



Модуль АЦП “USB-1402”.

Руководство по программированию.

Ревизия 1.0.

1. ПОРЯДОК РАБОТЫ С УСТРОЙСТВОМ.....	2
2. ФОРМАТ ДАННЫХ	2
3. ПРИМЕР РАБОТЫ С УСТРОЙСТВОМ.	3
3.1. ЗАПУСК СЕРВЕРА СБОРА ДАННЫХ.....	3
3.2. ФУНКЦИЯ ПОЛУЧЕНИЯ ОЧЕРЕДНОЙ ПОРЦИИ ДАННЫХ.....	4
3.3. ГЛАВНЫЙ ЦИКЛ	4
ПРИЛОЖЕНИЕ 1. СТРУКТУРА РАЗДЕЛЯЕМОГО БУФЕРА ПАМЯТИ.	5

Контакты:

<http://www.R-Technology.ru>

Info@R-Technology.ru

Sales@R-Technology.ru

Support@R-Technology.ru

- Общие вопросы

- Отдел продаж

- Техническая поддержка

1. Порядок работы с устройством.

Работа прикладного ПО с модулем АЦП USB-1402 осуществляется путём взаимодействия с сервером сбора данных *ADC1402_server.exe* через разделяемый буфер памяти в ОЗУ ПК.

Разделяемая буфер памяти состоит из двух областей: *область управления* и *область данных*. Полностью структура разделяемого буфера памяти описана в Приложении 1.

Через область управления происходит управление устройством: запуск/останов сбора данных и записи на диск, чтение флага готовности данных и статусов ошибок и т.д. Кроме того, в области управления содержится информация об устройстве (производитель, серийный номер и т.п.).

В область данных сервер сбора данных складывает непосредственно данные АЦП, поступающие от модуля USB-1402 по шине USB. Эта область разделена на N буферов, каждый из которых имеет длину S. Такая «многобуферность» связана с использованием сервером сбора данных механизма асинхронных запросов данных "overlapped I/O", при этом числа N и S выбираются сервером исходя из доступных системных ресурсов.

В начале работы прикладное ПО должно запустить сервер сбора данных *ADC1402_server.exe* с приоритетом реального времени.

Далее прикладное ПО запускает сбора данных путём записи в разделяемую память по адресу *SERVER_START* либо *SERVER_CYCLE*. Сбор данных может проходить в одном из 3х режимов:

- Однократный (сбор данных происходит до заполнения всей области данных и автоматически останавливается).
- Непрерывный.
- Непрерывный с записью в файл.

Далее прикладное ПО в цикле следит за флагом готовности данных по адресу *BUFFER_READY*, и как только видит готовность, может считывать очередную порцию поступивших с АЦП данных. Для этого ПО получает номер i готового буфера и количество готовых к считыванию байтов в этом i-м буфере.

Сразу после определения номера буфера и количества готовых байтов прикладное ПО должно сбросить флаг готовности данных.

Если запущен процесс записи данных на диск, в процессе записи прикладное ПО должно следить за статусом записи по адресу *WRITE_STATUS*. В случае, если статус не равен 0, произошла ошибка записи, код ошибки соответствует кодам ошибок Windows.

По окончании работы в *SERVER_CYCLE* записывается 0 и останавливается сервер сбора данных *ADC1402_server.exe*.

2. Формат данных

Данные представляют собой первичные 16-битные слова, младшие 14 бит в которых занимают данные АЦП (чётные слова - 1 канал, нечётные - 2-й), а старшие 2 бита - текущее состояние дискретных входов.

Соответствие кодов АЦП напряжению на входе:

0	-	-1 В
8192	-	0 В
16383	-	+1В

3. Пример работы с устройством.

Для иллюстрации принципа работы у устройством ниже приведены фрагменты листинга исходного текста примера USB-1402-example, написанного на языке Си в среде LabWindows CVI:

3.1. Запуск сервера сбора данных

```
/* Установим серверу сбора данных приоритет реального времени */
SetPriorityClass(GetCurrentProcess(), REALTIME_PRIORITY_CLASS);
dwPriClass = GetPriorityClass(GetCurrentProcess());

/* Запустим сервер сбора данных */
if (LaunchExecutableEx("ADC1402_server.exe", LE_HIDE, &g_exeHandle)) {
    printf("Cannot start ADC1402_server.exe\n");
    return 1;
}

/* Задержка 2 сек */
Delay(2.0);

/* Получим доступ к разделяемой памяти */
hMemMapFile = OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE, "AEshareMem");
if (!hMemMapFile) {
    printf("Device not found\n");
    return 2;
}

/* Получим указатель на начало разделяемой памяти*/
memPtr = (LPSTR)MapViewOfFile(hMemMapFile, FILE_MAP_ALL_ACCESS, 0, 0, 0);
if (!memPtr) {
    printf("MapViewOfFile failed\n");
    return 3;
}

/* Получим значения разделяемого буфера */
memPtrUI = (unsigned int *)memPtr; /* Указатель на область управления разделяемой памяти*/
memPtrUS = (unsigned short *)&memPtr[OUT_BUFFER]; // Указатель на начало области данных в
                                                    // разделяемой памяти*/
buffer_size = memPtrUI[BUFFER_SIZE]; /* Размер каждого из N буферов данных*/
n_buffers = memPtrUI[N_BUFFERS];      /* Количество N буферов данных*/

/* Запускаем сбор данных */
memPtrUI[SERVER_CYCLE] = SERVER_CYCLE_READ;
```

3.2. Функция получения очередной порции данных

```
static long get_data(unsigned short **data)
{
    long bytes, inp_ind;

    /* Ждём флаг готовности одного из N буферов данных */
    while (!memPtrUI[BUFFER_READY])
        Sleep(1);

    /* Получим индекс I готового буфера */
    inp_ind = memPtrUI[BUFFER_IDX];
    /* Получим количество готовых байт данных в буфере I с готовыми данными*/
    bytes = memPtrUI[BYTE_COUNTS + inp_ind];
    /* Получим указатель на буфер I*/
    *data = &memPtrUS[(buffer_size * inp_ind) / sizeof(short)];
    /* Сбросим флаг готовности данных */
    memPtrUI[BUFFER_READY] = 0;

    return bytes;
}
```

3.3. Главный цикл

```
for (;;) {
    /* Получение очередной порции данных от сервера */
    bytes = get_data(&data); /* bytes – кол-во готовых байт данных*/
    /* data - указатель на готовые данные */
    samples = bytes / (int)sizeof(short);
    count += samples / N_CHANNELS; /* Число сэмплов на канал */
    /* Обработываем данные - раскладываем по 2 каналам АЦП и перевод в Вольты */
    for (i = 0; i < samples; i++) {
        chan = i & 1; /* Определяем номер канала АЦП */
        itmp = data[i] + (chan ? ADC_OFFSET_A : ADC_OFFSET_B);
        tmp = (double)itmp / ADC_MAX_CODE; /* Напряжение в Вольтах */
    }
}
```

Приложение 1. Структура разделяемого буфера памяти.

Область управления				
Адрес	Имя	Тип	Доступ	Описание
0	VENDOR_ID	строка	r/o	Производитель устройства
80	PRODUCT_ID	строка	r/o	Идентификатор устройства
160	DESCRIPTOR	строка	r/o	Дескриптор устройства
240	S_N	строка	r/o	Серийный номер устройства
320	OUT_DIR	строка	r/w	Директория для записи файлов данных
2048	SERVER_PRIOR	int32	r/o	Приоритет сервера сбора данных
2052	SERVER_START	int32	r/w	1 - запуск однократного сбора данных, очищается после заполнения всех N буферов данных
2056	SERVER_CYCLE	int32	r/w	1 - запуск непрерывного сбора данных; 2 - запуск непрерывного сбора данных с записью в файл с названием вида ГГГГ_ММ_ДД_чч_мм_сс.bin, где "ГГГГ_ММ_ДД_чч_мм_сс" - системная дата/время на момент запуска (автоматически считывается сервером из ОС); 0 - останов сбора данных.
2060	SERVER_EXIT	int32	r/w	1 - прекращение работы сервера
2064	WRITE_STATUS	int32	r/o	Статус очередной операции записи в файл, см. коды ошибок Windows
2068	BUFFER_READY	int32	r/w	1 - готовность очередного буфера данных, очищается пользователем после обработки буфера
2072	BUFFER_SIZE	int32	r/o	Количество буферов данных N
2076	N_BUFFERS	int32	r/o	Размер буферов данных S
2080	BUFFER_IDX	int32	r/o	Индекс очередного готового буфера данных I (от 0 до N-1)
2112+I*4	BYTE_COUNTS	int32	r/o	Счётчики байтов для каждого из буферов данных
Область данных				
167936+(I*S)	OUT_BUFFER	unsigned short	r/o	Старт I-того буфера данных