BC7215 红外波形转换服务

本网络服务提供红外波形数据的转换功能，用户可以通过 HTTP POST 方法向特地服务地址上传红外波形数据，服务会将这些数据转换为可供 BC7215 芯片直接使用格式信息和发射指令，将返回的数据直接串口发送给 BC7215 芯片，即可令 BC7215 复制出与波形数据相同的红外信号。

服务地址

```
http://converter.bitcode.com.cn:17215/
https://converter.bitcode.com.cn:17216/
```

数据格式

输入数据格式

- **方法**：HTTP POST
- **数据类型**：二进制数据(application/octet-stream)
- **数据内容**：表示红外波形的脉冲时间，单位为微秒(us)
- **数据格式**：每个数据为 32 位无符号整数（Little-endian 格式）

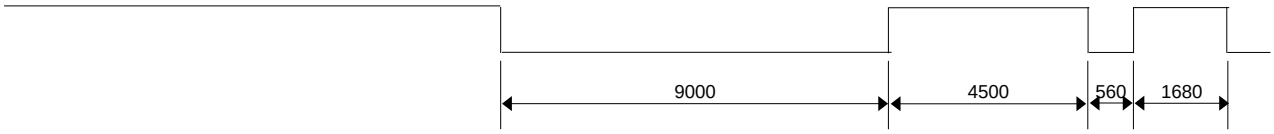
输入的数据格式与 Google 的 ConsumerIR 的数据格式兼容，唯一区别在于本服务输入数据为 32 位无符号整数，而 ConsumerIR 中为 32 位有符号整数。但因 IR 的数据不会出现负数，因此可以兼容通用。在 ConsumerIrManager 类的 transmit()方法中，

```
transmit(int carrierFrequency, int[] pattern);
```

其中 pattern 部分，即是需要输入到本服务的数据。

为便于不熟悉 ConsumerIR 协议的用户理解，下面详细解释如下:

本服务的输入数据，为红外信号的波形数据。例如，红外波形前两个完整脉冲为低电平 9000us，高电平 4500us，低电平 560us，高电平 1680us，



则输入数据前 4 个数据应为 9000，4500，560，1680, 对应十六进制表示为 0x00002328, 0x00001194, 0x00000230, 0x00000690，最终字节排列顺序为

```
0x28 0x23 0x00 0x00 0x94 0x11 0x00 0x00 0x30 0x02 0x00 0x00 0x90 0x06 0x00 0x00
```

对于输入数据，长度要求在 32 字节(4 个完整脉冲) - 4096(512 个完整脉冲)字节之间，如果长度超过这个范围，服务将返回 400 代码，“非法数据长度”错误。

输出数据格式

- **数据类型**：二进制数据(application/octet-stream)
- **第一个字节**：8 位有符号整数，值>=0 时(最高位为 0)表示转换成功，其值为服务软件版本号；值<0 时表示转换失败。
- **其余数据**：8 位无符号整数，是用于使 BC7215 芯片产生该波形数据的完整指令，包括 BC7215“加载格式信息”指令和“红外发射”指令。

“其余数据”前 35 个字节为 BC7215“加载格式信息”指令，以 0xF6 0x01 开头；后面紧接着是“红外发射”指令，以 0xF5 0x02 开头，红外发射指令长度依据红外波形数据不同而有长短变化。
除第一个字节外，所有数据均已经过字节填充编码处理(字节填充编码可能造成数据长度增加，如加载格式信息指令部分长度可能>35 字节，详情参见 BC7215 数据手册)，所以收到结果后可不经其它处理直接发送给 BC7215 芯片。

产生的格式数据默认设置为使用“37.5kHz 载波”，如果需要其他载波设置，请自行参考 BC7215 数据手册对格式信息部分数据进行设置。

示例程序

java 版：

这里假设波形数据由一个外部的 getIRData()函数获得。同时假设硬件电路连接同 BC7215 演示板，串口 CTS 连接 BC7215 BUSY 信号，DTR 连接 MOD 引脚。

```
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import com.fazecast.jSerialComm.SerialPort; // 使用 jSerialComm 库

// 假设 getIRData 是 ExternalIRDataProvider 类中的静态方法
import com.example.external.ExternalIRDataProvider;

public class InfraredWaveformConverter {

    public static void main(String[] args) {
        String url = "https://converter.bitcode.com.cn:17216/";
        byte[] inputData = ExternalIRDataProvider.getIRData();

        try {
            // 发送 HTTP POST 请求
            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection();
            con.setRequestMethod("POST");
            con.setDoOutput(true);
            DataOutputStream wr = new DataOutputStream(con.getOutputStream());
            wr.write(inputData);
            wr.flush();
```

```
            wr.close();

            int responseCode = con.getResponseCode();
            if (responseCode == 200) {
                InputStream in = new BufferedInputStream(con.getInputStream());
                byte[] response = in.readAllBytes();
                in.close();

                if (response[0] >= 0) {
                    System.out.println("转换成功，软件版本号：" + response[0]);

                    // 设置串口
                    SerialPort comPort = SerialPort.getCommPorts()[0]; // 假设使
用第一个串口
                    comPort.setComPortParameters(19200, 8,
SerialPort.TWO_STOP_BITS, SerialPort.NO_PARITY);
                    comPort.setFlowControl(SerialPort.FLOW_CONTROL_RTS_ENABLED |
SerialPort.FLOW_CONTROL_CTS_ENABLED);

                    if (comPort.openPort()) {
                        System.out.println("串口已打开");

                        // 将 DTR 信号置低电平，进入发射状态，并等待 2ms
                        comPort.setDTR(false);
                        Thread.sleep(2);

                        // 发送数据
                        OutputStream out = comPort.getOutputStream();
                        out.write(response, 1, response.length - 1); // 从第二个
字节开始发送
                        out.flush();

                        // 等待串口接收到 0x7a 字符
                        InputStream inSerial = comPort.getInputStream();
                        long startTime = System.currentTimeMillis();
                        boolean received = false;
                        while (System.currentTimeMillis() - startTime < 5000)
{ // 超时 5 秒
                            if (inSerial.available() > 0) {
                                int data = inSerial.read();
                                if (data == 0x7a) {
                                    received = true;
                                    break;
                                }
                            }
                        }
                        out.close();
                        inSerial.close();
                        comPort.closePort();

                        if (received) {
                            System.out.println("数据已发送并接收到确认字符，串口已关
闭");
                        } else {
                            System.out.println("未收到确认字符，发送超时");
                        }
                    } else {
                        System.out.println("无法打开串口");
                    }
```

```
                    // 保存返回数据到 .bin 文件
                    FileOutputStream fos = new FileOutputStream("output.bin");
                    fos.write(response);
                    fos.close();
                } else {
                    System.out.println("转换失败");
                }
            } else {
                System.out.println("HTTP 请求失败，响应代码：" + responseCode);
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Python 版：

假设波形数据由外部的 get_ir_data()函数获取，同时假设硬件电路连接同 BC7215 演示板，串口 CTS 连接 BC7215 BUSY 信号，DTR 连接 MOD 引脚。

```
import requests
import serial
import time

def get_ir_data():
    # 这里假设 get_ir_data 是一个外部由其他软件提供的函数
    # 在实际使用中，这个函数会由外部软件提供并返回所需的红外数据
    # 此处数据长度并不符合要求，用于转换会收到错误信息。
    return bytes([0x28, 0x23, 0x00, 0x00,  # 9000us
                  0x8C, 0x11, 0x00, 0x00,  # 4500us
                  0x30, 0x02, 0x00, 0x00,  # 560us
                  0x90, 0x06, 0x00, 0x00]) # 1680us

url = "https://converter.bitcode.com.cn:17216/"
input_data = get_ir_data()

response = requests.post(url, data=input_data)

if response.status_code == 200:
    output_data = response.content
    if output_data[0] >= 0:
        print("转换成功，软件版本号：", output_data[0])

        # 设置串口
        ser = serial.Serial()
        ser.port = 'COM1'  # 根据实际情况设置串口
        ser.baudrate = 19200
        ser.bytesize = serial.EIGHTBITS
        ser.parity = serial.PARITY_NONE
        ser.stopbits = serial.STOPBITS_TWO
        ser.rtscts = True
        ser.open()

        if ser.is_open:
            print("串口已打开")

            # 将 DTR 信号置低电平，进入发射状态，并等待 2ms
```

```
            ser.dtr = False
            time.sleep(0.002)

            # 发送数据
            ser.write(output_data[1:])  # 从第二个字节开始发送

            # 等待串口接收到 0x7a 字符，超时 5 秒
            start_time = time.time()
            received = False
            while time.time() - start_time < 5:
                if ser.in_waiting > 0:
                    data = ser.read(1)
                    if data == b'\x7a':
                        received = True
                        break
            ser.close()

            if received:
                print("数据已发送并接收到确认字符，串口已关闭")
            else:
                print("未收到确认字符，发送超时")
        else:
            print("无法打开串口")

        with open('output.bin', 'wb') as f:
            f.write(output_data)
    else:
        print("转换失败")
else:
    print("HTTP 请求失败，响应代码：", response.status_code)
```

手动测试

假设处于 linux 环境中，波形数据以二进制文件 input.bin 的形式提供，使用下面 curl 命令，可以手动测试转换服务，结果会保存在 output.bin 文件中。

```
curl -X POST https://converter.bitcode.com.cn:17216/ --data-binary @input.bin --output output.bin
```

使用下面命令可以查看 output.bin 的内容:

```
hexdump -C output.bin
```

* 随本说明书提供了 uPD6121.bin 文件，为标准 NEC 格式之红外波形文件，可用于使用手动方法测试转换服务。

数据存储

如果用户想保存转换的结果，对单个按键，直接保存除第一个字节外的所有数据即可。如果用户希望保存同一遥控器的多个按键，因为同一个遥控器的格式数据都是一致的（仅有测量误差所致的细微差别），因此可以将格式数据和按键数据分开存储，格式数据每种遥控器保存一份即可，达到节省存储空间的目的。