ezTCP 기술자료

# 시리얼 매니저 프로토콜

문서버전 2.5

솔내시스템㈜

http://www.sollae.co.kr

# 목 차

# 1 개요

- CSE, CIE, CSW 제품은 시리얼 포트를 통해서 제품의 환경변수를 변경할 수 있는 기능을 제공합니다. 이러한 기능을 시리얼 매니저라고 합니다.
- 본 문서는 시리얼 매니저 기능의 프로토콜에 대한 설명을 담고 있습니다.
- 본 문서에서 사용하는 표기법은 다음과 같습니다.

| 구분 | 표기법 |
|---|---|
| 16진수 | 0x를 붙여서 표시합니다. |
| 바이너리 데이터 | 16진수를 사용하여 표시합니다. |
| 일반 텍스트 | 영문 또는 숫자로 표시합니다. |
| 0x20 | <SP> |
| 0x0D | <CR> |
| 0x0A | <LF> |

- 제품(ezTCP)의 환경변수는 특별한 언급이 없으면 기본적으로 Little-Endian을 사용합니다. Big-Endian을 사용하는 환경변수는 Big-Endian을 사용한다고 명시되어 있습니다.

# 2 시리얼 매니저 상태

## 2.1 시리얼 매니저 상태로 변경

- 제품(ezTCP)에 시리얼 매니저를 사용하여 환경변수를 설정하려면 제품(ezTCP)의 작동 상태를 시리얼 매니저 상태로 변경해야 합니다.
- 제품(ezTCP)의 작동 상태를 시리얼 매니저 상태로 변경하는 방법은 다음과 같습니다.

| 구분 | 시리얼 매니저 상태로 변경하는 방법 |
|---|---|
| 모듈형 | 제품이 부팅되면 제품(ezTCP)의 ISP핀에 20밀리초 ~ 1,000밀리초 이내로 LOW 신호를 보내면 시리얼 매니저 상태로 변경됩니다. |
| 외장형 | 제품이 부팅되면 제품(ezTCP)의 ISP 버튼을 20밀리초 ~ 1,000밀리초 이내로 눌렀다 떼면 시리얼 매니저 상태로 변경됩니다. |

☞ *ISP 핀 또는 ISP 버튼에 지정된 시간 이상 신호를 입력하면 제품(ezTCP)은 펌웨어를 다운로드 하는 상태로 진입합니다.*

- 제품(ezTCP)의 상태가 시리얼 매니저 상태로 변경되면 제품(ezTCP)의 시리얼 포트 설정값은 다음과 같이 변경됩니다.

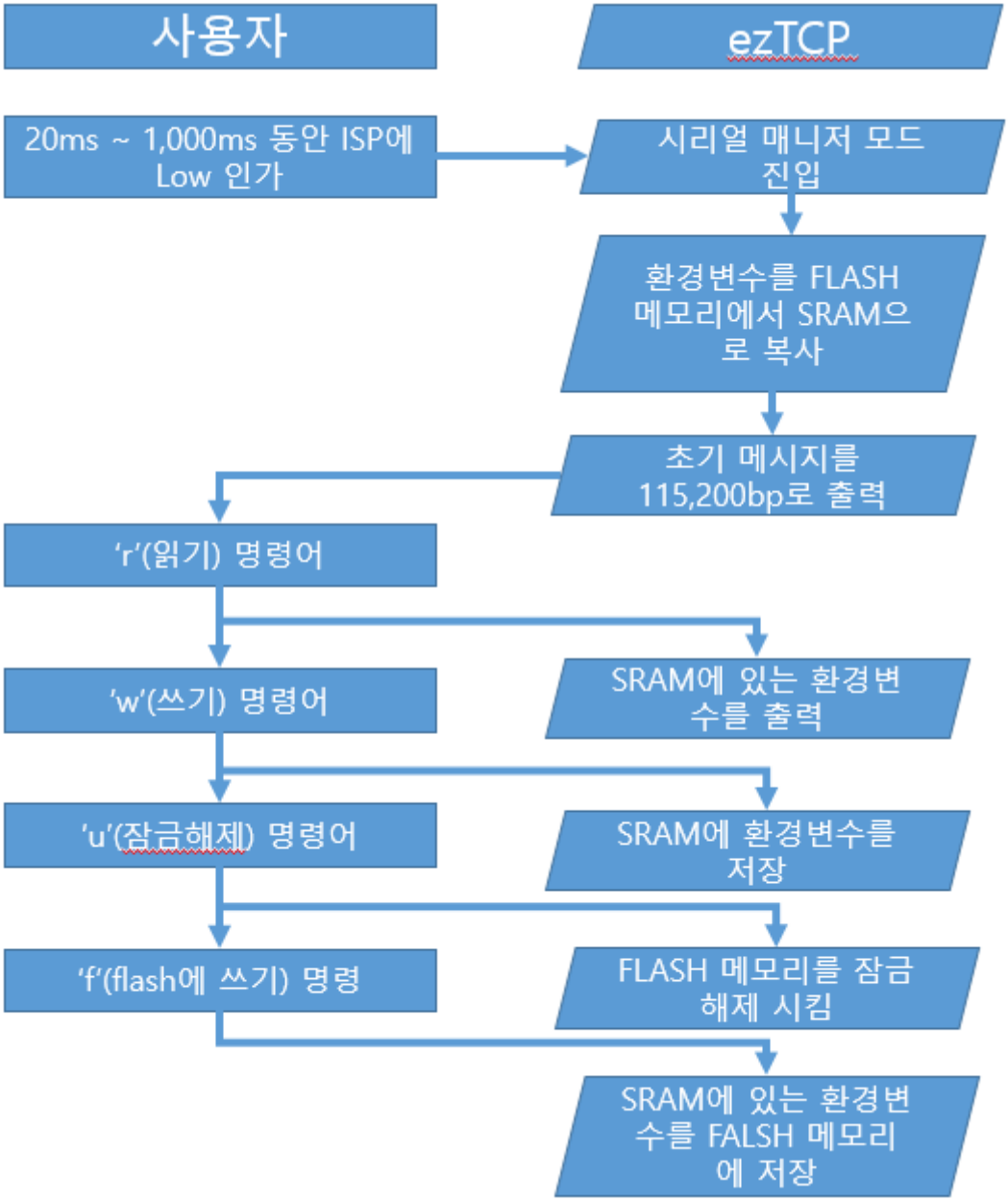
| | |
|---|---|
| 속도 | 115,200 bps |
| 데이터비트 | 8 |
| 정지비트 | 1 |
| 패리티 | 없음 |
| 흐름제어 | 없음 |

- 제품(ezTCP)은 시리얼 매니저 상태로 진입하면 다음과 같은 형태의 문자열을 시리얼 포트로 출력합니다.

| 항목 | 길이(바이트) | 바이너리 | | | 텍스트 |
|---|---|---|---|---|---|
| 메시지 코드 | 3 | 0x37 | 0x30 | 0x30 | 700 |
| <SP> | 1 | 0x20 | | | |
| 메시지 | N | | | | 임의의 문자열 |
| 메시지종료 코드(<CR><LF>) | 2 | 0x0D | | 0x0A | |

☞ *메시지 코드를 사용하여 시리얼 매니저 모드 진입 여부를 확인하면 됩니다.*

## 2.2 시리얼 매니저 작동 순서



# 3 시리얼 매니저 프로토콜

## 3.1 기본 규칙

- 제품(ezTCP)은 저장된 환경변수를 전송할 때 바이너리 형태의 데이터를 ASCII코드 형태로 바꾸고 각 데이터 사이에는 <SP>을 삽입합니다. 단, 마지막 데이터 다음에는 <CR><LF>를 삽입하여 데이터 전송의 마지막을 표시합니다.
- 다음은 길이가 2바이트인 바이너리 데이터를 ASCII코드 데이터로 변경한 예 입니다.

| 바이너리 데이터 | | ASCII코드 데이터 | | | | | 텍스트 |
|---|---|---|---|---|---|---|---|
| 0x61 | 0x7f | 0x36 | 0x31 | <SP> | 0x37 | 0x66 | 61<SP>7F |

## 3.2 ezTCP 환경변수

### 3.2.1 환경변수 구조

- 제품(ezTCP)의 환경변수는 전체 크기가 정해져 있지 않으며 제품마다 다른 값들이 저장되어 있습니다. 환경변수는 다음의 형식에 맞추어 제품 메모리에 바이너리 데이터 형태로 저장되어 있습니다.

| 항목 | 바이트 수 | 내용 |
|---|---|---|
| 코드 | 1 | 환경변수 종류를 나타냅니다. |
| 아이디 | 1 | 같은 코드의 데이터가 여러 개인 경우 아이디를 부여하여 데이터를 구분합니다. |
| 길이 | 2 | 코드부터 CRC까지 전체 길이를 나타냅니다.<br>**Little-Endian을 사용**합니다. |
| 데이터 | N | 코드, 아이디에 해당하는 데이터입니다. |
| 패드 | 0~N | 전체 길이를 4의 배수로 맞추는 데 필요한 경우 0x00을 삽입합니다. |
| CRC | 2 | 코드, 아이디, 길이, 데이터, 패드에 대한 CRC값 입니다.<br>**Little-Endian을 사용**합니다. |

● 코드와 아이디 정의는 다음과 같습니다.

| 코드 | 아이디 | 데이터 길이 (바이트) | 내용 |
|---|---|---|---|
| 0x02 | 0x01 | 2 | 제품 아이디<br>**Little-Endian을 사용**합니다.<br>**3.2.9. 절을 참고하세요.** |
| 0x03 | 0x00 | 4 | 제품 IP 주소.<br>**Big-Endian을 사용**합니다. |
| 0x03 | 0x01 | 4 | 서브넷마스크.<br>**Big-Endian을 사용**합니다. |
| 0x03 | 0x02 | 4 | 게이트웨이 IP 주소.<br>**Big-Endian을 사용**합니다. |
| 0x03 | 0x03 | 4 | DNS 서버 IP 주소.<br>**Big-Endian을 사용**합니다. |
| 0x04 | 0x00 | 18 | **IPv6 주소**<br>상위 16바이트는 IPv6 주소이고 하위 2바이트는<br>서브넷 접두사 길이입니다. |
| 0x04 | 0x02 | 16 | 게이트웨이 IPv6 주소입니다. |
| 0x04 | 0x03 | 16 | DNS 서버 IPv6 주소입니다. |
| 0x04 | 0x10 ~ | 16 | UART가 통신할 IPv6 주소입니다. |
| 0x05 | 0x00 | 4 | ezTCP 작동에 필요한 옵션.<br>**3.2.2. 절을 참고하세요.** |
| 0x06 | 0x00 | 10 | 비밀번호가 설정된 경우 현재 설정된 비밀번호입니다.<br>현재 **8바이트 만사용**하고 있습니다. |
| 0x07 | 0x00 | 64 | 제품 설명. |
| 0x07 | 0x01<br>~0x08 | 16 | 디지털 입력 포트에 대한 설명.<br>각 포트당 16바이트씩 사용 가능합니다.<br>**I/O제품만 사용 가능합니다.** |

| | | | |
|---|---|---|---|
| 0x07 | 0x09<br>~0x10 | 16 | 디지털 출력 포트에 대한 설명.<br>각 포트당 16바이트씩 사용 가능합니다.<br>**I/O제품만 사용 가능합니다.** |
| 0x08 | 0x00 | 48 | PPPoE, EAPoL을 위한 아이디, 비밀번호.<br>처음 32바이트가 아이디이고 다음 16바이트가 비밀번호입니다. |
| 0x08 | 0x01 | 48 | DDNS를 위한 아이디와 비밀번호.<br>처음 32바이트가 아이디이고 다음 16바이트가 비밀번호입니다. |
| 0x08 | 0x03 | 48 | WPA 설정이 EAP TLS, EAP TTLS, PEAP인 경우 ezTCP가 사용해야 할 아이디, 비밀번호.<br>처음 32바이트가 아이디이고 다음 16바이트가 비밀번호입니다. |
| 0x09 | 0x00 | 4 | 무선랜을 위한 환경변수.<br>**3.2.3. 절을 참고하세요.** |
| 0x09 | 0x01 | 32 | 액세스 포인트(무선랜장비)의 SSID.<br>**0x00으로 끝나는 ASCII데이터입니다**. 따라서 **최대 31바이트**까지 사용 가능 합니다. |
| 0x09 | 0x02 | 20 | 64비트 WEP키.<br>한 개의 키 길이는 5바이트이고, 4개의 킷값을 저장할 수 있습니다. 상위 5바이트가 키 아이디 0번입니다. |
| 0x09 | 0x03 | 52 | 128비트 WEP키.<br>한 개의 키 길이는 13바이트이고, 4개의 킷값을 저장할 수 있습니다. 상위 13바이트가 키 아이디 0번입니다. |

| | | | |
|---|---|---|---|
| 0x09 | 0x04 | 32 | WPA 암호문<br>**0x00으로 끝나는 ASCII데이터로 최대 31바이트**까지 사용 가능합니다. 최소 8바이트 이상 입력하십시오.<br>**3.2.10. 절을 참고하세요.** |
| 0x09 | 0x08 | 64 | WPA 암호문<br>**0x00으로 끝나는 ASCII데이터로 최대 63바이트**까지 사용 가능합니다. 최소 8바이트 이상 입력하십시오.<br>**3.2.10. 절을 참고하세요.** |
| 0x09 | 0x05 | 32 | WPA PSK<br>WPA 암호문과 SSID로 계산되는 WPA PSK 입니다.<br>**3.2.10. 절을 참고하세요.** |
| 0x09 | 0x09 | 32 | WPA PSK<br>WPA 암호문과 SSID로 계산되는 WPA PSK 입니다.<br>**3.2.10. 절을 참고하세요.** |
| 0x10,<br>0x14 | 0x00 ~ | 12 | UART의 하드웨어 설정값.<br>UART가 여러 개인 경우 아이디로 구분합니다. 아이디 0번이 첫 번째 UART를 가리킵니다.<br>**3.2.4. 절을 참고하세요.** |
| 0x20,<br>0x2c | 0x00 ~ | 20 | UART의 작동에 관련된 설정값.<br>UART가 여러 개인 경우 아이디로 구분합니다. 아이디 0번이 첫 번째 UART를 가리킵니다.<br>**3.2.5. 절을 참고하세요.** |
| 0x21 | 0x00 | 6 | 제품(ezTCP)에 접속 가능한 호스트를 MAC 주소로 제한할 때 사용합니다. |
| 0x22 | 0x00 | 8 | 제품(ezTCP)에 접속 가능한 호스트를 IP 주소와 서브넷 마스크로 제한할 때 사용합니다.<br>처음 4바이트가 IP 주소, 다음 4바이트가 서브넷 마스크입니다.<br>**Big-Endian을 사용**합니다. |

| | | | |
|---|---|---|---|
| 0x23 | 0x00 | 64 | UART 0번이 통신할 주소.<br>**0x00으로 끝나는 ASCII 데이터**입니다. **최대 63바이트**만 사용 가능합니다.<br>**3.2.5. 절을 참고하세요.** |
| 0x23 | 0x01 | 64 | UART 1번이 통신할 주소.<br>**0x00으로 끝나는 ASCII 데이터입니다. 최대 63바이트**만 사용 가능합니다.<br>**3.2.5. 절을 참고하세요.** |
| 0x23 | 0x02 | 64 | I/O 포트가 통신할 주소.<br>**0x00으로 끝나는 ASCII 데이터입니다. 최대 63바이트**만 사용 가능합니다.<br>**3.2.6. 절을 참고하세요.** |
| 0x23 | 0x80 | 64 | DynDNS서비스에 등록된 호스트 이름.<br>**0x00으로 끝나는 ASCII 데이터입니다. 최대 63바이트**만 사용 가능합니다.<br>**3.2.7. 절을 참고하세요.** |
| 0x23 | 0x81 | 64 | IP 주소 통보 기능을 사용하는 경우 IP 주소를 보낼 주소.<br>**0x00으로 끝나는 ASCII 데이터입니다. 최대 63바이트**만 사용 가능합니다.<br>**3.2.7. 절을 참고하세요.** |
| 0x24 | 0x00<br>~ 0x07 | 32 | 디지털 출력포트 매크로.<br>각 포트당 32바이트씩이며 아이디로 포트를 구별합니다. |
| 0x25 | 0x00 | 56 | I/O 제품을 위한 설정값.<br>**3.2.6. 절을 참고하세요.** |
| 0x27 | 0x00 | 12 | IP 주소 통보 기능을 위한 설정값.<br>**3.2.7. 절을 참고하세요.** |
| 0x28 | 0x00 | 10 | 제품(ezTCP)이 사용하는 TCP 포트번호.<br>I/O제품의 웹(HTTP) 포트 번호만 변경 가능합니다.<br>**3.2.8. 절을 참고하세요.** |

| | | | |
|---|---|---|---|
| 0x2a,<br>0x2d | 0x00 ~ | 6 | UART 구분자기능 설정값.<br>아이디 0번이 첫 번째 UART를 가리킵니다.<br>**3.2.11. 절을 참고하세요.** |
| 0x2b | 0x00 | 20 | 제품(ezTCP)에 접속 가능한 호스트를 IPv6 주소로 제한할 때 사용합니다.<br>상위 16바이트는 IPv6 주소이고 하위 4바이트는 서브넷 접두사 길이입니다. |

### 3.2.2 ezTCP 작동에 필요한 옵션

| 코드 | 아이디 |
|---|---|
| 0x05 | 0x00 |

- 총 4바이트이고 다음과 같은 구조로 되어있습니다.

```
struct opt_env
{
        unsigned int ezcfg_lock     : 1;
        unsigned int rcfg           : 1;
        unsigned int arp            : 1;
        unsigned int dhcp           : 1;
        unsigned int pppoe          : 1;
        unsigned int auto_ns        : 1;
        unsigned int pad0           : 2;    // 사용하지 않음. 값을 사용하지 마십시오.
        unsigned int ip6            : 1;
        unsigned int ip6_eui        : 2;
        unsigned int ip6_gua        : 2;
        unsigned int pad1           : 3;    // 사용하지 않음. 값을 사용하지 마십시오.

        unsigned int debug          : 1;
        unsigned int telnet         : 1;
        unsigned int ssl            : 1;
        unsigned int ssh            : 1;
        unsigned int http           : 1;
        unsigned int ddns           : 3;
        unsigned int t2smc          : 1;
        unsigned int secure         : 1;    // 읽기 전용. 값을 변경하지 마십시오.
        unsigned int mac_id         : 1;
        unsigned int ps             : 1;
        unsigned int pd             : 1;
        unsigned int pad3           : 3;    // 사용하지 않음. 값을 사용하지 마십시오.
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ezcfg_lock | rcfg | arp | dhcp | pppoe | auto_ns | pad0 | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| ip6 | ip6_eui | | ip6_gua | | pad1 | | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| debug | telnet | ssl | ssh | http | ddns | | |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| t2smc | secure | mac_id | ps | pd | pad3 | | |

- ezcfg_lock
  [입력/출력]
  ezcfg_lock 옵션을 설정하면 [코드:0x21, 아이디:0x00] 또는 [코드:0x22, 아이디:0x00]에 설정된 호스트만 ezManager 프로그램으로 제품(ezTCP)을 관리할 수 있습니다.

- rcfg
  [입력/출력]
  rcfg 옵션을 설정하면 IP 주소 검색 기능이 활성화 됩니다.
  원격지에서 제품(ezTCP)에 설정된 제품 IP 주소를 사용하여 제품(ezTCP)을 관리할 수 있습니다.

- arp
  [입력/출력]
  arp 옵션을 설정하면, 처음 수신된 패킷의 IP 주소 정보를 제품(ezTCP)의 IP 주소로 사용합니다.

- dhcp
  [입력/출력]
  dhcp 옵션을 설정하면 DHCP 프로토콜이 활성화 됩니다.

- pppoe
  [입력/출력]
  pppoe 옵션을 설정하면 PPPoE 프로토콜이 활성화 됩니다.

- auto_ns
  [입력/출력]
  auto_ns 옵션을 설정하면 DHCP 또는 PPPoE 프로토콜을 사용할 때 DNS 서버 IP 주소를 자동으로 설정합니다. 이 옵션을 설정하지 않으면 DNS 서버 IP 주소로 [코드:0x03, 아이디:0x03]에 설정된 값을 사용합니다.

- ip6

  [입력/출력]

  ip6를 설정하면 IP6 주소를 사용합니다.

- ip6_eui

  [입력/출력]

  링크-로컬 IP6 주소를 만드는 방법을 설정합니다.

| ip6_eui | 설명 |
|---|---|
| 0 | 제품(ezTCP) MAC 주소를 사용. |
| 1 | 임의로 생성. |

- ip6_gua

  [입력/출력]

  IP6 주소 방식

| ip6_gua | 설명 |
|---|---|
| 0 | 자동으로 IP 주소 받기. |
| 1 | 고정된 IP 주소 사용. |

- debug

  [입력/출력]

  debug 옵션을 설정하면, 제품(ezTCP)은 디버깅 정보를 담은 UDP 브로드 캐스트 패킷을 포트 번호 50006번으로 보냅니다.

- telnet

  [입력/출력]

  telnet 옵션을 설정하면, 제품(ezTCP)의 Telnet 콘솔에 접속할 수 있습니다. 이 옵션은 ssh 옵션과 같이 사용할 수 없습니다.

- ssl

  [입력/출력]

  ssl 옵션을 설정하면, TCP/IP 통신 시 SSL프로토콜을 사용하여 데이터를 암호화 합니다.

- ssh

  [입력/출력]

  ssh 옵션을 설정하면, TCP/IP 통신 시 SSH 프로토콜을 사용하여 데이터를 암호화 합니다.

- http

  [입력/출력]

  HTTP 프로토콜을 사용하여 제품(ezTCP)을 모니터링 할 수 있습니다.

  다음의 제품(ezTCP)만 지원합니다.

| LAN 종류 | 제품명 |
|---|---|
| 유선랜 | CIE-H10, CIE-M10, CIE-H12, CIE-H14 |
| 무선랜 | |

- ddns

  [입력/출력]

  IP 주소 통보 기능의 종류를 선택합니다.

| ddns | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | DDNS ( dyndns.org 서비스를 사용합니다 ) |
| 2 | TCP |
| 3 | UDP |

☞ *자세한 설명은 제품 사용자 설명서를 참고하십시오.*

- t2smc

  [입력/출력]

  "mux_type"을 T2S(TCP서버)로 사용하는 경우 다중접속 허용 여부를 선택합니다.

  이 변수는 아래의 제품만 유효 합니다.

| LAN 종류 | 제품명 |
|---|---|
| 유선랜 | CSE-M73, CSE-H25 |
| 무선랜 | |

- secure

  [출력]

  **읽기 전용 변수입니다.** 제품이 SSL 또는 SSH 기능 지원 여부를 나타냅니다.

- mac_id

  [입력/출력]

  mac_id 옵션을 설정하면, 제품은 TCP/IP 접속이 완료되면 원격지 호스트로 제품의 MAC 주소를 전송합니다.

### 3.2.3 무선랜을 위한 옵션

| 코드 | 아이디 |
|---|---|
| 0x09 | 0x00 |

- 총 4바이트이고 다음과 같은 구조로 되어있습니다.

```
struct wlan_opt
{
        unsigned int cctype             : 4;
        unsigned int channel            : 4;
        unsigned int wep                : 2;
        unsigned int wep_id             : 2;
        unsigned int pad0               : 1;
        unsigned int bg_scan            : 1;
        unsigned int auth               : 2;
        unsigned int wpa                : 3;
        unsigned int cipher             : 2;
        unsigned int pad1               : 3;    // 사용하지 않음. 값을 사용하지 마십시오.
        unsigned int antenna            : 1;
        unsigned int phy                : 3;
        unsigned int short_preamble     : 1;
        unsigned int short_slot         : 1;
        unsigned int cts_protection     : 1;
        unsigned int pad2               : 1;    // 사용하지 않음. 값을 사용하지 마십시오.
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| cctype | | | | channel | | | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| wep | | wep_id | | pad0 | bg_scan | auth | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| wpa | | | cipher | | 사용안함 | | |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| antenna | phy | | | SP | SS | CTS | pad2 |

- cctype

  [입력/출력] 무선랜 작동 방식

| cctype | 설명 |
|---|---|
| 0 | 애드혹 |
| 1 | 인프라스트럭처 |
| 2 | Soft AP |

- channel

  [입력/출력] 무선랜 채널번호

  cctype이 0(애드혹)인 경우에만 유효합니다.

- wep

  [입력/출력] 무선랜 암호화 방식

| wep | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | WEP – 64비트 키 |
| 2 | WEP – 128비트 키 |

☞ *WEP(Wired Equivalent Privacy)*

- wep_id

  [입력/출력] WEP키 아이디 (0, 1, 2, 3)

- auth

  [입력/출력] 무선랜 인증방식

| auth | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | 개방모드 |
| 2 | 공유모드 |
| 3 | 혼합모드 |

- wpa

  [입력/출력] WPA(Wi-Fi Protected Access) 인증방식

| wpa | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | EAP TLS |
| 2 | WPA-PSK |
| 3 | EAP TTLS |
| 4 | WPA2-PSK |
| 5 | PEAP |

- cipher

  [입력/출력] WPA 암호와 방법

| cipher | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | TKIP (Temporal Key Integrity Protocol ) |
| 2 | AES ( Advanced Encryption Standard ) |
| 3 | TKIP/AES |

- antenna

  [입력/출력]

| antenna | 설명 |
|---|---|
| 0 | 내장 안테나 |
| 1 | 외장 안테나 |

*** 안테나 설정은 CSW-M85만 지원 합니다.**

- passive, bg_scan, phy, short_preamble, short_slot, cts_protection

  [입력/출력]

  무선랜 작동에 관한 고급설정입니다. 자세한 내용은 해당 제품 사용자 설명서를 참고하십시오.

### 3.2.4 UART 의 하드웨어 설정값

| 코드 | 아이디 |
|---|---|
| 0x10 | 0x00 ~ |

- 총 12바이트이고 다음과 같은 구조로 되어있습니다.

```
struct uart_dev_env
{
unsigned int max_stype     : 2;      // 사용하지 않음. 값을 사용하지 마십시오.
unsigned int stype         : 2;
unsigned int databit       : 2;
unsigned int stopbit       : 2;
unsigned int parity        : 2;
unsigned int flowctrl      : 2;
unsigned int telcom        : 2;
unsigned int parity2       : 2;
unsigned int en_ttl        : 1;      // 읽기 전용. 값을 변경하지 마십시오.
unsigned int ttl           : 1;
unsigned int en_tx_delay   : 1;      // 읽기 전용. 값을 변경하지 마십시오.
unsigned int tx_delay      : 5;
unsigned int dtrdsr        : 1;
unsigned int pad0          : 1;
unsigned int pad3          : 6;      // 사용하지 않음. 값을 사용하지 마십시오.
DWORD max_baud;                      // 읽기 전용. 값을 변경하지 마십시오.
DWORD sio_baud;
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| max_stype | | stype | | databit | | stopbit | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| parity | | flowctrl | | telcom | | parity2 | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| en_ttl | ttl | en_tx_delay | tx_delay | | | | |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| dtrdsr | pad0 | pad3 | | | | | |

| 5th 바이트 |
|---|
| max_baud(Little-Endian) |

| 6th 바이트 |
|---|
| max_baud(Little-Endian) |

| 7th 바이트 |
|---|
| max_baud(Little-Endian) |

| 8th 바이트 |
|---|
| max_baud(Little-Endian) |

| 9th 바이트 |
|---|
| sio_baud(Little-Endian) |

| 10th 바이트 |
|---|
| sio _baud(Little-Endian) |

| 11th 바이트 |
|---|
| sio _baud(Little-Endian) |

| 12th 바이트 |
|---|
| sio _baud(Little-Endian) |

- stype

  [입력/출력] 시리얼 종류.

| stype | 설명 |
|---|---|
| 0 | RS-232 |
| 1 | RS-485 |
| 2 | RS-422 |

- databit

  [입력/출력] 데이터 비트.

| databit | 설명 |
|---|---|
| 0 | 5-비트 |
| 1 | 6-비트 |
| 2 | 7-비트 |
| 3 | 8-비트 |

- stopbit

  [입력/출력] 정지 비트.

| stopbit | 설명 |
|---|---|
| 0 | 1-비트 |
| 1 | 1.5-비트 |
| 2 | 2-비트 |

- parity

  [입력/출력] 패리티 비트.

| parity | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | EVEN 패리티 |
| 2 | ODD 패리티 |
| 3 | parity2 변수를 사용 |

- flowctrl

  [입력/출력] 흐름제어.

| flowctrl | 설명 |
|---|---|
| 0 | 사용 안 함 |
| 1 | RTS / CTS |
| 2 | Xon / Xoff |

- telcom

  [입력/출력]

  시리얼 포트 설정/상태 전송(RFC2217) 옵션 사용여부를 설정합니다.

  자세한 설명은 제품 사용자 설명서를 참고하십시오.

- parity2

  [입력/출력] 패리티 비트.

| parity2 | 설명 |
|---|---|
| 0 | Mark 패리티 |
| 1 | Space 패리티 |

- en_ttl

  [출력]

  **읽기 전용 변수입니다.** 제품의 TTL레벨 출력 지원 여부를 나타냅니다.

  이 변수는 아래의 제품만 유효 합니다.

| LAN 종류 | 제품명 |
|---|---|
| 유선랜 | CSE-M73(H/W 버전 1.3이상, F/W 버전 1.4a이상) |
| 무선랜 | |

- ttl

  [입력/출력]

  TTL레벨 출력 사용여부를 설정합니다.

  이 변수는 아래의 제품만 유효 합니다.

| LAN 종류 | 제품명 |
|---|---|
| 유선랜 | CSE-M73(H/W 버전 1.3이상, F/W 버전 1.4a이상) |
| 무선랜 | |

- en_tx_delay

  [출력]

  **읽기 전용 변수입니다.** 제품의 데이터 전송 간격 옵션 지원 여부를 나타냅니다.

- tx_delay

  [입력/출력]

  데이터 전송 간격을 설정 합니다. 자세한 설명은 제품 사용자 설명서를 참고하십시오.

- dtrdsr

  [입력/출력] DTR/DSR 흐름제어.

- max_baud

  [출력] 최대 시리얼 통신 속도.

  **읽기 전용 변수입니다.** 이 변수는 UART가 지원하는 최대 시리얼 통신 속도를 나타냅니다.

- sio_baud

  [입력/출력] 시리얼 통신 속도.

  UART의 통신 속도를 설정합니다.

  **max_baud 보다 큰 값을 설정하는 경우 제품이 동작하지 않을 수 있으므로 시리얼 통신 속도 설정 시 주의하여 주십시오.**

### 3.2.5 UART 의 작동에 관련된 설정값

| 코드 | 아이디 |
|---|---|
| 0x20 | 0x00 ~ |

- 총 20바이트이고 다음과 같은 구조로 되어있습니다.

```
struct uart_var_env
{
_u32 local_ip;              // 사용하지 않음. 값을 사용하지 마십시오.
_u32 peer_ip;
_u16 local_port;
_u16 peer_port;
_u8 mux_type;
_u8 no_delay     :1;
_u8 cod_listen   :1;        // 사용하지 않음. 값을 사용하지 마십시오.
_u8 secure       :2;
_u8 pad          :4;        // 사용하지 않음. 값을 사용하지 마십시오.
_u16 water_mark;
_u16 time_mark;
_u16 timeout;
};
```

| 1st 바이트 |
|---|
| local_ip(Big-Endian) |

| 2nd 바이트 |
|---|
| local_ip(Big-Endian) |

| 3rd 바이트 |
|---|
| local_ip(Big-Endian) |

| 4th 바이트 |
|---|
| local_ip(Big-Endian) |

| 5th 바이트 |
| --- |
| peer_ip(Big-Endian) |

| 6th 바이트 |
| --- |
| peer_ip(Big-Endian) |

| 7th 바이트 |
| --- |
| peer_ip(Big-Endian) |

| 8th 바이트 |
| --- |
| peer_ip(Big-Endian) |

| 9th 바이트 |
| --- |
| local_port(Little-Endian) |

| 10th 바이트 |
| --- |
| local_port(Little-Endian) |

| 11th 바이트 |
| --- |
| peer_port(Little-Endian) |

| 12th 바이트 |
| --- |
| peer_port(Little-Endian) |

| 13th 바이트 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| mux_type | | | | | | | |

| 14th 바이트 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| no_delay | cod_listen | secure | | pad | | | |

| 15th 바이트 |
| --- |
| water_mark(Little-Endian) |

| 16th 바이트 |
| --- |
| water_mark(Little-Endian) |

| 17th 바이트 |
| --- |
| time_mark(Little-Endian) |

| 18th 바이트 |
| --- |
| time_mark(Little-Endian) |

| 19th 바이트 |
| --- |
| timeout(Little-Endian) |

| 20th 바이트 |
| --- |
| timeout(Little-Endian) |

- peer_ip
  [입력/출력] 제품이 통신 할 서버의 IP 주소.
  "mux_type"이 COD(TCP 클라이언트)와 U2S(UDP)인 경우에만 사용됩니다.
  IP 주소 대신 DNS 이름을 사용하려면 peer_ip 값을 0으로 설정하고 [코드:0x23, 아이디:0x00]에 DNS 이름을 저장하면 됩니다.

- local_port
  [입력/출력] 제품 로컬포트.
  "mux_type"이 COD(TCP 클라이언트)와 U2S(UDP)인 경우에만 사용됩니다.

- peer_port
  [입력/출력] 제품이 통신할 서버의 포트번호.
  "mux_type"이 COD(TCP 클라이언트)와 U2S(UDP)인 경우에만 사용됩니다.

- mux_type
  [입력/출력] 제품의 동작방식을 설정.

| mux_type | 동작 | 설명 |
| --- | --- | --- |
| 0 | T2S | [TCP 서버]<br>제품은 TCP/IP 접속을 기다립니다. |
| 1 | ATC | [AT 명령]<br>AT명령어를 사용하여 TCP 서버 또는 TCP 클라이언트로 사용이 가능합니다. |
| 2 | COD | [TCP 클라이언트]<br>제품은 peer_ip와 peer_port를 사용하여 TCP/IP 접속을 시도합니다. |
| 3 | U2S | [UDP]<br>UDP를 사용하여 통신을 합니다. |

- no_delay
  [입력/출력]
  no_dealy 옵션을 설정하면 UART에서 수신한 데이터를 전송지연 기능 없이 네트워크로 전송합니다. 좀 더 빠른 데이터 전송을 원하는 경우에 사용하십시오.

- water_mark
  [입력/출력] TCP 접속 전 데이터 크기 [단위:바이트].
  제품의 시리얼 포트로 "water_mark" 이상의 데이터가 수신되면 TCP 접속을 시도하거나 네트워크로 데이터를 전송합니다.
  "mux_type"이 COD(TCP 클라이언트)와 U2S(UDP)인 경우에만 사용됩니다.

- time_mark
  [입력/출력] 데이터 프레임 간격 [단위:10ms]
  시리얼 포트로 마지막 데이터를 수신 한 후 "time_mark"동안 시리얼 포트로 데이터가 들어오지 않으면 이전에 수신된 데이터를 전송합니다.
  최소값은 4(40ms)입니다.

- timeout
  [입력/출력] 접속동료 대기시간 [단위:초].
  "mux_type"이 T2S(TCP서버), ATC(AT명령), COD(TCP클라이언트)인 경우 timeout에 지정된 시간만큼 TCP/IP 데이터 통신이 없으면 제품이 TCP/IP 접속을 먼저 해제합니다. 0이면 TCP/IP 접속을 계속 유지합니다.

### 3.2.6 I/O 제품을 위한 설정값

| 코드 | 아이디 |
|---|---|
| 0x25 | 0x00 |

- 총 56바이트이고 다음과 같은 구조로 되어있습니다.

```
struct io_var_env
{
        unsigned int modbus         : 1;
        unsigned int macro          : 1;
        unsigned int master         : 1;
        unsigned int active         : 1;
        unsigned int notify         : 1;
        unsigned int conns          : 3;
        unsigned int emacro         : 8;
        unsigned int query          : 1;
        unsigned int ctrl           : 1;
        unsigned int pad0           : 6;    // 사용하지 않음. 값을 사용하지 마십시오.
        unsigned int pad1           : 8;    // 사용하지 않음. 값을 사용하지 마십시오.
        _u32 peer_ip;
        _u16 peer_port;
        _u16 slave_id;
        _u16 input_addr;
        _u16 output_addr;
        _u32 init_output;
        _u32 poll_interval;
        _u16 input_valid_time[8];
        _u16 output_delay[8];
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| modbus | macro | master | active | notify | conns | | |

| 2nd 바이트 |
|---|
| emacro |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| query | ctrl | pad0 | | | | | |

| 4th 바이트 |
|---|
| pad1 |

| 5th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 6th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 7th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 8th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 9th 바이트 |
|---|
| peer_port(Little-Endian) |

| 10th 바이트 |
|---|
| peer_port(Little-Endian) |

| 11th 바이트 |
|---|
| slave_id(Little-Endian) |

| 12th 바이트 |
|---|
| slave_id(Little-Endian) |

| 13th 바이트 |
|---|
| input_addr(Little-Endian) |

| 14th 바이트 |
|---|
| input_addr(Little-Endian) |

| 15th 바이트 |
|---|
| output_addr(Little-Endian) |

| 16th 바이트 |
|---|
| output_addr(Little-Endian) |

| 17th 바이트 | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| init_output(Little-Endian) | | | | | | | |
| port 0 | port 1 | port 2 | port 3 | port 4 | port 5 | port 6 | port 7 |

| 18th 바이트 |
|---|
| init_output(Little-Endian) |

| 19th 바이트 |
|---|
| init_output(Little-Endian) |

| 20th 바이트 |
|---|
| init_output(Little-Endian) |

| 21st 바이트 |
|---|
| poll_interval(Little-Endian) |

| 22nd 바이트 |
|---|
| poll_interval(Little-Endian) |

| 23rd 바이트 |
|---|
| poll_interval(Little-Endian) |

| 24th 바이트 |
|---|
| poll_interval(Little-Endian) |

| 25th 바이트 |
|---|
| input_valid_time (Little-Endian) : port 0 |

| 26th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 0 |

| 27th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 1 |

| 28th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 1 |

| 29th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 2 |

| 30th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 2 |

| 31st 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 3 |

| 32nd 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 3 |

| 33rd 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 4 |

| 34th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 4 |

| 35th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 5 |

| 36th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 5 |

| 37th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 6 |

| 38th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 6 |

| 39th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 7 |

| 40th 바이트 |
| --- |
| input_valid_time (Little-Endian) : port 7 |

| 41st 바이트 |
| --- |
| output_delay (Little-Endian) : port 0 |

| 42nd 바이트 |
| --- |
| output_delay (Little-Endian) : port 0 |

| 43rd 바이트 |
| --- |
| output_delay (Little-Endian) : port 1 |

| 44th 바이트 |
| --- |
| output_delay (Little-Endian) : port 1 |

| 45th 바이트 |
| --- |
| output_delay (Little-Endian) : port 2 |

| 46th 바이트 |
| --- |
| output_delay (Little-Endian) : port 2 |

| 47th 바이트 |
| --- |
| output_delay (Little-Endian) : port 3 |

| 48th 바이트 |
| --- |
| output_delay (Little-Endian) : port 3 |

| 49th 바이트 |
| --- |
| output_delay (Little-Endian) : port 4 |

| 50th 바이트 |
| --- |
| output_delay (Little-Endian) : port 4 |

| 51st 바이트 |
| --- |
| output_delay (Little-Endian) : port 5 |

| 52nd 바이트 |
| --- |
| output_delay (Little-Endian) : port 5 |

| 53rd 바이트 |
| --- |
| output_delay (Little-Endian) : port 6 |

| 54th 바이트 |
| --- |
| output_delay (Little-Endian) : port 6 |

| 55th 바이트 |
| --- |
| output_delay (Little-Endian) : port 7 |

| 56th 바이트 |
| --- |
| output_delay (Little-Endian) : port 7 |

- modbus

  [입력/출력]

  Modbus 옵션을 설정하면 제품의 I/O포트를 Modbus/TCP 프로토콜을 사용하여 제어합니다.

- macro

  [입력/출력]

  macro 옵션을 설정하면 제품의 디지털 출력 포트를 매크로 기능을 사용하여 제어합니다.

- master

  [입력/출력]

| master | 설명 |
| --- | --- |
| 0 | Modbus/TCP 슬레이브 |
| 1 | Modbus/TCP 마스터 |

- active

  [입력/출력]

| active | 설명 |
|---|---|
| 0 | 수동접속 (TCP 서버) |
| 1 | 능동접속 (TCP 클라이언트) |

- notify

  [입력/출력] 입력포트 변경 알림.

  notify 옵션을 설정하면, 디지털 입력 포트에 입력 값이 변경되는 경우 마스터로 변경된 입력 포트 데이터를 Modbus/TCP를 사용하여 전송합니다.

- conns

  [입력/출력] Modbus/TCP 다중접속 개 수

  Modbus/TCP를 사용하는 경우 한번에 접속 가능한 TCP/IP 접속 개수를 선택할 수 있습니다. 최대 8개까지 가능합니다.

  *** 펌웨어 버전 1.3F 이상에서 지원합니다.**

- emacro

  [입력/출력]

  각 디지털 출력 포트 별로 매크로 기능 사용을 설정합니다.

  LSB부터 포트 0번이고 현재8비트만 사용합니다. 비트가 1이면 사용, 0이면 사용 안 함 입니다.

- query

  [입력/출력]

  master 변수를 1(Modbus/TCP 마스터)로 선택한 경우, 제품이 사용할 제어 명령어 종류를 선택합니다.

| query | 설명 |
|---|---|
| 0 | **FC 16(동시제어)**<br>Modbus/TCP 클래스0의 16번 함수인 write multiple registers를 이용해 출력 포트를 일괄적으로 제어하고 03번 함수인 read multiple registers를 이용해 입력포트를 감시합니다. |
| 1 | **FC 05(개별제어)**<br>입력포트 감시는 02번 함수인 read input discretes를 사용하고 출력포트 제어는 FC 05번 함수인 write coil을 사용합니다. 이 write coil 함수는 각각의 출력포트를 개별적으로 제어할 수 있게 해 줍니다. |

- ctrl

  [입력/출력]

  master 변수를 1(Modbus/TCP 마스터)로 선택한 경우, 제품 출력포트 제어방식을 선택합니다.

| ctrl | 설명 |
|---|---|
| 0 | 논리곱 |
| 1 | 논리합 |

- peer_ip

  [입력/출력]

  active 변수를 1(능동접속)로 선택한 경우 제품이 통신 할 서버의 IP 주소를 설정합니다. IP 주소 대신 DNS 이름을 사용하려면 peer_ip 값을 0으로 설정하고 [코드:0x23, 아이디:0x02]에 DNS 이름을 저장하면 됩니다.

- peer_port

  [입력/출력]

  active 변수를 1(능동접속)로 선택한 경우 제품이 통신 할 서버의 포트번호를 설정합니다.

  active 변수를 0(수동접속)로 선택한 경우 제품의 로컬 포트번호로 사용됩니다.

- slave_id

  [입력/출력] 유니트 아이디

  master 변수에 따라서 다음의 의미를 가집니다.

| master | 설명 |
|---|---|
| 0 - Modbus/TCP 슬레이브 | 제품의 유니트 아이디 |
| 1 - Modbus/TCP 마스터 | 원격장비의 유니트 아이디 |

- input_addr

  [입력/출력] 입력 포트 주소

  master 변수에 따라서 다음의 의미를 가집니다.

| master | 설명 |
|---|---|
| 0 - Modbus/TCP 슬레이브 | 제품의 입력 포트 주소 |
| 1 - Modbus/TCP 마스터 | 슬레이브 장비의 입력 포트 주소 |

☞ *입력 포트 주소와 출력 포트 주소는 8이상 차이가 나야 합니다.*

- output_addr
  [입력/출력] 출력 포트 주소
  master 변수에 따라서 다음의 의미를 가집니다.

| master | 설명 |
|---|---|
| 0 - Modbus/TCP 슬레이브 | 제품의 출력 포트 주소 |
| 1 - Modbus/TCP 마스터 | 슬레이브 장비의 출력 포트 주소. |

☞ *입력 포트 주소와 출력 포트 주소는 8이상 차이가 나야 합니다.*

- init_output
  [입력/출력] 출력 포트의 초기상태.
  디지털 출력 포트의 초기상태를 설정합니다. LSB부터 포트 0번이고 현재8비트만 사용합니다. 비트가 1이면 켜지고 0이면 꺼집니다.

- poll_interval
  [입력/출력]
  슬레이브 장비로부터 데이터를 읽는 주기를 설정합니다.[단위:밀리 초]

- input_valid_time
  [입력/출력]
  이 변수에 대한 자세한 설명은 제품 사용자 설명서를 참고하십시오

- output_delay
  [입력/출력]
  이 변수에 대한 자세한 설명은 제품 사용자 설명서를 참고하십시오.

### 3.2.7 IP 주소 통보 기능을 위한 설정값

| 코드 | 아이디 |
|---|---|
| 0x27 | 0x00 |

- 총 12바이트이고 다음과 같은 구조로 되어있습니다.

```
struct ip_trap_env
{
        unsigned int level   : 3;
        unsigned int pad0    : 13;      // 사용하지 않음. 값을 사용하지 마십시오.
        unsigned int pad1    : 16;      // 사용하지 않음. 값을 사용하지 마십시오.
        _u32 peer_ip;
        _u16 peer_port;
        _u16 interval;
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| level | | | pad0 | | | | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| pad0 | | | | | | | |

| 3rd 바이트 |
|---|
| pad1 |

| 4th 바이트 |
|---|
| pad1 |

| 5th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 6th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 7th 바이트 |
|---|
| peer_ip(Big-Endian) |

| 8th 바이트 |
| --- |
| peer_ip(Big-Endian) |

| 9th 바이트 |
| --- |
| peer_port(Little-Endian) |

| 10th 바이트 |
| --- |
| peer_port(Little-Endian) |

| 11th 바이트 |
| --- |
| interval_port(Little-Endian) |

| 12th 바이트 |
| --- |
| interval_port(Little-Endian) |

- level

  [입력/출력]

  "opt_env" 구조체의 "ddns"변수가 2(TCP) 또는 3(UDP)인 경우 제품이 전송하는 IP 주소 관련 데이터의 형식을 지정합니다.

| level | 설명 |
| --- | --- |
| 0 | ASCII |
| 1 | 바이너리 |

- peer_ip

  [입력/출력] 제품의 IP 주소 정보를 수신 할 호스트의 IP 주소.

  IP 주소 대신 DNS 이름을 사용하려면 peer_ip 값을 0으로 설정합니다.

  그리고, 3.2.2 절의 ddns 값이 1(DDNS)인 경우에는 [코드:0x23, 아이디:0x80]에 DNS 이름을 저장하고, 2(TCP) 또는 3(UDP)인 경우에는 [코드:0x23, 아이디:0x81]에 DNS이름을 저장합니다.

- peer_port

  [입력/출력] 제품의 IP 주소 정보를 수신 할 호스트의 포트번호.

- interval

  [입력/출력] 제품의 IP 주소 정보를 전송하는 주기 [단위 : 분].

  3.2.2 절의 ddns 값이 2(TCP) 또는 3(UDP)인 경우에만 사용됩니다.

### 3.2.8 제품(ezTCP)이 사용하는 TCP 포트번호

| 코드 | 아이디 |
|---|---|
| 0x28 | 0x00 |

- ezTCP 시스템에서 사용하는 TCP 포트번호를 변경할 수 있습니다. 현재 I/O 제품의 HTTP번호만 변경가능 합니다.
- 총 10바이트이고 다음과 같은 구조로 되어있습니다.

```
struct port_map_env
{
        _u16 http_port;
        _u16 reserved1;     // 사용하지 않음. 값을 사용하지 마십시오.
        _u16 reserved2;     // 사용하지 않음. 값을 사용하지 마십시오.
        _u16 reserved3;     // 사용하지 않음. 값을 사용하지 마십시오.
        _u16 reserved4;     // 사용하지 않음. 값을 사용하지 마십시오.
};
```

| 1st 바이트 |
|---|
| http_port(Little-Endian) |

| 2nd 바이트 |
|---|
| http_port(Little-Endian) |

| 3rd 바이트 |
|---|
| - |

| 4th 바이트 |
|---|
| - |

| 5th 바이트 |
|---|
| - |

| 6th 바이트 |
|---|
| - |

| 7th 바이트 |
| --- |
| - |

| 8th 바이트 |
| --- |
| - |

| 9th 바이트 |
| --- |
| - |

| 10th 바이트 |
| --- |
| - |

- http_port

  [입력/출력] HTTP 포트번호

  이 변수는 아래의 제품만 유효 합니다.

| LAN 종류 | 제품명 |
| --- | --- |
| 유선랜 | CIE-H10, CIE-M10, CIE-H12, CIE-H14 |
| 무선랜 | - |

### 3.2.9 제품(eTCP) 아이디

| 코드 | 아이디 |
|---|---|
| 0x02 | 0x01 |

- 제품명은 아래와 같습니다

| 제품아이디 | 제품(ezTCP)명 |
|---|---|
| 0x20 | CIE-H10 |
| 0x21 | CIE-M10 |
| 0x22 | CIE-H14 |
| 0x23 | CSE-M32 |
| 0x24 | CSE-H20 |
| 0x25 | CSE-H21 |
| 0x26 | CSE-M73 |
| 0x27 | CSW-H80 |
| 0x29 | CSE-H25 |
| 0x2b | CSE-M53 |
| 0x2c | CSE-H53 |
| 0x2d | CSW-M83 |
| 0x2e | CSW-M85 |
| 0x2f | CSE-H55 |
| 0x30 | CSC-HR2 |
| 0x31 | CSE-M24 |
| 0x33 | CSC-H64 |
| 0x34 | CIE-H12 |
| 0x35 | CSW-H85 |
| 0x36 | CSE-T32 |
| 0x37 | CSE-M53NI |
| 0x39 | CSE-T16 |
| 0x3a | CSE-T48 |
| 0x3b | CSE-H53A |
| 0x3d | CSE-M53N |
| 0x3e | CSE-H53N |
| 0x3f | CSE-H55N |
| 0x60 | IDIS-200 |
| 0x61 | CSC-H61 |

| 0x62 | CSC-H62 |
|---|---|
| 0x63 | CSW-H85N |

### 3.2.10 WPA 관련 설정값

| 코드 | 아이디 |
|---|---|
| 0x09 | 0x04 |

- WPA 암호문

- **CSW-H80만 사용합니다.**
- WPA PSK 키를 만들 때 사용되는 암호문으로 32**바이트 크기**입니다.
- 암호문은 **0x00으로 끝나는 ASCII데이터로 최대 31바이트**까지 사용할 수 있습니다.
- 암호문에는 A~Z, a~z, 0~9만 사용할 수 있습니다.
- 최소 8바이트 이상 입력해야 합니다.

| 코드 | 아이디 |
|---|---|
| 0x09 | 0x05 |

- WPA PSK(Pre-shared Key)

- **CSW-H80만 사용합니다.**
- SSID [ 코드:0x09, 아이디:0x01 ] 와 WPA 암호문 [ 코드:0x09, 아이디:0x04 ] 을 사용하여 계산됩니다. PSK는 **3.8 PSK 생성 명령어로** 계산할 수 있습니다.
- PSK 생성에는 CSW-H80이 사용하는 CPU로 약 30초의 시간이 필요합니다.
- 다음의 웹 페이지에서 PSK를 확인해 볼 수 있습니다..
  http://www.wireshark.org/tools/wpa-psk.html

| 코드 | 아이디 |
|---|---|
| 0x09 | 0x08 |

- WPA 암호문

- **CSW-M83 / M85, CSC-H64만 사용합니다.**
- WPA PSK 키를 만들 때 사용되는 암호문으로 64**바이트 크기**입니다.
- 암호문은 **0x00으로 끝나는 ASCII데이터로 최대 63바이트**까지 사용할 수 있습니다.
- 암호문에는 A~Z, a~z, 0~9만 사용할 수 있습니다.
- 최소 8바이트 이상 입력해야 합니다.

| 코드 | 아이디 |
|---|---|

| | |
|---|---|
| 0x09 | 0x09 |

- WPA PSK(Pre-shared Key)

- **CSW-M83 / M85, CSC-H64만 사용합니다.**
- SSID [ 코드:0x09, 아이디:0x01 ] 와 WPA 암호문 [ 코드:0x09, 아이디:0x08 ] 을 사용하여 계산됩니다. PSK는 **3.8 PSK 생성 명령어로** 계산할 수 있습니다.
- PSK 생성에는 CSW-M83 / M85 가 사용하는 CPU로 약 5초의 시간이 필요합니다.
- 다음의 웹 페이지에서 PSK를 확인해 볼 수 있습니다..
  http://www.wireshark.org/tools/wpa-psk.html

### 3.2.11 UART 의 구분자 기능

| 코드 | 아이디 |
|---|---|
| 0x2a, 2d | 0x00 ~ |

- 총 6바이트이고 아래와 같은 구조로 되어있습니다.

| 1st 바이트 |
|---|
| 구분자 0번 |

| 2nd 바이트 |
|---|
| 구분자 1번 |

| 3rd 바이트 |
|---|
| 구분자 2번 |

| 4st 바이트 |
|---|
| 구분자 3번 |

| 5st 바이트 | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 구분자 길이 | | | | 사용 안 함 | | | |

| 6st 바이트 | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 사용 안 함 | | | | 구분자 동작방식 | | | |

- 구분자 동작방식은 다음과 같습니다.

| 구분자 동작방식 | 설명 |
|---|---|
| 0 | 구분자까지 전송 |
| 1 | 구분자 + 1바이트 전송 |
| 2 | 구분자 + 2바이트 전송 |

## 3.3 읽기 명령어

### 3.3.1 읽기 명령어 구조

- 읽기 명령은 영문 r 또는 R을 사용합니다. 읽기 명령어 인자로 읽고자 하는 환경변수의 코드와 아이디를 입력하면 해당하는 환경변수를 출력합니다.
- 읽기 명령어의 구조는 다음과 같습니다.

| 데이터 | 데이터 길이 (바이트) | 값 |
|---|---|---|
| 명령어 | 1 | r 또는 R |
| 공백 | 1 | <SP> |
| 코드 | 2 | 읽고자 하는 환경변수의 코드 |
| 공백 | 1 | <SP> |
| 아이디 | 2 | 읽고자 하는 환경변수의 아이디 |
| 명령어 끝 | 1 | <CR> |

- 예) 제품아이디 [코드:0x02, 아이디:0x01]를 읽는 명령어.

| 읽기명령어 | 공백 | 코드 | | 공백 | 아이디 | | 읽기 명령어 끝 |
|---|---|---|---|---|---|---|---|
| r 또는 R | <SP> | 0 | 2 | <SP> | 0 | 1 | <CR> |
| 0x72 또는 0x52 | 0x20 | 0x30 | 0x32 | 0x20 | 0x30 | 0x31 | 0x0D |

### 3.3.2 읽기 명령어 응답코드

- 읽기 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 읽기 명령이 성공한 경우. |
| | 3 | 801 | 읽기 명령이 실패한 경우. |
| | 3 | 802 | 읽기 명령은 성공했으나 요청한 코드 또는 아이디를 찾지 못 한 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

- 응답코드가 701<CR><LF> 인 경우에는 응답코드 끝 다음에 환경변수 내용과 환경변수 끝 코드가 출력됩니다.
- 다음은 예는 제품아이디 [코드:0x02, 아이디:0x01] 읽기 명령에 대한 응답입니다.

```
37 30 31 0D 0A 30 32 20 30 31 20 30 38 20 30 30   701..02 01 08 00
20 32 45 20 30 30 20 30 35 20 42 30 0D 0A          2E 00 05 B0..
```

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 읽기 명령 성공 |
| 응답코드 끝 | 2 | <CR><LF> | 응답코드 끝 |
| 환경변수 | 23 | 02<SP>01<SP>08<SP>00<SP> 2E<SP>00<SP>05<SP>B0 | 제품아이디 |
| 환경변수 끝 | 2 | <CR><LF> | 환경변수 끝 |

## 3.4 쓰기 명령어

### 3.4.1 쓰기 명령어 구조.

● 쓰기 명령어는 다음과 같이 3가지 명령어가 사용됩니다.

| 명령어 | 설명 |
|---|---|
| w 또는 W | 전송 받은 환경변수를 제품(ezTCP)의 SRAM 메모리에 임시로 저장합니다. |
| u 또는 U | 임시로 저장된 환경변수를 제품(ezTCP)의 FLASH 메모리에 저장하기 위해서 메모리를 잠금 해제합니다. |
| f 또는 F | 임시로 저장된 환경변수를 FLASH 메모리에 저장합니다.. |

☞ *f 또는 F 명령어를 사용하기 전에 반드시 u 또는 U 명령어로 FLASH 메모리를 잠금 해제해야 합니다.*

☞ *f 또는 F 명령어를 사용한 후에는 반드시 명령어에 대한 응답코드를 확인해야 합니다. 응답코드를 확인하기 전에 제품에 입력된 전원을 차단하거나 제품을 재시동하면, FLASH 메모리에 저장된 제품 환경변수가 지워질 가능성이 있습니다.*

● w 또는 W 명령어 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | w 또는 W |
| 공백 | 1 | <SP> |
| 코드 | 2 | 환경변수의 코드 번호 |
| 공백 | 1 | <SP> |
| 아이디 | 2 | 환경변수의 아이디 번호 |
| 공백 | 1 | <SP> |
| 환경변수 내용 | N | 환경변수 내용. **3.2 절을 참고하세요.** |
| 명령어 끝 | 1 | <CR> |

● 예)제품 IP 주소[코드:0x03, 아이디:0x00]에 192.168.0.168을 저장하는 명령

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | w 또는 W |
| 공백 | 1 | <SP> |

| 코드 | 2 | 03 |
|---|---|---|
| 공백 | 1 | <SP> |
| 아이디 | 2 | 00 |
| 공백 | 1 | <SP> |
| 환경변수 | 35 | 03<SP>00<SP>0C<SP>00<SP><br>C0<SP>A8<SP>00<SP>A8<SP><br>00<SP>00<SP>C5<SP>A6 |
| 명령어 끝 | 1 | <CR> |

● 환경변수의 내용은 다음과 같습니다

| 환경변수 | 내용 |
|---|---|
| 03 | 코드 |
| 00 | 아이디 |
| 0C 00 | 데이터 전체 길이(Little Endian) |
| C0 A8 00 A8 | 제품 IP 주소(Big Endian) : 192.168.0.168 |
| 00 00 | 패드 |
| C5 A6 | CRC |

● u 또는 U 명령어 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | u 또는 U |
| 공백 | 1 | <SP> |
| 잠금 해제 코드 | 9 | aa55<SP>cc33 |
| 명령어 끝 | 1 | <CR> |

☞ *FLASH 메모리 잠금 해제 코드는 항상 aa55<sp>cc33 9바이트입니다. 이 코드가 맞지 않으면 FLASH 메모리에 데이터를 저장할 수 없습니다.*

● f 또는 F 명령어 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | f 또는 F |
| 공백 | 1 | <SP> |
| 코드 | 4 | a5c3 |
| 명령어 끝 | 1 | <CR> |

☞ *FLASH 메모리에 저장 코드는 항상 a5c3 4바이트입니다. 이 코드가 맞지 않으면 FLASH 메모리에 저장되지 않습니다.*

☞ *f 또는 F 명령어를 사용한 후에는 반드시 명령어에 대한 응답코드를 확인해야 합니다.*

*응답코드를 확인하기 전에 제품에 입력된 차단하거나 제품을 재시동하면, FLASH 메모리에 저장된 제품 환경변수가 지워질 가능성이 있습니다.*

### *3.4.2* 쓰기 명령어 응답코드

- w 또는 W 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 쓰기 명령이 성공한 경우. |
| | 3 | 801 | 쓰기 명령이 실패한 경우. |
| | 3 | 802 | 쓰기 명령은 성공했으나 요청한 코드 또는 아이디를 찾지 못한 경우. |
| | 3 | 803 | 쓰기 명령으로 전송한 환경변수 내용이 잘 못 된 경우. CRC가 맞지 않으면 발생할 수 있습니다. |
| | 3 | 805 | 쓰기 명령으로 전송한 환경변수 내용이 너무 긴 경우. |
| | 3 | 806 | 쓰기 명령으로 전송한 환경변수 내용이 16진수가 아닌 경우. |
| | 3 | 807 | 쓰기 명령으로 전송한 환경변수 내용의 길이가 잘 못된 경우. |
| | 3 | 808 | 쓰기 명령을 요청한 코드, 아이디와 쓰기 명령으로 전송한 환경변수 내용의 코드, 아이디가 일치하지 않는 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

- u 또는 U 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 잠금 해제 명령이 성공한 경우. |
| | 3 | 801 | 잠금 해제 명령이 실패한 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |

| 응답코드 끝 | 2 | <CR><LF> | |
|---|---|---|---|

☞ *잠금 해제 코드가 aa55<SP>cc33 이 아니더라도 701 응답 메시지를 보냅니다. 하지만 정상적으로 쓰기 작동이 되지는 않습니다.*

● f 또는 F 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | FLASH에 저장 명령이 성공한 경우. |
| | 3 | 804 | FLASH에 저장 명령이 실패한 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

☞ *FLASH 메모리가 잠금 상태면 804응답코드를 반환합니다.*

☞ *f 또는 F 명령어를 사용한 후에는 반드시 명령어에 대한 응답코드를 확인해야 합니다. 응답코드를 확인하기 전에 제품에 입력된 전원을 차단하거나 제품을 재시동하면, FLASH 메모리에 저장된 제품 환경변수가 지워질 가능성이 있습니다.*

## 3.5 반향(echo) 명령어

### 3.5.1 반향(echo) 명령어 구조

- 반향(echo) 명령은 영문 e 또는 E 을 사용합니다. 반향(echo) 명령어의 인자는 사용자로부터 ezTCP로의 입력 문자 반향 사용 여부를 결정합니다.
- 반향(echo) 기능을 사용하면 ezTCP로 입력된 데이터를 사용자 장비로 받아볼 수 있습니다.
- 반향(echo) 명령어의 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | e 또는 E |
| 공백 | 1 | <SP> |
| 플래그 | 1 | 0 : 반향기능을 중지합니다.<br>1 : 반향기능을 시작합니다. |
| 명령어 끝 | 1 | <CR> |

'0'은 입력 문자 반향(echo)을 안 하며 제품 기본 설정 상태입니다. ezTCP로 입력된 문자를 사용자 장비로 반향(echo) 받으려면 '0' 이외의 값을 입력하십시오.

- 예) 반향(echo) 기능을 활성화 시키는 명령.

| 명령어 | 공백 | 플래그 | 명령어 끝 |
|---|---|---|---|
| e 또는 E | <SP> | 1 | <CR> |
| 0x65 또는 0x45 | 0x20 | 0x31 | 0x0D |

### 3.5.2 반향(echo) 명령어 응답코드

- 반향(echo) 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 반향(echo)명령이 성공한 경우 |
| | 3 | 801 | 반향(echo)명령이 실패한 경우 |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

# 3.6 리부팅 명령어

## 3.6.1 리부팅 명령어 구조

- 리부팅 명령은 영문 g 또는 G를 사용합니다.
- 명령어의 구조는 다음과 같습니다.

| 데이터 | 데이터길이 (바이트) | 값 |
|---|---|---|
| 명령어 | 1 | g 또는 G |
| 공백 | 1 | <SP> |
| 코드 | 1 | 0 |
| 명령어 끝 | 1 | <CR> |

**리부팅 명령을 입력받으면 제품(ezTCP)은 리부팅 됩니다.**

- 예) 시리얼 매니저 모드에서 제품(ezTCP)을 리부팅.

| 명령어 | 공백 | 코드 | 명령어 끝 |
|---|---|---|---|
| g 또는 G | <SP> | 0 | <CR> |
| 0x67 또는 0x47 | 0x20 | 0x30 | 0x0D |

## 3.6.2 리부팅 명령어 응답코드

- 리부팅 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 801 | 리부팅 명령이 실패한 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

- 리부팅 명령어는 정상적으로 수행되면 응답코드 없이 제품(ezTCP)은 리부팅 합니다.

# 3.7 제품(ezTCP) 정보 보기

## 3.7.1 제품(ezTCP) 정보 보기 명령어 구조

- 제품(ezTCP) 정보 보기 명령은 영문 c 또는 C 를 사용합니다.
  제품(ezTCP) 아이디와, MAC주소, 펌웨어 버전을 조회할 수 있습니다.
- 명령어의 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | c 또는 C |
| 공백 | 1 | <SP> |
| 코드 | 1 | 0 |
| 명령어 끝 | 1 | <CR> |

- 예) 시리얼 매니저 모드에서 제품(ezTCP) 정보 보기.

| 명령어 | 공백 | 코드 | 명령어 끝 |
|---|---|---|---|
| c 또는 C | <SP> | 0 | <CR> |
| 0x63 또는 0x43 | 0x20 | 0x30 | 0x0D |

## 3.7.2 제품(ezTCP) 정보 보기 명령어 응답코드

- 제품(ezTCP) 정보 보기 명령에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 명령이 성공한 경우. |
| | 3 | 801 | 명령이 실패한 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

- 명령이 성공한 경우, 응답코드에 이어서 위에서 설명한 제품(ezTCP) 정보 데이터와 <CR><LF>이 수신 됩니다.
- 제품(ezTCP) 정보는 총 16바이트이며 다음과 같은 구조로 되어 있습니다.

☞ *제품(ezTCP) 정보는 환경변수와 달리 코드, 아이디, 길이, CRC가 없으며 연속된 16바이트 데이터가 ASCII데이터로 변환되어 수신됩니다.*

| 데이터 | 데이터길이(바이트) |
|---|---|
| 제품(ezTCP) 아이디 | 1 |
| 펌웨어 버전 (Major) | 1 |

| | |
|---|---|
| 펌웨어 버전 (Minor) | 1 |
| 펌웨어 버전 (Revision) | 1 |
| Pad | 4 |
| MAC 주소 | 6 |
| Pad | 2 |

☞ *Revision은 0부터 시작하고, 0부터 A로 표시하면 됩니다.*

● 다음은 제품(ezTCP) 정보 보기 예 입니다.

```
Request: 2009-03-11 오전 11:44:45.534125064
 63 20 30 0D                                        c 0.
Answer: 2009-03-11 오전 11:44:46.081000064 (+0.0000000000 seconds)
 37 30 31 0D 0A 31 35 20 30 31 20 30 30 20 30 31    701..15 01 00 01
 20 30 30 20 30 30 20 30 30 20 30 30 20 30 30 20     00 00 00 00 00
 33 30 20 46 39 20 30 41 20 38 30 20 32 45 20 34    30 F9 0A 80 2E 4
 34 20 38 31 0D 0A                                  4 81..
```

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 명령성공. |
| 응답코드 끝 | 2 | <CR><LF> | 응답코드 끝. |
| 제품(ezTCP) 아이디 | 2 | 15<SP> | CSW-H80 |
| 펌웨어 버전 | 6 | 01<SP>00<SP>01<SP> | 1.0B |
| 패드 | 8 | 00<SP>00<SP>00<SP>00<SP> | |
| MAC 주소 | 12 | 00<SP>30<SP>F9<SP> 0A<SP>80<SP>2E<SP> | 00:30:F9:0A:80:2E |
| 패드 | 3 | 44<SP>81 | |
| 데이터 끝 | 2 | <CR><LF> | 제품 정보 끝 |

# 3.8 PSK 생성 명령어

## 3.8.1 CSW-H80 의 PSK 생성 명령어 구조

- PSK 생성 명령은 영문 p 또는 P 를 사용합니다.

- SSID [ 코드:0x09, 아이디:0x01 ]와 WPA암호문 [ 코드:0x09, 아이디:0x04 ]을 가지고 PSK를 계산하여 반환합니다.
- **계산 값은 제품의 환경변수에 적용되지 않으므로 함수의 반환 값을 직접 입력해야 합니다.**
- PSK는 32바이트 이며, 약 30초의 시간이 소요 됩니다.

- 명령어의 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | p 또는 P |
| 공백 | 1 | <SP> |
| 코드 | 1 | 0 |
| 명령어 끝 | 1 | <CR> |

## 3.8.2 CSW-H80 의 PSK 생성 명령어 응답코드

- PSK 생성 명령어에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 명령이 성공하고 PSK 생성이 시작된 경우. |
| | 3 | 802 | SSID (코드 0x09, 아이디 0x01) 또는 WPA암호문 (코드 0x09, 아이디 0x04)이 환경 변수 안에 정의 되어 있지 않는 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

☞ *명령이 성공한 경우, 응답코드에 이어서 PSK 데이터와 <CR><LF>가 수신됩니다.*

- 다음은 PSK 생성 명령 예 입니다.
  SSID는 "sollae", PSK 암호문은 "1234567890123456789"이 이미 입력되어 있는 상태입니다.

```
Request: 2009-03-27 오후 1:33:54.328125064 (+17.3125000000 seconds)
70 20 30 0D                                              p 0.
Answer: 2009-03-27 오후 1:33:55.515625064 (+0.0000000000 seconds)
37 30 31 0D 0A 45 46 20 38 34 20 33 43 20 30 46   701..EF 84 3C 0F
20 34 36 20 35 35 20 34 37 20 42 35 20 37 44 20    46 55 47 B5 7D
41 44 20 45 31 20 35 39 20 30 36 20 31 38 20 37   AD E1 59 06 18 7
38 20 41 37 20 43 42 20 39 32 20 37 45 20 33 42   8 A7 CB 92 7E 3B
20 33 31 20 42 46 20 32 34 20 45 33 20 41 35 20    31 BF 24 E3 A5
33 30 20 37 38 20 32 32 20 46 42 20 39 37 20 30   30 78 22 FB 97 0
38 20 35 41 0D 0A                                 8 5A..
```

| 데이터 | 데이터길이 (바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 명령 성공. |
| 응답코드 끝 | 2 | <CR><LF> | 응답코드 끝 |
| WPA PSK | 95 | EF<SP>84<SP>3C<SP>0F<SP>46<SP>55<SP>47<SP>B5<SP>7D<SP>AD<SP>E1<SP>59<SP>06<SP>18<SP>78<SP>A7<SP>CB<SP>92<SP>7E<SP>3B<SP>31<SP>BF<SP>24<SP>E3<SP>A5<SP>30<SP>78<SP>22<SP>FB<SP>97<SP>08<SP>5A | WPA PSK |
| 데이터 끝 | 2 | <CR><LF> | 데이터 끝 |

### 3.8.3 CSW-M83 / M85, CSC-H64 의 PSK 생성 명령어 구조

- PSK 생성 명령은 영문 p 또는 P 를 사용합니다.
- SSID [ 코드:0x09, 아이디:0x01 ]와 WPA암호문 [ 코드:0x09, 아이디:0x08 ]을 가지고 PSK를 계산합니다.
- **계산 값은 제품의 환경변수에 바로 적용이 됩니다.**
- PSK는 32바이트 이며, 약 5초의 시간이 소요 됩니다.

- 명령어의 구조는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 |
|---|---|---|
| 명령어 | 1 | p 또는 P |
| 공백 | 1 | <SP> |
| 코드 | 1 | 0 |
| 명령어 끝 | 1 | <CR> |

### 3.8.4 CSW-M83 / M85, CSC-H64 의 PSK 생성 명령어 응답코드

- PSK 생성 명령어에 대한 응답코드는 다음과 같습니다.

| 데이터 | 데이터길이(바이트) | 값 | 설명 |
|---|---|---|---|
| 응답코드 | 3 | 701 | 명령이 성공하고 PSK 계산이 끝난 후 환경변수에 저장이 완료된 경우. |
| | 3 | 802 | SSID (코드 0x09, 아이디 0x01) 또는 WPA암호문 (코드 0x09, 아이디 0x08)이 환경 변수 안에 정의 되어 있지 않는 경우. |
| | 3 | 809 | 명령어 구문이 잘 못된 경우. |
| 응답코드 끝 | 2 | <CR><LF> | |

# 4 프로그래밍 참고

## 4.1 CRC 계산

● CRC 계산은 아래의 함수와 같이 계산하면 됩니다.

```
unsigned short crc16(unsigned char *buf, int len)
{
	unsigned long crc = 0xffff0000L;
	int bit, byte;

	for(byte = 0; byte < len + 2; byte++)
	{
		if(byte < len)
			crc |= ((unsigned long)buf[byte] << 8);
		for(bit = 0; bit < 8; bit++)
			crc = (crc & 0x80000000L) ? (crc << 1) ^ 0x10210000L : (crc << 1);
	}
	return (unsigned short)(crc >> 16);
}
```

● 자료형

| 자료형 | 길이(비트) |
|---|---|
| unsigned short | 16bit |
| unsigned long | 32bit |
| unsigned char | 8bit |

● 함수 인자

| 함수 인자 | 방향 | 설명 |
|---|---|---|
| unsigned char *buf | 입력 | CRC를 계산할 데이터 |
| int len | 입력 | CRC를 계산할 데이터의 길이 |

● 함수 리턴 값

계산된 16비트의 CRC값을 리턴 합니다. CRC값은 Little-Endian을 사용합니다.

## 4.2 시리얼 매니저 데이터 만들기

- ezManager 프로그램에서 사용하는 데이터 만드는 함수는 다음과 같습니다.

```
#define ENV_HEAD_SIZE		4

struct data_head
{
	_u8 code;
	_u8 id;
	_u16 len;
};

int get_pad_size(int data_size)
{
	int pad_size = 0;
	int reminder = data_size % 4;

	switch(reminder)
	{
	case 0:
		pad_size = 2;
		break;
	case 1:
		pad_size = 1;
		break;
	case 2:
		pad_size = 0;
		break;
	case 3:
		pad_size = 3;
		break;
	}
	return pad_size;
}
```

```
void make_data(_u8* buf, _u8 code, _u8 id, _u8 *data, int data_size, int *len)
{
//+----0------+-----1-----+----2----+----3----+----------n-----------+--- 0~n ---+-----2------+
// | CODE   |     ID    |     LENGTH     |        DATA         |    PAD    |    CRC    |
//+-----------+-----------+--------------------+----------------------+------------+-----------+
        struct data_head head;
        _u16 crc;
        _u8* ptr;

        int pad_size        = 0;
        int len1            = 0;

        ptr = buf;

        head.code           = code;
        head.id             = id;

        pad_size  = get_pad_size(data_size);

        head.len  = ENV_HEAD_SIZE + data_size + pad_size + 2;

        // head
        memcpy(ptr, &head, ENV_HEAD_SIZE);
        ptr         += ENV_HEAD_SIZE;
        len1        += ENV_HEAD_SIZE;

        // data
        if ( data_size != 0 )
        {
                memcpy(ptr, data, data_size);
                ptr         += data_size;
                len1        += data_size;
        }

        // pad
        if ( pad_size != 0 )
        {
```

```
                for ( int I = 0 ; I < pad_size ; i++ )
                {
                        *ptr = 0x00;
                        ptr++;
                        len1++;
                }
        }

        // crc
        crc = crc16(buf, len1);
        memcpy(ptr, &crc, 2);
        ptr     +=2;
        len1    +=2;

        // length
        *len = len1;
}
```

- 자료형

| 자료형 | 길이(비트) |
|---|---|
| _u8 | 8bit |
| _u16 | 16bit |
| _u32 | 32bit |
| int | 32bit |

- 함수 인자

| 함수 인자 | 방향 | 설명 |
|---|---|---|
| _u8 *buf | 출력 | 변환된 데이터를 담을 버퍼 |
| _u8 code | 입력 | 환경변수 코드 |
| _u8 id | 입력 | 환경변수 아이디 |
| _u8 *data | 입력 | 환경변수 |
| int data_size | 입력 | 환경변수 길이 |
| int *len | 출력 | 변환된 데이터의 사이즈 |

# 5 주의사항

제품(ezTCP)에 환경변수 저장 시 잘못된 값을 저장하거나 "읽기전용" 또는 "사용하지 않음" 항목을 변경하는 경우 제품(ezTCP)이 정상작동하지 않을 수 있습니다. 환경변수 설정 시 주의하여 주십시오.

# 6 문서 변경 이력

| 작성일 | 버전 | 변경 내용 | 작성자 |
|---|---|---|---|
| 2008.07.29 | 0.5 | ○ 본 문서 최초 작성. | 김형준 |
| 2008.11.21 | 0.6 | ○ 제품 개발 내역 반영. | 김형준 |
| 2009.02.24 | 0.7 | ○ E, G 명령어 추가 일부 오타 수정. | 김형준 |
| 2009.02.27 | 1.0 | ○ 용어 정리 및 본 문서 릴리즈. | 김형준 |
| 2009.03.11 | 1.1 | ○ 참고용 프로그램 코드 추가.<br>○ C 명령어 추가. | 김형준 |
| 2009.03.18 | 1.2 | ○ 환경변수 구조 중 비트필드 설명 추가. | 김형준 |
| 2009.03.27 | 1.3 | ○ WPA관련 설정값 설명 추가. | 김형준 |
| 2009.07.17 | 1.4 | ○ cod_listen 변수 추가. | 김형준 |
| 2009.08.31 | 1.5 | ○ 1. 3.2.2 ezTCP 작동에 필요한 옵션에 mac_id, cmt_id, flash 추가.<br>○ 전체적인 문서 내용 검토 및 수정 | 김형준 |
| 2009.12.15 | 1.6 | ○ 전체적인 문서 내용 검토 및 수정 | 김형준 |
| 2010.04.30 | 1.7 | ○ 문서 스타일 변경 | 이인 |
| 2011.02.07 | 1.8 | ○ 제품 아이디 내용 추가. | 김형준 |
| 2011.08.08 | 1.9 | ○ 새로운 환경변수들 추가. | 김형준 |
| 2012.01.16 | 2.0 | ○ 명령어에 대한 응답코드 추가.<br>○ F 명령어 사용에 대한 경고문구 추가.<br>○ 변경된 내용 반영 | 김형준 |
| 2012.08.24 | 2.1 | ○ 변경된 내용 반영 | 김형준 |
| 2012.11.16 | 2.2 | ○ 무선랜 옵션 passive 삭제. | 김형준 |
| 2013.01.11 | 2.3 | ○ Soft AP 내용 추가.<br>○ 코드 0x08, 아이디 0x03 추가. | 김형준 |
| 2016.01.25 | 2.4 | ○ 4. 프로그래밍 참고 내용 수정. | 김형준 |
| 2018.2.19 | 2.5 | ○ 3.2.9 제품 아이디 내용 추가. | 김형준 |