

УТВЕРЖДАЮ

Генеральный директор

ООО «Лаборатория Микроприборов»

_____ А.С. Тимошенко
«__» _____ 20__ г.

МЭМС-АКСЕЛЕРОМЕТР МА-10

ПРОТОКОЛ

информационного взаимодействия с изделием
ЛМАП.402131.039Д1

Разработал

_____ С.А. Анчутин
«__» _____ 20__ г.

Нормоконтролер

_____ Н.А. Соломкина
«__» _____ 20__ г.

Содержание

Введение.....	3
Подключение	3
Интерфейс обмена.....	3
Представление данных	3
Способ передачи данных.....	4
Структура пакета данных.....	4

Введение

Цифровой МЭМС-акселерометр МА-10 предназначен для измерения проекции кажущегося ускорения на измерительную ось. МА-10 состоит из чувствительного элемента и преобразователя, размещенных в металлостеклянном корпусе. МА-10 содержит встроенный датчик температуры.

Подключение

Подключение МА-10 осуществляется в соответствии с таблицей 1.

№ вывода	Цепь	Описание
5	TxOUT	Линия данных, выход UART
9	RST	Сигнал перезагрузки (активный низкий уровень)
10	GND	Общий вывод
11	VDD	Цепь питания, 3.3В
15	SHLD	Выход на корпус
1-4, 6-8, 12-14	-	Не подключать

Таблица 1. Подключение МА-10

Напряжение питания МА-10 должно быть 3.3В \pm 5%.

Интерфейс обмена

Информационное взаимодействие с изделием МА-10 обеспечивается посредством цифрового последовательного асинхронного двухпроводного интерфейса UART. Параметры передачи данных: 8 бит данных, без бита четности, 1 стоп-бит и скорость обмена данными 115200 бит/с.

Уровни сигналов не более 3.3 В (VDD):

- логическая единица 2.4В..3.3В
- логический ноль 0В..1В.

Представление данных

Пакет данных представляет собой упакованную структуру. Типы данных:

uint8_t – беззнаковое целое число, 8бит

uint16_t – беззнаковое целое число, 32бит

uint32_t – беззнаковое целое число, 32бит

char – символы в формате ASCII, 8бит

float – число с плавающей точкой одинарной точности в формате IEEE 754, 32бит.

Способ передачи данных

МА-10 преобразует измеряемое линейное ускорение в цифровой код с частотой 90Гц. После этого МА-10 передает пакет данных с результатами преобразования по интерфейсу UART.

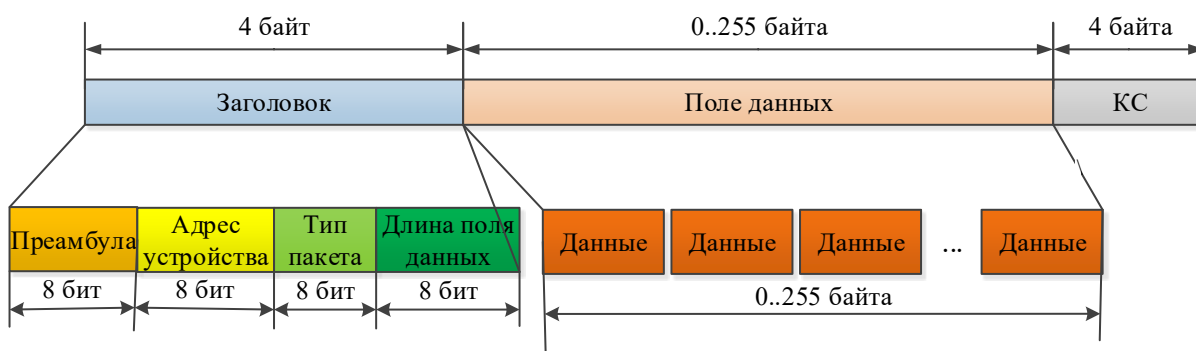
Структура пакета данных

Структура пакета: заголовок фиксированной длины, поле данных переменной длины и контрольная сумма (см. рисунок 1).

Контрольная сумма (32 бит) служит для проверки целостности переданного пакета данных. Расчет производится над массивом заголовков плюс поле данных. Значение добавляется к пакету в формате little-endian. Полином 32-й степени для расчета контрольной суммы имеет вид:

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$. Пример расчета контрольной суммы CRC32 на языке программирования C приведен ниже.

Рисунок 1. Структура пакета данных.



Описание структуры пакета на языке C и значения неизменных полей:

```
#pragma pack(1)
typedef struct
{
    uint8_t preamble; // = 255 (Заголовок)
    uint8_t address; // = 0 (Адрес устройства)
    uint8_t type; // = 19 (Тип пакета)
    uint8_t length; // = 16 (Длина пакета)
    uint8_t data[sizeof(ACCEL_T_data_packet)]; // Данные
    uint32_t crc32; // Контрольная сумма
} T_Basic_Packet;
#pragma pack()
```

Поле данных содержит байты структуры **ACCEL_T_data_packet** - результаты преобразования линейного ускорения и показания встроенного датчика температуры.

```

#pragma pack(1)
typedef struct
{
    float status;          // Статус
    float sample_counter; // Счетчик пакетов
    float temper;         // Температура, °C
    float accel;          // Ускорение, g
} ACCEL_T_data_packet;
#pragma pack()

```

Через 5 сек после включения МА-10 однократно отправляет два служебных пакета основное использование – передача серийного номера и настроек для ПО QInertSys.

Описание структуры служебного пакета (ID) на языке C:

```

#pragma pack(1)
typedef struct
{
    uint8_t preamble; // = 255 (Заголовок)
    uint8_t address;  // = 0 (Адрес устройства)
    uint8_t type;     // = 5 (Тип пакета)
    uint8_t length;   // = 43 (Длина пакета)
                    // Данные
    uint16_t boot_version; // = 4
    uint16_t software_version; // = 1
    uint32_t manufacture_date; // Дата производства в формате UTC
    char serial_num[16]; // Строка с серийным номером
    char name_of_device[16]; // = "МА-10" (Строка с названием)
    uint8_t device_mode; // = 0
    uint8_t status; // = 0
    uint32_t crc32; // Контрольная сумма
} T_DeVID_Packet;
#pragma pack()

```

Описание структуры служебного пакета (settigns) на языке C:

```

#pragma pack(1)
typedef struct
{
    uint8_t preamble; // = 255 (Заголовок)
    uint8_t address;  // = 0 (Адрес устройства)
    uint8_t type;     // = 39 (Тип пакета)
    uint8_t length;   // = 5 (Длина пакета)
                    // Данные
    uint8_t num_of_params; // = 4
    uint8_t type_of_params[4]; // = 0, 0, 0, 0
}

```

```

uint32_t crc32; // Контрольная сумма
} T_Settings_Packet;
#pragma pack()

```

Дата производства в формате UTC преобразована по правилам описанным в <https://www.epochconverter.com>

Функция на языке C для расчета контрольной суммы.

```

const uint32_t crc32_tabl[] =
{
    0x00000000, 0x77073096, 0xEE0E612C, 0x990951BA, 0x076DC419,
    0x706AF48F, 0xE963A535, 0x9E6495A3,
    0x0EDB8832, 0x79DCB8A4, 0xE0D5E91E, 0x97D2D988, 0x09B64C2B,
    0x7EB17CBD, 0xE7B82D07, 0x90BF1D91, 0x1DB71064,
    0x6AB020F2, 0xF3B97148, 0x84BE41DE, 0x1ADAD47D, 0x6DDDE4EB,
    0xF4D4B551, 0x83D385C7, 0x136C9856, 0x646BA8C0,
    0xFD62F97A, 0x8A65C9EC, 0x14015C4F, 0x63066CD9, 0xFA0F3D63,
    0x8D080DF5, 0x3B6E20C8, 0x4C69105E, 0xD56041E4,
    0xA2677172, 0x3C03E4D1, 0x4B04D447, 0xD20D85FD, 0xA50AB56B,
    0x35B5A8FA, 0x42B2986C, 0xDBBBC9D6, 0xACBCF940,
    0x32D86CE3, 0x45DF5C75, 0xDCD60DCF, 0xABD13D59, 0x26D930AC,
    0x51DE003A, 0xC8D75180, 0xBFDD06116, 0x21B4F4B5,
    0x56B3C423, 0xCFBA9599, 0xB8BDA50F, 0x2802B89E, 0x5F058808,
    0xC60CD9B2, 0xB10BE924, 0x2F6F7C87, 0x58684C11,
    0xC1611DAB, 0xB6662D3D, 0x76DC4190, 0x01DB7106, 0x98D220BC,
    0xEFD5102A, 0x71B18589, 0x06B6B51F,
    0x9FBFE4A5, 0xE8B8D433, 0x7807C9A2, 0x0F00F934, 0x9609A88E,
    0xE10E9818, 0x7F6A0DBB, 0x086D3D2D, 0x91646C97,
    0xE6635C01, 0x6B6B51F4, 0x1C6C6162, 0x856530D8, 0xF262004E,
    0x6C0695ED, 0x1B01A57B, 0x8208F4C1, 0xF50FC457,
    0x65B0D9C6, 0x12B7E950, 0x8BBEB8EA, 0xFCB9887C, 0x62DD1DDF,
    0x15DA2D49, 0x8CD37CF3, 0xFBD44C65, 0x4DB26158,
    0x3AB551CE, 0xA3BC0074, 0xD4BB30E2, 0x4ADFA541, 0x3DD895D7,
    0xA4D1C46D, 0xD3D6F4FB, 0x4369E96A, 0x346ED9FC,
    0xAD678846, 0xDA60B8D0, 0x44042D73, 0x33031DE5, 0xAA0A4C5F,
    0xDD0D7CC9, 0x5005713C, 0x270241AA, 0xBE0B1010,
    0xC90C2086, 0x5768B525, 0x206F85B3, 0xB966D409, 0xCE61E49F,
    0x5EDEF90E, 0x29D9C998, 0xB0D09822, 0xC7D7A8B4,
    0x59B33D17, 0x2EB40D81, 0xB7BD5C3B, 0xC0BA6CAD, 0xEDB88320,
    0x9ABFB3B6, 0x03B6E20C, 0x74B1D29A, 0xEAD54739,
    0x9DD277AF, 0x04DB2615, 0x73DC1683, 0xE3630B12, 0x94643B84,
    0x0D6D6A3E, 0x7A6A5AA8, 0xE40ECF0B, 0x9309FF9D,
    0x0A00AE27, 0x7D079EB1, 0xF00F9344, 0x8708A3D2, 0x1E01F268,
    0x6906C2FE, 0xF762575D, 0x806567CB, 0x196C3671,

```

```

    0x6E6B06E7, 0xFED41B76, 0x89D32BE0, 0x10DA7A5A, 0x67DD4ACC,
    0xF9B9DF6F, 0x8EBEEFF9, 0x17B7BE43, 0x60B08ED5,
    0xD6D6A3E8, 0xA1D1937E, 0x38D8C2C4, 0x4FDF252, 0xD1BB67F1,
    0xA6BC5767, 0x3FB506DD, 0x48B2364B, 0xD80D2BDA,
    0xAF0A1B4C, 0x36034AF6, 0x41047A60, 0xDF60EFC3, 0xA867DF55,
    0x316E8EEF, 0x4669BE79, 0xCB61B38C, 0xBC66831A,
    0x256FD2A0, 0x5268E236, 0xCC0C7795, 0xBB0B4703, 0x220216B9,
    0x5505262F, 0xC5BA3BBE, 0xB2BD0B28, 0x2BB45A92,
    0x5CB36A04, 0xC2D7FFA7, 0xB5D0CF31, 0x2CD99E8B, 0x5BDEAE1D,
    0x9B64C2B0, 0xEC63F226, 0x756AA39C, 0x026D930A,
    0x9C0906A9, 0xEB0E363F, 0x72076785, 0x05005713, 0x95BF4A82,
    0xE2B87A14, 0x7BB12BAE, 0x0CB61B38, 0x92D28E9B,
    0xE5D5BE0D, 0x7CDCEFB7, 0x0BDBDF21, 0x86D3D2D4, 0xF1D4E242,
    0x68DDB3F8, 0x1FDA836E, 0x81BE16CD, 0xF6B9265B,
    0x6FB077E1, 0x18B74777, 0x88085AE6, 0xFF0F6A70, 0x66063BCA,
    0x11010B5C, 0x8F659EFF, 0xF862AE69, 0x616BFFD3,
    0x166CCF45, 0xA00AE278, 0xD70DD2EE, 0x4E048354, 0x3903B3C2,
    0xA7672661, 0xD06016F7, 0x4969474D, 0x3E6E77DB,
    0xAED16A4A, 0xD9D65ADC, 0x40DF0B66, 0x37D83BF0, 0xA9BCAE53,
    0xDEBB9EC5, 0x47B2CF7F, 0x30B5FFE9, 0xBDBDF21C,
    0xCABAC28A, 0x53B39330, 0x24B4A3A6, 0xBAD03605, 0xCDD70693,
    0x54DE5729, 0x23D967BF, 0xB3667A2E, 0xC4614AB8,
    0x5D681B02, 0x2A6F2B94, 0xB40BBE37, 0xC30C8EA1, 0x5A05DF1B,
    0x2D02EF8D
};

```

```

uint32_t crc32_compute(const void *buf, unsigned long size)
{
    uint32_t crc=0;
    const uint8_t *p = (const uint8_t *)buf;
    crc = crc ^ 0xFFFFFFFFUL;

    while(size--)
        crc = crc32_tabl[(crc ^ *p++) & 0xFF] ^ (crc >> 8);

    return crc ^ 0xFFFFFFFFUL;
}

```