

# 用户自定义数据

## API使用说明

### 重置用户自定义数据

```
// 重置舵机的用户资料
FSUS_STATUS FSUS_ResetUserData(Usart_DataTypeDef *usart, uint8_t servoId);
```

- `usart` 舵机控制对应的串口数据对象 `Usart_DataTypeDef`
- `servoId` 舵机的ID

#### 使用示例

```
uint8_t servoId = 0;          // 连接在转接板上的串口舵机ID号
FSUS_ResetUserData(servoUsart, servoId);
```

### 读取用户自定义数据

```
// 读取数据
FSUS_STATUS FSUS_ReadData(Usart_DataTypeDef *usart, uint8_t servoId, uint8_t
address, uint8_t *value, uint8_t *size);
```

- `usart` 舵机控制对应的串口数据对象 `Usart_DataTypeDef`
- `servoId` 舵机的ID
- `address` 用户自定义数据的ID号
- `value` 读取到的数值
- `size` 读取到的数值的长度

#### 使用示例

```
// 读取用户自定义数据
// 数据表里面的数据字节长度一般为1个字节/2个字节/4个字节
// 查阅通信协议可知,舵机角度上限的数据类型是有符号短整型(Ushort, 对应STM32里面的int16_t),长
// 度为2个字节
// 所以这里设置value的数据类型为int16_t
int16_t value;
uint8_t dataSize;
// 传参数的时候, 要将value的指针强行转换为uint8_t
FSUS_ReadData(servoUsart, servoId, FSUS_PARAM_ANGLE_LIMIT_HIGH, (uint8_t
*)&value, &dataSize);
```

## 写入用户自定义数据

```
// 写入数据
FSUS_STATUS FSUS_WriteData(Usart_DataTypeDef *usart, uint8_t servoId, uint8_t
address, uint8_t *value, uint8_t size);
```

- `usart` 舵机控制对应的串口数据对象 `Usart_DataTypeDef`
- `servoId` 舵机的ID
- `address` 地址位
- `value` 写入的数据
- `size` 写入的数据的长度

### 使用示例

```
uint8_t servoId = 0;           // 连接在转接板上的串口舵机ID号
float angleLimitLow = -90.0;    // 舵机角度下限（默认值-135）
value = (int16_t)(angleLimitLow*10); // 舵机角度下限 转换单位为0.1度
statusCode = FSUS_WriteData(servoUsart, servoId, FSUS_PARAM_ANGLE_LIMIT_LOW,
(uint8_t *)&value, 2);
```

## 重置用户自定义数据

用户自定义数据重置。

```
#include "stm32f10x.h"
#include "usart.h"
#include "sys_tick.h"
#include "fashion_star_uart_servo.h"

// 使用串口1作为舵机控制的端口
// <接线说明>
// STM32F103 PA9(Tx) <----> 串口舵机转接板 Rx
// STM32F103 PA10(Rx) <----> 串口舵机转接板 Tx
// STM32F103 GND <----> 串口舵机转接板 GND
// STM32F103 V5 <----> 串口舵机转接板 5V
// <注意事项>
// 使用前确保已设置usart.h里面的USART1_ENABLE为1
// 设置完成之后，将下行取消注释
Usart_DataTypeDef* servoUsart = &usart1;

// 使用串口2作为日志输出的端口
// <接线说明>
// STM32F103 PA2(Tx) <----> USB转TTL Rx
// STM32F103 PA3(Rx) <----> USB转TTL Tx
// STM32F103 GND <----> USB转TTL GND
// STM32F103 V5 <----> USB转TTL 5V（可选）
// <注意事项>
// 使用前确保已设置usart.h里面的USART2_ENABLE为1
Usart_DataTypeDef* loggingUsart = &usart2;

// 重定向C库函数printf到串口，重定向后可使用printf函数
int fputc(int ch, FILE *f)
{
```

```

while((loggingUsart->pUSARTx->SR&0x40)==0){}
/* 发送一个字节数据到串口 */
USART_SendData(loggingUsart->pUSARTx, (uint8_t) ch);
/* 等待发送完毕 */
// while (USART_GetFlagStatus(USART1, USART_FLAG_TC) != SET);
return (ch);
}

// 连接在转接板上的串口舵机ID号
uint8_t servoId = 0;
// 状态码
FSUS_STATUS statusCode;

int main (void)
{
    // 嘀嗒定时器初始化
    SysTick_Init();
    // 串口初始化
    Usart_Init();
    // 发送重置用户数据指令
    statusCode = FSUS_ResetUserData(servoUsart, servoId);
    printf("====reset user data====\r\n status code: %d\r\n", statusCode);
    if (statusCode == FSUS_STATUS_SUCCESS){
        printf("sucess\r\n");
    }else{
        printf("fail\r\n");
    }
}

// 死循环
while (1);
}

```

## 读取用户自定义数据

读取用户自定义数据 - 读取舵机角度上限。

```

#include "stm32f10x.h"
#include "usart.h"
#include "sys_tick.h"
#include "fashion_star_uart_servo.h"

// 使用串口1作为舵机控制的端口
// <接线说明>
// STM32F103 PA9(Tx) <----> 串口舵机转接板 Rx
// STM32F103 PA10(Rx) <----> 串口舵机转接板 Tx
// STM32F103 GND <----> 串口舵机转接板 GND
// STM32F103 V5 <----> 串口舵机转接板 5V
// <注意事项>
// 使用前确保已设置usart.h里面的USART1_ENABLE为1
// 设置完成之后，将下行取消注释
Usart_DataTypeDef* servoUsart = &usart1;

// 使用串口2作为日志输出的端口
// <接线说明>
// STM32F103 PA2(Tx) <----> USB转TTL Rx

```

```

// STM32F103 PA3(Rx) <----> USB转TTL Tx
// STM32F103 GND <----> USB转TTL GND
// STM32F103 V5 <----> USB转TTL 5V (可选)
// <注意事项>
// 使用前确保已设置usart.h里面的USART2_ENABLE为1
Usart_DataTypeDef* loggingUsart = &usart2;

// 重定向c库函数printf到串口, 重定向后可使用printf函数
int fputc(int ch, FILE *f)
{
    while((loggingUsart->pUSARTx->SR&0x40)==0){}
    /* 发送一个字节数据到串口 */
    USART_SendData(loggingUsart->pUSARTx, (uint8_t) ch);
    /* 等待发送完毕 */
    // while (USART_GetFlagStatus(USART1, USART_FLAG_TC) != SET);
    return (ch);
}

uint8_t servoId = 0; // 连接在转接板上的串口舵机ID号
FSUS_STATUS statusCode; // 状态码
float angleLimitHigh = 0; // 舵机角度上限

int main (void)
{
    // 嘀嗒定时器初始化
    SysTick_Init();
    // 串口初始化
    Usart_Init();

    // 读取用户自定义数据
    // 数据表里面的数据字节长度一般为1个字节/2个字节/4个字节
    // 查阅通信协议可知,舵机角度上限的数据类型是有符号短整型(ushort, 对应STM32里面的
    int16_t),长度为2个字节
    // 所以这里设置value的数据类型为int16_t
    int16_t value;
    uint8_t dataSize;
    // 传参数的时候, 要将value的指针强行转换为uint8_t
    statusCode = FSUS_ReadData(servoUsart, servoId, FSUS_PARAM_ANGLE_LIMIT_HIGH,
    (uint8_t *)&value, &dataSize);
    printf("====read user data=====\r\n status code: %d\r\n", statusCode);
    if (statusCode == FSUS_STATUS_SUCCESS){
        // 用户自定义表里面, 角度的单位是0.1度. 需要改写为度
        angleLimitHigh = value / 10.0;
        printf("read success, angle limit high: %f\r\n", angleLimitHigh);
    }else{
        printf("fail\r\n");
    }

    // 死循环
    while (1);
}

```

输出日志

```
=====read user data=====
status code: 0
read sucess, angle limit high: 135.000000
```

## 写入用户自定义数据

写入用户自定义数据 - 写入舵机角度上限与下限，并开启舵机角度限制开关。

```
#include "stm32f10x.h"
#include "usart.h"
#include "sys_tick.h"
#include "fashion_star_uart_servo.h"

// 使用串口1作为舵机控制的端口
// <接线说明>
// STM32F103 PA9(Tx) <----> 串口舵机转接板 Rx
// STM32F103 PA10(Rx) <----> 串口舵机转接板 Tx
// STM32F103 GND <----> 串口舵机转接板 GND
// STM32F103 V5 <----> 串口舵机转接板 5V
// <注意事项>
// 使用前确保已设置usart.h里面的USART1_ENABLE为1
// 设置完成之后，将下行取消注释
Usart_DataTypeDef* servoUsart = &usart1;

// 使用串口2作为日志输出的端口
// <接线说明>
// STM32F103 PA2(Tx) <----> USB转TTL Rx
// STM32F103 PA3(Rx) <----> USB转TTL Tx
// STM32F103 GND <----> USB转TTL GND
// STM32F103 V5 <----> USB转TTL 5V (可选)
// <注意事项>
// 使用前确保已设置usart.h里面的USART2_ENABLE为1
Usart_DataTypeDef* loggingUsart = &usart2;

// 重定向c库函数printf到串口，重定向后可使用printf函数
int fputc(int ch, FILE *f)
{
    while((loggingUsart->pUSARTx->SR&0x40)==0){}
    /* 发送一个字节数据到串口 */
    USART_SendData(loggingUsart->pUSARTx, (uint8_t) ch);
    /* 等待发送完毕 */
    // while (USART_GetFlagStatus(USART1, USART_FLAG_TC) != SET);
    return (ch);
}

uint8_t servoId = 0; // 连接在转接板上的串口舵机ID号
FSUS_STATUS statusCode; // 状态码
// 限位角度
float angleLimitHigh = 90.0; // 舵机角度上限 (默认值 135)
float angleLimitLow = -90.0; // 舵机角度下限 (默认值-135)
uint8_t angleLimitswitch = 0x01; // 0x01: 开启限位; 0x00: 关闭限位

uint16_t value;
int main (void)
```

```

{
    // 嘀嗒定时器初始化
    SysTick_Init();
    // 串口初始化
    Usart_Init();

    // 写入舵机角度上限
    value = (int16_t)(angleLimitHigh*10); // 舵机角度上限 转换单位为0.1度
    statusCode = FSUS_WriteData(servoUsart, servoId,
    FSUS_PARAM_ANGLE_LIMIT_HIGH, (uint8_t *)&value, 2);
    printf("write angle limit high = %f, status code: %d\r\n", angleLimitHigh,
    statusCode);

    // 写入舵机角度下限制
    value = (int16_t)(angleLimitLow*10); // 舵机角度下限 转换单位为0.1度
    statusCode = FSUS_WriteData(servoUsart, servoId, FSUS_PARAM_ANGLE_LIMIT_LOW,
    (uint8_t *)&value, 2);
    printf("write angle limit low = %f, status code: %d\r\n", angleLimitLow,
    statusCode);

    // 打开舵机角度限位开关，配置生效
    statusCode = FSUS_WriteData(servoUsart, servoId, FSUS_PARAM_ANGLE_LIMIT_LOW,
    &angleLimitswitch, 1);
    printf("enable angle limit mode, status code: %d\r\n", statusCode);

    while (1){
        // 控制舵机角度
        FSUS_SetServoAngle(servoUsart, servoId, 90.0, 2000, 0, 1);
        FSUS_SetServoAngle(servoUsart, servoId, -90.0, 2000, 0, 1);
        // 写入的舵机角度超出了预设的限位，查看效果
        // FSUS_SetServoAngle(servoUsart, servoId, 135.0, 2000, 0, 1);
        // FSUS_SetServoAngle(servoUsart, servoId, -135.0, 2000, 0, 1);
    }
}

```