메시지를 재전송하려고 시도한다. 작업이 실패했을 때 끊임없이 재시도하기 보다는(이 경우 리더 자원을 계속 소모한다), 리더는 시도 횟수가 NotifyRetryCount에 도달하면 중단한다. 이 시점에 리더는 NotifyMode를 끈다.

리더는 실패한 공지 메시지를 순서대로 늘어 세우며(최대 1000개 메시지), 다시 연결되는 대로 모두 전송하는데 내재한 네트워크 문제를 해결한 후에 전송하거나 다른 NotifyAddress를 구성하여 전송한다. 이 기능은 NotifyMode "kill switch"가 필요할 상황이 없도록 예방하며, 따라서 NotifyRetryCount 기본값은 이러한 리더에서는 -1(재시도를 중지하지 않음)이 된다.

- 허용된 값: Integer(-1..32767)
- 기본값: -1 (재시도를 포기하지 않음)
- 재시도 간격은 NotifyRetryPause(아래)로 지정한다.

NotifyRetryCount 예제	
Command Response	>NotifyRetryCount? NotifyRetryCount = 3
Command Response	>NotifyRetryCount = 0 NotifyRetryCount = 0

NotifyRetryPause

F800 | 9900 | 9680 | 9650

리더가 네트워크 공지 전송을 시도했다가 실패하면, NotifyRetryCount로 지정한 횟수만큼 다시 시도한다. 재시도 간격은 NotifyRetryPause로 지정한다.

- 허용된 값: Integer(0..32767)
- 기본값: 10 sec

NotifyRetryPause 예제	
Command Response	>NotifyRetryPause? NotifyRetryPause = 10
Command Response	>NotifyRetryPause = 60 NotifyRetryPause = 60

NotifyQueueLimit

F800 | 9900 | 9680 | 9650

NotifyQueueLimit 명령어를 사용하면 리더가 나중에 전송하도록 늘어 세울 때 실패한 공지 메시지가 몇 개인지를 명시할 수 있다. TCP 소켓이나 메일 서버로 공지를 보낼 때, 실패한 전송 내용은 리더 내에 줄 세우고 일정 간격이 되면 재시도한다(NotifyRetryPause로 명시함). 최대 1000개의 공지를 줄 세울 수 있으며, 일단 큐 한계에 도달하면 오래된 메시지부터 제거하고 새 메시지를 채운다.

NotifyQueueLimit를 더 작은 값으로 설정하면 큐에 있는 메시지 최대수가 줄어들며, 그 수만큼의 실패한 메시지가 공지 채널이 재구성된 뒤에 전송한다. NotifyQueueLimit를 0으로 설정하면 실패한 메시지가 재전송되지 않는다.

- 허용된 값: Integer(0..1000)
- 기본값: 1000

NotifyQueueLimit 예제	
Command Response	>NotifyQueueLimit? NotifyQueueLimit = 1000
Command Response	>NotifyQueueLimit = 0 NotifyQueueLimit = 0

NotifyInclude

F800 | 9900 | 9680 | 9650

NotifyInclude 명령어는 정상적인 TagList 대신 또는 그에 추가로 공지 메시지에 현재 IOList를 포함시킬 것인지 여부를 NotifyMode 엔진에 명시한다. 다음 값 중 하나를 포함할 수 있다.

NotifyInclude	설명
Tags	공지 메시지에는 TagList만 포함된다. 전형적인 공지 메시지에 해당되며, 기본 설정값이다.
DI	공지 메시지는 디지털 입력 이벤트만 포함한다.
DO	공지 메시지는 디지털 출력 이벤트만 포함한다.
DIO	공지 메시지는 디지털 입력/출력 이벤트만 포함한다(상호 배치됨).
All	공지 메시지는 표준 TagList와 모든 디지털 I/O 이벤트를 포함한다.

- 기본값은 "Tags"이다.
- 공지 메시지가 태그 및 I/O 데이터를 모두 포함할 경우, 태그 데이터가 항상 우선한다.
- TagList와 IOList 모두의 형식을 NotifyFormat 명령어로 지정한다.
- 다음은 샘플 데이터로, NotifyFormat=Text, NotifyInclude=All 조건이다. TagList와 IOList 간에는 명확한 구분이 없다는 것에 유의해야 한다.

```
#Alien RFID Reader Auto Notification Message
#ReaderName: Alien RFID Reader
#ReaderType: Alien RFID Tag Reader, Model: ALR-9800 (Four Antenna / Multi-Protocol / 902-928 MHz)
#IPAddress: 10.10.82.72
#CommandPort: 23
#MACAddress: 00:80:66:10:11:6A
#Time: 2007/01/22 11:25:45
#Reason: TEST MESSAGE
#StartTriggerLines: 0
#StopTriggerLines: 0
Tag: E200 3411 B801 0108 1209 0054, Disc:2007/01/22 11:25:22, Last:2007/01/22 11:25:43, Count:20, Ant:0, Proto:2
IO:DI, Time:2007/01/22 11:25:22.343, Data:1
IO:DO, Time:2007/01/22 11:25:22.361, Data:2
#End of Notification Message
```

- The same data, with NotifyFormat=XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-Reader-Auto-Notification>
  <ReaderName>Alien RFID Reader</ReaderName>
  <ReaderType>Alien RFID Tag Reader, Model: ALR-9900 (Four Antenna / Multi-Protocol /
902-928 MHz)</ReaderType>
  <IPAddress>10.10.82.72</IPAddress>
  <CommandPort>23</CommandPort>
  <MACAddress>00:80:66:10:11:6A</MACAddress>
  <Time>2007/01/22 11:25:45</Time>
  <Reason>TEST MESSAGE</Reason>
  <StartTriggerLines>0</StartTriggerLines>
  <StopTriggerLines>0</StopTriggerLines>
    <Alien-RFID-Tag-List>
        <Alien-RFID-Tag>
        <TagID>E200 3411 B801 0108 1209 00546</TagID>
        <DiscoveryTime>2007/01/22 11:25:22</DiscoveryTime>
        <LastSeenTime>2007/01/22 11:25:22</LastSeenTime>
        <Antenna>0</Antenna>
        <ReadCount>2</ReadCount>
        <Protocol>2</Protocol>
        </Alien-RFID-Tag>
    </Alien-RFID-Tag-List>
    <Alien-RFID-IO-List>
        <Alien-RFID-IO>
        <Type>DI</Type>
        <Time>2007/01/22 11:25:22.343</Time>
        <Data>1</Data>
        </Alien-RFID-IO>
        <Alien-RFID-IO>
         <Type>DO</Type>
         <Time>2007/01/22 11:25:22.361</Time>
         <Data>2</Data>
        </Alien-RFID-IO>
    </Alien-RFID-IO-List>
</Alien-RFID-Reader-Auto-Notification>
```

NotifyInclude 예제	
Command	>NotifyInclude?
Response	NotifyInclude = Tags
Command	>NotifyInclude = All
Response	NotifyInclude = All

NotifyNow

F800 | 9900 | 9680 | 9650

NotifyNow 명령어는 NotifyAddress 명령어로 현재 설정한 주소로 TagList의 공지를 즉각 전송하도록 리더에 지시한다.

NotifyNow 예제	
Command	>NotifyNow
Response	Issuing Notify Trigger...

CHAPTER 5
Tag Programming

모든 Alien RFID 태그는 프로그래밍 가능한 ID 번호를 지원한다. 이 장에서는 EPC 준수 ID 코드와 사용자 정의 ID 코드를 Alien RFID 태그 내에서 프로그래밍하는데 필요한 일련의 명령어들을 설명하며, 프로그래밍을 성공적으로 완수하기 위해 필요한 물리적 조건들의 세부 정보도 제시한다.

태그 메모리 구조

태그 하위집합을 얻기 위해 태그를 프로그래밍하고 마스크 명령어를 사용하는 데 있어 태그 메모리 구조를 이해하는 것은 필수 조건이다. 이 장에서는 모든 Alien 태그 시스템들이 지원하는 기본 메모리 형식에 대해 기술한다.

클래스 1/Gen 2 태그 메모리

클래스1 Gen2 데이터 개체 크기는 기본적으로 1바이트(8비트)가 아니라 한 단어(16비트)이다. 클래스 1/Gen 2 태그는 최대 4개의 메모리 뱅크를 포함한다.

G2 Bank	설명
Bank 0 RESERVED	'Kill'과 'Access' 패스워드를 포함한다. 길이로는 각각 두 단어씩이다(4 바이트). 태 그는 이 패스워드들은 실행할 필요가 없는데, 이 경우에 값은 0000 0000이고, 이에 해당하는 미사용 메모리 위치는 존재하지 않는다. 뱅크 0으로의 마스킹(Masking)은 허용되지 않는다.
Bank 1 EPC	하나의 CRC 단어, 하나의 PC 단어, 그에 따른 유동적인 수의 EPC 단어들을 포함한다. 구조에 관한 자세한 정보는 아래를 참조할 것.
Bank 2 TID	태그 확인 정보를 포함하며, 여기에는 할당 클래스 식별자(EPCglobal 등), 제조업체 정보, 태그 모델과 버전 정보 등이 해당된다.
Bank 3 USER	사용자 별 데이터 저장을 위한, 구조화하지 않은 스크래치(scratch) 공간 태그는 USER 뱅크를 실행할 필요가 없으며, 메모리 용량은 태그 수행에 따라 달라진다.

클래스 1 Gen 2 구조

현재 클래스 1/Gen 2 태그는 보통 EPC 메모리 뱅크에 프로그래밍 가능한 메모리를 96 또는 128비트 포함하나, 사양에는 최대 496 EPC비트까지로 허용한다. EPC 코드에 더해, EPC 메모리 뱅크에도 CRC 체크섬 16비트와 프로토콜-컨트롤(PC) 16비트를 포함한다.

	CRC	PC	EPC Code (or User ID Code)						
Word	0	1	0	1	2	3	4	5	…
Bit	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127	…

128-bit Class 1, Gen 2 Tag Memory Structure

EPC 코드 어드레스 방향은 좌측에서 우측이며, 이때 좌측단 비트는 (가장 유의미한 비트) 32비트이고 EPC 길이만큼 연장된다. 이 태그 구획에 있는 데이터에는 제한이 가해지지 않는다.

체크섬은 PC의 16비트와 전체 EPC 코드 상에서 태그에 의해 자동으로 계산된다. 체크섬은 리더가 자동으로 계산하여 태그로 프로그래밍 한다.

프로토콜-컨트롤(PC) 단어는 EPC 단어 수를 인코딩하고(비트 0-4), 두 개의 추가 비트(비트 5-6)와 넘버링 시스템 식별자(비트 7-15)가 뒤따른다. 넘버링 시스템 식별자는 EPCglobal™ 헤더(EPC™ 태그 데이터 표준으로 정의함) 또는 애플리케이션 제품군 식별자(ISO/IEC 15961로 정의함)을 지정한다.

거리 & 동력 레벨 프로그래밍

주의사항: 태그 프로그래밍 전에 이 섹션을 숙독하고 이해할 것을 권장한다.

이 매뉴얼에서 "프로그래밍"이라는 용어는 태그 메모리를 바꾸는 작업을 의미한다. 여기에는 Erase, ProgramEPC, Lock, Kill 명령어가 포함된다. 이러한 명령어는 이 장 후반부에서 자세히 다룰 것이다. 기기 작동 및 소프트웨어 운용의 관점에서 보면, 태그 ID를 프로그래밍하는 일은 단순하며, 대부분의 경우 몇 차례 버튼을 클릭하거나 명령어 한 번이면 해결된다. 그러나 몇 가지 변수가 프로그래밍 신뢰도에 영향을 미칠 수 있으며 각 애플리케이션에 주소 지정을 정확히 해 주어야 한다. 어떤 애플리케이션에서든 프로그램이 성공적이 되기 위해 사용자가 제어해야 할 세 가지 요소가 있는데, 바로 애플리케이션 소프트웨어, 통신 신호 감쇠, 태그의 물리적 위치이다.

프로그래밍 할당 동력

태그를 프로그래밍하는 데는 태그를 읽는 것보다 훨씬 더 많은 동력이 필요하다. 그 결과 태그 프로그래밍 범위는 읽기 범위보다 훨씬 더 작을 수 밖에 없다. 프로그래밍 명령어는 명령어를 실행할 때 충분한 동력을 받고 있던 태그 전체에 영향을 미친다. 그 결과 프로그래밍하는 태그는 프로그래밍하도록 의도하지 않은 태그들과는 물리적으로 분리되어 있어야 하며, 마스크를 하나의 태그와만 부합되도록 설정해야 한다. 마찬가지로 인접 태그에 영향을 미치지 않도록 해당 환경 내 태그를 프로그래밍할 때에도 최소한의 동력으로만 프로그래밍해야 한다. 프로그래밍 동력이 낮을 수록 태그의 물리적 분리 거리도 더 짧아진다.

태그가 받는 동력은 안테나에 공급하는 동력에 따라 결정되며, 거리 태그는 안테나에서 받는다. 환경에 따른 신호 반사 수준이나 태그 대상 객체도 고려해야 한다. 환경적 반사도 태그 근처의 로컬 동력 무효화(null)를 일으킬 수 있으며, 이 경우 동력이 불충분해져 프로그래밍을 할 수 없게 된다. 태그는 가능한 지속적으로 안테나에서 분리해야 하며, 이로써 동력 변동을 최소화하고 반사로 인한 무효화를 피해야 한다. 안테나와 관련해 태그의 방향도 제어해야 한다. 특히 선형 안테나를 사용할 때 태그는 안테나 극성과 같은 방향으로 배치해야 한다. 안테나에 공급되는 동력은 감쇠기를 사용해 제어하여 신호 강도를 줄여야 한다. 감쇠기 값은 상기 기술한 대로 다른 변수들을 제어한 다음에 환경에 따라 정밀 측정값을 토대로 선정한다. 가변 감쇠기는 최종 감쇠값을 설정할 때 유용한 도구이다.

프로그래밍 문제

프로그래밍이 잘 되지 않을 때 가장 공통된 문제는 글로벌 스크롤이나 인벤토리 모드에서 태그 읽기가 더 이상 작동하지 않는 것이다. 이는 태그 메모리가 지워졌거나 프로그래밍이 불완전하기 때문이다. 이런 상태에 있는 태그는 'Verify Tag' 명령어를 사용해 읽거나 'Program Tag' 명령어를 사용해 프로그래밍할 수 있다.

일반적으로 프로그래밍이 잘 돼지 않은 채로 태그가 지워지는 일은(Verify Tag 명령어가 항상 0을 반환함) 동력이 불충분할 때 발생하는데, 삭제 과정은 프로그래밍 과정에 비해 에너지가 덜 필요하기 때문이다. 동력은 감쇠를 줄이거나 태그를 안테나에 더 가깝게 배치하고 프로그래밍 과정을 반복하면 늘어난다.

프로그래밍이 제대로 되지 않은 태그는 프로그래밍 동력이 불충분하거나 과도한 것이 원인일 수 있다. 이 경우 태그 위치와 신호 감쇠를 재평가한 다음 프로그래밍 과정을 반복해 본다.

스텝 감쇠기(step attenuator)는 프로그래밍 조건을 평가하는데 강력한 도구이다. 감쇠기를 사용할 때 태그 읽기가 신뢰할 만한 수준이 되는 최고값으로 설정한다. 그런 다음 이 값에서 약 10dB 정도 감쇠를 줄여서(동력을 키움)

프로그래밍 할 때 적합한 감쇠 산정치로 삼는다. 감쇠기 값을 조절해 가면서 자신의 애플리케이션 내 다른 변수들을 고려해 최적의 프로그래밍 조건을 결정할 수 있다.

프로그래밍 명령어 요약

Command	설명	F800	9900+	9680	9650
ProgProtocol	차후 프로그래밍 작업에 사용할 단일 프로토콜(0, 1, 2)	✓	✓	✓	✓
ProgAntenna	프로그래밍 명령어를 전송할 안테나	✓	✓	✓	✓
ProgramEPC ("p")	태그에 EPC 코드를 작성함	✓	✓	✓	✓
ProgramAndLockEPC	작성 후 EPC 뱅크를 잠금	✓	✓		
ProgramAccessPwd	Gen2 접속 패스워드를 작성함	✓	✓	✓	✓
ProgramKillPwd	Gen2 Kill 패스워드를 작성함	✓	✓	✓	✓
ProgramUser	Gen2 태그에 사용자 데이터를 작성함	✓	✓	✓	✓
ProgramAndLockUser	작성 후 사용자 뱅크를 잠금	✓	✓		
LockEPC	태그의 EPC 뱅크 잠금	✓	✓	✓	✓
LockAccessPwd	Gen2 태그 접속 패스워드를 잠금	✓	✓	✓	✓
LockKillPwd	Gen2 태그 Kill 패스워드를 잠금	✓	✓	✓	✓
LockUser	Gen2 태그 사용자 메모리를 잠금	✓	✓	✓	✓
LockUserBlocks	Gen2 사용자 메모리의 개별 블록을 영구 잠금	✓	✓		
HideAlienUserBlocks	사용자 메모리의 개별 블록을 숨김/표시함 (Higgs3 & Higgs4 태그만 해당).	✓	✓		
UnlockEPC	Gen2 태그의 EPC 뱅크 잠금해제	✓	✓	✓	✓
UnlockAccessPwd	Gen2 태그 접속 패스워드를 잠금해제	✓	✓	✓	✓
UnlockKillPwd	Gen2 태그 Kill 패스워드를 잠금해제	✓	✓	✓	✓
UnlockUser	Gen2 태그 사용자 메모리를 잠금해제	✓	✓	✓	✓
ProgG2LockType	Gen2 태그 잠금 시 사용할 잠금 유형 (Lock, PermaLock, PermaUnlock)	✓	✓	✓	✓
ProgEPCData	태그를 자동으로 프로그래밍할 때 사용할 다음 EPC "ProgramID"라 칭함.	✓	✓	✓	✓
ProgG2NSI	ProgramEPC 명령어로 프로그래밍할 때 PC 단어의 NSI 파트를 설정할 값	✓	✓	✓	✓
ProgG2AccessPwd	Gen2 접속 패스워드를 자동으로 작성할 때 사용할 접속 패스워드 데이터	✓	✓	✓	✓
ProgG2KillPwd	Gen2 Kill 패스워드를 자동으로 작성할 때 사용할 Kill 패스 워드 데이터	✓	✓	✓	✓
ProgUserData	Gen2 태그 사용자 메모리를 자동 프로그래밍할 때 사 용하는 사용자 데이터	✓	✓	✓	✓
ProgEPCDataInc	ProgEPCData가 자동으로 증분하게 하는 EPC 프로그래밍 결과	✓	✓	✓	✓
ProgUserDataInc	ProgEPCData가 자동으로 증분하게 하는 사용자 프로그래밍 결과	✓	✓	✓	✓
ProgEPCDataIncCount	ProgEPCData 증분 횟수 제한	✓	✓		
ProgUserDataIncCount	ProgUserData 증분 횟수 제한	✓	✓		

Command	설명	F800	9900+	9680	9650
G2Erase	Gen2 태그 메모리의 특정 구획 지움	✓	✓	✓	✓
Erase	태그 지움(EPC에 0을 작성).	✓	✓	✓	✓
Kill	지정된 Kill 패스워드로 태그를 없앰	✓	✓	✓	✓
ProgAttempts	태그 프로그래밍을 시도하는 횟수	✓	✓	✓	✓
ProgSuccessFormat	성공적인 프로그램 작업의 결과 형식을 결정함	✓	✓	✓	✓
ProgSingulate	프로그래밍 작업을 수행하기 전에 식별 (singulation) 체크를 활성화/비활성화함	✓	✓		
TagInfo	태그에 관한 정보 불러오기	✓	✓	✓	✓
G2Read	Gen2 태그 메모리의 저수준 읽기 수행	✓	✓	✓	✓
G2Write	Gen2 태그 메모리의 저수준 쓰기 수행	✓	✓	✓	✓
ProgDataUnit	Gen2 태그 작성 시 데이터 단위의 크기 지정	✓	✓	✓	✓
ProgBlockSize	블록 모드를 작성할 때 데이터 블록의 크기를 단어로 지정	✓	✓		
ProgBlockAlign	블록 경계들에서 블록 쓰기를 수행할 때 도움을 받음	✓	✓		
ProgramAlienImage	전체 Alien 태그의 메모리 이미지를 프로그래밍함		✓		
ProgAlienImageMap	전체 Alien 태그 메모리 이미지를 프로그래밍할 때 사용하는 메모리 맵 유형		✓		
ProgAlienImageNSI	전체 Alien 태그 메모리 이미지를 프로그래밍할 때 사용하는 NSI 단어		✓		

Program, Erase, and Verify Functions

ProgramXXXX 명령어는 태그 메모리에 사용자 지정 데이터를 프로그래밍함 Erase 명령어는 태그 메모리 안에 모든 데이터를 지우고 모든 태그 EPC 데이터를 0으로 설정함

프로그래밍 기능은 언제나 ProgAntenna가 지정한 단일 안테나 상에서 작동함. 다중 프로토콜 리더에서는 ProgProtocol 명령어로 프로그래밍할 때 사용할 단일 태그 프로토콜을 사용자가 지정한다.

ProgramEPC

F800 | 9900 | 9680 | 9650

ProgramEPC 명령어를 사용하면 새로운 EPC (태그 ID)를 태그 안에 작성할 수 있다. 일단 명령어를 전송했으면, 리더는 태그의 존재를 확인하고, 프로그래밍한 다음 태그 메모리로 반환하여 읽어서 프로그램 명령어가 제대로 작동했음을 확증한다.

- 잠긴 EPC가 있는 태그는 재프로그래밍할 수 없다. 태그의 EPC 뱅크는 우선 반드시 잠금해제해야 한다.
- ProgramEPC 명령어의 경우, 사용자는 프로그래밍할 태그 메모리에 대하여 16진수 바이트를 사이띄기로 분리한 짝수를 제공해야 한다.
- Class1/Gen2 태그에는 최대 EPC 길이가 있으나, 비교적 작은 EPC도 프로그래밍할 수 있다.
- ProgramEPC 명령어에 응답하는 형식은 ProgSuccessFormat 명령어로 변경할 수있다.
- 프로그래밍은 AutoAction=Program 설정을 통해 AutoMode로 수행 가능하다.
- 초기의 리더 펌웨어에서는 이 명령어를 단순히 "Program"이라고 칭했다. 초기의 리더 모델에서는 이 명령어를 단순히 "Program Tag"라고 칭했다. 이러한 명령어 명칭은 후향적 호환성 유지를 위해 아직 사용한다.
- ProgramEPC 명령어에 인수가 없을 경우, 리더는 대신 ProgEPCData 명령어가 지정하는 데이터를 사용한다(아래 참조).
- 프로그래밍할 EPC가 이미 태그에 수록된 EPC와 부합한다면, ProgramEPC 명령어는 성공했다는 응답을 반환할 것이며, 이때 태그가 잠겨 있고 프로그래밍 할 수 없는지 여부는 관계없다.

ProgramEPC 예제	
Condition	**시야 내에 정확히 하나의 태그가 있음. 태그 읽기가 강력함. 새 64-bit ID로 Class I 태그를 프로그래밍함.**
Command Response	>ProgramEPC = 01 02 03 04 05 06 07 08 09 0A 0B 0C ProgramEPC = 01 02 03 04 05 06 07 08 09 0A 0B 0C
Condition	**읽기 범위 가장자리에 태그가 있음**
Command Response	> ProgramEPC = 01 02 03 04 05 06 07 08 09 0A 0B 0C Error 134: No tag found.
Condition	**태그가 잠김**
Command Response	> ProgramEPC = 01 02 03 04 05 06 07 08 09 0A 0B 0C Error 137: Tag Is locked.

ProgramAndLockEPC

F800 | 9900 | 9680 | 9650

ProgramAndLockEPC 명령어를 사용하면 새 EPC 코드를 작성할 수 있으며 태그의 EPC를 한 단계 잠글 수 있다. ProgG2LockType으로 지정하여 선택한 Class1/Gen2 잠금 유형이 '잠금(Lock)' 상태인 경우(PermaLock이나 PermaUnlock이 아님), 접속 패스워드가 태그도 작성된다. 이는 단순히 EPC 뱅크를 잠그는 것만으로는 데이터를 보호할 수 없기 때문이며, 0 이외의 접속 패스워드가 태그 내에 있어야 한다. 리더는 접속 패스워드를 작성할 때 ProgG2AccessPwd로 제시된 값을 사용한다. AcqG2AccessPwd를 같은 값으로 설정하여 이 이중적인 작업을 효율화할 것을 권장한다.

- 사용자는 태그의 새 EPC 코드를 ProgramAndLockEPC 명령어의 인수로 제공한다. 데이터가 없으면 리더는 대신 ProgEPCData 값을 사용한다.
- ProgramAndLockEPC 작업은 AutoAction=ProgramAndLockEPC 설정을 통해 AutoMode 로 수행 가능하다.
- 프로그래밍할 EPC가 이미 태그에 수록된 EPC와 부합한다면, ProgramAndLockEPC 명령어는 성공했다는 응답을 반환할 것이며, 이때 태그가 잠겨 있고 프로그래밍 할 수 없는지 여부는 관계 없다.

ProgramAndLockEPC 예제

```
                // Set up
                >ProgG2LockType = Lock
                >ProgG2AccessPwd = F1 F2 F3 F4
                >AcqG2AccessPwd = F1 F2 F3 F4

                // Command with argument
Command         >ProgramAndLockEPC = 01 02 03 04 05 06 07 08 09 0A 0B 0C
Response        ProgramAndLockEPC = 0102 0304 0506 0708 090A 0B0C
                // With no argument
                >ProgEPCData = 01 02 03 04 05 06 07 08 09 0A 0B 0C

Command         >ProgramAndLockEPC
Response        ProgramAndLockEPC = 0102 0304 0506 0708 090A 0B0C
```

ProgramAccessPwd

F800 | 9900 | 9680 | 9650

ProgramAccessPwd 명령어를 사용하면 새로운 접속 패스워드를 Class1/Gen2 태그 안에 작성할 수 있다. 0 이외의 접속 패스워드가 있는 태그는 메모리 뱅크를 잠그거나 잠금해제할 수 없고 보호 영역 내에 있는 데이터를 변경할 수도 없다. 단 현재 접속 패스워드를 사용자가 제공하는 경우는 예외이다(AcqG2AccessPwd 명령어를 통해서).

- 사용자는 태그의 새 접속 패스워드(4바이트)를 ProgramAccessPwd 명령어의 인수로 제공한다. 데이터가 없으면 리더는 대신 ProgG2AccessPwd 값을 사용한다.
- Class1/Gen2 태그로는 접속 패스워드용 메모리를 제공할 수 없다. 이러한 태그는 접속 패스워드가 "0000 0000"인 것처럼 작동한다.
- 타인으로부터 접속 패스워드를 숨기려면 접속 패스워드를 작성한 후에 AccessPwd 필드를 잠궈야 한다.

ProgramAccessPwd 예제	
	// With an argument
Command	>ProgramAccessPwd = 01 02 03 04
Response	ProgramAccessPwd = 0102 0304
	// Without an argument
Command	>ProgG2AccessPwd = 01 02 03 04
Command	>ProgramAccessPwd
Response	ProgramAccessPwd = 0102 0304

ProgramKillPwd

F800 | 9900 | 9680 | 9650

ProgramKillPwd 명령어를 사용하면 새로운 Kill 패스워드를 Class1/Gen2 태그 안에 작성할 수 있다. Kill 패스워드가 없는 태그는 없앨 수 없으며, 0 이외의 Kill 패스워드를 가진 태그는 보관된 패스워드를 Kill 명령어로 제시했을 때만 없앨 수 있다.

- 사용자는 태그의 새 Kill 패스워드(4바이트)를 ProgramKillPwd 명령어의 인수로 제공한다. 데이터가 없으면 리더는 대신 ProgG2KillPwd값을 사용한다.
- 모든 Class1/Gen2 태그가 Kill 패스워드를 지원하는 것은 아니다. 이러한 태그는 없앨 수 없다.
- 타인으로부터 Kill 패스워드를 숨기려면 Kill 패스워드를 작성한 후에 KillPwd 필드를 잠궈야 한다.

ProgramKillPwd 예제	
	// With an argument
Command	> ProgramKillPwd = FA FB FC FD
Response	ProgramKillPwd = FAFB FCFD
	// Without an argument
	>ProgG2KillPwd = FA FB FC FD
Command	> ProgramKillPwd
Response	ProgramKillPwd = FAFB FCFD

ProgramUser

F800 | 9900 | 9680 | 9650

ProgramUser 명령어를 사용하면 지정한 사용자 패스워드를 Class1/Gen2 태그 안에 작성할 수 있다.

- 사용자 메모리가 태그 상에 존재하는지 확인하는 것은 가능하나(PC 단어의 비트 15), 이 방법은 0 이외의 데이터가 사용자 뱅크의 첫 단어에 있을 때만 가능하다. 사용자 메모리에 접속하려고 해보지 않고 태그에 사용자 메모리가 얼마나 있는지를 측정할 방법은 없다.
- 사용자는 태그의 새 사용자 메모리를 ProgramUser 명령어의 인수로 제공한다. 제공된 데이터가 없으면 리더는 대신 ProgUserData 값을 사용한다.
- 사용자 메모리는 단어(2바이트) 단위로 늘려가며 작성해야 하며, 따라서 제공한 데이터는 짝수 바이트로 구성되어야 한다.

ProgramUser 예제	
	// With an argument
Command	>ProgramUser = DE AD BE EF CA FE C0 ED
Response	ProgramUser = DEAD BEEF CAFÉ C0ED
	// Without an argument
	>ProgUserData = F1 F2 F3 F4 F5 F6 F7 F8
Command	> ProgramUser
Response	ProgramUser = F1F2 F3F4 F5F6 F7F8
	// Can read user data with generic "G2Read" command.
	// Bank=3, wordPtr=0, wordCnt=4
Command	>G2Read = 3, 0, 4
Response	G2Read = F1 F2 F3 F4 F5 F6 F7 F8

ProgramAndLockUser

F800 | 9900 | 9680 **|** 9650

ProgramAndLockUser 명령어를 사용하면 새 사용자 데이터를 작성할 수 있으며 태그의 사용자 뱅크를 한 단계 잠글 수 있다.

ProgG2LockType으로 지정하여 선택한 Class1/Gen2 잠금 유형이 '잠금(Lock)' 상태인 경우(PermaLock이나 PermaUnlock이 아님), 접속 패스워드가 태그도 작성된다. 이는 단순히 EPC 뱅크를 잠그는 것만으로는 데이터를 보호할 수 없기 때문이며, 0 이외의 접속 패스워드가 태그 내에 있어야 한다. 리더는 접속 패스워드를 작성할 때 ProgG2AccessPwd로 제시된 값을 사용한다. AcqG2AccessPwd를 같은 값으로 설정하여 이 이중적인 작업을 효율화할 것을 권장한다.

- 사용자는 태그의 새 사용자 데이터를 ProgramAndLockUser 명령어의 인수로 제공한다. 제공된 데이터가 없으면 리더는 대신 ProgUserData 값을 사용한다.

ProgramAndLockUser 예제	
	// Command with argument
Command	>ProgramAndLockUser = F1 F2 F3 F4 F5 F6 F7 F8
Response	ProgramAndLockUser = F1F2 F3F4 F5F6 F7F8
	// With no argument
	>ProgUserData = F1 F2 F3 F4 F5 F6 F7 F8
Command	> ProgramAndLockUser
Response	ProgramAndLockUser = F1F2 F3F4 F5F6 F7F8

G2Erase

F800 | 9900 | 9680 | 9650

G2Erase 명령어로 태그 메모리의 특정 구획을 지울 수 있다(00으로 설정). 어떤 메모리 뱅크를 선택할지, 처음 지울 단어의 워드 페인터, 지울 단어들의 수를 표시할 수 있다. 태그가 Gen2 "BlockErase" 명령어를 실행하면, 표시한 메모리 위치에 0을 입력한다.

G2Erase 명령어의 형식은 다음과 같다:

```
G2Erase = <bank> <wordPtr> <wordLen>
```

- <bank>는 반드시 0-3이어야 한다.
- <wordPtr>은 반드시 0-20971510이어야 한다.
- <wordLen>은 반드시 1-32이어야 한다(더 많이 지우려면 명령어를 연속하여 전송한다).
- G2Erase는 Erase(아래 참조)와 유사한데, 단 Erase는 전체 EPC에만 0을 입력할 수 있다. 하지만 이 경우에는 Gen1 태그에도 적용된다.
- 사용자는 잠금 상태인 메모리를 지울 수 없는데, 단 정확한 접속 패스워드를 가지고 있으면 가능하다.

- ProgDataUnit=Word인 조건에서, G2Erase 명령어는 기본적으로 모든 0의 G2Write 명령어를 쓸 때와 동일하다. ProgDataUnit=Block일 때, Gen2 "BlockErase" 명령어를 대신 전송하면 더 빠르기는 하나 모든 태그에서 지원되는 것은 아니다.

G2Erase 예제	
	`// Erase both the Kill and Access passwords (four words in bank 0)`
Command	`>G2Erase = 0 0 4`
Response	`G2Erase = Success!`
	`// Compare with Program___Pwd, and G2Write`
Command	`>ProgramKillPwd = 00 00 00 00`
Command	`>ProgramAccessPwd = 00 00 00 00`
Command	`>G2Write = 0 0 00 00 00 00 00 00 00 00`

Erase

F800 | 9900 | 9680 | 9650

Erase 명령어는 리더의 시야 내에 있는 태그의 EPC 메모리를 지우려고 시도한다. 이 명령어의 영향을 받는 태그는 전체 EPC 태그 메모리를 0으로 설정한 태그이다. 일단 명령어를 전송했으면, 리더는 태그의 존재를 확인하고, 지운 다음 태그 메모리로 반환하여 읽어서 Erase 명령어가 제대로 작동했음을 확증한다.

- Class1/Gen2에서는, 둘 이상의 태그가 있더라도 하나의 태그만 지워진다.
- 잠긴 EPC 뱅크가 있는 태그는 지울 수 없다.
- 이 명령어를 G2Erase 명령어(상기)와 혼동해서는 안 된다. G2Erase는 Erase보다 더 유동적인 Gen2 명령어이다.
- 이전 리더 모델들은 "Erase Tag" 명령어를 사용하였고, 현재는 후향 호환성을 위해 이 명령어도 수용한다.

Erase 예제	
Command	`>Erase`
Response	`Erase = Success!`
	`// Tag's EPC is locked`
Command	`>Erase`
Response	`Error 137: Tag is locked.`

Lock, Unlock, and Kill Functions

Class1/Gen2 태그는 태그의 각 메모리 뱅크/필드(지원되는 경우)에서 암호 기반 잠금(Lock, PermaLock, PermaUnlock)의 다음 3가지 모드를 지원한다. Kill Pwd, Access Pwd, EPC, User. 잠긴 필드를 잠금 해제할 수도 있는데, 단 정확한 접속 패스워드를 제시해야 한다. 이때 필드는 PermaLock 또는 PermaUnlock 잠금 유형으로 잠그지 않아야 한다. 일부 태그는 사용자 메모리의 PermaLocking 개별 블록을 지원하기도 하며, Alien Higgs3 & Higgs4 태그를 사용하면 사용자는 접속 패스워드를 알지 못하고는 읽을 수 없는 방식으로 사용자 메모리 블록을 잠근다. Gen2 태그를 없애려면 0 이외의 Kill 패스워드가 있어야 하며 정확한 Kill 패스워드를 제시해야 한다. Gen2 태그를 없애면 영구적으로 작동이 불가능해진다.

LockEPC

F800 | 9900 | 9680 | 9650

태그의 EPC를 잠그면 메모리 변경을 차단할 수 있다.

- Class1/Gen2 LockEPC 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. 태그의 접속 패스워드 필드에 작성하려면, ProgramAccessPwd 명령어(아래)를 사용하거나 ProgramAndLockEPC(상기)를 사용한다. LockEPC 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- 일단 Class 1/Gen 2 태그의 EPC가 잠기면, 잠금해제를 하기 위해서는 원래 잠금 유형이 PermaLock 또는 PermaUnlock이 아니어야 하며, 정확한 접속 패스워드를 제시해야 한다.

- Class 1/Gen2 프로토콜은 3가지 마스크 유형인 Lock, PermaLock, PermaUnlock을 허용한다. 잠금 유형은 ProgG2LockType 명령어(아래)로 제시한다.

- 이전 리더 펌웨어는 "Lock Tag" 명령어를 사용하였는데, 후향 호환성을 위해 이 명령어도 아직 수용한다.
- 리더 펌웨어의 이전 버전들도 "LockEPC" 대신 "Lock" 명령어를 사용했는데, 전자의 경우도 아직 수용한다.

C1G2 LockEPC 예제	
Command Response	// With the current Access Pwd as an argument >LockEPC = 01 02 03 04 LockEPC = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 >LockEPC LockEPC = Success!

LockAccessPwd

F800 | 9900 | 9680 | 9650

Class1/Gen2 태그의 접속 패스워드를 잠그면 접속 패스워드가 바뀌지 않게 차단할 수 있으며(Access Key로 동일 패스워드를 제시하지 않는 한), 또는 저수준 읽기 명령어로 태그에서 다시 읽어 들이지 못하게 차단한다.

- Class1/Gen2 LockAccessPwd 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. LockAccessPwd 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- 태그의 접속 패스워드 필드에 작성하려면, ProgramAccessPwd 명령어를 사용한다.
- 모든 Class1/Gen2 태그가 접속 패스워드를 실행하는 것은 아니다.

LockAccessPwd 예제	
Command Response	// With the current Access Pwd as an argument >LockAccessPwd = 01 02 03 04 LockAccessPwd = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 >LockAccessPwd LockEPC = Success!

LockKillPwd

F800 | 9900 | 9680 | 9650

Class1/Gen2 태그의 Kill 패스워드를 잠그면 Kill 패스워드가 바뀌지 않게 차단할 수 있으며(정확한 접속 키를 제시하지 않는 한), 또는 저수준 읽기 명령어로 태그에서 다시 읽어 들이지 못하게 차단한다.

- Class1/Gen2 LockKillPwd 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. LockKillPwd 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- 태그의 Kill 패스워드 필드에 작성하려면, ProgramKillPwd 명령어를 사용한다.
- 모든 Class1/Gen2 태그가 Kill 패스워드를 실행하는 것은 아니다.

LockKillPwd 예제

Command Response	// With the current Access Pwd as an argument > LockKillPwd = 01 02 03 04 LockKillPwd = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 > LockKillPwd LockKillPwd = Success!

LockUser

F800 | 9900 | 9680 | 9650

Locking the User bank of a Class1/Gen2 tag prevents the User memory from being changed (unless you provide the same password as the Access Key). Locking the User bank does not prevent other users from reading the User data.

- Class1/Gen2 LockUser 명령어는 4바이트 접속 패스워드를 취하며, 이 패스워드는 태그 메모리에 존재하는 그에 상응하는 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. LockUser 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- 태그의 사용자 데이터를 작성하려면, ProgramUser 또는 ProgramAndLockUser 명령어를 사용한다.
- 모든 Class1/Gen2 태그가 사용자 메모리를 사용하는 것은 아니다.

LockUser 예제

Command Response	// With the current Access Pwd as an argument > LockUser = 01 02 03 04 LockUser = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 > LockUser LockUser = Success!

LockUserBlocks

F800 | 9900 | 9680 | 9650

Gen2 프로토콜의 선택 기능 중에는 사용자 메모리의 개별 블록들을 **permalock(영구잠금)**하는 것이 있다 (BlockPermaLock이라 칭함). Alien Higgs3와 Higgs4 태그가 이 기능을 지원하며, 사용자는 LockUserBlocks 리더 명령어를 사용해 permalocks를 설정할 수 있다.

이 기능 및 사용자 메모리 블록의 영구잠금에 대한 세부 정보 및 를 원한다면 Gen2 프로토콜 사양서를 참조해 보기 바란다. 어떤 대상을 영구잠금 했다면, 다시는 변경하거나 잠금해제할 수가 없으며, 블록의 영구잠금과 사용자 뱅크 잠금 간의 상호작용이 복잡해질 수 있다.

Gen2 프로토콜을 사용하면 태그 공급자(tag vendors)가 사용자 메모리 뱅크를 불록들로 나눌 수 있으며(아래, HideAlienUserBlocks 명령어에서 참조한 것과 동일한 블록), 각 블록에 얼마나 많은 데이터가 들어갈지는 공급자가 결정한다. Alien Higgs3와 Higgs4 태그에는 4단어 크기(8바이트 또는 64비트)의 블록이 있어서, 이용 가능한 사용자 메모리의 최대 512비트를 8개 블록이 나뉜다.

LockUserBlocks 명령어는 3개 인수를 취하여, 2진수 블록 포인터, 그 뒤에 사용자 메모리 각 블록의 원하는 영구잠금 상태를 나타내는 비트마스크의 16진수 2바이트가 뒤따른다. "1"비트는 관련된 사용자 블록을 영구잠금하기 원한다는 의미이다. "0"비트는 현재 영구잠금 상태를(잠금이든 잠금해제이든) 그대로 유지한다는 말이다.

```
LockUserBlocks = <blockPtr> <blockmask1> <blockmask2>
```

이 Gen2 명령어를 사용하면 리더는 사용자 블록 permalocks를 한번에 16개 블록 단위로 구성하므로, LockUserBlocks 명령어는 2개의 완전한 <blockmask> 바이트(총 16비트)가 필요하다. <blockPtr>를 사용하면 사용자는 더 상위의 16 블록 그룹 주소를 지정할 수 있으며, 따라서 16 블록 단위로 여러 개 있어야 한다(0, 16, 32, 등).

Alien Higgs3 태그는 사용자 데이터를 8개 블록만 실행하므로(Higgs4는 4 블록), <blockPtr>은 항상 0이어야 하고, <blockmask2> 값 또한 0이어야 한다. 존재하지 않는 사용자 블록을 영구잠금하려고 시도하면 태그 "memory overrun" 오류가 발생하며 전체 작업이 실패한다.

- 모든 Class1/Gen2 태그가 사용자 메모리나 BlockPermaLock 기능을 사용하는 것은 아니다.
- 사용자가 동일한 사용자 블록 집합에 반복적으로 LockUserBlocks 명령어를 전송할 수도 있는데, 이렇게 하면 매번 개별 블록들이 잠긴다.
- 블록 영구잠금은 돌이킬 수 없으므로 사용 시 주의를 요한다!

LockUserBlocks 예제

	// Permalock the first two User blocks (C0 = 11000000)
Command	>LockUserBlocks = 0 C0 00
Response	LockUserBlocks = Success!
	// Write 3 blocks
Command	>ProgramUser = A1 A2 A3 A4 A5 A6 A7 A8 B1 B2 B3 B4 B5 B6 B7 B8 C1 C2 C3 C4 C5 C6 C7 C8
Response	Error 137: Tag is locked.
	// Still can write to block #3 (word #8)
Command	>G2Write = 3 8 C1 C2 C3 C4 C5 C6 C7 C8
Response	G2Write = Success!
Command	>G2Read = 3 0 0
Response	G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3 C4 C5 C6 C7 C8
	// Now additionally lock the 3rd User block (20 = 00100000)
Command	>LockUserBlocks = 0 20 00
Response	LockUserBlocks = Success!
	// The first three User blocks are now permalocked.

HideAlienUserBlocks

F800 | 9900 | 9680 **|** 9650

Alien Higgs3와 Higgs4 태그가 실행하는 기능으로 사용자 메모리의 개별 블록들을 잠글 수 있으며, 이렇게 하면 접속 패스워드 없이는 다시 읽을 수가 없다. 이 "read locks(읽기 잠금)"으로 태그 내에 민감한 데이터를 저장할 수 있으며, 믿을 만한 동료들만 읽거나 수정할 수 있게 된다.

Alien Higgs3 태그는 사용자 메모리를 4 단어 단위 블록을 나눈다(상기 LockUserBlocks 명령어 섹션에서 참조한 것과 같은 블록). Higgs3 태그에 포함시킬 수 있는 사용자 메모리의 최대 용량은 512 비트이며, 이는 8개 블록에 해당된다. Alien Higgs4 태그는 사용자 메모리를 2단어 단위의 블록으로 나눈다. Higgs4 태그에 포함시킬 수 있는 사용자 메모리의 최대 용량은 128 비트이며, 이는 4개 블록에 해당된다.

HideAlienUserBlocks는 인수로 16진수 1바이트를 취하며, 8개 사용자 블록 모두에서 읽기잠금을 하는 비트마스크에 해당된다. 이 비트마스크에 있는 최우선 비트는 첫 번째 사용자 블록이며, 다시 말하면 블록 #1에 읽기 잠금을 하려면 이 비트마스크가 01 ($00000001_{2진수}$)가 아닌 80 ($10000000_{2진수}$)이어야 한다.

- 읽기 잠금은 태그가 0 이외의 접속 패스워드를 가진 경우에만 실행 가능하다.
- Alien Higgs3와 Higgs4 태그만 이 기능을 실행한다.

HideAlienUserBlocks 예제

```
                // Read 1st three blocks (12 words) of User data
Command         >G2Read = 3 0 12
Response        G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
                C4 C5 C6 C7 C8

                // Read-lock the 1st and 3rd blocks (A0 = 10100000)
Command         >HideAlienUserBlocks = A0
Response        HideAlienUserBlocks = Success!

                // No Access Pwd yet, so it is still readable:
Command         >G2Read = 3 0 12
Response        G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
                C4 C5 C6 C7 C8

                // Write non-zero Access Pwd and read again
Command         >ProgramAccessPwd = 11 22 33 44
Response        ProgramAccessPwd = 1122 3344

Command         >G2Read = 3 0 12
Response        Error 260: (Tag Error) Memory locked.

                // Can read block #2 (starting at word #4)
Command         >G2Read = 3 4 4
Response        G2Read = 21 22 23 24 25 26 27 28

                // …but not blocks #1,3
Command         >G2Read = 3 0 4
Response        Error 260: (Tag Error) Memory locked.
Command         >G2Read = 3 8 4
Response        Error 260: (Tag Error) Memory locked.

                // Supplying the correct Access Pwd lets us see everything
Command         >AcqG2AccessPwd = 11 22 33 44
Response        AcqG2AccessPwd = 11 22 33 44

                >G2Read = 3 0 12
Command         G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
Response        C4 C5 C6 C7 C8
                // Reveal the hidden blocks
Command         >HideAlienUserBlocks = 00
Response        HideAlienUserBlocks = Success!

                // Don't need the password now
Command         >AcqG2AccessPwd = 0 0 0 0
Response        AcqG2AccessPwd = 00 00 00 00

Command         >G2Read = 3 0 12
Response        G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
                C4 C5 C6 C7 C8
```

UnlockEPC

F800 | 9900 | 9680 | 9650

UnlockEPC 명령어는 Class1/Gen2 태그의 EPC 메모리 뱅크를 잠금해제한다.

- UnlockEPC 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. UnlockEPC 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- EPC 뱅크를 PermaLock 잠금 유형으로 잠근 Class 1/Gen 2 태그는 잠금해제할 수 없다.

UnlockEPC 예제	
Command Response	// With the current Access Pwd as an argument >UnlockEPC = 01 02 03 04 UnlockEPC = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 >UnlockEPC UnlockEPC = Success!

UnlockAccessPwd

F800 | 9900 | 9680 | 9650

UnlockAccessPwd 명령어는 Class1/Gen2 태그의 접속 패스워드 필드를 잠금해제한다. 이 명령어는 Class1/Gen1 태그에는 적용되지 않는다.

- UnlockAccessPwd 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 상응하는 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. UnlockAccessPwd 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- 접속 패스워드를 PermaLock 잠금 유형으로 잠근 Class 1/Gen 2 태그는 잠금해제할 수 없다.

- 접속 패스워드는 Class1/Gen2의 필수 기능은 아니며, 몇몇 태그는 이를 실행하지 않을 수도 있다.

UnlockAccessPwd 예제	
Command Response	// With the current Access Pwd as an argument > UnlockAccessPwd = 01 02 03 04 UnlockAccessPwd = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 > UnlockAccessPwd UnlockAccessPwd = Success!

UnlockKillPwd

F800 | 9900 | 9680 | 9650

UnlockKillPwd 명령어는 Class1/Gen2 태그의 Kill 패스워드 필드를 잠금해제한다. 이 명령어는 Class1/Gen1 태그에는 적용되지 않는다.

- UnlockKillPwd 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 상응하는 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. UnlockKillPwd 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- Kill 패스워드를 PermaLock 잠금 유형으로 잠근 Class 1/Gen 2 태그는 잠금해제할 수 없다.

- Kill 패스워드는 Class1/Gen2의 필수 기능은 아니며, 몇몇 태그는 이를 실행하지 않을 수도 있다.

UnlockKillPwd 예제	
Command Response	// With the current Access Pwd as an argument > UnlockKillPwd = 01 02 03 04 UnlockKillPwd = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 > UnlockKillPwd UnlockKillPwd = Success!

UnlockUser

F800 | 9900 | 9680 | 9650

The UnlockUser command unlocks the User bank of a Class1/Gen2 tag. 이 명령어는 Class1/Gen1 태그에는 적용되지 않는다.

- UnlockUser 명령어는 4바이트 접속 패스워드를 취하며, 이때 패스워드는 태그 메모리 내 이미 존재한 상응하는 접속 패스워드와 부합해야 한다. 이 접속 패스워드는 태그 메모리에 입력되지 않은 것이나, 태그의 내장 보안 수단을 극복할 수 있어야 한다. UnlockUser 명령어에 인수가 없다면, 리더는 대신 AcqG2AccessPwd이 제시한 값을 사용한다.

- 사용자 뱅크를 PermaLock 잠금 유형으로 잠근 Class 1/Gen 2 태그는 잠금해제할 수 없다.

- 사용자 메모리는 Class1/Gen2의 필수 기능은 아니며, 몇몇 태그는 이를 실행하지 않을 수도 있다.

UnlockUser 예제	
Command Response	// With the current Access Pwd as an argument > UnlockUser = 01 02 03 04 UnlockUser = Success!
Command Response	// Without the current Access Pwd argument >AcqG2AccessPwd = 01 02 03 04 > UnlockUser UnlockUser = Success!

HideAlienUserBlocks

F800 | 9900 | 9680 | 9650

Alien Higgs3 태그가 실행하는 기능으로 사용자 메모리의 개별 블록들을 잠글 수 있으며, 이렇게 하면 접속 패스워드 없이는 다시 읽을 수가 없다. 이 "read locks(읽기 잠금)"으로 태그 내에 민감한 데이터를 저장할 수 있으며, 믿을 만한 동료들만 읽거나 수정할 수 있게 된다.

Alien Higgs3 태그는 사용자 메모리를 4 단어 단위 블록을 나눈다(상기 LockUserBlocks 명령어 섹션에서 참조한 것과 같은 블록). Higgs3 태그에 포함시킬 수 있는 사용자 메모리의 최대 용량은 512 비트이며, 이는 8개 블록에 해당된다.

HideAlienUserBlocks는 인수로 16진수 1바이트를 취하며, 사용자 블록 모두에서 읽기잠금을 하는 비트마스크에 해당된다. 이 비트마스크에 있는 최우선 비트는 첫 번째 사용자 블록이며, 다시 말하면 블록 #1에 읽기 잠금을 하려면 이 비트마스크가 01 (000000012진수)가 아닌 80 (100000002진수)이어야 한다.

- 읽기 잠금은 태그가 0 이외의 접속 패스워드를 가진 경우에만 실행 가능하다.

- Alien Higgs3와 Higgs4 태그만 이 기능을 실행한다.

HideAlienUserBlocks 예제

```
                // Read 1st three blocks (12 words) of User data
Command         >G2Read = 3 0 12
Response        G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
                C4 C5 C6 C7 C8

                // Read-lock the 1st and 3rd blocks (A0 = 10100000)
Command         >HideAlienUserBlocks = A0
Response        HideAlienUserBlocks = Success!

                // No Access Pwd yet, so it is still readable:
Command         >G2Read = 3 0 12
Response        G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
                C4 C5 C6 C7 C8

                // Write non-zero Access Pwd and read again
Command         >ProgramAccessPwd = 11 22 33 44
Response        ProgramAccessPwd = 1122 3344

Command         >G2Read = 3 0 12
Response        Error 260: (Tag Error) Memory locked.

                // Can read block #2, but not blocks #1,3
Command         >G2Read = 3 4 4
Response        G2Read = 21 22 23 24 25 26 27 28

Command         >G2Read = 3 0 4
Response        Error 260: (Tag Error) Memory locked.

Command         >G2Read = 3 8 4
Response        Error 260: (Tag Error) Memory locked.

                // Supplying the correct Access Pwd lets us see it
Command         >AcqG2AccessPwd = 11 22 33 44
Response        AcqG2AccessPwd = 11 22 33 44

Command         >G2Read = 3 0 12
Response        G2Read = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 C1 C2 C3
                C4 C5 C6 C7 C8
```

Kill

F800 | 9900 | 9680 | 9650

Class1/Gen2 태그에 적용하면 이 태그는 영구적으로 비활성된다. 이후 어떤 리더 명령에도 반응하지 않을 것이다. Class1/Gen2 태그는 Kill 패스워드를 실행했을 때, 현재 Kill 패스워드가 0이 아닐 때, 사용자가 Kill 명령어에 정확한 Kill 패스워드를 입력했을 때만 없앨 수 있다.

이전 리더 모델들은 "Kill Tag" 명령어를 사용하였고, 현재는 후향 호환성을 위해 이 명령어도 수용한다.

- 사용자는 전체 태그 EPC를 지정하고 그 다음에는 ProgramKillPwd 명령어로 태그에 앞서 작성했던 4바이트 Kill 패스워드가 뒤따른다. 또는 EPC를 생략하고 4바이트 Kill 패스워드만 제시할 경우에는, 리더가 처음 찾은 Class1/Gen2 태그를 없애려고 할 것이다. 사용자가 인수 없이 Kill 명령어를 사용하면, 리더는 처음 찾는 Class1/Gen2 태그를 없애려고 할 것이며, ProgG2KillPwd 명령어로 사용자가 지정한 Kill 패스워드를 사용할 것이다.
- 접속 키는 필요하지 않으며 Kill 패스워드만 있으면 된다.
- 태그에는 0이 아닌 Kill 패스워드가 이미 입력되어 있어야 한다.
- 태그는 영구적으로 작동 불능 상태가 된다.

C1G2 Kill 예제	
Condition	*EPC is B0 B0 11 11 22 22 33 33 44 44 55 55* *Programmed Kill password is 0A 0B 0C 0D* *// Specify EPC to kill and its kill password*
Command Response	>Kill = B0 B0 11 11 22 22 33 33 44 44 55 55 0A 0B 0C 0D Kill = Success!
Command Response	*// Specify just the kill password (kills first tag it finds)* >Kill = 0A 0B 0C 0D Kill = Success!
Command Response	*// Preload kill password to use, then Kill without arguments* >ProgG2KillPwd = 0A 0B 0C 0D ProgG2KillPwd = 0A 0B 0C 0D
Command Response	>Kill Kill = Success!

프로그래밍 구성 및 데이터 저장 명령어

다음 "ProgXXXX" 명령어를 사용하면 모든 프로그래밍 작업에 직접 관계된 핵심 매개변수를 구성할 수 있으며, 사용자 데이터 또는 EPC의 자동 증분 등 "인수가 없는" 프로그래밍 작업이 가능한 사전 설정된 데이터 필드도 구성할 수 있다. 이러한 명령어를 사용하면 AutoMode에서 프로그래밍 작업이 가능하다. 이때 리더는 독립적으로 작업할 수 있도록 필요한 데이터를 모두 앞자리에 배치해야 한다.

이 명령어들 중 가장 중요한 두 개는 ProgProtocol과 ProgAntenna 명령어이다.

ProgProtocol

F800 | 9900 | 9680 | 9650

두 개이 상의 프로토콜에서 프로그래밍을 지원하는 리더라면, 반드시 프로그래밍 작업에 사용할 프로토콜을 미리 알고 있어야 한다. 이를 위해 사용자는 ProgProtocol 명령어를 사용한다. TagType 명령어와는 달리(태그 읽기 중 사용할 프로토콜의 비트 마스크), ProgProtocol은 사용할 단일 프로토콜을 가리키는 값을 취한다.

다음 값들이 한때 사용되었으나, 현재 Alien 리더들은 다음의 Class 1/Gen 2 표준만 지원한다.

0 = Class 0
1 = Class 1/Gen 1
2 = Class 1/Gen 2 (기본값)

- ProgProtocol을 변경할 때에는 선택한 ProgProtocol을 포함하도록 TagType 설정값을 변경할 것을 권한다. 프로그래밍 작업을 수행한 후, 차후에 확인을 위해 "get TagList"를 사용할 계획이라면 프로그래밍에 사용한 것과 동일한 프로토콜을 참조해야 할 것이다.

ProgProtocol 예제	
Command Response	>ProgProtocol? ProgProtocol = 2
Command Response	>ProgProtocol = 1 ProgProtocol = 1

ProgAntenna

F800 | 9900 | 9680 | 9650

태그를 프로그래밍하려면 제어가 가능하고 반복되는 RF 환경이 구성되어야 한다("프로그래밍 거리 및 동력 수준" 제하의 이전 내용 참조). 대부분의 경우, 프로그래밍 명령어는 잠재적으로 RF 필드 내 모든 태그에 영향을 미칠 수 있다. 이 때문에 어떤 안테나에 프로그래밍 언어를 전송할지 표시하기 위해 AntennaSequence를 사용하기 보다는 ProgAntenna 속성을 이용하는 것이 좋다.

모든 프로그래밍 명령어는 현재 AntennaSequence 값과는 무관하게 ProgAntenna가 지정하는 안테나 포트를 통해 전송된다.

- 허용된 값: 정수(0..3) (안테나 최대값은 리더 모델에 따라 달라질 수 있음)
- 기본값: 0
- ProgAntenna에는 단 하나의 안테나 값만 허용되는데, 이는 안테나 번호 목록을 취할 수 있는 AntennaSequence와 차이가 있다.
- ProgAntenna를 설정할 때, 선택한 ProgAntenna를 포함하는 AntennaSequence 값을 설정할 것을 권한다. 프로그래밍 작업을 수행한 후, 차후에 확인을 위해 "get TagList"를 사용할 계획이라면 프로그래밍에 사용한 것과 동일한 안테나를 참조할 것이다.

ProgAntenna 예제	
Command Response	>ProgAntenna? ProgAntenna = 0
Command Response	>ProgAntenna = 2 ProgAntenna = 2

ProgG2NSI

F800 | 9900 | 9680 | 9650

ProgG2NSI 매개변수는 ProgramEPC 명령어를 사용할 때 태그의 PC 단어의 사용자 정의 프로그래밍 NSI값을 갖도록 허용한다. 이 매개변수 값은 16진수 형식의 2바이트로 구성되며 NSI 9비트에 상응한다. NSI 값이 0x01FF보다 크면 이는 PC 단어의 NSI 파트를 변경할 필요가 없음을 의미한다. 형식은 다음과 같다.

```
ProgG2NSI = <hexbyte1> <hexbyte2>
```

여기서 `<hexbyte1,2>` 는 0x00과 0xFF 사이이다. `<hexbyte1>` 값이 1을 초과하면(예를 들어, NSI 값이 0x01FF을 초과), PC단어의 NSI 파트는 프로그래밍 중에 변경되지 않는다. 기본값은 `'FF FF'`이다.

- 제공한 값은 16진법으로 2바이트여야 한다.
- ProgG2NSI의 기본값은 `'FF FF'` 이다.
- 첫 바이트 값이 1을 초과하면, PC단어의 NSI 파트는 프로그래밍 중에 변경되지 않는다.

ProgG2NSI 예제	
Command Response	>ProgG2NSI? ProgG2NSI = FF FF

Command
Response
	>ProgG2NSI = 01 23
	ProgG2NSI = 01 23
	
	// look at existing NSI value (bank #1, word #1)
	>G2Read = 1 1 1
	G2Read = 30 00
	// NSI is all zeros
	
	// program the tag
	>ProgramEPC
	ProgramEPC = 0102 0304 0506 0708 090A 0B0C
	
	// verify that the NSI part of the PC word has changed
	>G2Read = 1 1 1
	G2Read = 31 23
	// NSI is set to 0x123

ProgEPCData

F800 | 9900 | 9680 | 9650

ProgEPCData 값은 ProgramEPC와 ProgramAndLockEPC 명령어를 인수 없이 사용할 때 사용한다. 또한 AutoMode에서는 "ProgramEPC" 또는 "ProgramAndLockEPC" 등 AutoAction으로 프로그램 작업을 선택한 경우에도 사용한다. 이 명령어를 사용하면 다음에 프로그래밍할 EPC 코드를 태그 안에 지정할 수 있다.

- 제공되는 값은 Class1/Gen2에 대하여 짝수 바이트여야 한다(EPC 최대 길이는 태그 실행 특성에 따라 달라진다).
- ProgEPCData 기본값은 00 00 00 00 00 00 00 00 00 00 00 00이다.
- 이 값은 ProgEPCDataInc 설정에 따라 작업 후에 증분될 수 있다(16진수 형태로).

- 이전 리더는 "Program ID" 명령어를 대신 사용하였는데, 후향 호환성을 위해 이 명령어도 아직 수용한다.
- "ProgEPCData = None"의 특수한 경우에는 리더가 프로그래밍할 때 태그의 EPC를 "지운다." 이 경우 EPC 길이가 0으로 설정되며, 따라서 태그에는 EPC가 없다(EPC 태그의 공장 설정 상태이며, Gen2 사양서에 따른다). 이 태그도 인벤토리에 반응하나, TagList 항목은 EPC 코드를 표시하지 않는다.

ProgEPCData 예제	
Command Response	>ProgEPCData = 01 23 45 67 89 AB CD EF ProgEPCData = 01 23 45 67 89 AB CD EF
Command Response	>ProgEPCData? ProgEPCData = 01 23 45 67 89 AB CD EF

ProgUserData

F800 | 9900 | 9680 | 9650

ProgUserData 값은 ProgramUser 명령어를 인수 없이 사용할 때 사용한다. 동일한 데이터를 매번 입력할 필요 없이 여러 태그에 동일한 사용자 데이터를 작성할 수 있게 해주는 단축키 플레이스홀더라고 할 수있다.

- 제공되는 값은 짝수 바이트여야 한다. 최대 길이는 태그 실행 특성에 따라 결정된다.

- ProgUserData 기본값은 00 00이다.
- 이 값은 ProgUserDataInc 값에 따라 ProgramUser 작업 후에 증분될 수 있다(16진수 형태로).

ProgUserData 예제	
Command Response	>ProgUserData? ProgUserData = 00 00
Command Response	>ProgUserData = 01 23 45 67 89 AB CD EF ProgUserData = 01 23 45 67 89 AB CD EF

ProgEPCDataInc

F800 | 9900 | 9680 | 9650

ProgEPCDataInc 명령어를 사용하면 어떤 조건 하에서 ProgEPCData 값이 증분되는지를 지정할 수 있으며, ProgramEPC 명령어 뒤에 오거나 "ProgramEPC" 등과 같이 AutoAction로 선택한 프로그램 작업에서 AutoMode를 사용할 때에도 이 명령어를 사용한다.

값	설명
OFF	ProgEPCData가 절대 증분하지 않는다.
Success	ProgEPCData는 ProgramEPC가 성공할 때에만 증분한다.
Fail	ProgEPCData는 ProgramEPC가 실패할 때에만 증분한다.
Always	ProgEPCData는 ProgramEPC 명령어가 전송될 때마다 증분한다.
Write	ProgEPCData는 쓰기 작업을 시도할 때에만 증분한다. 필드에 태그가 없거나 태그가 식별되지 않았을 경우에는 데이터 값이 증분되지 않으므로 '거짓' 증분 수가 감소한다. **이 옵션은 ALR-9680 & 9650에서는 지원되지 않는다.**

이 명령어는 이전의 미진했던 ProgIncrementOnFail 명령어보다 더 많은 기능성을 제공하나, ProgIncrementOnFail 은 후향 호환성을 위해 보유해 두고 있다. 이 두 명령어는 쌍을 이루는데, 여기서 사용자가 실패를 초래하는 ProgEPCDataInc 값을 증분되도록 설정하면("Fail" 또는 "Always"), ProgIncrementOnFail 이 ON으로 설정되고, 실패를 초래하는 ProgEPCDataInc 값을 증분되지 않게 설정하면("OFF" 또는 "Success"), ProgIncrementOnFail 은 OFF로 설정된다. 이 경우에 ProgIncrementOnFail 은 ProgEPCDataInc 값에 상응하는 영향을 미칠 수 있다.

- ProgEPCDataInc 기본값은 'Success'이다.
- ProgEPCDataInc 은 16진수로 증분한다.

ProgEPCDataInc 예제	
Command Response	>ProgEPCDataInc? ProgEPCDataInc = Success
Command Response	>ProgEPCDataInc = Always ProgEPCDataInc = Always

ProgUserDataInc

F800 | 9900 | 9680 | 9650

ProgUserDataInc 명령어를 사용하면 ProgramUser 명령어를 전송할 때 어떤 조건 하에서 ProgUserData 값이 증분할 지를 지정할 수 있다.

값	설명
OFF	ProgUserData는 결코 증분하지 않는다.
Success	ProgUserData 는 ProgramUser 가 성공할 때에만 증분한다.
Fail	ProgUserData 는 ProgramUser 가 성공할 때에만 증분한다.
Always	ProgUserData 는 ProgramUser 명령어가 전송될 때마다 증분한다.
Write	ProgUserData는 쓰기 작업을 시도할 때에만 증분한다. 필드에 태그가 없거나 태그가 식별되지 않았을 경우에는 데이터 값이 증분되지 않으므로 '거짓' 증분 수가 감소한다. **이 옵션은 ALR-9680 & 9650에서는 지원되지 않는다.**

- 기본값은 'Success'이다.
- ProgUserData는 16진수로 증분한다.

ProgUserDataInc 예제	
Command Response	>ProgUserDataInc? ProgUserDataInc = Success
Command Response	>ProgUserDataInc = OFF ProgUserDataInc = OFF

ProgEPCDataIncCount

F800 | 9900 | 9680 | 9650

ProgEPCDataIncCount 명령어는 ProgEPCData 증분값을 다음과 같이 제한한다.

```
ProgEPCDataIncCount = <count>
```

여기서, <count> 값은 다음과 같다.

```
-1 (default)  : 무제한으로 계속 증분한다.
 0            : 증분을 중지하고 카운터가 -1 나 양수로 재설정될 때까지 프로그래밍을 허용하지 않는다.

 1 to 86400000: 남은 증분 횟수
```

카운터를 'Save' 명령어로 저장할 수 없다. 항상 리더 시동 시에 -1 로 초기화된다.

카운터의 현재 값은 남은 증분 횟수를 가리킨다. ProgEPCData 가 증분할 때마다, 0에 도달할 때까지 ProgEPCDataInc 가 하나씩 줄어든다.

수작업으로 프로그래밍하여(ProgramEPC 명령어를 사용) 카운터가 0이 되면, 태그를 프로그래밍하려고 시도했을 때 ProgEPCDataInc 을 OFF으로 설정해 두지 않는 한 오류가 발생할 것이다. AutoMode로 프로그래밍하고 카운터가 0이 되면, 카운터가 0에 도달했다는 공지가 전송되며(NotifyMode 는 ON이어야 함) AutoMode는 OFF가 된다.

AutoMode는 사용자의 명령 없이도 OFF 가 된다. 이는 NotifyRetryCount 재시도 횟수 이후에 공지 전송이 중단될 때 NotifyRetryCount 의 동작과 유사하다.

카운터를 재설정하지 않고 AutoMode 가 ON 이 되면, AutoAction 주기를 한 차례 수행한 다음 OFF가 된다.

이 동작은 AutoAction = ProgramXXXX 옵션에만 적용된다.

ProgEPCDataIncCount 예제	
Command Response	> ProgEPCDataIncCount? ProgEPCDataIncCount = -1
Command Response	> ProgEPCDataIncCount = 100 ProgEPCDataIncCount = 100

ProgUserDataIncCount

F800 | 9900 | 9680 **|** 9650

ProgUserDataIncCount 명령어는 ProgUserData 증분값을 다음과 같이 제한한다.

```
ProgUserDataIncCount = <count>
```

여기서, <count> 값은 다음과 같다.

-1 (default) : 무제한으로 계속 증분한다.
0 : 증분을 중지하고 카운터가 -1 나 양수로 재설정될 때까지 프로그래밍을 허용하지 않는다.

1 to 86400000: 남은 증분 횟수

카운터를 'Save' 명령어로 저장할 수 없다. 항상 리더 시동 시에 -1 로 초기화된다.

카운터의 현재 값은 남은 증분 횟수를 가리킨다. ProgUserData 가 증분할 때마다, 0 에 도달할 때까지 ProgUserDataInc 가 하나씩 줄어든다.

수작업으로 프로그래밍하여(ProgramUser 명령어를 사용) 카운터가 0이 되면, 태그를 프로그래밍하려고 시도했을 때 ProgUserDataInc 을 OFF으로 설정해 두지 않는 한 오류가 발생할 것이다. AutoMode로 프로그래밍하고 카운터가 0이 되면, 카운터가 0에 도달했다는 공지가 전송되며(NotifyMode 는 ON이어야 함) AutoMode는 OFF가 된다.

AutoMode는 사용자의 명령 없이도 OFF 가 된다. 이는 NotifyRetryCount 재시도 횟수 이후에 공지 전송이 중단될 때 NotifyRetryCount 의 동작과 유사하다.

카운터를 재설정하지 않고 AutoMode 가 ON 이 되면, AutoAction 주기를 한 차례 수행한 다음 OFF가 된다.

이 동작은 AutoAction = ProgramUser 옵션에만 적용된다.

ProgUserDataIncCount 예제	
Command Response	> ProgUserDataIncCount? ProgUserDataIncCount = -1
Command Response	> ProgUserDataIncCount = 100 ProgUserDataIncCount = 100

ProgG2AccessPwd

F800 | 9900 | 9680 | 9650

ProgG2AccessPwd 값은 ProgramAccessPwd 명령어를 인수 없이 사용할 때 사용된다. 또한 AutoMode에서 ProgramAndLockEPC의 AutoAction이 사용될 때도 사용된다(ProgG2LockType=Lock일 때, 접속 패스워드는 태그를 잠글 때 자동으로 작성된다).

이 명령어는 유사한 듯 보이는 AcqG2AccessPwd 명령어와는 다르다. ProgG2AccessPwd는 Access Pwd 값을 태그에 작성하기 위한 플레이스홀더인 반면, AcqG2AccessPwd는 이미 패스워드로 보호하는 Class 1/Gen2 태그에서 작업할 때 사용하는 태그의 현재 접속암호를 제공한다.

- 제공한 값은 4-바이트 값이어야 한다.
- ProgG2AccessPwd 기본값은 00 00 00 00이다.

ProgG2AccessPwd 예제	
Command	>ProgG2AccessPwd = 1 2 3 4
Response	ProgG2AccessPwd = 01 02 03 04
Command	>ProgG2AccessPwd?
Response	ProgG2AccessPwd = 01 02 03 04

ProgG2KillPwd

F800 | 9900 | 9680 | 9650

ProgG2KillPwd 값은 ProgramKillPwd 명령어를 인수 없이 사용할 때 사용한다. 동일한 패스워드를 매번 입력할 필요 없이 여러 태그에 동일한 Kill 패스워드를 작성할 수 있게 해주는 단축키 플레이스홀더라고 할 수있다.

- 제공한 값은 4-바이트 값이어야 한다.
- ProgG2KillPwd 기본값은 00 00 00 00이다.

ProgG2KillPwd 예제	
Command	>ProgG2KillPwd = F1 F2 F3 F4
Response	ProgG2KillPwd = F1 F2 F3 F4
Command	>ProgG2KillPwd?
Response	ProgG2KillPwd = F1 F2 F3 F4

ProgG2LockType

F800 | 9900 | 9680 | 9650

Class 1/Gen 2 프로토콜은 태그 메모리 뱅크/필드를 각각 세 가지 잠금 유형 중 한 가지로 잠글 수 있게 해준다. ProgG2LockType 명령어를 사용하면 다음 잠금 작업에 사용할 잠금 유형을 지정할 수 있다.

잠금 유형	설명
Lock	데이터를 잠그고 향후 잠금해제할 수 있다.
PermaLock	데이터를 잠그고 향후 잠금해제하거나 변경할 수 없다.
PermaUnlock	데이터를 잠그해제하고 향후 다시 잠글 수 없다.

- ProgG2LockType 기본값은 Lock(잠금)이다.

- 기술적으로는 Class 1/Gen 2 프로토콜 "잠금해제"로 정의한 4번째 잠금 유형이 있다. "Unlocked 잠금 유형으로" 잠그는 대신 Alien 리더는 더 명확한 잠금해제 명령어들을 제공한다.

- PermaLock과 PermaUnlock 유형은 영구적이다.
- EPC나 사용자 메모리 뱅크를 잠궈도 읽지 못하는 것은 아니다.
  Kill 패스워드나 접속 패스워드를 잠그면 이 필드들을 읽을 수 없게 된다.

- 태그 내에 0 이외의 접속 패스워드가 있지 않는 한, EPC 또는 사용자 메모리 뱅크를 잠궈도 변경하지 못하는 것은 아니다.
- 0 이외의 접속 패스워드가 태그 내에 있더라도, 잠근 뱅크는 정확한 접속 키만 제공된다면 변경이 가능하다(AcqG2AccessPwd 명령어로). 한 가지 예외는 영구잠금된 데이터로, 정확한 접속 키가 있더라도 변경할 수 없다.

ProgG2LockType 예제	
Command Response	>ProgG2LockType? ProgG2LockType = Lock
Command Response	>ProgG2LockType = PermaLock ProgG2LockType = PermaLock

ProgDataUnit

F800 | 9900 | 9680 | 9650

데이터를 Gen2 태그에 작성하는 일반적인 방법은 데이터를 단어 단위로(16비트) 작성한 다음, 그 다음 단어를, 또 그 단어를 작성하는 것으로, 다시 읽어 보면서 각 단어를 확인할 수 있을 것이다. Gen2 태그에 작성하는 다른 방법으로 "block write"가 있는데, 이 경우 리더는 순번대로 작성할 단어의 목록을 메모리에 전달한다. 이런 블록 쓰기는 단어 별로 작성할 때보다 현저하게 더 빠르다.

Alien 리더는 두 가지 쓰기 유형에 필요한 세부사항을 모두 고려하며, 사용자는 어떤 모드를 사용할 것인지를 ProgDataUnit 명령어로 알려주기만 하면 된다. 기존의 프로그래밍 명령어들, 이를테면 ProgramEPC, ProgramUser, G2Write, AutoMode 내 프로그래밍 등은 전부 'block-write' 방법을 지정하면 이 방법으로 자동 전환한다.

BlockWrite 명령어는 Gen2의 선택 기능이므로, 모든 태그가 지원하는 것은 아니다. Alien Higgs3와 Higgs4 태그는 이 기능을 수행하는 첫 Alien 태그들이다. **BlockWrite를 실행하지 않는 태그에서 블록 쓰기를 수행하려고 하면 작업이 실패할 것이다. 태그가 이를 지원하는 것이 확실할 때에만 BlockWrite를 사용하기 바란다.**

- 허용된 값: "Word" | "Block"
- 기본값: "Word"

ProgDataUnit 예제	
Command Response	>ProgDataUnit? ProgDataUnit = Word
Command Response	>ProgDataUnit = Block ProgDataUnit = Block

ProgBlockSize

F800 | 9900 | 9680 | 9650

블록 모드로 프로그래밍할 때(`ProgDataUnit = block`), 리더는 각 트랜잭션과 함께 더 많은 데이터를 전송함으로써 더 빨리 태그에 데이터를 작성할 수 있다. `ProgBlockSize` 명령어를 사용하면 블록의 크기를

지정할 수 있다. Alien Higgs 태그를 사용하면 한 번에 32개 단어까지 쓸 수 있으며, 다른 태그 제조자의 경우에는 2개 단어만 허용한다.

블록 경계를 넘어서는 데이터를 쓰려고 하면 쓰기 작업이 실패한다(이어지는 ProgBlockAlign 명령어 참조).

`ProgBlockSize` 의 기본값은 0(i.e. "default")이며, 이는 32 단어에 상응한다.

- 허용된 값: 0, 2, 4, 8, 16, 32
- 기본값: 0 (리더는 합리적인 값, 현재로서는 32를 자동 선택한다)

ProgBlockSize 예제	
Command Response	>ProgBlockSize? ProgBlockSize = 0
Command Response	>ProgBlockSize = 2 ProgBlockSize = 2

ProgBlockAlign

F800 | 9900 | 9680 **|** 9650

블록 모드로 프로그래밍할 때, 사용자는 블록 경계를 벗어나 작성하지 않도록 주의해야 하는데(블록 크기는 태그 제조자들마다 다르다), 그렇지 않으면 태그에 오류가 발생할 것이다. `ProgBlockAlign=On/Off` 명령어를 켜면 사용자가 태그에 작성하기 원하는 데이터를 자동으로 분할하도록 리더에 지시할 것이며, 이로써 각각의 조각을 태그에 별도로 블록 쓰기 할 것이다. 이렇게 하면 블록 경계를 넘어서 쓸 때의 오류를 피할 수 있다.

Alien Higgs 태그의 경우, 블록 크기는 32 단어이며, 가장 큰 메모리 뱅크(USER) 크기보다 더 작은 일은 없다. 때문에 사용자는 다른 제조자의 태그를 사용하지 않는 한 블록 경계를 넘는 문제는 결코 없을 것이다.

- 허용된 값: On | Off
- 기본값: Off

ProgBlockAlign 예제	
Command Response	>ProgBlockAlign? ProgBlockAlign = Off
Command Response	>ProgBlockAlign = On ProgBlockAlign = On

ProgAttempts

F800 | 9900 | 9680 | 9650

ProgAttempts 속성은 프로그램 태그 작업 중 태그 프로그래밍을 리더가 시도하는 횟수를 명시한다.

- 허용된 값: 정수(1..7)
- 기본값: 3

ProgAttempts 예제	
Command Response	>ProgAttempts? ProgAttempts = 3
Command Response	>ProgAttempts = 5 ProgAttempts = 5

ProgSuccessFormat

F800 | 9900 | 9680 | 9650

"Program Tag" 명령어는 프로그래밍 시도가 성공할 경우 태그 내에 프로그래밍할 ID를 반환한다. 펌웨어의 이전 버전들은 단순히 "Success!"라고만 반환했다. 사용자는 ProgSuccessFormat 값으로 리더의 동작을 선택할 수 있다.

0 = "Success!"
1 = 프로그래밍한 태그 ID (기본값)

ProgSuccessFormat 예제	
Command Response	>ProgSuccessFormat? ProgSuccessFormat = 0 >Program Tag = 11 22 33 44 55 66 77 88 Program Tag = Success!
Command Response	>ProgSuccessFormat = 1 ProgSuccessFormat = 1 >Program Tag = 11 22 33 44 55 66 77 88 Program Tag = 11 22 33 44 55 66 77 88

ProgSingulate

F800 | 9900 | 9680 **|** 9650

프로그래밍 작업을 시도했고 필드에 둘 이상의 태그가 있을 경우 결과는 예측할 수 없다. 여러 태그가 동시에 반응하므로 프로그래밍은 성공할 수도 있고 실패할 수도 있다.
ProgSingulate 옵션을 켜면, 프로그래밍 작업을 시도할 때마다 프로그래밍 직전에 짧은 인벤토리를 수행하며, 필드에 둘 이상의 태그가 있으면, 프로그래밍하려는 시도 없이 "Singulation Error 149"라는 메시지를 반환한다.

형식은 다음과 같다:

ProgSingulate = ON | OFF

이 명령어는 G2Write와 G2Erase를 제외한 모든 Gen2 프로그래밍 명령어에 적용된다(AutoMode AutoActions 포함). 기본값은 'OFF'이다.

- 허용된 값: "ON" | "OFF"
- 기본값: "OFF"

ProgSingulate 예제	
Command Response	>ProgSingulate? ProgSingulate = OFF
Command Response	// set to check-if-singulated before an attempt to program >ProgSingulate = ON ProgSingulate = ON >ProgramEPC Error 149: Singulation error > t Tag:E200 3411 B801 0108 6500 6464, ... Tag:E200 3411 B801 0108 6500 6465, ...

Low-Level Programming – TagInfo, G2Read, G2Write

Class 1/Gen 2 태그에는 여러 메모리 필드가 포함되어 있으며, 사용자는 앞서 기술한 명령어들을 사용해 이 필드에 편리하게 작성할 수 있다(ProgramEPC, ProgramAccessPwd, ProgramKillPwd, and ProgramUser). 단, 태그 메모리의 특정 구획에 임의로 작성해야 할 필요가 있을지 모른다. 이를 위해서는 저수준 G2Write 명령어를 사용할 수 있다.

또한 리더는 TagList 데이터 내에 있는 태그의 EPC 필드만 노출한다(또한 사용자 정의 TagList 표시를 사용할 경우 PC 단어). 접속 및 Kill 패스워드, 사용자 데이터, NSI 필드 등의 태그 쿼리를 위해서는 저수준 G2Read 명령어를 사용해야 한다.

G2Read와 G2Write 명령어는 둘 다 태그 메모리의 단어 경계에서 작동하며, 사용자는 데이터의 개별 비트나 바이트를 일거나 쓸 수 없다. 또한 태그 메모리에 뱅크 및 워드 포인터를 지정하여야 하므로, 사용자는 태그 메모리 구조에 대해 충분히 이해해야 한다. 각 뱅크 내에서 뱅크 번호와 워드 포인터는 0부터 시작한다.

기억할 점:

G2 bank 0 - RESERVED (Kill과 Access 패스워드)
G2 bank 1 - EPC (CRC + PC + EPC)
G2 bank 2 - TID (Tag 식별자)
G2 bank 3 - USER (사용자 지정 데이터로 전체 태그에서 지원되지 않음)

TagInfo는 TID 뱅크와 기타 다른 태그 구성요소들을 읽으며, 사용자에게 다양한 필드의 잠금 상태, 제조자와 태그 모델, PC워드, 사용자 메모리 가용성 등을 요약하여 반환한다.

이러한 명령어는 현재 수집 마스크에 부합하는 태그에서만 작동하며, 태그가 지원하는 태그 메모리 뱅크 접속을 가능케 한다. 리더는 현재 ProgAntenna에서 이러한 명령을 수행한다.

접속하려는 메모리 뱅크가 잠금 상태이면, 정확한 접속 패스워드를 제시해야 한다(AcqG2AccessPwd로).

TagInfo

F800 | 9900 | 9680 | 9650

TagInfo 명령어는 제조자 ID, 태그 모델 및 버전 등의 태그 정보를 제시한다. Alien 태그의 경우, TagInfo는 확장되어 메모리 뱅크의 잠금 상태, 고유 ID, PC 워드 등의 정보도 포함한다.

이 명령어의 형식은 다음과 같다:

```
TagInfo?
```

리더는 태그를 찾아 정보를 검색한다. 이 명령을 수행할 때에는 읽기 영역을 한정하거나 특정 태그에 마스크를 생성하는 것이 가장 좋다. 리더는 태그 유형에 따라 가능한 많은 정보를 보고한다.

Alien Higgs3 & Higgs4:
TagInfo = M:xxxxxxxx KP:xx AP:xx EPC:xx,n USER:xx,n UPL:xxxx URL:xxxx PC:xxxx
UID:xxxx xxxx xxxx xxxx

Alien Higgs2:
TagInfo = M:xxxxxxxx MAP:x KP:xx AP:xx EPC:xx,n User:xx PC:xxxx

기타 태그:
TagInfo = M:xxxxxxxx

여기서
M:xxxxxxxx 는 TID의 처음 두 단어이며, EPCglobal 태그인지, 태그의 제조자와 모델/버전 코드를 표시한다.

KP:xx 는 Kill 패스워드의 잠금 상태이다.
AP:xx 는 접속 패스워드의 잠금 상태이다.
EPC:xx,n 은 EPC 뱅크의 잠금 상태와 길이(단어)이다.
USER:xx,n 은 USER 뱅크의 잠금 상태와 길이이다(단어).
UPL:xxxx 는 사용자 블록 영구잠금 비트이다.
URL:xxxx 는 사용자 읽기 잠금 비트이다.
PC:xxxx 는 PC 워드이다.
MAP:x 는 Higgs2 메모리 맵 번호이다.
UID:xxxx xxxx xxxx xxxx 는 Higgs 태그마다 보장되어 있는 네 개의 고유 단어이다.

TagInfo 예제	
Command Response	// Standard Higgs3 tag (6 EPC words, 32 USER words) >taginfo? TagInfo = M:E2003412 KP:UL AP:UL EPC:UL,6 USER:UL,32 UPL:0000 URL:0000 PC:3000 UID:012F F200 0044 F817
Command Response	// Reprogrammed Higgs3 tag (31 EPC words, 4 USER words) >taginfo? TagInfo = M:E2003412 KP:UL AP:UL EPC:UL,31 USER:UL,4 UPL:0000 URL:0000 PC:F800 UID:0133 F000 0486 6128
Command Response	// Higgs2 tag >taginfo? TagInfo = M:E2003411 MAP:4 KP:UL AP:UL EPC:UL,6 PC:3000
Command Response	// Non-Alien Higgs tag >taginfo? TagInfo = M:E2006004

G2Read

F800 | 9900 | 9680 | 9650

G2Read 명령어는 하나의 Class 1/Gen 2 태그에서 저수준 메모리 읽기를 수행한다. 사용자는 메모리 뱅크, 뱅크 내 워드 위치, 읽을 워드의 수를 지정해야 한다. 태그 메모리의 일부 구획은 읽을 수 없을 수도 있다. 예를 들어, 잠긴 상태의 Access 또는 Kill 패스워드 필드는 다시 읽지 못할 수 있다.

G2Read 명령어 구문은 다음과 같다:
G2Read = <bank> <wordPtr> <wordLen>
여기서:

<bank> = 0 – 3
<wordPtr> = 처음 읽을 단어의 위치 (0-2097151, base 10)
<wordLen> = 읽을 단어의 수(0-32, base 10)

사용자가 G2Read 명령어에서 제공하는 매개변수는 사이띄기나 콤마 조합으로 분리할 수 있다. 반환된 데이터는 성공하거나 오류 메시지를 보낼 때 사이띄기로 분리한 16진수 짝수 바이트이다.

<wordLen>=0 조건으로 인해 태그가 <wordPtr>에서 뱅크 끝으로 <bank> 내 모든 메모리를 반환하게 하는 특별한 경우 이 경우를 "zero-length read(길이가 0인 읽기)"라 칭하며, 특정 메모리 뱅크의 크기를 모를 때 유용할 수 있다. 반환된 데이터의 양이 32단어보다 크면, 오류 메시지가 보일 수 있다. 어떤 경우에는 길이가 0인 G2Read가 보통 0이 아닌 길이의 읽기에서는 보지 못할 태그 데이터를 반환할 수도 있다. 태그가 메모리에서 "sliding partitions(슬라이딩 파티션)"을 사용하는 경우에는 "the end of the bank(뱅크 끝)"이 명확히 정의되지 않을 수도 있다.

G2Read 예제

Command Response	// Read the CRC and PC words (bank 1, words 0-1) >G2Read = 1 0 2 G2Read = 42 E7 30 00 (CRC=42E7, PC=3000)

Command
Response
 	
Command
Response
	// Read the 96-bit EPC (bank 1, starting at word #2, 6 words)
	>G2Read = 1 2 6
	G2Read = CF F7 CC 4D 22 26 FE BF 00 00 00 00
	
	// Compare to a "get TagList" response
	>get TagList
	Tag:CFF7 CC4D 2226 FEBF 0000 0000, Disc:2007/07/09 14:56:30...

Command Response	// Read two words of user memory (tag must support this) >G2Read = 3, 0, 2 G2Read = DE AD BE EF
Command Response	// Read Access password (bank 0, words #2-3 - must not be locked) >G2Read = 0, 2, 2 G2Read = 01 02 03 04
Command Response	// Try to read a memory location that doesn't exist // (tag not supporting User memory) >G2Read = 3 0 2 Error 259: (Tag Error) Memory overrun or unsupported PC value.
Command Response	// Read all of the TID bank (zero-length read!) >G2Read = 2, 0, 0 G2Read = E2 00 34 12 01 61 00 00 00 00 00 00 03 1D 01 60 70…

G2Write

F800 | 9900 | 9680 | 9650

G2Write 명령어는 하나의 Class 1/Gen 2 태그에서 저수준 메모리 쓰기를 수행한다. 사용자는 메모리 뱅크, 뱅크 내 워드 위치, 쓰기 할 데이터를 지정해야 한다(짝수 바이트). 태그 메모리의 일부 구획은 쓸 수 없을 수도 있다. 예를 들어, 태그 EPC의 CRC나 잠긴 필드(일반적으로)는 쓰지 못할 수 있다. G2Write 명령어의 구문은 다음과 같다.

```
G2Write = <bank> <wordPtr> <data>…
```

where: <bank> = 0-3
<wordPtr> = 처음 쓸 단어의 위치 (0-2097151, base 10)
<data>… = 쓸 단어의 수(1-32, base 10)

사용자가 G2Write 명령어에서 제공하는 매개변수는 사이띄기나 콤마 조합으로 분리할 수 있다. 리더는 "Success!" 또는 오류 메시지로 응답할 것이다.

G2Write 예제	
	// Write the EPC "the hard way" (bank 1, beginning at word #2)
Command	// This is **not** a good way to write the EPC!
Response	>G2Write = 1, 2, 11 22 33 44 55 66 77 88 99 10 11 12
	G2Write = Success!
Command	// Double-check with an inventory
Response	>get TagList
	Tag:1122 3344 5566 7788 9910 1112, Disc:2006/10/19 14:56:30...
	// Write to User memory (bank 3 – must be supported by the tag)
Command	>G2Write = 3, 0, DE AD BE EF CA FE
Response	G2Write = Success!
	// Read that User memory back (3 words)
Command	>G2Read = 3, 0, 3
Response	G2Read = DE AD BE EF CA FE
	// Try to write to a memory location that doesn't exist
	// (tag not supporting User memory)
Command	>G2Write = 3, 0, DE AD BE EF CA FE
Response	Error 259: (Tag Error) Memory overrun or unsupported PC value.

Programming an Entire Tag Image (Alien Higgs Tags Only)

다음 명령어는 Alien Higgs2와 Higgs3 태그에만 존재하는 사용자 정의 Class 1/Gen 2 태그 명령어를 노출한다. Alien 태그 이미지를 프로그래밍할 때에는 하나의 명령어에 태그 메모리 전부에 작성한다(패스워드, EPC, 사용자 데이터 등). 전체 태그 이미지를 프로그래밍하면 태그 메모리 뱅크를 별도로 작성할 때 보다 훨씬 더 빠르다. 현재 리더 수행 조건에서는 작성한 태그의 모든 필드/뱅크가 잠금해제 상태로 남아 있다.

리더는 현재 ProgAntenna에서 ProgramAlienImage 작업을 수행하며, 태그는 현재 수집 마스크에 부합해야 한다. 또한 Alien Higgs2 태그의 경우에는 태그가 공장 기본값 상태의 "Higgs2_96" 메모리 맵에 있어야 하며, 필드나 뱅크가 잠긴 상태여서는 안 된다. Alien Higgs3 태그는 이러한 제약이 없으며, 단순화된 메모리 맵을 제시한다.

ProgramAlienImage

F800 | **9900** | 9680 | 9650

ProgramAlienImage 명령어는 전체 태그의 메모리 맵 작성을 위한 기능이다. 인수는 전혀 필요하지 않다. 리더가 ProgEPCData, ProgUserData, ProgG2AccessPwd, ProgG2KillPwd 명령어에서 값을 취하여 리더로 보낼 이미지 바이트의 적절한 순서를 자동으로 구성하며, 차후 설명할 ProgAlienImageMap과 ProgAlienImageNSI 명령어에 정보를 제공한다. 데이터 필드 중에는 선택한 메모리 맵에 따라 무관한 것들도 있을 수 있다.

ProgEPCData와 ProgUserData 값은 ProgramAlienImage 명령어 전송 후에 자동으로 증분될 수 있으며, 이때 ProgEPCDataInc과 ProgUserDataInc 속성의 설정과 작동 결과에 따라 달라진다. ProgUserData는 사용자 데이터가 태그에 작성되었을 때에만 증분한다.

- ProgramAlienImage 명령어가 성공했을 때의 응답으로 태그에 보낸 전체 이미지 바이트 시퀀스가 표시된다.
- Alien Higgs2 태그는 공장 기본값 Higgs2_96 메모리 맵 상태 그대로 있고 필드가 잠기지 않았을 때에만 ProgramAlienImage 기능을 허용한다. 다른 맵 중 하나를 한번만 변경할 있으며, 또는 Higgs2_96 맵을 계속 다시쓰기 할 수도 있다.
- Alien Higgs3 태그 이미지는 필요할 때마다 변경할 수 있다.
- Alien Higgs4 태그는 ProgramAlienImage 기능을 지원하지 않는다.
- 태그 내 필드 중 잠긴 것이 있으면 이미지 맵을 쓰기 전에 잠금해제해야 한다. 현재로서는 이 태그는 ProgramAlienImage 후에 완전히 잠금해제된 상태를 유지한다. 향후 사용자가 여러 뱅크와 메모리 블록의 원하는 잠금 상태를 지정할 수도 있을 것이며 ProgramAlienImage를 사용해 구성할 수도 있을 것이다.

ProgramAlienImage 예제	
	// Preset the data to write (only needs to be done initially)
Command	Alien>ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66
Response	ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66
Command	Alien>ProgG2KillPwd = AA AA AA AA
Response	ProgG2KillPwd = AA AA AA AA
Command	Alien>ProgG2AccessPwd = BB BB BB BB
Response	ProgG2AccessPwd = BB BB BB BB
	// See following section for other ProgAlienImageMap examples
Command	Alien>ProgAlienImageMap = Higgs2_96
Response	ProgAlienImageMap = Higgs2_96
	// See following sections regarding ProgAlienImageNSI
Command	Alien>ProgAlienImageNSI = 00 00
Response	ProgAlienImageNSI = 00 00
	// Program the entire tag image
Command	Alien>ProgramAlienImage
Response	AAAA AAAA BBBB BBBB 1111 2222 3333 4444 5555 6666 3000 03B8
	// Verify the EPC
Command	Alien>t
Response	Tag:1111 2222 3333 4444 5555 6666, Disc:2007/07/06 13:05:34…
	// Verify the Kill and Access passwords
Command	Alien>G2Read = 0 0 4
Response	G2Read = AA AA AA AA BB BB BB BB
	// Check that the ProgEPCData field automatically incremented
Command	Alien>ProgEPCData?
Response	ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 67

ProgAlienImageMap

F800 | **9900** | 9680 | 9650

Alien Class 1/Gen 2 태그는 몇 가지 종류의 칩과 Higgs2, Higgs3, Higgs4를 포함한다. Higgs2와 Higgs3만 ProgramAlienImage 기능과 관련 속성들을 지원한다.

HIGGS2

Alien Higgs2 태그는 세 가지 메모리 맵 중 하나로 구성이 가능하다. 이 태그 내에 메모리의 총량이 일정한데 반해, 각 메모리 맵은 Kill과 Access 패스워드 중 하나 혹은 둘이 확장됨에 따라 EPC 길이와 사용자 메모리 옵션 조합을 다수 제공한다. 리더는 ProgAlienImageMap 속성과 관련 ProgEPCData, ProgUserData 등이 지정한 이미지 맵을 자동으로 구성한다.

Higgs2 태그는 다음과 같이 TID 뱅크(뱅크 2)의 두 번째 단어에서 0x3411을 찾아 식별할 수 있다.

```
Alien>G2Read = 2 1 1
G2Read = 34 11
```

Higgs2 태그로 작업할 때 ProgAlienImageMap에 허용된 값은 다음과 같다:

ProgAlienImageMap (Higgs2)	EPC (bits)	USER (bits)	Access Pwd	Kill Pwd
Higgs2_96	96	0	✓	✓
Higgs2_128	128	0		✓
Higgs2_96U64	96	64		

- Alien Higgs2 태그는 공장 기본값 Higgs2_96 메모리 맵 상태 그대로 있고 필드가 잠기지 않았을 때에만 ProgramAlienImage 기능을 허용한다. 다른 맵 중 하나를 한번만 변경할 있으며, 또는 Higgs2_96 맵을 계속 다시쓰기할 수도 있다. 필드 중 잠긴 것이 있으면 이미지 맵을 쓰기 전에 잠금해제해야 한다.
- 이전 펌웨어 버전들은 이 메모리 맵핑을 "EPC96", "EPC128", "EPC96USER64"로 노출하였다. 리더는 후향 호환성을 위해 아직 이처럼 오래된 메모리 맵 명칭도 수용한다.

HIGGS3

Alien Higgs3 태그는 훨씬 더 유동적인 메모리 아키텍처를 적용한다. Access와 Kill 패스워드는 항상 있으며, EPC 메모리의 최소 96비트와 사용자 메모리의 64비트가 있다. EPC 데이터와 사용자 데이터 간 구획 분리는 정해져 있지 않으므로, 사용자가 두 뱅크 간에 태그 메모리를 분할하는 방법에 있어 어느 정도 재량을 갖는다.

Higgs3 태그는 다음과 같이 TID 뱅크(뱅크 2)의 두 번째 단어에서 0x3412을 찾아 식별할 수 있다.

```
Alien>G2Read = 2 1 1
G2Read = 34 12
```

Alien Higgs3 태그는 Alien Higgs2 태그처럼 고정된 메모리 맵핑이 없으며, 따라서 사용자가 각 뱅크마다 상이한 양의 데이터를 작성함으로써 EPC+사용자 메모리 구획을 변경하면 된다. Higgs3 태그 메모리 로딩을 위해서는, ProgAlienImageMap 명령어에서 "Higgs3" 맵핑을 선택하면 된다.

표준 이미지 로드보다 훨씬 더 빠른 Higgs3 메모리를 로딩하는 다른 방법이 있다(각각의 뱅크를 별도로 작성할 때보다 훨씬 더 빠름). 이 수단은 "fast load(빠른 로딩)"로 알려져 있으며, Access & Kill 패스워드와 96비트 EPC만 작성한다. 다른 EPC 메모리와 사용자 메모리는 모두 지운다. 사용자는 ProgAlienImageMap 명령어에서 "Higgs3_96" 맵핑을 선택함으로써 이 모드를 선택할 수 있다.

ProgAlienImageMap (Higgs3)	EPC (bits)	USER (bits)	Access Pwd	Kill Pwd
Higgs3	96-496	512-64	✓	✓
Higgs3_96	96	-	✓	✓

EPC+사용자 뱅크에 할당하는 메모리 총량은 608비트를 초과할 수 없으며, 충족해야 할 워드/블록 경계 조건들도 있다. 허용된 구획은 아래 약술하였다.

EPC Size (bits)	USER Max (bits)
0-96	512
112-160	448
176-224	384
240-288	320
304-352	256
368-416	192
432-480	128
496	64

ProgAlienImageMap 예제 – Higgs2 128-bit EPC	
	// Program a new Higgs2 tag with a 128-bit EPC
Command	>ProgAlienImageMap = Higgs2_128
Response	ProgAlienImageMap = Higgs2_128
Command	>ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66 77 77 88 88
Response	ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66 77 77 88 88
Command	>ProgG2KillPwd = AA AA AA AA
Response	ProgG2KillPwd = AA AA AA AA
	// Program the tag
Command	>ProgramAlienImage
Response	AAAA AAAA 1111 2222 3333 4444 5555 6666 7777 8888 4000 81B8
	// Verify the new 128-bit EPC
Command	>t
Response	Tag:1111 2222 3333 4444 5555 6666 7777 8888, Disc:2007/07/06…
	// Verify the Kill password
Command	>G2Read = 0 0 2
Response	G2Read = AA AA AA AA

ProgAlienImageMap 예제 – Higgs2 w/User Data	
	// Program a new Higgs2 tag with User memory
Command	>ProgAlienImageMap = Higgs2_96U64
Response	ProgAlienImageMap = Higgs2_96U64
Command	>ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66
Response	ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66
Command	>ProgUserData = AA AA BB BB CC CC DD DD
Response	ProgUserData = AA AA BB BB CC CC DD DD
	// Program the tag
Command	>ProgramAlienImage
Response	AAAA BBBB CCCC DDDD 1111 2222 3333 4444 5555 6666 3000 41B8
	// Verify the new EPC
Command	>t
Response	Tag:1111 2222 3333 4444 5555 6666, Disc:2007/07/06 14:14:24…
	// Verify the User data
Command	>G2Read = 3 0 4
Response	G2Read = AA AA BB BB CC CC DD DD

ProgAlienImageMap 예제 – Higgs3 Full Load	
	// Program a Higgs3 tag with a 496-bit EPC and 64-bits of User
Command	>ProgAlienImageMap = Higgs3
Response	ProgAlienImageMap = Higgs3
Command	>ProgEPCData = 11 12 13 14 15 16 17 18 21 22 23 24 25 26 27 28 31 32 33 34 35 36 37 38 41 42 43 44 45 46 47 48 51 52 53 54 55 56 57 58 61 62 63 64 65 66 67 68 71 72 73 74 75 76 77 78 81 82 83 84 85 86
Response	ProgEPCData = 11 12 13 14 15 16 17 18 21 22…
Command	>ProgUserData = A1 A2 A3 A4 A5 A6 A7 A8
Response	ProgUserData = A1 A2 A3 A4 A5 A6 A7 A8
Command	>ProgG2KillPwd = AA AA AA AA
Response	ProgG2KillPwd = AA AA AA AA
Command	>ProgG2AccessPwd = BB BB BB BB
Response	ProgG2AccessPwd = BB BB BB BB
	// Program the tag
Command	>ProgramAlienImage
Response	ProgramAlienImage = BBBB BBBB AAAA AAAA F800 1112 1314…
	// Verify the new 496-bit EPC
Command	>t
Response	Tag:1112 1314 1516 1718 2122 2324 2526 2728 3132 3334 3536 3738 4142 4344 4546 4748 5152 5354 5556 5758 6162 6364 6566 6768 7172 7374 7576 7778 8182 8384 8586, Disc:2008/08/12 13:43:16, Last:2008/08/12 13:43:16, Count:1, Ant:0, Proto:2
	// Verify the Kill & Access passwords
Command	>G2Read = 0 0 0
Response	G2Read = AA AA AA AA BB BB BB BB
	// Verify User memory
Command	>G2Read = 3 0 0
Response	G2Read = A1 A2 A3 A4 A5 A6 A7 A8

ProgAlienImageNSI

F800 | **9900** | 9680 | 9650

모든 Class 1/Gen 2 태그의 PC (Protocol Control) 단어는 EPC 직전 워드 #1 뱅크 #1에 있다. PC 워드의 상위 5비트(비트 0-4, 0이 최우선)는 EPC 데이터 길이를 지정하며, EPC를 프로그래밍할 때 리더가 자동으로 작성한다. 다음 2비트(비트 5,6)는 EPC Class 1/Gen 2 프로토콜이 향후 사용을 위해 예비로 확보해 두며 항상 0이어야 한다. 마지막 9비트(비트 7-15)는 NIS(Numbering System Identifier)를 포함한다.

NSI의 첫 비트(PC 워드의 비트#7)는 토글 비트로서, EPC 값을 EPCglobal™ 번호부여 시스템을 사용해 인코딩했는지(비트 #7가 0인 경우), 또는 ISO/IEC 15961 인코딩인지(비트#7이 1인 경우)를 표시한다. EPCglobal 인코딩인 경우, NSI의 나머지는 확보해 두며, 모두 0이어야 한다. ISO/IEC 15961 인코딩인 경우, 나머지 비트는 AFI(Application Family Identifier) 정보를 제공한다. NSI의 기본값은 모두 0이다(EPCglobal 인코딩을 표시함).

사용자가 특정 NSI 값으로 태그를 인코딩해야 할 경우, ProgAlienImageNSI 속성으로 이 정보를 제공할 수 있다. 리더는 ProgramAlienImage 명령어를 실행하는 중에 Alien 태그 이미지 맵을 구성할 때 이 값을 자동으로 사용한다. 사용자가 이미지 맵 24비트를 손수 제공하여 ProgramAlienImage 명령어를 사용하는 경우, PC 워드의 일부로 이미지 맵 데이터 내에 원하는 NSI 값을 직접 삽입해야 한다.

ProgAlienImageNSI는 하나의 단어로 지정하며(2바이트), 단 실제로는 9개의 최하위 비트를 사용한다(처음 바이트의 하위 비트와 두 번째 바이트의 8비트 모두).

ProgAlienImageNSI 예제

```
            // First, look at existing NSI (bank #1, word #1, ignore 1st nibble)
Command     Alien>G2Read = 1 1 1
Response    G2Read = 30 00
            // NSI is all zeros → EPCglobal encoding

Command     Alien>ProgAlienImageMap = Higgs2_96
Response    ProgAlienImageMap = Higgs_95

            // Program an NSI that is all ones
Command     Alien>ProgAlienImageNSI = 01 FF
Response    ProgAlienImageNSI = 01 FF

Command     Alien>ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66
Response    ProgEPCData = 11 11 22 22 33 33 44 44 55 55 66 66

            // Program the tag
Command     Alien>ProgramAlienImage
Response    0000 0000 0000 0000 1111 2222 3333 4444 5555 6666 31FF 03B8

            // Verify the new custom NSI
Command     Alien>G2Read = 1 1 1
Response    G2Read = 31 FF
            // NSI is all ones
```

Programming Tags in AutoMode

프로그래밍 기능들은 리더의 작업 자율 모드로 완전히 지원된다. AutoMode를 사용하면 사용자가 태그 프로그래밍, 지우기, 잠그기를 리더에 배정할 수 있다. 아래 기술한 바와 같이, AutoAction 속성을 적절한 동작에 설정하기만 하면 된다.

일부 AutoMode 기능들, 이를테면 "ProgramEPC"와 "ProgramAndLockEPC"는 아래 기술한 대로 우선 몇 가지 추가적인 값을 지정해야 한다.

AutoAction = ProgramEPC (was “Program”)

F800 | 9900 | 9680 | 9650

ProgramEPC의 AutoAction을 AutoMode에서 실행할 때, 리더는 AutoMode에서 새로운 EPC로 태그를 자동 프로그래밍할 수 있으며, 이 동작은 무한정 반복하면서 매번 프로그래밍을 위해 EPC를 증분한다. 이렇게 하면 AutoMode에 있는 동안 프로그래밍된 태그 그룹에 순서대로 태그 ID가 배정된다.

AutoMode가 EPC를 프로그래밍하게 하려면 다음 명령어를 전송한다.

```
AutoAction = ProgramEPC
```

"ProgramEPC" 대신 이전의 리더들은 "Program"을 사용했으며, 후향 호환성을 위해 이 값도 승인한다.

이 "work" 단계에서 태그를 읽는 대신(자세한 정보는 2장에서 AutoMode 상태 도표 참조), 리더는 ProgEPCData 값으로 지정한 필드 내 태그를 EPC로 프로그래밍 한다(상기 내용 참조).

프로그래밍 이벤트가 성공하면 AutoMode 평가 단계가 진행되어 `true`(참)인지를 평가한다. 프로그래밍 이벤트가 실패하면 `false`(거짓)으로 평가한다.

리더는 ProgEPCDataInc 설정값에 따라서 다음에 프로그래밍할 EPC를 증분하기로 선택할 수 있다.

AutoAction = ProgramAndLockEPC

F800 | 9900 | 9680 **|** 9650

ProgramAndLockEPC의 AutoAction으로 AutoMode를 실행할 때에는, 리더가 새로운 EPC로 태그를 자동 프로그래밍할 수 있으며, 그 다음에는 패스워드로 잠근다. 이 이벤트는 무한정 반복할 수 있으며, 매번 하나씩 EPC가 증분한다.

AutoMode가 EPC를 프로그래밍하게 하고 잠그게 하려면 다음 명령어를 전송한다.

```
AutoAction = ProgramAndLockEPC
```

"ProgramAndLockEPC" 대신 이전의 리더들은 "Program and Lock"을 사용했으며, 후향 호환성을 위해 이 값도 승인한다.

이 "work" 단계에서 태그를 읽는 대신(자세한 정보는 2장에서 AutoMode 상태 도표 참조), 리더는 ProgEPCData 값으로 지정한 태그를 EPC로 프로그래밍하고 잠근다. Class 1/Gen 2 태그의 경우, ProgG2AccessPwd는 ProgG2LockType=Lock일 때에만 태그에 작성한다. PermaLock과 PermaUnlock 잠금 유형의 경우에는 ProgG2AccessPwd를 작성하지 않는다(잠금이 영구적인 본질 때문에 접속 패스워드가 필요하지 않다).

ProgramAndLockEPC 이벤트가 성공하면 AutoMode 평가 단계가 진행되어 `true`(참)인지를 평가한다. ProgramAndLockEPC 이벤트가 실패하면 `false`(거짓)으로 평가한다.

리더는 ProgEPCDataInc 설정값에 따라서 다음에 프로그래밍할 EPC를 증분하기로 선택할 수 있다.

AutoAction = ProgramUser

F800 | 9900 | 9680 | 9650

ProgramUser의 AutoAction을 AutoMode에서 실행할 때, 리더는 AutoMode에서 새로운 사용자 데이터로 태그를 자동 프로그래밍할 수 있으며, 이 동작은 무한정 반복하면서 매번 프로그래밍을 위해 사용자 데이터를 증분한다. 이렇게 하면 AutoMode에 있는 동안 프로그래밍된 태그 그룹에 순서대로 사용자 데이터가 배정된다.

AutoMode가 사용자 데이터를 프로그래밍하게 하려면 다음 명령어를 전송한다.

```
AutoAction = ProgramUser
```

이 "work" 단계에서 태그를 읽는 대신(자세한 정보는 2장에서 AutoMode 상태 도표 참조), 리더는 ProgUserData 값으로 지정한 필드 내 태그를 사용자 데이터로 프로그래밍 한다(상기 내용 참조).

프로그래밍 이벤트가 성공하면 AutoMode 평가 단계가 진행되어 `true`(참)인지를 평가한다. 프로그래밍 이벤트가 실패하면 `false`(거짓)으로 평가한다.

리더는 ProgUserDataInc 설정값에 따라서 다음에 프로그래밍할 사용자 데이터를 증분하기로 선택할 수 있다.

AutoAction = ProgramAndLockUser

F800 | 9900 | 9680 **|** 9650

ProgramAndLockUser의 AutoAction으로 AutoMode를 실행할 때에는, 리더가 새로운 사용자 데이터로 Class1/Gen2 태그를 자동 프로그래밍할 수 있으며, 그 다음에는 패스워드로 잠근다. 이 이벤트는 무한정 반복할 수 있으며, 매번 하나씩 사용자 데이터가 증분한다.

AutoMode가 사용자 데이터를 프로그래밍하게 하고 잠그게 하려면 다음 명령어를 전송한다.

```
AutoAction = ProgramAndLockUser
```

이 "work" 단계에서 태그를 읽는 대신(자세한 정보는 2장에서 AutoMode 상태 도표 참조), 리더는 ProgUserData 값으로 지정한 태그를 새 사용자 데이터로 프로그래밍하고 잠근다. ProgG2AccessPwd는ProgG2LockType=Lock일 때에만 태그에 작성한다. PermaLock과 PermaUnlock 잠금 유형의 경우에는 ProgG2AccessPwd를 작성하지 않는다(잠금이 영구적인 본질 때문에 접속 패스워드가 필요하지 않다).

ProgramAndLockUser 이벤트가 성공하면 AutoMode 평가 단계가 진행되어 `true`(참)인지를 평가한다. ProgramAndLockUser 이벤트가 실패하면 `false`(거짓)으로 평가한다.

리더는 ProgUserDataInc 설정값에 따라서 다음에 프로그래밍할 사용자 데이터를 증분하기로 선택할 수 있다.

AutoAction = ProgramAlienImage

F800 **| 9900 |** 9680 **|** 9650

ProgramAlienImage의 AutoAction을 AutoMode에서 실행할 때, 리더는 AutoMode에서 완전히 새로운 이미지로 Alien Class 1/Gen 2태그를 자동 프로그래밍할 수 있으며, 이 동작은 무한정 반복하면서 매번 프로그래밍을 위해 EPC(및 관련된 사용자 데이터)를 증분한다. 이렇게 하면 AutoMode에 있는 동안 프로그래밍된 태그 그룹에 순서대로 태그 ID가 배정된다.

AutoMode가 새 이미지로 Alien G2 태그를 프로그래밍하게 하려면 다음 명령어를 전송한다.

```
AutoAction = ProgramAlienImage
```

이 "work" 단계에서 태그를 읽는 대신(자세한 정보는 2장에서 AutoMode 상태 도표 참조), 리더는 ProgEPCData 값으로 지정한 필드 내 태그를 완전히 새로운 메모리 이미지로 프로그래밍 하며, 이는 ProgAlienImageMap, ProgEPCData, ProgG2KillPwd, ProgG2AccessPwd, ProgUserData, and ProgAlienImageNSI 값으로 지정되어 있다. 이처럼 선택한 ProgAlienImageMap과 관계된 속성들만 사용한다.

프로그래밍 이벤트가 성공하면 AutoMode 평가 단계가 진행되어 `true`(참)인지를 평가한다. 프로그래밍 이벤트가 실패하면 `false`(거짓)으로 평가한다.

리더는 ProgEPCDataInc 설정값에 따라서 다음에 프로그래밍할 EPC를 증분하기로 선택할 수 있다. 선택한 ProgAlienImageMap에 사용자 데이터가 포함되어 있으면, 리더는 ProgUserDataInc 설정에 따라 ProgUserData 값을 증분할 수도 있다.

AutoAction = Erase

F800 | 9900 | 9680 | 9650

'Erase'의 AutoAction으로 AutoMode를 실행할 때, 리더는 필드에 있는 태그를 자동으로 지울 수 있으며, 이 동작은 무한정 반복할 수 있다.

AutoMode가 태그를 지우게 하려면 다음 명령어를 전송한다.

```
AutoAction = Erase
```

이 "work" 단계에서 태그를 수집하는 대신(자세한 정보는 2장에서 AutoMode 상태 도표 참조), 리더는 필드 내 태그를 지울 수 있다.

Erase 이벤트가 성공하면 AutoMode 평가 단계가 진행되어 `true`(참)인지를 평가한다. Erase 이벤트가 실패하면 `false`(거 짓)으로 평가한다.

Command	>NotifyNow
Response	Issuing Notify Trigger...

CHAPTER 6
LLRP

LLRP(Low Level Reader Protocol)는 RFID 리더와 제어 소프트웨어 또는 하드웨어 간에 표준 네트워크 인터페이스를 위한 사양이다. LLRP는 2007년 4월에 EPCglobal가 제작 및 인가하였으며 최근 버전은 1.1이다. LLRP의 목표는 최종 사용자를 위해 애플리케이션 개발을 단순화하여 각 RFID 공급자가 제공하고 소유권이 있는 여러 통신 프로토콜을 따라야 할 필요를 없애는 것이다. 여러 다른 소프트웨어 "어댑터"를 사용할 필요가 더 이상 없어지는 것이다. 이렇게 하면 이론적으로 개별 RFID 솔루션을 개발하는데 소비할 시간을 줄일 수 있으며, 한 두 RFID 공급자에게만 의지해야 하는 위험성도 없애준다.

LLRP는 ALR-9900/+ 리더에서만 지원된다. 현재로서는 ALR-F800, ALR-9680, ALR-9650 리더에서는 실행되지 않는다.

LLRP and ARP Coexistence

LLRP 프로토콜에는 클라이언트와 리더 두 종점이 존재한다. 클라이언트는 리더와의 연결을 시작하고 “specs”을 전달하며, 리더가 구성되어야 할 방식의 세부사항과 에어 프로토콜 별 속성, 어떤 안테나를 언제 사용할 것인지, RFID 데이터를 클라이언트에 어떻게 보고할지 등을 지정한다. LLRP는 또한 개별 리더 공급자가 개발한 "사용자 정의" specs도 고려하며, 이로써 소유권이 있는 특정 리더 모델에서 작업하는 것도 가능하다. 안타깝게도 이 방식은 통합형 표준의 "한번 쓰면 어디서나 실행"한다는 약속과는 거리가 있다.

LLRP 서비스를 실행 중일 때, 리더는 외부로부터의 LLRP 연결을 위해 LLRP TCP 포트(5084)를 청취할 것이나, 다른 통신 채널인 포트 23, 포트 2300(내부), 직렬 인터페이스에서는 ARP(Alien Reader Protocol) 연결을 계속 청취하기도 한다. LLRP가 실행 중일 때, 리더는 ARP 채널 상에서 전송한 Alien 명령어를 승인하나, 라디오 서브시스템과 직접 관련된 Alien 명령어는 LLRP 서비스와의 충돌을 피하기 위해 차단될 것이다. Alien 펌웨어는 동시에 ARP와 LLRP 인터페이스를 모두 지원하기에 충분할 만큼 유동적이므로, 사용자는 이를테면 서비스 시작 및 정지, 네트워크 설정 변경, 펌웨어 업그레이드 등을 위해 ARP 상에서 리더를 제어할 수 있다.

LLRP 서비스를 중단한 후에는 리더를 재부팅하고 FactorySettings 명령을 수행할 것을 권한다.

Controlling the LLRP Service

Alien 리더 프로토콜의 "서비스" 명령을 사용하여 리더 상에서 LLRP 서비스를 시작, 정지, 활성화, 비활성화한다. 리더는 한번에 하나씩 LLRP 연결을 승인하지만, 이후 연결 시도가 있을 때는 리더가 기존 연결을 테스트하여 아직 유효한지 확인하고, 유효하다면 새 연결은 거부한다. 현재 연결이 좋지 않다면, 오래된 연결은 해제하고 새 연결을 승인한다.

Example:

```
// Check the status (stopped, autostart disabled, startup priority 15)
Alien>service llrp status
s d 15 llrp

// Start the LLRP service
Alien>service llrp start
R d 15 llrp

// While LLRP is running, can't read tags and do some other functions here
Alien>t
(No Tags)

Alien>BlinkLED = 0 255 100 5
Error 516: (Driver error) Some ARP commands are not available while LLRP is active.

// Force LLRP to disconnect any connected client
Alien>service llrp close
service llrp RUNNING

// Stop the LLRP service, fully reenabling ARP control of the reader
Alien>service llrp stop
s d 15 llrp

// Cause LLRP to automatically start when the reader boots
Alien>service llrp enable
s A 15 llrp

// Prevent LLRP from automatically starting when the reader boots
Alien>service llrp disable
s d 15 llrp
```

LLRP References

EPCglobal: LLRP Specification, v1.0.1
This is the first release of LLRP.
http://www.epcglobalinc.org/standards/llrp/llrp_1_0_1-standard-20070813.pdf

EPCglobal: LLRP Specification, v1.1.0
This is the newer release of LLRP, and supported by Alien LLRP.
http://www.epcglobalinc.org/standards/llrp/llrp_1_1-standard-20101013.pdf

EPCglobal: UHF Class 1, Gen 2 (C1G2) Standard
This is the C1G2 standard. You should be familiar with this before attempting LLRP.
http://www.epcglobalinc.org/standards/uhfc1g2/uhfc1g2_1_2_0-standard-20080511.pdf

LLRP Toolkit (LTK) Software Development Kit
LLRP Toolkit is a set of free SDKs in C, C++, .Net, Java, and Perl.
http://www.llrp.org/

LLRP Version 1.0.1 vs. Version 1.1.0

오늘날 사용 중인 대부분의 LLRP 실행 예는 이전의 v1.0 프로토콜을 실행한다. Alien 펌웨어는 더 최근의 LLRP v1.1 프로토콜을 지원한다.

LLRP v1.1은 새로운 옵션 기능을 추가했으나, 다른 대부분의 변경사항들은 특정 specs 내에 추가적인 매개변수가 사용되며, 필드값의 여러 목록에도 확장된 값을 적용한다. 후향 호환성을 위해 리더는 v1.0 모드로 모든 새로운 연결에 응답한다. 일단 클라이언트가 프로토콜 버전을 v1.1로 변경하면, 리더는 연결 기간 동안 v1.1 프로토콜로 작동한다. 프로토콜 버전은 연결 당 1회만 설정할 수 있다.

VERSION NEGOTIATION

LLRP v1.1은 다음 네 개의 메시지를 도입한다.

1. GET_SUPPORTED_VERSION
2. GET_SUPPORTED_VERSION_RESPONSE: 현재 설정되어 있는 프로토콜 버전과 리더가 지원하는 최상 버전을 반환한다.
3. SET_PROTOCOL_VERSION: 이후로는 사용하기 원하는 LLRP 프로토콜 버전을 포함한다(연결 당 한 번만 전송한다).
4. SET_PROTOCOL_VERSION_RESPONSE

NEW C1G2 FEATURES

일부 새로운 C1G2 v1.2.0 기능들은 LLRP v1.1에서 지원되며, 여기에는 BlockPermalock과 태그 리커미셔닝이 포함된다. 이들을 지원할 것인지는 GET_READER_CAPABILITIES_RESPONSE 메시지에 있는 C1G2LLRPCapabilities 매개변수에 표시된다.

NEW COMMUNICATION STANDARDS

새로운 지역 통신 표준들을 현재 프로토콜에서 열거하고 있다.

ENHANCED REPORT TRIGGERS

현재 v1.1에 새롭게 추가된 다음 두 가지 시간 기반 보고 트리거가 있다.

5. Upon N seconds or end of AISpec
6. Upon N seconds or end of of ROSpec

LOOPSPECS

LLRP v1.1은 ROSpec 내에서 안테나 인벤토리의 실행을 위한 수단을 정의한다.

Alien LLRP Capabilities and Optional Features

LLRP 프로토콜의 여러 특징은 공급자가 정의 또는 지정하게 되어 있다. 다음 표는 Alien LLRP 실행 시 지원하는 여러 기능과 옵션들이다.

LLRP Feature or Capability	ALR-9900
General Device Capabilities	
MaxNumberOfAntennaSupported	4
CanSetAntennaProperties	-
HasUTCClockCapability	Y
# ReceiveSensitivityTableEntries	1
GPIOCapabilities	4 In, 8 Out
PerAntennaAirProtocols	Class1Gen2
LLRP Capabilities	
CanDoRFSurvey	-
CanReportBufferFillWarning	Y
SupportsClientRequestOpSpec	-
CanDoTagInventoryStateAwareSingulation	Y
SupportsEventAndReportHolding	Y
MaxPriorityLevelSupported	7
ClientRequestOpSpecTimeout	-
MaxNumROSpecs	64
MaxNumSpecsPerROSpec	16
MaxNumInventoryParameterSpecsPerAISpec	1
MaxNumAccessSpecs	256
MaxNumOpSpecsPerAccessSpec	8
Regulatory Capabilities	

# Transmit Power Levels	151
Min Transmit Power	16.6
Max Transmit Power	31.6
# RF Modes	5
C1G2 LLRP Capabilities	
CanSupportBlockErase	Y
CanSupportBlockWrite	Y
MaxNumSelectFiltersPerQuery	8
Custom Capabilities	
Separate Transmit Power when Writing	Y
Higgs Dynamic Authentication	Y
Hide User Blocks	Y

Alien LLRP RF Modes

LLRP 인터페이스에서 제공하는 RF 모드는 ARP 인터페이스에서 보는 것과 동일한 RF 모드를 반영한다. 다음 표는 이러한 모드와 그 세부정보를 보여준다.

Alien RF Modes	25M4 (DRM)	25FM0 (STD)	06FM0 (HS)	12M4	06M4
Mode Table Index (US-FCC)	**0**	**1**	**2**	**3**	**4**
Mode Identifier	**108**	**105**	**107**	**109**	**110**
DR Value	DRV_64_3	DRV_64_3	DRV_8	DRV_64_3	DRV_8
EPC HAG TC Conformance	0	0	0	0	0
M Value	MV_4	MV_FM0	MV_FM0	MV_4	MV_4
Forward Link Modulation	PR_ASK	PR_ASK	PR_ASK	PR_ASK	PR_ASK
Spectral Mask Indicator	DI	DI	MI	MI	MI
BDR Value	250000	250000	250000	250000	250000
PIE Value	1750	1750	1750	1750	1750
Min Tari Value	25000	25000	6250	12500	6250
Max Tari Value	25000	25000	6250	12500	6250
Step Tari Value	0	0	0	0	0

안테나에서 RF 모드를 설정할 때, LLRP 사양은 '0'에 기반한 모드표 인덱스 또는 공급자가 정의한 모드 식별자를 사용자가 제시할지에 있어 다소 모호한 면이 있으며, 공급자 실행 내용도 이 면에서는 차이가 있다. Alien는 이러한 이중적인 해석을 모두 수용한다. 사용자는 모드표 인덱스나 모드 식별자를 사용해 모드를 설정할 수 있다.

예를 들어, DRM 모드를 설정하려면 “1” (모드표 인덱스) 또는 “108” (모드 식별자) 중 무엇이든 사용할 수 있다. Alien 모드 식별자 값은 위 표 에 명시하였다.

유의사항: RF 모드표 항목들은 리더의 지역에 따라 다르므로, 동일한 모드표인덱스 값이 지역 별로 다른 모드를 나타낼 수도 있다. 이용 가능한 모드와 인덱스 범위를 결정하는 가장 좋은 방법은 리더 기능 정보를 읽어서 전체 RF 모드표를 가져오기 하는 것이다.
상기 표에 있는 모드표 인덱스값들은 US-FCC에서 구성한 리더에 해당된다.

Alien LLRP Extensions

LLRP는 프로토콜 전체에 걸쳐 다수의 "사용자정의 확장 지점"을 제공하며, 이에 공급자들은 리더와 주고 받는 메시지에서 사용자정의 매개변수를 삽입하도록 허용된다. 또한 LLRP에 지정되지 않은 기능도 제공하며 리더와 태그의 고유 기능들도 전시할 수 있다. 어느 정도는 이러한 확장성이 표준화된 프로토콜을 유지하려는 기본 목적과는 위배되긴 하지만, Alien은 프로토콜 내 새로운 공급자 별(및 리더 별) 옵션들이 RFID 사용자들 간에 넘쳐나게 하기 보다는 핵심 프로토콜을 실행하는데 최대한 중점을 두었다(여러 ROSpecs & 우선순위, TagStateAware 인벤토리, 마스킹, 필터, AISpec 루핑의 전적인 지원).

따라서 핵심 표준에서 멀어지더라도 그만한 가치가 있다고 Alien가 느끼는 몇 가지 기능과 옵션은 있을 수 있다. Alien LLRP 서비스는 LLRP 확장 매개변수를 통해 다음 세 가지 기능을 지원한다.

Alien 기능	설명
동적 인증	Higgs3 & Higgs4 태그의 인증을 위해 추가적인 인증 단계들을 수행한다. 인증 결과와 모든 태그의 태그 제조자 정보를 보고한다.
쓰기 파워 제어	태그로 쓰기 작업을 수행할 때 사용할 전달 동력을 각기 다르게 지정할 수 있다.
사용자 메모리 읽기 잠금	Higgs3 & Higgs4 내 사용자 메모리의 개별 블록들을 읽지 못하 게 감출 수 있다.

Alien의 고유 제조자 ID는 17996이다. 이 값과 각각의 사용자 정의 매개변수 "유형" 값은 사용자의 LLRP 메시지 속에 사용자 정의 매개변수들을 구성하여 삽입할 때 필요하다.

동적 인증

Alien Higgs3 & Higgs4 태그에는 리더가 인증 시 확인할 수 있게 해주는 기능이 있다. 리더는 추가적인 인증 점검을 (Higgs3/4 태그에) 수행하고 각 태그의 제조자와 태그 모델을 보고한다. `ROReportSpec`에 있는 사용자 정의 매개변수를 통해 정보를 요청함으로써 Alien LLRP 내 이러한 기능을 사용할 수 있으며, 이때 `TagReportContentSelector`의 나머지를 구성한다. 외부 정보는 리더가 생성한 `TagReportData` 에서 사용자 정의 매개변수로 나타난다.

ALIENTAGREPORTCONTENTSELECTOR PARAMETER

`AlienTagReportContentSelector` 를 사용하면 사용자는 `TagReportData`에 포함시킬 Alien만의 데이터 필드를 지정할 수 있다. 이를 `ROReportSpec` 의 `TagReportContentSelector` 의 연장이라고 생각하면 된다. `enableDynamicAuthentication` 필드는 동적 인증 기능을 제어하며, 태그 제조자 정보를 `TagReportData` 에 포함시킬지 여부도 제어한다.

`AlienTagReportContentSelector` 는 `ROReportSpec`의 사용자 정의 확장 지점에 첨부한다.

AlienTagReportContentSelector (paramType #218)

EnableAlienDynamicAuthentication: Boolean

ALIENTAGREPORTDYNAMICAUTHENTICATION PARAMETER

리더가 태그 인증 또는 제조자 정보를 수집하면 이를 사용자정의 매개변수로 해당 태그의 다른 데이터와 함께 `TagReportData` 매개변수에 첨부한다.

AlienTagReportDynamicAuthentication (paramType #219)

TID: 서명하지 않은 정수(4바이트) - 태그 TID의 처음 두 단어; EPCglobal 태그인지 여부와 태그의 제조자 및 모델 번호를 표시한다.

AuthenticateResult: 서명하지 않은 문자(1바이트)
Possible Values:

Value	Definition
0	Authentication Succeeded
1	Authentication Failed (possible fake!)
2	Alien Tag, but authentication not supported
3	Non-Alien Tag
4	Authentication Operation Failed (e.g. tag lost)

쓰기 파워 제어

태그 읽기와는 다르게 태그를 작성할 때 다른 전송 동력을 사용할 수 있으므로, 태그 프로그래밍 지역을 한정지어야 할 때 유용하다. 태그 식별 시 낮은 전송 동력을 사용할 수 있고, 읽기 영역을 작게 유지할 수 있으며, 태그에 입력할 때 더 높은 동력으로 바꾸어 쓰기 작업의 성공 가능성을 확보한다.

ALIENWRITEPOWERCONTROL PARAMETER

리더는 태그 상에서 쓰기, 잠금, Kill 작업을 수행할 때 `AlienWritePowerControl` 매개변수가 제시한 동력 수준으로 자동 전환한다. 사용자가 지정한 값은 RFTransmitter 매개변수에 있는 `TransmitPower` 필드에서처럼 리더의 전송 동력 수준표에 대한 인덱스값이다.

`The AlienWritePowerControl` 매개변수는 `C1G2AirProtocolInventoryCommand`의 사용자 정의 확장 지점에 첨부된다.

AlienWritePowerControl (paramType #100)

WriteTransmitPower: 서명하지 않은 짧은 정수(2바이트) - 전송동력표 인덱스값

사용자 읽기 잠금

Align Higgs3 & Higgs4 태그를 사용하면 사용자 메모리의 개별 블록을 정확한 접속 패스워드가 없는 타인에게서 감출 수 있다. 이 "read locks(읽기 잠금)"으로 태그 내에 민감한 데이터를 저장할 수 있으며, 믿을 만한 동료들만 읽거나 수정할 수 있게 된다. 읽기 잠금을 위해 Higgs3와 Higgs4 태그는 사용자 메모리를 크기가 다른 블록들로 분할한다. Higgs3는 네 단어 길이(총 64비트)의 블록을 사용하고, Higgs4는 두 단어(32비트) 길이 블록을 사용한다. 사용자 뱅크에서 읽기 잠금을 설정할 때, 동시에 모든 블록에 대해 잠금을 설정할 수 있다. 다른 블록들의 상태를 재지정하지 않고 하나의 블록 상태만 변경할 수는 없다.

ALIENHIDEBLOCKS PARAMETER

`AlienHideBlocks` 매개변수는 프로토콜 내 다른 `C1G2OpSpec`과 마찬가지로 작동하며, 특정 `AccessSpec`에 첨부된다. `AccessSpec`의 `C1G2TagSpec` 매개변수에 부합하는 태그가 있을 때, `AccessSpec` 내에 정의된 모든 `OpSpecs` 가 실행되며, 여기에는 `AlienHideBlocks` 매개변수도 포함된다. `AlienHideBlocks` 매개변수에는 `OpSpecID` 필드가 포함되어 있어서 사용자는 생성된 `TagReportData` 내에 동작 결과가 어떠한지를 확인할 수 있다. 또한 메모리 뱅크가 동작할 필드(뱅크 3만 지원됨), 필요한 접속 패스워드, 잠금 상태의 바이트도 포함된다.

Higgs3/4 태그 사용자 메모리 내에 있는 각 블록은 순서대로 번호가 부여되며, 잠금 상태 설정에 사용하는 ReadLockMask 내에 그에 해당하는 비트가 있다. 첫 블록은 마스크 내 첫(최상위 순위) 비트와 상응하며, 다음 순위 비트에는 두 번째 블록이 뒤따르는 방식이다. ReadLockMask가 8비트 길이이므로, 사용자 메모리의 첫 블록만 감추기 원한다면 0x80(2진수: 10000000)를 0x01 (2진수: 00000001) **대신** 사용하면 된다.

`AlienHideBlocks` 사용자정의 매개변수가 `AccessCommand`의 사용자정의 확장 지점(LLRP v1.0 또는 1.1)에 첨부되거나, `AccessCommand`의 `OpSpec` 목록에 직접 첨부된다(LLRP v1.1만 해당).

AlienHideBlocks (paramType #300)

OpSpecId: 서명하지 않은 짧은 정수(2바이트)

Bank: 서명하지 않은 문자(1바이트) - 현재 뱅크 3만 지원된다.

AccessPassword: 서명하지 않은 정수(4바이트)

ReadLockMask: 서명하지 않은 문자(1바이트)

ALIENHIDEBLOCKSRESULT PARAMETER

`AccessSpec` 이 `AlienHideBlocks` 매개변수를 포함하여 실행 중일 때, 읽기 잠금 작업 결과는 인벤토리 중 태그에서 얻는 다른 데이터와 함께 태그의 `TagReportData`, 사용자정의 매개변수 내에 위치하며, 수집된 다른 `OpSpecResult`도 마찬가지이다.

The `AlienHideBlocksResult` 매개변수는 `TagReportData`의 사용자 정의 확장 지점에 첨부된다.

AlienHideBlocksResult (paramType #301)

OpSpecId: 서명하지 않은 짧은 정수(2바이트)

Result: 서명하지 않은 문자(1바이트)
가능한 값:

Value	Definition
0	Succeeded.
1	Insufficient power at the tag.
2	Non-specific tag error.
3	No response from tag.
4	Non-specific reader error.
5	Incorrect password.
6	Tag memory overrun.
7	Tag memory locked.

부록 A
XML 데이터 구조용 DTD

리더는 세 가지 유형의 XML형식 문서를 생성할 수 있다. 이 부록은 이 XML 문서 각각에 대한 문서유형정의(DTD)를 제시한다.

Heartbeat DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Alien-RFID-Reader-Heartbeat [
  <!ELEMENT Alien-RFID-Reader-Heartbeat (ReaderName, ReaderType,
            IPAddress, CommandPort, HeartbeatTime, MACAddress?, ReaderVersion?)>
    <!ELEMENT ReaderName    (#PCDATA)>
    <!ELEMENT ReaderType    (#PCDATA)>
    <!ELEMENT IPAddress     (#PCDATA)>
    <!ELEMENT CommandPort   (#PCDATA)>
    <!ELEMENT HeartbeatTime (#PCDATA)>
    <!ELEMENT MACAddress    (#PCDATA)>
    <!ELEMENT ReaderVersion (#PCDATA)>
]>
```

TagList DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Alien-RFID-Tag_List [
  <!ELEMENT Alien-RFID-Tag_List (Alien-RFID-Tag*)>
    <!ELEMENT Alien-RFID-Tag (TagID, DiscoveryTime, LastSeenTime,
              Antenna, ReadCount, Protocol?, D1?, D2?, D3?, D4?)>
      <!ELEMENT TagID         (#PCDATA)>
      <!ELEMENT DiscoveryTime (#PCDATA)>
      <!ELEMENT LastSeenTime  (#PCDATA)>
      <!ELEMENT Antenna       (#PCDATA)>
      <!ELEMENT ReadCount     (#PCDATA)>
      <!ELEMENT Protocol      (#PCDATA)>
      <!ELEMENT D1            (#PCDATA)>
      <!ELEMENT D2            (#PCDATA)>
      <!ELEMENT D3            (#PCDATA)>
      <!ELEMENT D4            (#PCDATA)>
]>
```

Notification DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Alien-RFID-Reader-Auto-Notification [
  <!ELEMENT Alien-RFID-Reader-Auto-Notification (ReaderName, ReaderType,
            IPAddress, CommandPort, Time, Reason, StartTriggerLines?,
            StopTriggerLines?, Alien-RFID-Tag-List)>
    <!ELEMENT ReaderName        (#PCDATA)>
    <!ELEMENT ReaderType        (#PCDATA)>
    <!ELEMENT IPAddress         (#PCDATA)>
    <!ELEMENT CommandPort       (#PCDATA)>
    <!ELEMENT MACAddress        (#PCDATA)>
    <!ELEMENT Time              (#PCDATA)>
    <!ELEMENT Reason            (#PCDATA)>
    <!ELEMENT StartTriggerLines (#PCDATA)>
    <!ELEMENT StopTriggerLines  (#PCDATA)>
]>
```

부록 B
리더 펌웨어 업그레이드 (GUI)

Alien Technology는 정기적으로 리더 내에 실행되는 펌웨어 업그레이드와 버그 수정을 제공한다.
이러한 업그레이드는 펌웨어 업그레이드 파일로 배포되며, 리더 웹 인터페이스를 통해서나 Gateway Demonstration 소프트웨어를 사용해 리더에 업로드 된다.

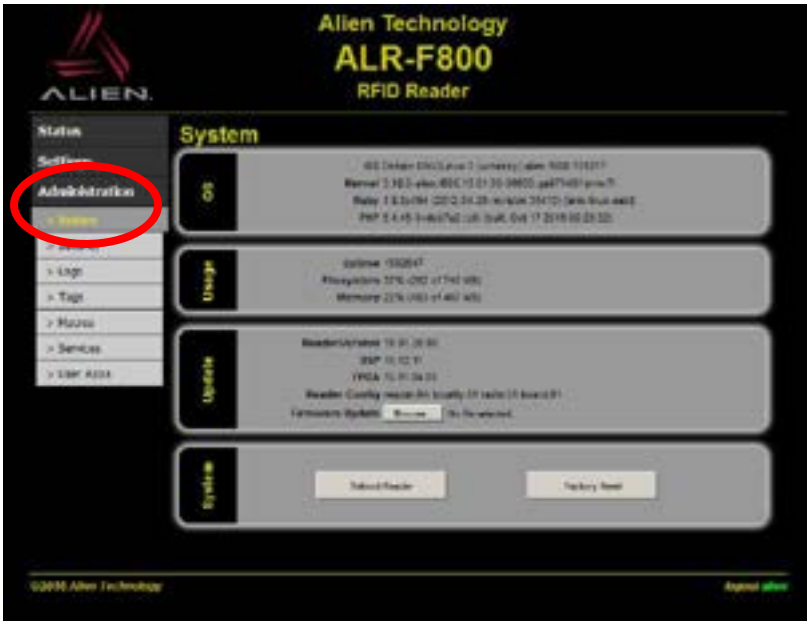
펌웨어 업그레이드를 수행하면 일반적으로 리더의 구성을 기본 상태로 재설정하므로 업그레이드 전에 리더 상태를 기록해 두었다가 리더가 업그레이드 전 상태로 돌아가도록 필요한 명령을 리더에 전송해야 한다. DHCP, IP 주소 등의 네트워크 설정은 영향을 받지 않는다.

리더는 Gateway Demonstration 소프트웨어를 사용해 네트워크 상에서 업그레이드할 수 있다. 현재 리더는 직렬 인터페이스를 통해서는 업그레이드를 지원하지 않는다.

리더는 내장형 웹 인터페이스를 통해서도 업그레이드할 수 있다. 이 과정은 모든 리더에서 동일하나 그래픽 인터페이스는 다르게 보일 수 있다.

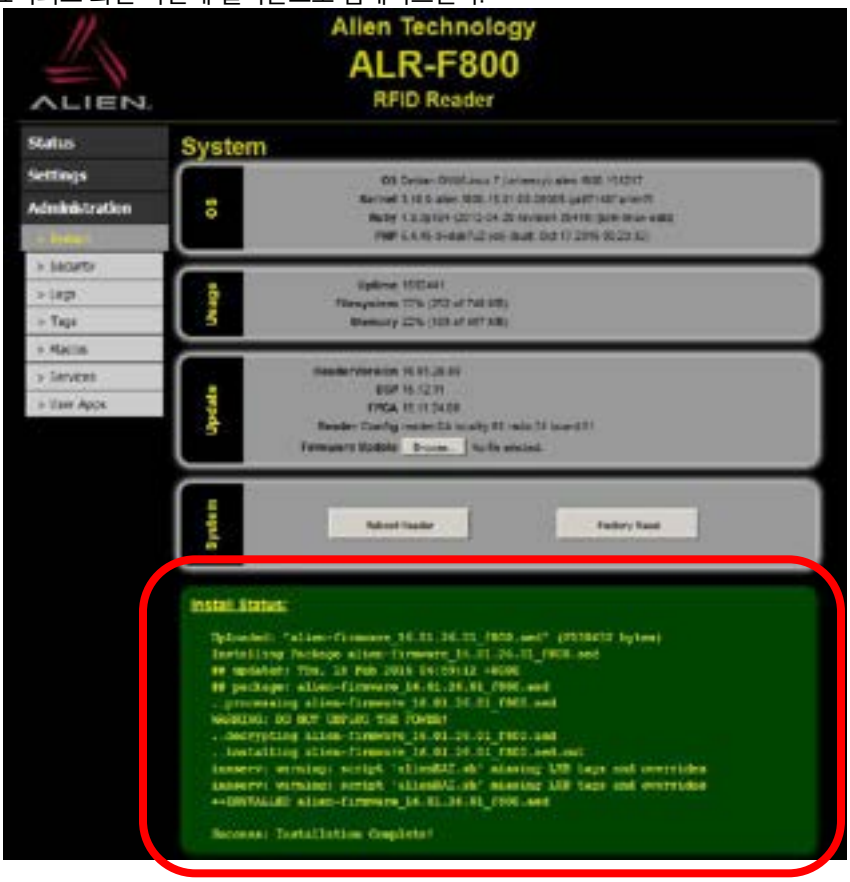
ALR-F800 Web Interface

리더의 웹 서버를 검색하여 Admin → System 페이지를 찾는다. 이때 사용자 자신의 TCP 로그인 사용자명과 패스워드를 입력하여(기본값은 "alien" / "password") admin 페이지에 접속한다.



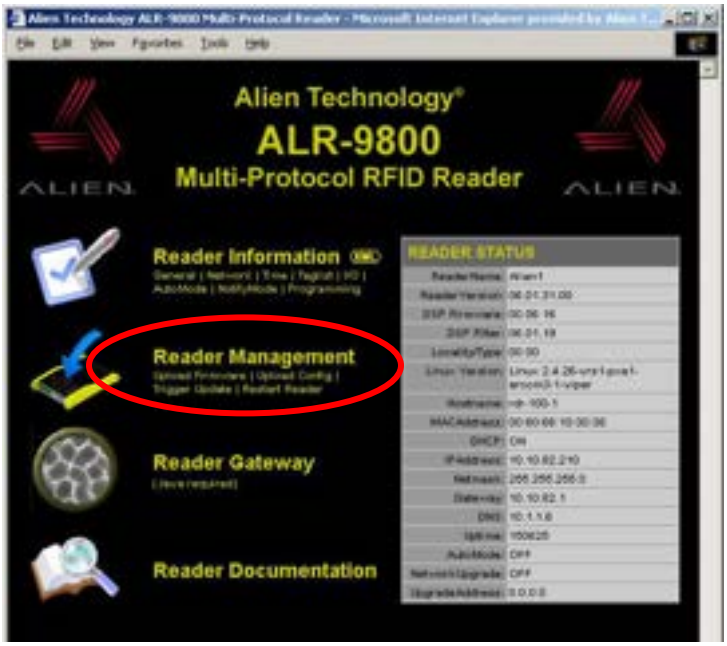
'Browse…' 버튼을 클릭하고 업로드할 펌웨어 파일을 선택한 다음(또는 리더 매크로) '설치' 버튼을 클릭한다.

설치 진행상황이 표시되고 화면 하단에 실시간으로 업데이트된다.

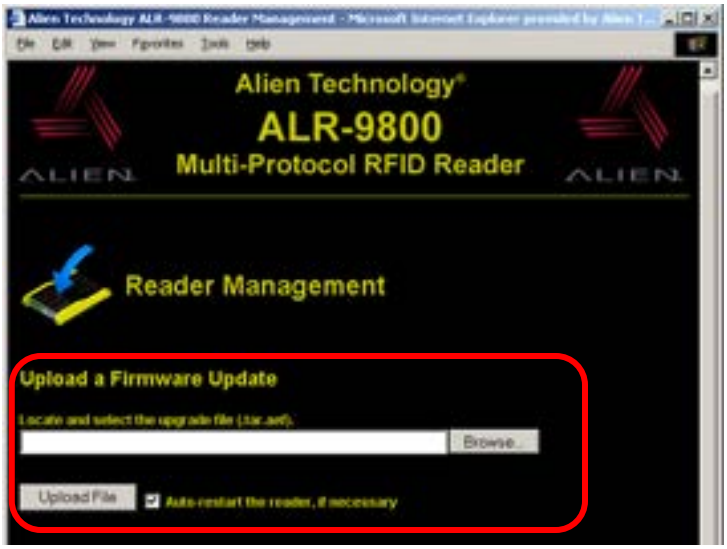


ALR-9900/9680/9650 Web Interface

리더의 웹 서버를 검색하여 리더 관리 페이지를 찾는다. 이때 사용자 자신의 TCP 로그인 사용자명과 패스워드를 입력하여(기본값은 "alien" / "password") 리더 관리 페이지에 접속한다.



페이지에 있는 양식을 사용하여 업그레이드 파일을 선택하고 양식을 제출한다.



Firmware Updates 가져오기

리더는 또한 커맨드 라인을 통해서 업데이트를 가져오도록 구성하거나 시작되게 할 수도 있다. 다음 부록에서 "업그레이드 풀링(Pulling Upgrades)" 제하의 섹션을 참조할 것.

부록 C
리더 펌웨어 업그레이드 (프로그램 방식)

이 부록은 표준형 HTTP POST 프로토콜을 사용하여 리더에 업그레이드를 보내는 방법과
FTP 또는 HTTP 서버로부터 업그레이드를 자동 풀링하도록 리더를 구성하는 방법에 대해 기술한다.

업그레이드 파일 구조

고객에게 제공되는 업그레이드 파일은 일반적으로 암호화되어 있다. 파일명은 다음 형식과 같다.

ALRx800_fw_050921.tar.aef

첫 구획에 "ALRx800"은 대상 리더 플랫폼을 가리킨다. 전체 모델 라인에 적용되더라도 "ALRx800"으로 표시될 수 있는데, 이는 현재 리더가 모두 ALR-9800 플랫폼에 기초해 있으며 동일한 펌웨어 구조를 사용하기 때문이다.

파일명의 두 번째 구획인 "fw"는 업그레이드하는 리더의 구성요소를 가리킨다("펌웨어").

세 번째 구획은 업그레이드 날짜로, YYMMDD의 형식이다. 일반적으로 ReaderVersion 명령어가 제시하는 YY.MM.DD 값에 부합한다.

마지막으로, 두 확장자는 패키지가 tarball (.tar)임을 보여주며, Alien의 고유 암호화 알고리즘(.aef = Alien Encrypted File)을 사용해 암호화하였음을 가리킨다.

업그레이드 푸싱

부록 B에서 이미 리더의 웹 인터페이스를 사용해 리더에 업그레이드를 보내는 방법을 설명하였다. 동일한 인터페이스로 표준형 HTTP POST 메커니즘으로 프로그래머가 사용할 수도 있다. 리더 펌웨어 업그레이드와 리더 매크로 파일(.arm)은 모두 이 수단을 사용해 리더로 업로드할 수 있다.

HTTP Post 요청

리더로 파일을 업로드하려면 다음 URL에서 리더의 웹 서버로 POST 요청을 전송하기만 하면 된다:

http://<reader's IP>/cgi-bin/upgrade.cgi

요청을 보낼 때, 다음 요건을 충족해야 한다. 이 요건들을 실제로 충족하는지 여부에 따라 실행 상태가 달라진다. Java로 작성한 예제 프로그램을 참고하도록 포함하였다.

- 요청 방법은 반드시 POST를 통해야 한다.
- 요청의 콘텐츠 형식은 "multipart/form-data"이어야 한다.
- 요청 인증은 "Basic <BASE64-encoded user:pass>"이어야 한다.
- 파일 첨부 시 콘텐츠 처리: "form-data"
- 파일 첨부 시 명칭: "uploadedFile"
- 파일 첨부 시 파일명: <localFilePath>
- 파일 첨부시 콘텐츠 유형: "text/plain"

HTTP Post 응답

파일을 업그레이드 CGI로 보낸 뒤에 리더는 일련의 종료 코드로 응답하며, 업그레이드 설치 후 리더의 최종 처리가 진행된다. 응답의 각 행은 하나의 줄바꿈 문자로 끝난다.

응답의 첫 행은 파일 업로드 결과를 표시한다. 파일이 수신되고 리더 파일시스템에 성공적으로 작성되면, "0" 문자를 반환하고, 그렇지 않으면 오류코드로 "0"이 아닌 문자를 반환한다. 이때 바이트 값이나 0x00, 0x01 등이 아닌, ASCII 문자라는 점에 유의해야 한다.

파일 수신 중 오류가 발생하면, 응답 2행 문자열이 문제의 특성을 기술하고, 리더 응답이 이렇게 마친다.

업로드가 성공하면 리더는 압축을 해제하고 업그레이드를 설치한다. 응답의 다음 행은 이 단계 결과를 표시한다. 반환된 결과 코드는 다음 중 하나이다.

0 – 업그레이드 성공; 재부팅 필요함
1-127 – 업그레이드 성공하지 못함.
-256 – 업그레이드 성공함; 재부팅 불필요.

합충해제나 설치가 실패하면, 리더 응답의 세 번째 행 문자열이 문제의 특성을 설명한다. 또는 리더가 다음 중 하나를 반환한다.

Done – 업그레이드 완료; 재부팅 불필요.
Rebooting – 업그레이드 완료; 리더가 재부팅함

마지막으로, 리더는 단일 null(0x00) 값으로 응답을 종료한다.

Sample Java Implementation

```
import java.net.*;
import java.io.*;

public class HTTPFilePost {
  private static final String READER_URL = "http://192.168.1.100/cgi-bin/upgrade.cgi";
  private static final String FILE_PATH = "C:\\9800upgrades\\ALR9800_fw_test.tar.aef";
  private static final String CONTENT_BOUNDARY = "pflxm";
  private static final String EOL = "\r\n";
  private static final String username = "alien";
  private static final String password = "password";

  public static final void main(String args[]) {
    FileInputStream fileInputStream = null;
    OutputStream outputToReader = null;
    BufferedReader inputFromReader = null;

    try {
      // Open the connection
      URL testPost = new URL(READER_URL);
      URLConnection conn = testPost.openConnection();

      // Deal with authentication
      String userpass = username + ":" + password;
      String encoding = new sun.misc.BASE64Encoder().encode(userpass.getBytes());
      conn.setRequestProperty("Authorization", "Basic " + encoding);
```

```
      // This makes it a POST request
      conn.setDoOutput(true);
      conn.setRequestProperty("Content-Type",
                              "multipart/form-data; boundary=" + CONTENT_BOUNDARY);
      outputToReader = conn.getOutputStream();

      // Setup the file and start the http request
      File uploadFile = new File(FILE_PATH);
      fileInputStream = new FileInputStream(uploadFile);
      String requestHeader = "--" + CONTENT_BOUNDARY + EOL
                           + "Content-Disposition: form-data; name=\"uploadedFile\";"
                           + " filename=\"" + FILE_PATH + "\"" + EOL
                           + "Content-Type: text/plain" + EOL + EOL;
      System.out.println("Request Header:\n" + requestHeader);
      outputToReader.write(requestHeader.getBytes());

      // Read from the upgrade file and write it to the reader
      byte[] fileBuffer = new byte[512];
      int totalBytesRead = 0;
      int numBytesRead = fileInputStream.read(fileBuffer, 0, 512);
      while (numBytesRead != -1) {
        totalBytesRead += numBytesRead;
        outputToReader.write(fileBuffer, 0, numBytesRead);
        numBytesRead = fileInputStream.read(fileBuffer, 0, 512);
      }
      System.out.println("Wrote " + totalBytesRead + " bytes from the upgrade file.\n");

      // Close the local upgrade file and finish the POST request
      fileInputStream.close();
      fileInputStream = null;
      String requestFooter = EOL + "--" + CONTENT_BOUNDARY + "--" + EOL;
      System.out.println("Request Footer:\n" + requestFooter);
      outputToReader.write(requestFooter.getBytes());

      // Construct the BufferedReader to get the reader's response
      inputFromReader = new BufferedReader(new InputStreamReader(conn.getInputStream()));

      // Read back the server's response
      String responseLine = inputFromReader.readLine();
      while (responseLine != null) {
        System.out.println("Reader Response: " + responseLine);
        responseLine = inputFromReader.readLine();
      }

      // Close everything
      inputFromReader.close();
      outputToReader.close();
      inputFromReader = null;
      outputToReader = null;
    } catch (Exception ex) {
      ex.printStackTrace();
    } finally {
      if (fileInputStream != null) {
        try { fileInputStream.close(); } catch (Exception ignore_it) {}
      }
      if (inputFromReader != null) {
        try { inputFromReader.close(); } catch (Exception ignore_it) {}
      }
```

```
      if (outputToReader != null) {
        try { outputToReader.close(); } catch (Exception ignore_it) {}
      }
    }
  }

} // End of class HTTPFilePost
```

Sample output from this program:

```
Request Header:
--pflxm

Content-Disposition: form-data; name="uploadedFile"; filename =
"C:\9800upgrades\ALR9800_fw_test.tar.aef"
Content-Type: text/plain

Wrote 39 bytes from the upgrade file.

Request Footer:
--pflxm--

Reader Response: 0
Reader Response: 0
Reader Response: Rebooting
Reader Response: [null]
```

Upgrades 가져오기

리더를 구성하여 업그레이드 호스트에 접속하고 자체적으로 업데이트를 다운로드하고 설치하게 할 수도 있다. 이때 요건은 리더가 업데이트를 검색할 수 있게 해줄 HTTP 또는 FTP 서버뿐이다. HTTPS(secure HTTP)도 지원된다.

앞서 기술한 "push" 모델과는 달리(호스트가 리더와의 연결을 시작하여 업그레이드 파일을 업로드함), "pull" 모델은 수동적인 호스트가 있어 리더가 업그레이드 파일에 접속하고 다운로드하게 한다.

리더 구성

업그레이드 풀링(pulling)을 촉진하도록 리더에 구성해야 할 두 가지 속성이 있다.

UPGRADEADDRESS

UpgradeAddress는 업그레이드 파일을 접속할 수 있는 디렉터리 URL을 지정한다. HTTP와 FTP 프로토콜 모두 지원하며, HTTP의 안전한 버전인 HTTPS도 지원한다. 여기서 지정할 것은 디렉터리이므로 URL은 모두 사선(/)으로 끝나야 한다. 예를 들면 다음과 같다.

```
Alien>UpgradeAddress = http://192.168.1.1/<path>/
UpgradeAddress = http://192.168.1.1/<path>/

Alien>UpgradeAddress = ftp://ftp.mydomain.com/<path>/
UpgradeAddress = ftp://ftp.mydomain.com/<path>/

Alien>UpgradeAddress = https://secureHTTPhost/<path>/
UpgradeAddress = https://secureHTTPhost/<path>/
```

SPECIFYING USERNAME AND PASSWORD

인증이 필요한 프로토콜의 경우, 사용자명과 패스워드는 UpgradeAddress URL 도메인 부분 앞에 붙이고, 패스워드와 사용자명은 콜론으로 분리한다. "@" 기호는 도메인에서 이들을 분리할 때 사용한다. 예를 들면 다음과 같다.

```
Alien>UpgradeAddress = ftp://user:password@ftpserver/<path>/
UpgradeAddress = ftp://user:password@ftpserver/<path>/
```

Upgrade 가져오기 활성화

업그레이드 풀링 메커니즘을 켜려면, NetworkUpgrade 속성을 ON으로 설정하기만 하면 된다.

```
Alien>NetworkUpgrade = On
NetworkUpgrade = On
```

이때 리더를 구성한 다음에는 "save" 명령어를 전송하여 플래시에 새로운 설정값을 작성해야 한다. 그렇지 않으면 리더가 재부팅할 때 이 설정값들이 이전 값으로 돌아가버린다.

Upgrade Host 설정하기

업그레이드 호스트는 리더의 UpgradeAddress 속성에 명시한 프로토콜에 해당하는 적절한 서비스를 실행해야 한다.

리더가 사용하기 원하는 업그레이드 파일을 UpgradeAddress가 표시하는 업그레이드 디렉터리 내에 둔다. 디렉터리 안에는 다른 파일도 있을 수 있다. 업그레이드 중에 리더가 풀링하는 특정한 .tar.aef 파일은 "control"이라는 파일명으로 열거해야 하며, 이러한 파일은 동일한 업그레이드 디렉터리 안에 있어야 한다.

NETWORKUPGRADE CONTROL FILE

업그레이드 디렉터리 안에 "control"이라는 이름의 파일을 생성한다. 서버가 확장자가 없는 파일들을 받기 꺼린다면, 파일명을 "control.txt"로 정할 수도 있다. 이 파일은 리더가 다운로드하는 특정 업그레이드 파일들의 파일명을 나열한다(보통은 업그레이드 용 파일이 하나뿐이다). 업그레이드 파일을 열거하는 행마다 하나의 줄바꿈 문자("\n" = 0x0A)로 종료해야 한다.

이 제어 파일을 사용하면 업그레이드 디렉터리 안에 리더 소프트웨어의 여러 버전을 구비할 수 있으며, 제어 파일 내에 항목을 바꾸기만 하면 어떤 버전을 활성화할지를 제어할 수 있다. 리더가 항상 접속 가능한 파일을 가리키는지 확인하고, 새 버전이 이용 가능할 때는 네트워크 호스트를 한번 변경하기만 하면 구성되어 있는 리더들에서 이 새 파일에 접속할 수가 있다. 리더는 새 업그레이드의 특정 이름을 알 필요가 없으며, 그저 제어 파일이 있는 곳만 알면 된다.

제어 파일들은 또한 리더 매크로를 가리킬 수도 있다(.arm 파일). 이 파일들이 리더 안에 다운로드 및 설치되어 같은 이름의 기존 매크로를 대신한다.

제어 파일에 참조한 펌웨어 파일은 모두 제어 파일과 동일한 디렉터리 안에 있어야 한다. 다른 디렉터리들, 이를테면 "../ALR-x800_fw_070131.tar.aef" 또는 "/alienFirmware/ALR- x800_fw_070131.tar.aef"등을 지정하는 것은 현재는 지원되지 않는다.

작동 원리

리더는 서버에서 업그레이드파일 풀링 스크립트를 실행하여 설치한다. 리더가 부팅하면 자동으로 스크립트를 실행하나, UpgradeNow 명령어를 사용하거나 웹 인터페이스를 통해 시작할 수도 있다.

업데이트 스크립트는 우선 NetworkUpgrade 속성이 "ON"으로 설정되어 있는지 확인한다. 그렇지 않다면 스크립트가 있어서 Alien 리더 서브시스템이 일반적인 방식으로 시동한다.

NetworkUpgrade 플래그가 "On"이면, 리더는 UpgradeAddress 속성으로 지정된 URL에 위치한 제어 파일을 다운로드 하려고 할 것이다. 제어 파일이 없다면, 스크립트가 있어서 Alien 리더 서브시스템이 일반적인 방식으로 시동한다.

제어 파일이 있으면, 스크립트는 파일 속에서 파일명을 찾고 파일명에서 버전 정보를 확인한다. 예를 들어, 일반적인 업그레이드 파일명 "ALR9800_fw_060217.tar.aef"은 리더버전 06.02.17에 해당한다. 이 스크립트는 버전을 리더에 설치되어 있는 것과 비교하고, 만약 다르면, 업그레이드를 다운로드하여 압축해제하고 설치한다.

이 방식으로 다운그레이드도 가능하다. 업그레이드 버전이 설치된 것과 다르면, 업그레이드가(또는 다운그레이드가) 진행된다.

부록 D
EN 300 220 작동 주기

유럽연합에서 작동하도록 구성된 ALR-9900+ 리더(**ALR-9900+EMA**)는 EN 302 220 사양에 부합하게 실행한다. EN 300 220 사양에 따른 Alien 리더 운용 시 리더는 RF 배출 시 10% 작동주기를 유지해야 한다.

정상 작동 조건 하에서는 리더가 동기 모드에서 사용되는 횟수가 비교적 적을 경우에는 IO 기반으로 AutoMode 작업을 하거나 AutoMode를 연속된 작동주기로 구성함으로써 이 요건을 충족할 수 있다.
이 부록은 연속된 작동주기를 수행하는 AutoMode에서 리더를 실행하는 방법을 제시한다.

AutoMode로 작동주기 제어

리더가 켜져 있거나 꺼져 있는 상대적 시간은 Appropriate AutoStopTimer, AutoTruePause, AutoFalsePause 값을 사용하여 제어할 수 있다.
10% 작동주기를 위해서는(이 경우 리더는 시간의 10%만 켜져 있음), 다음 AutoMode 매개변수를 설정해야 한다:

AutoStopTimer	= 100	(평가 전 ~100ms까지 실행)
AutoTruePause	= 900	
AutoFalsePause	= 900	(AutoTruePause과 같게 설정)

이 값들의 배수(500/4500/4500 or 1000/9000/9000)도 사용할 수 있다.

부록 E
EN 302 208 작동

유럽연합에서 작동하도록 구성된 ALR-F800, ALR-9900+, ALR-9680 리더(**ALR-F800-EMA**, **ALR-9900+EMA** and **ALR-9680-EMA**)는 EN 302 208 사양에 부합하게 실행한다.
기본 작동 모드는 600 kHz로 공간을 설정한 4채널 간 호핑(hopping)이다.

이러한 리더는 또한 도일 채널에서 다시 전송하기 전 100ms 동안 정지할 때까지 최대 연속 전송 시간 4초 조건으로 고정 주파수에서 작동할 수 있다.

고정주파수 작동 설정 (ALR-F800/9900+/9680 EMA 만 해당)

리더가 기본 호핑 모드에서 또는 단일 고정 주파수 채널에서 작동하게 하려면 다음 명령어를 사용한다.

`RFChannel = <channel>`

- 허용된 값: `0-3` 또는 `-1` (hopping)
- 기본값: `-1` (hopping)
- 100ms 채널의 유지시간(dwell time)은 리더가 제어한다. 이 작동 모드에서 리더 작동주기를 수동으로 조절할 필요는 없다.

RFChannel 예제	
Command Response	Alien>**RFChannel?** RFChannel = -1 // hopping
Command Response	// set fixed frequency channel 1 Alien>**RFChannel** = 1 RFChannel = 1

부록 F
오류 코드

다음 표는 리더가 생성한 에러 코드 및 에러 메시지를 열거하며,
생성 방법 및 이유를 간략하게 기술한다.

일반 오류 (1-49)

오류 #	오류 메세지 세부사항
1	**Command not understood.** 전송된 명령어를 리더가 이해하지 못함.
4	**Invalid use of command.** 유효한 명령어를 부적절하게 사용함(읽기 전용 명령어를 변경. 필요하면 "get" 또는 "set" 없이 함)
5	**Timeout waiting for user command.** 사용자가 명령어를 입력하기 시작하였으나 60초 이내에 끝내지 못함.
7	**Command requires a value in the format Command=Value.** 값이 필요한 명령어에 값을 제공하지 않음.
10	**Value out of range.** 제공한 인수가 유효 범위 내에 있지 않음(문자열 길이, 정수가 필요한 곳에 정수가 아닌 수, 최소/최대 범위를 벗어남).
19	**Invalid Username and/or Password.** TCP 로그인에서 사용자명/패스워드 확인에 실패함.
22	**Invalid IP address.** 제시된 IP 주소 인수가 유효한 IP 주소로 보이지 않음.
24	**Invalid format.** 제시한 인수 구조가 잘못되었음.
27	**Invalid context.** 선택한 태그 프로토콜이 데이터 크기(예: EPC 길이) 또는 명령어(예: ProgProtocol != 2일 때 잠금해제)를 지원하지 않음.
28	**Unknown error.** (내부 오류)
29	**Parameter required.** 공지를 시도했으나 NotifyAddress가 지정되지 않음.
34	**Invalid mask.** EPC보다 큰 bitPtr + bitLen 또는 bitLen이 제시한 마스크 바이트에 부합하지 않음.
36	**Unable to deliver notification.** 공지 메시지 전송 시 문제가 발생함.
37	**Input buffer overflow.** 커맨드 라인에 너무 많은 문자를 입력하였음.
39	**Timeout waiting for command response.** 명령어를 내부 리더 구성요소에 전송하였으나 60초 내에 응답을 수신하지 못함.
38	**Invalid parameter.** 명령어 매개변수가 유효하지 않음. 예를 들면 매크로 이름에 상응하는 매크로가 없음.

Macro 오류 (50-60)

오류 #	오류 메세지 세부사항
51	**Macro name is missing.** 이 명령어는 매크로 이름이 필요하나 빠져 있음.
53	**Macro name is too long.** 매크로 이름이 64 문자를 초과한다.
54	**Macro does not exist.** 찾을 수 없는 매크로를 실행하거나 삭제하려고 하였음.
55	**The supplied macro is empty.** 명령어가 없는 매크로를 실행하려 하였음.
60	**Macro access error.** 매크로를 실행 또는 삭제하려고 하였으나, 리더가 필요한 파일시스템 승인을 하지 못함.

DSP 오류 (128-255)

오류 (dec)	오류 (hex)	오류 메세지 세부사항
133	0x85	**Memory overrun.** 존재하지 않는 메모리에 접속하였음.
134	0x86	**No tag found.** 프로그램 작동을 시도하였으나 이 필드에서 태그를 찾지 못함.
135	0x87	**Erase failed.** 태그를 지우려는 시도가 실패함.
136	0x88	**Program failed.** 프로그램 데이터를 태그에 쓰려는 시도가 실패함.
137	0x89	**Tag is locked.** 잠긴 태그 데이터를 프로그래밍하려고 시도하였음.
138	0x8A	**Kill failed. Check the kill password.** 태그를 없애려는 시도가 실패함. 태그를 없앨 수 없거나 잘못된 패스워드를 사용하였음.
139	0x8B	**Lock attempt failure.** 태그 데이터를 잠그려는 시도가 실패함.
140	0x8C	**Tag data memory size mismatch.** 작성하려는 데이터 양이 부정확함.
141	0x8D	**Hardware error.** 무선신호 하드웨어 구성요소 접속 문제.
146	0x92	**Lock failed CRC error.** 잠금 후에 태그 CRC 점검 오류.
147	0x93	**A previously found tag is not responding to program operations.**
148	0x94	**Kill Pwd could not be verified during Lock, no Lock performed.** C1 잠금 작업 중 Kill 패스워드 쓰기 실패.
149	0x95	**Singulation error** 다수 태그가 명령어에 응답하였다.

154	**0x9A**	**Read error.** 알 수 없는 태그 오류.
157	**0x9D**	**No tag found matching the provided EPC.** Kill 작업에서 명시한 EPC는 필드에 태그가 없음.
158	**0x9E**	**Access failed.** 제시한 AcqG2AccessPwd가 태그의 접속 패스워드에 부합하지 않음.
159	**0x9F**	**Malformed Tag response on write.** 쓰기 작업에 대한 태그의 비동시적 응답을 디코딩할 수 없음.

G2 Tag 오류 (256-511)

저수준 G2Read 및 G2Write 명령어들 중 하나를 수행하는 중에 G2가 반환한 원시 오류 코드가 있음.

오류 #	오류 메세지 세부사항
256	**(Tag Error) Other error.** G2 태그 오류가 다른 오류 코드에 해당되지 않음.
259	**(Tag Error) Memory overrun or unsupported PC value.** 기존에 없는 메모리 위치에 접속하려 했음. 또는 원하는 PC 값을 태그가 지원하지 않음.
260	**(Tag Error) Memory locked.** 잠긴 태그 메모리에 접속하려고 시도하였음.
267	**(Tag Error) Insufficient power.** 태그가 메모리 쓰기 작업을 하기에 필요한 동력이 부족함.
271	**(Tag Error) Non-specific error.** 태그 오류(에러 별 코드를 태그가 지원하지 않음).