# Study 2회차 정리

```
// SERIAL_RX 가 SPEKTRUM 이 정의되어 있으면 정의되는데
// def.h 에 COPTERTEST == 4 가 쿼드콥터에 해당하므로
// SPEKTRUM 은 2048 로 값이 정의되어 있으므로 SERIAL_RX 가 활성화 됨
#if defined(SERIAL_RX)
  if (spekFrameFlags == 0x01) readSerial_RX();
#endif
```

수신기와 통신하는 부분을 알아보자 SERIAL_RX가 define 되어 있을 때 readSreial_RX()함수를 호출 하고 있다. 이제 SERIAL_RX가 정의 되었는지 살펴보자.

```
#if defined(SPEKTRUM) || defined(SBUS) || defined(SUMD)
  #define SERIAL_RX
#endif
```

def.h 파일에서 SERIAL_RX가 정의되어있는 부분을 보니 SPEKTRUM, SBUS, SUMD세개중 하나만 정의 되어있어도 정의 된다는 것을 알 수 있다.

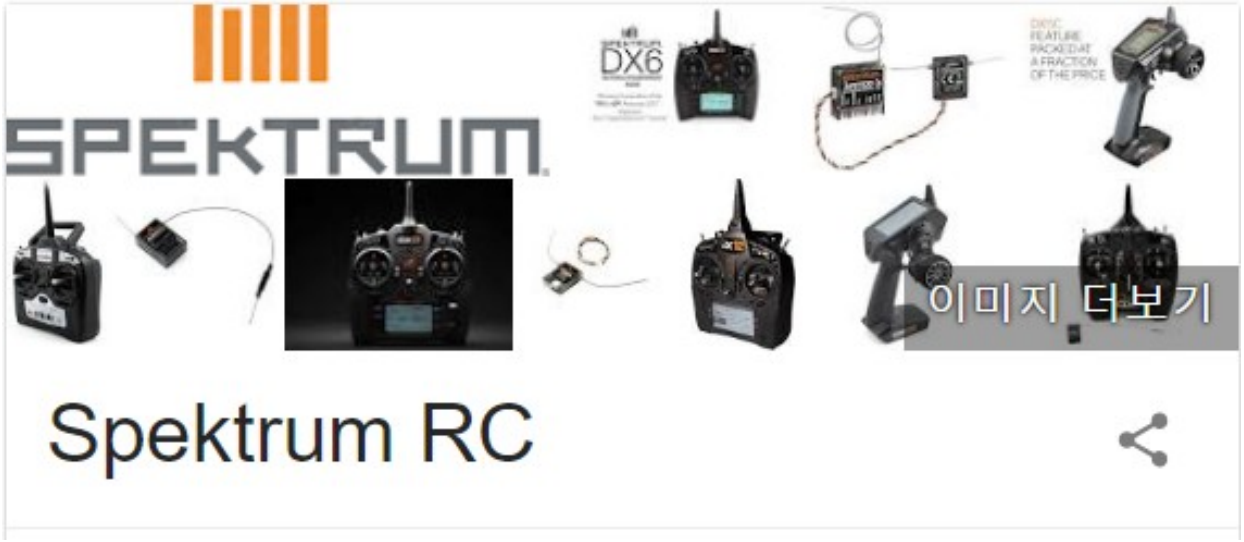
```
#elif COPTERTEST == 4
  #define QUADX
  #define CRIUS_SE
  #define SPEKTRUM 2048
  #define LED_RING
  #define GPS_SERIAL 2
  #define NMEA
  #define LOG_VALUES 2
  #define LOG_PERMANENT
  #define LOG_PERMANENT_SERVICE_LIFETIME 36000
```

```
#elif COPTERTEST == 7
  #define HELI_120_CCPM
  #define YAW_COLL_PRECOMP 15
  #define YAW_COLL_PRECOMP_DEADBAND 130
  #define NANOWII
  #define FORCE_ACC_ORIENTATION(X, Y, Z)
  #define FORCE_GYRO_ORIENTATION(X, Y, Z)
  #define A32U4_4_HW_PWM_SERVOS
  #define SERVO_RFR_RATE  200    // 200 for
  #define SERVO_PIN5_RFR_RATE  165    // In
  #define SPEKTRUM 1024
  #define BUZZER
```

SPEKTRUM은 def.h 파일에 COPTERTEST가 4일때와 7일때로 다르게 정의되어있다. 우리는 쿼드 콥터이므로 COPTERTEST가 4일때이므로

SPEKTRUM은 2048로 정의된다.

그렇다면 SPEKTRUM은 무엇을 의미할까? 다음 슬라이드를 통해 알아보자.




SPEKTRUM을 검색해보니 쿼드콥터 조종기등 여러가지 종류의 조종기를 파는 회사 였다.

SPEKTRUM에 관한 문서를 찾아보다가 2개를 발견 할 수 있었다.

하나는 SPEKTRUM 회사에서 제공해주는 문서이고, 나머지 다른 하나는 개인이 SPEKTRUM회사에서 제공해주는 문서를 정리해놓은 문서 였다.

AM, FM, PCM and now DSM. Spektrum's newly released DSM (Digital Spectrum Modulation) system advances RC radio technology to the next generation. Based on an optimized version of Direct Sequencing Spread Spectrum, DSM offers pure digital control providing an impenetrable radio link that's immune to all types of interference. And with 4096 bit resolution and 5.6ms response time, the DSM system offers seemingly infinite accuracy and instantaneous response for a totally connected driving experience.

문서를 찾아보던 중에 DSM이 뭔지에 대해 의문을 가지고 있었는데 SPEKTRUM 홈페이지에 나와 있었다.

첫번째 밑줄친 부분을 보면 SPEKTRUM사에서 RC radio 시스템에서 사용하던 AM,FM,PCM 방식말고 새로 DSM 방식을 만들었다는 것을 알 수 있으며 DSM이 Digital Spectrum Modulation의 약자인것과 4096 bit의 resolution을 가진 것을 알 수 있다.

SPEKTRUM사는 DSM 이후에 DSM2와 DSMX를 발표하였다.

DSM SYSTEM SPECIFICATIONS

- Frequency Band 2.400–2.4835GHz
- Channels 79
- Channel Spacing 1MHz
- Range 3000 ft
- Latency 5.6 ms
- Resolution/Channel 4096 steps

DSM 시스템에 대해서 간략하게 나와있다.

**SPEKTRUM1024**
1024 resolution을 갖는 PROTOCOL로써 Spektrum의 DSM2 22ms 방식에서만 사용한다.

**SPEKTRUM2048**
2048 resolution을 갖는 PROTOCOL로써 Spektrum의 DSM2 22ms를 제외하고는 모두 2048을 사용한다.

드디어 우리가 궁금해하던 SPEKTRUM 2048의 정체를 알게 되었다. 위의 글에서 나와 있듯 SPEKTRUM 1024는 DSM2 22ms 방식에서만 사용하고 SPEKTRUM 2048는 DSM 22ms를 제외하고 즉 11ms에서 사용한다는 것이다.

우리는 SPEKTRUM 2048이였으니 DSM2 11ms 이다. 11ms와 22ms가 무슨 의미인지 다음 슬라이드에서 설명 하겠다.

11ms 나 22ms 마다 16-byte data packet를 전송한다.
UART 전송 속도
 . BaudRate  : 125000 bps or 115200 bps
 . Data bits : 8 bits
 . Parity    : No Parity
 . Stop bits : 1 stop


전 슬라이드에서 말했던 11ms의 의미는 Receiver에서 출력되는 16 byte data packet이 11ms 마다 UART 통신으로 전송된다는 의미였던 것이다.

그러면 이제 저 16byte data packet에 어떠한 data가 들어가 있는지 알아보기 전에 SPEKTRUM사에서 제공되는 문서를 보고 가자.

To put a receiver into bind mode, within 200ms of power application the host device needs to issue a series of falling pulses. The number of pulses issued selects which bind types will be accepted from the transmitter. Selecting the 11ms mode automatically allows either 11ms or 22ms protocols to be used. Selecting DSMX will automatically allow DSM2 to be used if the transmitter is unable to

**DSMX Bind Modes:**

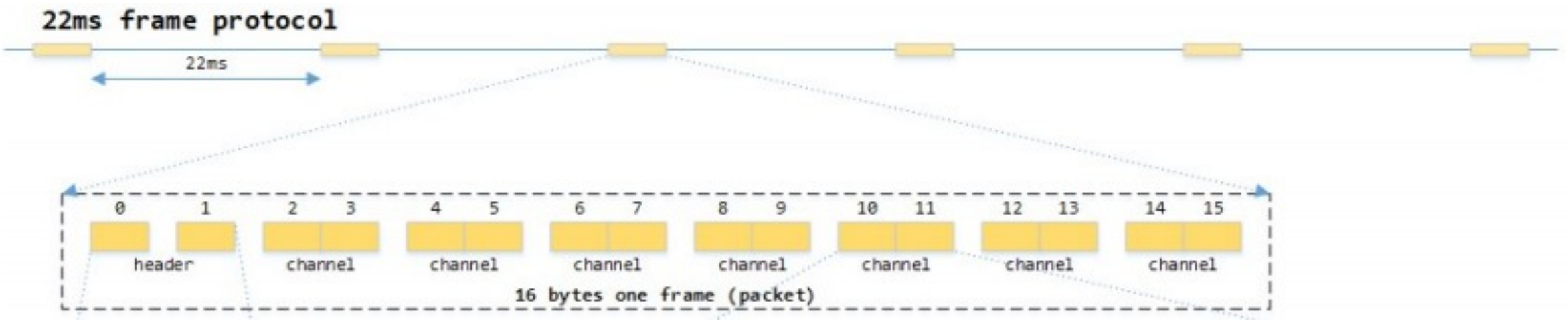
| Pulses | Mode | Protocol Type |
|---|---|---|
| 7 | Internal | DSMx 22ms |
| 8 | External | DSMx 22ms |
| 9 | Internal | DSMx 11ms |
| 10 | External | DSMx 11ms |

**DSM2 Bind Modes (not recommended):**

| Pulses | Mode | Protocol Type |
|---|---|---|
| 3 | Internal | DSM2 22ms |
| 4 | External | DSM2 22ms |
| 5 | Internal | DSM2 11ms |
| 6 | External | DSM2 11ms |

밑줄 친 부분을 읽어보면 수신기에 전원이 공급되고 200ms 이내에 host에서 falling pulse를 발생시키면 옆에 나와있듯이 Pulse 수에 따라 Bind Mode를 선택 할 수 있다.

우리는 Internal mode에 protocol Type이 DSM2 11ms이므로 Pulse 수는 5개를 발생시켜야 한다.




22ms 마다 16 Byte data packet을 보내는 protocol을 예로 들어 살펴보자 참고로 우리는 11ms마다 data packet을 보내는 protocol 이다.

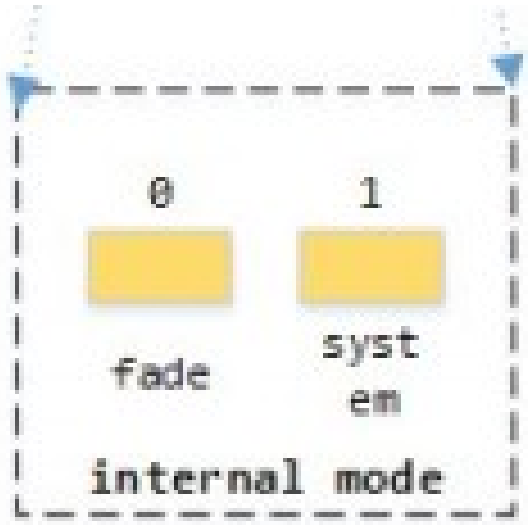
16Byte data 내용을 살펴보면 앞의 두 바이트 0,1 중에서 가장 먼저오는 Byte는 사용될 Protocol 및 data forma을 알 수 있는 System type이 들어간다 그리고 두번째 Byte에는 잃어버린 fram 개수를 표시하고

나머지 14Byte에는 data 가 들어간다.

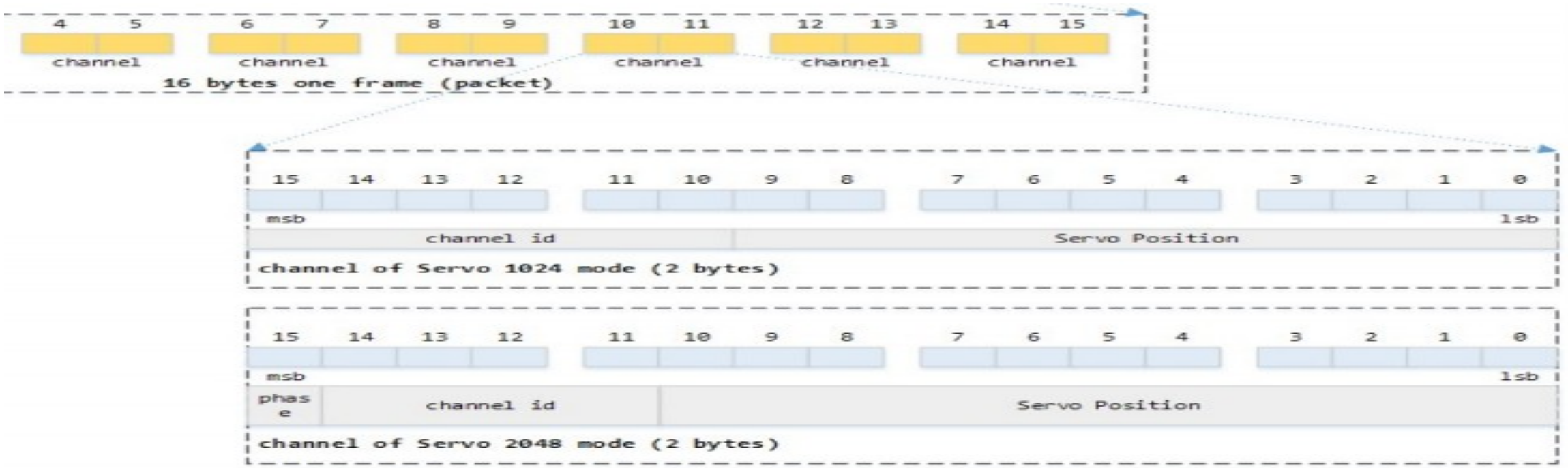
다음 슬라이드에서 그림과 Message Structure을 보며 이해해보자.

## 8.3 Message Structure Internal Remote

```
typedef stuct
{
        UINT8 fades;
        UINT8 system;
        UINT16 servo[7];
} INT_REMOTE_STR;
```




왼쪽은 16Byte Data packet Structure 이고 오른쪽은 전 슬라이드에서 말했던 16Byte data packet의 맨 앞 두 바이트이다.




앞에서 protocol 과 data format등을 정하였으므로 이제 data format 이 어떻게 생겼는지 알아보자 위의 그림에서 우리는 2048 mode이므로 두번째 그림을 보아야한다.

1개의 16byte packet 안에서 14byte가 data를 가지고 있는데 이 data는 2byte 단위로 채널이 나눠져 있고 한 채널안에는 phase,channel id, servo Position을 알려주는 정보가 담겨있다.

```
    SBUS_MID_OFFSET == 988 */
rcValue[0]  = ((sbus[1]|sbus[2]<< 8) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[1]  = ((sbus[2]>>3|sbus[3]<<5) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[2]  = ((sbus[3]>>6|sbus[4]<<2|sbus[5]<<10) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[3]  = ((sbus[5]>>1|sbus[6]<<7) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[4]  = ((sbus[6]>>4|sbus[7]<<4) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[5]  = ((sbus[7]>>7|sbus[8]<<1|sbus[9]<<9) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[6]  = ((sbus[9]>>2|sbus[10]<<6) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[7]  = ((sbus[10]>>5|sbus[11]<<3) & 0x07FF)/2+SBUS_MID_OFFSET; // & the other
//The following lines: If you need more than 8 channels, max 16 analog + 2 digital.
rcValue[8]  = ((sbus[12]|sbus[13]<< 8) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[9]  = ((sbus[13]>>3|sbus[14]<<5) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[10] = ((sbus[14]>>6|sbus[15]<<2|sbus[16]<<10) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[11] = ((sbus[16]>>1|sbus[17]<<7) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[12] = ((sbus[17]>>4|sbus[18]<<4) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[13] = ((sbus[18]>>7|sbus[19]<<1|sbus[20]<<9) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[14] = ((sbus[20]>>2|sbus[21]<<6) & 0x07FF)/2+SBUS_MID_OFFSET;
rcValue[15] = ((sbus[21]>>5|sbus[22]<<3) & 0x07FF)/2+SBUS_MID_OFFSET;
// now the two Digital-Channels
```

우리는 전 슬라이드에서 phase,channel id, servo Position 정보를 2byte로 수신기에서 보내준다는 것을 알았다. 그리고 수신기와는 UART통신을 하고 있다는 것을 알고있다.

하지만 UART통신은 한번에 1byte 씩보내는데 이는 수신기에서 보내는 16byte와 맞지않다. 그래서 바로 위의 코드처럼 16byte를 1byte로 변환하여 보내는 알고리즘이 필요하다.

# 참고

http://www.modulabs.co.kr/?module=file&act=procFileDownload&file_srl=18221&sid=2f1a4223f39de2f2e3bcea526428e039&module_srl=18196