

Библиотека LDSP для работы с устройствами АЦП/ЦАП

Иван Горинов, L-Card

Библиотека LDSP для работы с устройствами АЦП/ЦАП

Иван Горинов

Copyright © 2001 L-Card

Содержание

1. Введение	1
2. Использование библиотеки	2
Программирование	3
Последовательность работы с устройством	4
Открытие устройства	4
Работа с цифровыми линиями	4
Работа с одиночными отсчетами	5
Перевод значения отсчета АЦП в напряжение	5
Установка таблицы каналов	6
Запуск АЦП/ЦАП	6
Одновременный ввод/вывод	7
3. Функции	8
4. Структуры данных	29

Глава 1. Введение

LDSP - библиотека для работы с устройствами измерения данных, содержащих АЦП, ЦАП, цифровые линии (TTL). Используется для работы с платами L-Card в Linux, некоторые функции перенесены в Windows для E-440.

Интерфейс является универсальным для всех устройств L-Card, программы не требуют изменения при переходе с одного устройства на другое (например, ISA платы на PCI, или LPT на USB).

Глава 2. Использование библиотеки

Использование библиотеки LDSP в программах

Программирование

Прототипы функций и структуры данных описаны во включаемом файле **ldsp/ldsp.h**.

```
/* myprogram.c */  
  
#include <ldsp/ldsp.h>  
  
int main(char **argv)  
{  
    ...  
}
```

При сборке нужно добавить библиотеку опцией компоновщика **-LDSP**:

```
cc -o myprogram myprogram.c -LDSP
```

Последовательность работы с устройством

Открытие устройства

Имена устройств в Linux: `/dev/xdsp0`, `/dev/xdsp1`, ...

```
#include <ldsp/ldsp.h>

char *DeviceName = "/dev/xdsp0";
LDSP_HANDLE dev;
int st;

st = ldspOpen(&dev, DeviceName, LO_ADC);
if(st < 0)
{
    printf("Ошибка при открытии устройства: %s\n", DeviceName);
    exit(1);
}
```

Работа с цифровыми линиями

Ввод с цифровых линий - функция `ldspDigitalIn`.

```
int ttl;

st = ldspDigitalIn(dev, &ttl);
if(st < 0)
{
    puts("Ошибка при считывании!");
}
else
{
    printf("Цифровые линии: %d\n", ci.sample);
    for(i = 0; i < 32; i++)
    {
        printf(" Линия %2d: %d\n", i + 1, ttl & 1);
        ttl >>= 1;
    }
}
```

Каждый бит соответствует цифровой линии: бит 0 - IN1, 1 - IN2, ... 15 - IN16.

Работа с одиночными отсчетами

Для одиночного ввода/вывода не требуется устанавливать временные параметры и таблицу каналов.

```
sample_inf ci;

ci.mode = LCH_DIFFERENTIAL;
ci.input = 0;
ci.gain = 0;
x = ldspSampleIn(Dev, &ci);
if(x < 0)
{
    puts("Ошибка при вводе с АЦП!");
}
else
{
    printf("Значение сигнала: %d\n", ci.sample);
}
```

Перевод значения отсчета АЦП в напряжение

Значение отсчета возвращается в кодах АЦП. Эти коды могут быть переведены в напряжение, используя данные диапазона `ldsp_range`.

```
channel_inf ci;
ldsp_range range;
double fx, fa, fc;
int x;

x = ldspGetRange(Dev, &ci, &range);
fa = (range.umax - range.umin);
fa /= (range.max - range.min);
fc = (m * range.min) - range.umin;

x = ldspSampleIn(Dev, &ci);
if(x < 0)
{
    puts("Ошибка при вводе с АЦП!");
}
else
{
    printf("Код АЦП: %d\n", ci.sample);
    fx = (ci.sample * fa) + fc;
}
```

```
printf("Значение напряжения: %lf\n", fx);  
}
```

Установка таблицы каналов

Сначала устанавливается размер таблицы с помощью функции `ldspChannels`.

Затем каждый канал в таблице настраивается отдельным вызовом `ldspSetChannel`.

```
LDSP_HANDLE dev;  
channel_inf ci;  
int Channels;  
  
x = ldspChannels(dev, &Channels);  
for(i = 0; i < Channels; i++)  
{  
    ci.index = i;  
    ci.mode = LCH_DIFFERENTIAL;  
    ci.input = input[i];  
    ci.gain = 0;  
    x = ldspSetChannel(dev, &ci);  
    if(x < 0) break;  
}
```

Некоторые устройства поддерживают изменение режимов отдельных каналов в процессе непрерывного сбора без изменения общего числа каналов.

Запуск АЦП/ЦАП

Потоковый ввод/вывод данных начинается с запуска АЦП/ЦАП с помощью функции `ldspSetTrigger`.

Флаг `LDSP_ADC` запускает АЦП, `LDSP_DAC` - запускает ЦАП.

```
LDSP_HANDLE dev;  
int mask;  
  
mask = LDSP_MASK_ADC;  
x = ldspSetTrigger(dev, &mask);  
while(1)  
{  
    ldspRead(dev, buffer, fragment_size);  
}
```

```
    process_data(buffer);  
}
```

Одновременный ввод/вывод

Некоторые устройства позволяют одновременно вводить/выводить данные. Ввод и вывод могут быть запущены одновременно вызовом `ldspSetTrigger` с одновременно установленными `LDSP_MASK_ADC` и `LDSP_MASK_DAC`.

Флаг `LDSP_MASK_ADC` запускает АЦП, `LDSP_MASK_DAC` - запускает ЦАП.

```
LDSP_HANDLE dev;  
int mask;  
int x;  
  
ldspOpen(&dev, DevName, LDSP_ADC | LDSP_DAC | LDSP_NONBLOCK);  
  
mask = LDSP_MASK_ADC | LDSP_MASK_DAC;  
x = ldspSetTrigger(dev, &mask);  
while(1)  
{  
    mask = LDSP_MASK_ADC | LDSP_MASK_DAC;  
    x = ldspWait(dev, &mask, Timeout);  
    if(x < 0) break;  
    if(mask & LDSP_MASK_ADC)  
    {  
        ldspRead(dev, in_buffer, fragment_size);  
        process_data(in_buffer);  
    }  
    if(mask & LDSP_MASK_DAC)  
    {  
        get_output_data(out_buffer);  
        ldspWrite(dev, out_buffer, fragment_size);  
    }  
}
```

Глава 3. Функции

Функции библиотеки LDSP

ldspOpen — открытие устройства

ldspOpen

```
int ldspOpen(LDSP_HANDLE *Handle, char *DeviceName, int Flags);
```

Handle Указатель на переменную, в которую возвращается хендл устройства.

Name Имя файла устройства. Первому устройству соответствует имя "/dev/xdsp0".

Flags Режим доступа к устройству:

LO_ADC	Режим чтения данных с АЦП
LO_DAC	Режим вывода данных на ЦАП
LO_NONBLOCK	Режим ввода/вывода без ожидания

Функция используется для открытия устройства. После работы с устройством его нужно закрыть с помощью `ldspClose`.

Возвращает 0, отрицательное значение в случае ошибки.

ldspClose — закрытие устройства

ldspClose

```
int ldspClose (LDSP_HANDLE Device);
```

Device Хендл открытого устройства.

Закрывает устройство, открытое с помощью ldspOpen.

`ldspChannels` — установка числа каналов

`ldspChannels`

```
int ldspChannels(LDSP_HANDLE Device, int *n);
```

`Device` Хендл открытого устройства.

`n` Число элементов в таблице каналов.

Функция устанавливает размер таблицы каналов. Каждый элемент таблицы настраивается вызовом функции `ldspSetChannel`.

`ldspRead` — ввод данных с устройства

`ldspRead`

```
int ldspRead(LDSP_HANDLE Device, void *Data, int Length);
```

`Device` Хендл открытого устройства.

`Data` Буфер для ввода данных.

`Length` Размер блока данных в байтах.

Функция вводит блок данных с АЦП в режиме непрерывного ввода. Одиночные отсчеты вводятся функцией `ldspSampleIn`.

ldspWrite —вывод данных на устройство

ldspWrite

```
int ldspWrite(LDSP_HANDLE Device, void *Data, int Length);
```

Device Хендл открытого устройства.

Data Буфер с выводимыми данными.

Length Размер блока данных в байтах.

Функция выводит блок данных на ЦАП в режиме непрерывного ввода. Одиночные отсчеты выводятся функцией `ldspSampleOut`.

`ldspSetTrigger` — запуск/остановка потокового ввода

`ldspSetTrigger`

```
int ldspSetTrigger(LDSP_HANDLE Device, int *Mask);
```

Device Хендл открытого устройства.

Mask Маска потоков - комбинация флагов:

`LDSP_MASK_ADC` Запуск ввода данных с АЦП

`LDSP_MASK_DAC` Запуск вывода данных на ЦАП

Функция запускает или останавливает потоковый ввод/вывод на устройстве. Если маска потоков равна 0, ввод/вывод останавливается. Одиночные отсчеты выводятся функцией `ldspSampleOut`.

Вызов функции необходим только для потокового ввода данных, она не нужна при работе с одиночными отсчетами.

ldspMapBuffer —отображение буфера

ldspMapBuffer

```
void *ldspWrite(LDSP_HANDLE Device, int Stream, long Offset, long Size);
```

Device Хендл открытого устройства.

Stream Номер потока: 0 - вывод на ЦАП, 1 - ввод с АЦП.

Offset Смещение от начала буфера.

Size Длина отображаемой области в байтах.

Функция отображает буфер в адресное пространство процесса.

Возвращает указатель на отображенный буфер, 0 в случае ошибки.

ldspSetChannel —установка параметров канала

ldspSetChannel

```
int ldspSetChannel(LDSP_HANDLE Device, channel_inf ci);
```

Device Хендл открытого устройства.

ci Информация о канале - структура channel_inf. Поле *index* определяет индекс в таблице каналов.

Функция изменяет элемент с номером *index* в таблице каналов. Индекс первого элемента равен 0. Перед установкой элементов таблицы нужно задать ее размер функцией ldspChannels.

Если *index* = -1, функция просто вычисляет поле *channel_code*.

`ldspSampleIn` — ввод одиночного отсчета АЦП

`ldspSampleIn`

```
int ldspSampleIn(LDSP_HANDLE Device, channel_inf ci);
```

`Device` Хендл открытого устройства.

`ci` Информация о канале - структура `channel_inf`.

Функция считывает одиночный отсчет с канала АЦП. Значение отсчета возвращается в поле `ci.sample`.

ldspSampleOut – вывод одиночного отсчета на ЦАП

ldspSampleOut

```
int ldspSampleOut(LDSP_HANDLE Device, channel_inf ci);
```

Device Хендл открытого устройства.

ci Информация о канале - структура channel_inf.

Функция устанавливает напряжение на канале ЦАП. Значение отсчета передается в поле *ci.sample*, номер канала - в поле *ci.input*.

Значение передается в отсчетах ЦАП. Для преобразования напряжения в отсчеты ЦАП нужно использовать таблицы в техническом описании платы.

`ldspDigitalOut` — считывание цифровых линий

`ldspDigitalOut`

```
int ldspDigitalIn(LDSP_HANDLE Device, long mask);
```

`Device` Хендл открытого устройства.

`mask` Возвращает состояние входных цифровых линий.

Функция считывает состояние входные цифровые (TTL) линий. Каждый бит *mask* соответствует состоянию одной входной линии.

`ldspDigitalOut` — установка цифровых линий

`ldspDigitalOut`

```
int ldspDigitalOut(LDSP_HANDLE Device, long mask);
```

`Device` Хендл открытого устройства.

`mask` Маска выходных цифровых линий.

Функция устанавливает выходные цифровые (TTL) линии. Каждый бит *mask* соответствует состоянию выходной линии.

ldspSetTiming —установка временных параметров

ldspSetTiming

```
int ldspSetChannel(LDSP_HANDLE Device, frame_timing ft);
```

Device Хендл открытого устройства.

ft Временные параметры - частота АЦП и частота следования кадров, структура `frame_timing`.

Функция устанавливает временные параметры потокового сбора данных. Перед установкой временных параметров нужно установить размер таблицы каналов функцией `ldspChannels`.

ldspSetInBuf — установка размера входного буфера и фрагмента

ldspSetInBuf

```
int ldspSetInBuf (LDSP_HANDLE Device, fragment_info *fi);
```

Device Хендл открытого устройства.

fi Размер фрагмента и буфера, структура fragment_info.

Функция устанавливает размер входного буфера драйвера и фрагмента. Перед началом потокового ввода данных необходимо установить эти параметры.

ldspSetOutBuf — установка размера выходного буфера и фрагмента

ldspSetOutBuf

```
int ldspSetOutBuf(LDSP_HANDLE Device, fragment_info *fi);
```

Device Хендл открытого устройства.

fi Размер фрагмента и буфера, структура `fragment_info`.

Функция устанавливает размер выходного буфера драйвера и фрагмента. Перед началом потокового вывода данных необходимо установить эти параметры.

ldspSetSync — установка параметров синхронизации

ldspSetSync

```
int ldspSetSync (LDSP_HANDLE Device, sync_param *sp);
```

Device Хендл открытого устройства.

sp Параметры синхронизации - структура sync_param.

Функция устанавливает параметры синхронизации для потокового сбора данных. После запуска платы функцией `ldspSetTrigger` начинается ожидание условия синхронизации. Затем начинается непрерывный ввод (синхронизация старта) или вводится один кадр (покадровая синхронизация).

`ldspGetRange` — информация о диапазоне канала

`ldspGetRange`

```
int ldspGetRange(LDSP_HANDLE Device, channel_inf *ci, ldsp_range *range);
```

`Device` Хендл открытого устройства.

`ci` Информация о канале - структура `channel_inf`.

`range` Возвращает информацию о диапазоне - структура `ldsp_range`.

Функция возвращает информацию о диапазоне значений АЦП и соответствующих напряжений. Коэффициент усиления должен быть записан в `ci.gain`.

ldspWait —ожидание события

ldspWait

```
int ldspWait(LDSP_HANDLE Device, int EventMask, int msTimeout);
```

Device Хендл открытого устройства.

EventMask Маска событий - комбинация флагов:

LDSP_MASK_ADC Поступили данные с АЦП

LDSP_MASK_DAC Готовность вывода данных на ЦАП

msTimeout Время ожидания в миллисекундах.

Функция ожидает готовности ввода/вывода устройства. Возвращает 0, если истекло время ожидания, > 0, если устройство готово к вводу/выводу, отрицательное значение в случае ошибки.

ldspLoad —прямой доступ к памяти DSP

ldspLoad

```
int ldspLoad(LDSP_HANDLE Device, int Command, void *Data, int Words, int
WordSize, int Address);
```

Device Хендл открытого устройства.

Command Команда доступа к памяти:

LC_WRITE_PM	Запись в память программ DSP.
LC_READ_PM	Чтение из памяти программ DSP.
LC_WRITE_DM	Запись в память данных DSP.
LC_READ_DM	Чтение из памяти данных DSP.
LC_WRITE_EEPROM	Запись в память EEPROM.
LC_READ_EEPROM	Чтение из памяти EEPROM.

Data Указатель на массив слов.

Words Количество слов.

WordSize Размер слова в байтах.

Address Начальный адрес.

Функция осуществляет доступ к памяти сигнального процессора или EEPROM. Возвращает количество записанных/прочитанных слов, отрицательное значение в случае ошибки.

ldspMsTime — считывание таймера

ldspMsTime

```
unsigned long ldspMsTime ();
```

Функция возвращает значение таймера в миллисекундах.

Глава 4. Структуры данных

Структуры данных библиотеки LDSP

channel_inf — информация о канале

channel_inf

index Параметр задает индекс в таблице каналов. Используется функцией `ldspSetChannel`. Игнорируется при работе с одиночными отсчетами.

input Параметр задает номер входной/выходной линии устройства.

Для АЦП 0 соответствует входам $X1-Y1$ в дифференциальном режиме, $X1$ в одноконечном режиме.

Для ЦАП 0 соответствует выходу $DAC1, 1 - DAC2$.

mode Режим работы канала - комбинация флагов:

LCH_DIFFERENTIAL Дифференциальный режим

LCH_SINGLE_END Одноконечный режим (с общей землей)

LCH_CALIBR Режим калибровки нуля АЦП с заданным коэффициентом усиления. В поле *gain* должен быть установлен индекс коэффициента усиления, остальные поля не имеют значения.

LCH_RAW Установка логического номера канала, указанного в *log_channel*. Значения *number*, *gain* и остальных флагов игнорируются.

ctrl Маска цифровых линий. Значения, выводимые на цифровые линии во время срабатывания канала в таблице. Поддерживаются не всеми устройствами.

sample Значение сигнала. Значение, выводимое на ЦАП функцией `ldspSampleOut`. Значение, введенное с АЦП функцией `ldspSampleIn`.

`frame_timing` —временные параметры

`frame_timing`

channel_rate

Параметр задает частоту работы АЦП/ЦАП в Hz.

frame_rate

Параметр задает частоту следования кадров в Hz. Кадр состоит из массива отсчетов по всем каналам, установленным в таблице.

fragment_info —размер буфера и фрагмента

fragment_info

fragment_size Параметр задает размер фрагмента в байтах. Фрагмент - количество данных, которое передается при обработке аппаратного прерывания от платы.

buffer_size Параметр задает размер буфера в байтах. Размер буфера должен быть кратным размеру фрагмента. Если это поле установлено в 0, используется максимальное значение.

`sync_param` — параметры синхронизации

`sync_param`

`sync_type`

Тип синхронизации.

<code>SYNC_TYPE_NONE</code>	Синхронизация отключена.
<code>SYNC_TYPE_DIGITAL</code>	Синхронизация по цифровым каналам. Маска каналов задается в поле <code>sync_channel</code> .
<code>SYNC_TYPE_ANALOG</code>	Синхронизация по аналоговому каналу. Код канала задается в поле <code>sync_channel</code> . Код канала можно получить с помощью вызова <code>ldspSetChannel</code> с параметром <code>channel_inf.index = -1</code> , код канала возвращается в поле <code>channel_inf.channel_code</code> .
<code>SYNC_MODE_START</code>	Синхронизация старта. При срабатывании начинается непрерывный сбор данных. используется совместно с <code>SYNC_TYPE_DIGITAL</code> или <code>SYNC_TYPE_ANALOG</code> .
<code>SYNC_MODE_FRAME</code>	Покадровая синхронизация. При каждом срабатывании вводится один кадр АЦП. используется совместно с <code>SYNC_TYPE_DIGITAL</code> или <code>SYNC_TYPE_ANALOG</code> .
<code>SYNC_MODE_HIGH</code>	Аналоговая синхронизация старта по высокому уровню сигнала. Устанавливается вместе с <code>SYNC_TYPE_ANALOG</code> и <code>SYNC_MODE_START</code> . Если в момент вызова <code>ldspSetTrigger</code> значение сигнала выше порогового значения, сбор начинается сразу.
<code>SYNC_MODE_LOW</code>	Аналоговая синхронизация старта по низкому уровню сигнала. Устанавливается вместе с <code>SYNC_TYPE_ANALOG</code> и <code>SYNC_MODE_START</code> . Если в момент вызова <code>ldspSetTrigger</code> значение сигнала ниже порогового значения, сбор начинается сразу.

`SYNC_RISING_EDGE` Аналоговая синхронизация по восходящему фронту сигнала. Срабатывает при переходе порогового значения "снизу вверх". Устанавливается вместе с `SYNC_TYPE_ANALOG`.

`SYNC_FALLING_EDGE` Аналоговая синхронизация по нисходящему фронту сигнала. Срабатывает при переходе порогового значения "сверху вниз". Устанавливается вместе с `SYNC_TYPE_ANALOG`.

sync_channel

Код логического канала для аналоговой синхронизации.

Маска цифровых каналов для цифровой синхронизации.

sync_threshold

Пороговое значение сигнала (в отсчетах АЦП) для аналоговой синхронизации.

Значение цифровых каналов для цифровой синхронизации.

Параметры синхронизации устанавливаются функцией `ldspSetSync`.

`ldsp_range` — информация о диапазоне

`ldsp_range`

<code>gain_code</code>	Код коэффициента усиления.
<code>min</code>	Минимальное значение отсчета.
<code>max</code>	Максимальное значение отсчета.
<code>umin</code>	Напряжение (μV), соответствующее минимальному значению отсчета.
<code>umax</code>	Напряжение (μV), соответствующее максимальному значению отсчета.

Значения напряжения передаются в микровольтах, целые числа.