Описание протокола взаимодействия между платой АЦП farasystem_ain и персональным компьютером (ПК).

Далее по тексту в место “плата АЦП farasystem_ain” будет использоваться “устройство” или “ain”

Обмен информации между устройством и ПК происходит в виде запроса и обязательного ответа от устройства.

Формат запроса/ответа имеет одинаковую структуру, Табл. 1.

Табл. 1. Формат запроса/ответа.

Размер, байт	Описание поля
2	Команда (при запросе) / Статус (при ответе)
2	размер пакета
1016	Данные
2	Контрольная сумма CRC16

Порядок следования байт от старшего к младшему (big-endian).

Первое поле, размером 2 байта, при запросе от ПК содержит код команды. При ответе от устройства поле содержит статус устройства.

"Размер пакета" - количество байт начиная с команды и до контрольной суммы включительно

"Контрольная сумма" - вычисляется согласно алгоритму CRC16 начиная с первого байта кадра "Команда"/”Статус” и до последнего байта в поле "Данные". Размер блока для вычисления CRC16 равен полю "Размер пакета" - 2.

Коды ошибок / статус

При ответе устройство в случае сбоев может в поле статуса поместить значение об ошибке. Коды ошибок отражены в таб.2.

Табл. 2. Коды ошибок и статуса.

Код	Описание
0xFF01	Переполнение буфера приёма
0xFF02	Не совпала контрольная сумма CRC при приёме
0xFF03	Не достаточно параметров в команде запроса
0xFF04	Ошибка в команде запроса
0xAAAA	Команда успешно выполнена

Коды команд

Запрос версии - данные строка
Код: 0x0001
Данных в запросе нет

Запрос версии - бинарные данные
Код: 0x0081
Данных в запросе нет

Структура данных в ответе

Размер, байт	Описание поля	Тип данных
2	Версия платы	unsigned short
1	Версия ПО, старший байт (в запросе “Запрос версии - данные строка” - целая часть)	unsigned char
1	Версия ПО, младший байт (в запросе “Запрос версии - данные строка” - дробная часть)	unsigned short

Чтение времени
Код: 0x0002
Данных в запросе нет

Чтение конфигурации
Код: 0x0003
Данных в запросе нет

Чтение статуса буфера
Код: 0x0008
Данных в запросе нет

Ответ:

Размер, байт	Описание поля	Тип данных
2	Подтверждение команды (0xAAAA).	unsigned short int
4	Заполнение буфера Flash (количество выборок).	unsigned long
4	Размер выделяемой Flash памяти под буфер в Байтах.	unsigned long
2	Размер выборки в Байтах.	unsigned short int

2	CRC	unsigned short int

Энергонезависимый буфер, построенный на Flash-памяти организован по принципу FIFO - первый вошёл, первый вышел. Например, в буфер последовательно помешаются 3 выборки A, B и C. Причём, первой сохраняется A, потом B и далее C. При чтении сохраняется такой же порядок. Сперва будет прочитана выборка A, потом B и далее C. Т.е. при работе с буфером нет указателей и индексов, но сохраняется порядок помещения в буфер.

У Flash-буфера есть 3 параметра. Первый - это заполнение буфера. При ответе идёт во втором поле. Если в слове конфигурации выставлен бит ***adc_run,*** то начинается процесс измерений и выборки сохраняются во Flash-буфере, т.е. буфер будет заполнятся. Заполнение буфера будет идти до его максимального размера. На рис. 1 обозначено как “Use sample”.

Размер буфера определяется с помощью 2х параметров - размера физически выделенной Flash-памяти под буфер, в ответе передаётся в третьем поле, и размером самой выборки. Деление размера выделенной Flash-памяти на размер строки с отбрасыванием дробной части даёт размер буфера. На рис. 1 обозначено как “Total samples”.

Если буфер будет заполнен, т.е. не будет места для записи очередной выборки, то выставится бит ***flash_full***, сигнализирующий о полном заполнении буфера. На рис. 1 обозначено как “BufFlash”.

При полном заполнении буфера новые выборки не сохраняются. Есть 2 варианта освобождения буфера. Полная очистка, либо чтение с автоматическим удалением выборки.

Чтобы сделать полную очитку надо послать команду “Очистка буфера” (код 0x0010A). При этом все данные в буфере будут безвозвратно потеряны.

Чтобы сделать чтение с автоматическим удалением необходимо предварительно в конфигурацию записать в бит ***delet_sample*** значение “1”. В этом случае при выполнении команды “Чтение выборки из буфера” выборка будет удалена из буфера, а сам буфер освободится на 1 элемент.

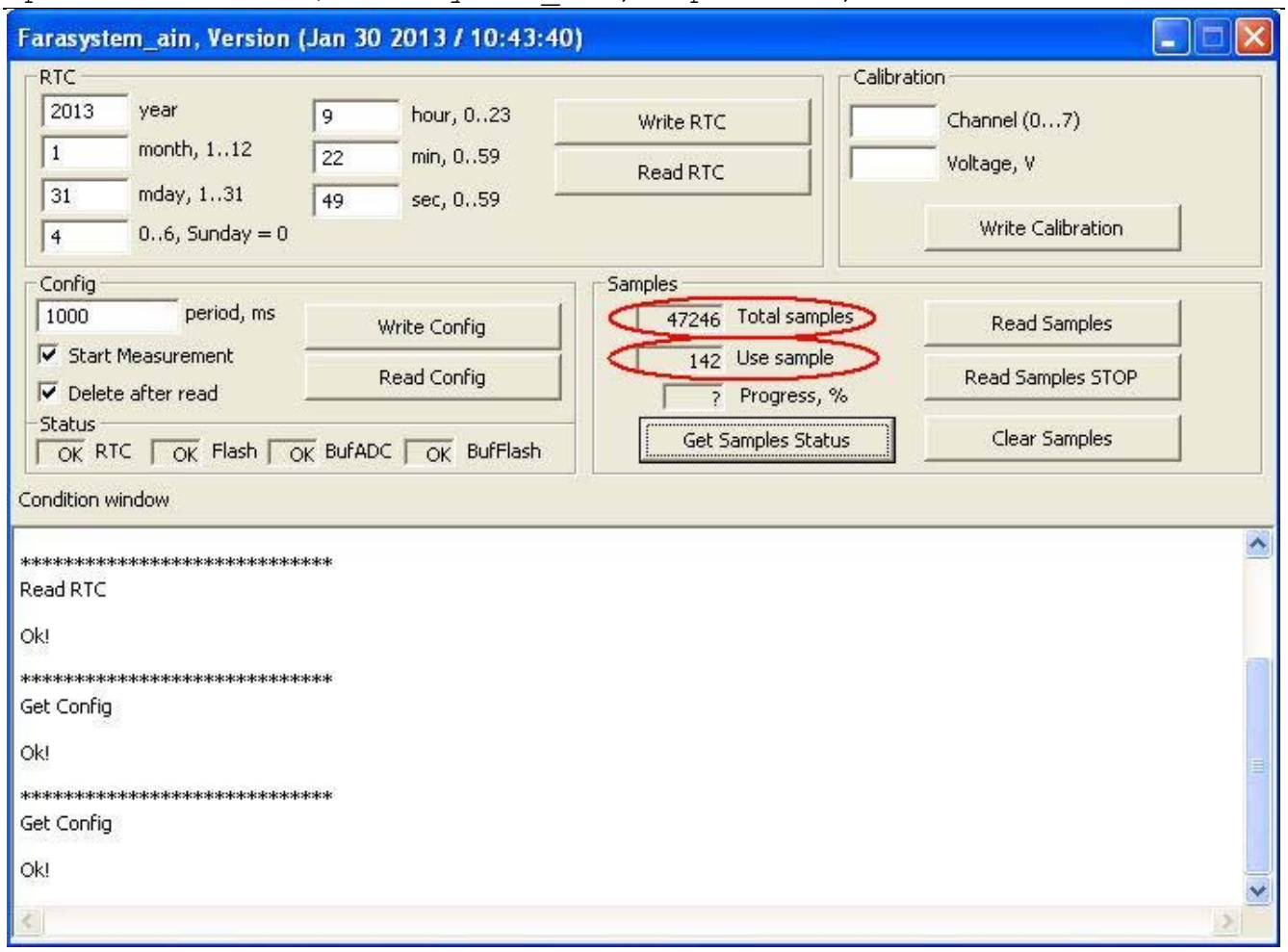

Рис.1. Статус буфера.

Чтение выборки из буфера
Код: 0x0009
Данных в запросе нет

Ответ:

Размер, байт	Описание поля	Тип данных
2	Год	unsigned short int
1	Месяц (1...12)	unsigned char
1	День (1...31)	unsigned char
1	Час (0...23)	unsigned char
1	Минута (0...59)	unsigned char
1	Секунда (0...59)	unsigned char
4	Канал 1	float
4	Канал 2	float
4	Канал 3	float
4	Канал 4	float
4	Канал 5	float
4	Канал 6	float
4	Канал 7	float
4	Канал 8	float

4	Канал 9 (температура)	float
2	CRC	unsigned short int

Чтение N-ой выборки из буфера

Код: 0x0011

Структура данных в запросе

Размер, байт	Описание поля	Тип данных
4	Номер выборки	unsigned long int

Нумерация выборок начинается с 1.

Если номер выборки в запросе больше заполнения буфера, то в ответе будет код ошибки 0xFF04.

Установка времени

Код: 0x0102

Структура данных в запросе

Размер, байт	Описание поля	Тип данных
2	Год	unsigned short int
1	Месяц (1...12)	unsigned char
1	День (1...31)	unsigned char
1	День недели (0..6, Воскресение = 0)	unsigned char
1	Час (0...23)	unsigned char
1	Минута (0...59)	unsigned char
1	Секунда (0...59)	unsigned char
1	Зима/Лето (0-Зима)	unsigned char

Установка конфигурации

Код: 0x0103

Структура данных в запросе

Размер, байт	Описание поля	Тип данных
2	Флаги	unsigned short int
4	Период выборок, мс	unsigned long int

Поле флагов описывается следующей структурой:

```
typedef struct {

    unsigned short int    adc_run           :1;   // запущено измерение
    unsigned short int    adc_work          :1;   // идёт сняти выборки
    unsigned short int    adc_cal           :1;   // режим калибровки
    unsigned short int    adc_full          :1;   // переполнение буфера АЦП
    unsigned short int    delet_sample      :1;   // удалить выборку после чтения
```

```
    unsigned short int    flash_err         :1;   // ошибка флэш памяти
    unsigned short int    rtc_err           :1;   // ошибка часов
    unsigned short int    flash_full        :1;   // буфер Flash-памяти заполнен

} AIN_FLAGS;
```

adc_run - этот бит запускает процесс измерений и сохранения выборок в памяти. Под выборкой понимается показания по 9-ти каналам и временной штамп. Если бит не установлен, то измерения не происходят и в памяти выборки не сохраняются. Для того, чтобы начать процесс измерений и сохранения выборок в памяти надо выставить данный бит, т.е. записать “1”.

adc_work – этот бит участвует во внутренней работе устройства. Он определяет момент начала снятия выборки, т.е. запускает процесс измерения выборки. При установке конфигурации может принимать любое значение.

adc_cal – этот бит запускает процесс калибровки. Данный бит выставляется при выполнении команды 0x0106 - “Установка калибровки”. Сбрасывается бит автоматически. При установке конфигурации данный бит должен принимать значение “0”.

adc_full – этот бит участвует во внутренней работе устройства. Сбрасывается и устанавливается автоматически. При установке конфигурации данный бит должен принимать значение “0”.

delet_sample – этот бит определяет, будет ли выборка удалена автоматически после чтения или нет. Если бит выставлен, то после чтения выборка удалится из памяти. Если бит сброшен, то после команды чтения выборки из буфера выборка останется в буфере.

flash_err – этот бит участвует во внутренней работе устройства. Если бит выставлен, то это сигнализирует о сбое во Flash памяти. Нормальная работа устройства возможна, когда бит имеет значение “0”. При установке конфигурации данный бит должен принимать значение “0”, т.е. необходимо сбросить бит.

rtc_err – этот бит участвует во внутренней работе устройства. Если бит выставлен, то это сигнализирует о сбое в часах. Например, это может произойти при смене батарейки. Нормальная работа устройства возможна, когда бит имеет значение “0”. При установке конфигурации данный бит должен принимать значение “0”, т.е. необходимо сбросить бит.

Установка калибровки

Код: 0x0106
Структура данных в запросе

Размер, байт	Описание поля	Тип данных
1	канал (0-7)	unsigned char
4	Напряжение, В	float

Очистка буфера
Код: 0x0010A
Данных в запросе нет

Приложение 1.
Пример программы расчёта CRC-16 CCITT на языке Си.

```
/*
  Name  : CRC-16 CCITT
  Poly  : 0x1021    x^16 + x^12 + x^5 + 1
  Init  : 0xFFFF
  Revert: false
  XorOut: 0x0000
  Check : 0x29B1 ("123456789")
  MaxLen: 4095 байт (32767 бит) - обнаружение
    одинарных, двойных, тройных и всех нечетных ошибок
*/
unsigned short Crc16(unsigned char *pcBlock, unsigned short len)
{
    unsigned short crc = 0xFFFF;
    unsigned char i;

    while (len--)
    {
        crc ^= *pcBlock++ << 8;

        for (i = 0; i < 8; i++)
            crc = crc & 0x8000 ? (crc << 1) ^ 0x1021 : crc << 1;
    }

    return crc;
}
```

Пример программы табличного (быстрого) расчёта CRC-16 CCITT на языке Си.

```
/*
  Name  : CRC-16 CCITT
  Poly  : 0x1021    x^16 + x^12 + x^5 + 1
  Init  : 0xFFFF
  Revert: false
  XorOut: 0x0000
  Check : 0x29B1 ("123456789")
  MaxLen: 4095 байт (32767 бит) - обнаружение
    одинарных, двойных, тройных и всех нечетных ошибок
*/
const unsigned short Crc16Table[256] = {
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
    0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
    0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
    0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
    0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
    0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
    0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
    0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
    0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
    0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
    0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
    0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
    0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};

unsigned short Crc16(unsigned char * pcBlock, unsigned short len)
{
    unsigned short crc = 0xFFFF;

    while (len--)
        crc = (crc << 8) ^ Crc16Table[(crc >> 8) ^ *pcBlock++];
```

```
    return crc;
}
```