ИЗМЕРИТЕЛЬ ВРЕМЕННЫХ ИНТЕРВАЛОВ В-471 "TiMeS"

УПРАВЛЕНИЕ ИЗМЕРИТЕЛЕМ В РЕЖИМЕ TiMeS-Cnt.

РУКОВОДСТВО ПРОГРАММИСТА

ФДБИ.106.00.00.00 РП

СОДЕРЖАНИЕ

ИСТОРИЯ ИЗМЕНЕНИЙ

v.2.2 (10.06.2010)

Добавлен таймаут обновления данных. Если в течение 1 с не произведено ни одного чтения данных из буфера измерителя, производится принудительное чтение.

В более ранних версиях чтение производится только при заполнении буферной памяти канала, как минимум, наполовину (> 128 событий), что при неинтенсивном потоке событий (< 1 Hz) приводит к большим задержкам обновления данных.

v.2.1 (13.11.2007)

Исправлены:

BUG – инверсия входных сигналов и соотв., инверсия фронтов событий (при выбранном фронте фиксировался спад и наоборот);

BUG – перепутаны источники событий, запуск происходил по стоповому событию, а фиксировались стартовые события.

Добавлена возможность выбора входа синхронизации в качестве события (как стартового, так и стопового). Изменения – в B471Drv.dll. В описании структуры **B471CntChanSettings** пеализовано допустимое значение SS_TI = 2. Опция SS_TO = 3, по-прежнему, не реализована и недоступна.

v.2.0 (09.10.2007)

Первая рабочая версия драйвера и документации

Руководство программиста предназначено для ознакомления с принципом управления измерителем временных интервалов В-471 (далее измерителем) в составе измерительных установок и комплексов, при встраивании его в линию контроля производственного цикла, а также в целях автоматизации научных исследований.

При создании программного обеспечения для управления измерителем необходимо иметь представление о принципе работы измерителя, описанным в РЭ (раздел «Устройство и работа»). Отдельное внимание следует уделить различию понятий «входной разъем», «аналоговый тракт» и «измерительный канал». Каждый из аналоговых трактов 1 и 2 может быть (программно) подключен или к соответствующему входному разъему (I и II), или к выходу калибратора. Каждый из измерительных каналов 1 и 2 может независимо получать стартовые или стоповые события с выхода любого (программно-устанавливаемого) аналогового тракта.

Перечисленные функции, в большинстве своем (если не указано иное), возвращают код ошибки. Описание кодов ошибок приведено в Приложении А. Ноль (NO_ERRORS) означает отсутствие ошибок, нормальное функционирование. Отрицательное значение – ошибка,– работа невозможна. Положительное значение – предупреждение,– работа возможна, но заявленные параметры могут не выполняться из-за ошибок при инициализации отдельных узлов измерителя, выполнении одной или нескольких операций в ходе выполнения команды или получении калибровочных данных из измерителя.

Информативные для программиста и конечного пользователя значения возвращаемого кода ошибки перечислены в описании каждой функции. Остальные значения свидетельствуют о наличии конкретной неисправности аппаратуры измерителя или проблемы в программном обеспечении и предназначены для передачи в службу поддержки. Далее они будут обозначаться символом «+».

Настоящая версия программного обеспечения поддерживает работу только с одним измерителем. Одновременная работа нескольких измерителей возможна только при обеспечении загрузки соответствующего числа копий dll.

1 ПРИНЦИП УПРАВЛЕНИЯ ИЗМЕРИТЕЛЕМ

Для корректной работы приложения с аппаратурой измерителя и памятью необходимо вызвать функцию инициализации устройства Initialize() при запуске приложения и функцию Deinitialize() при выходе из приложения.

Примечание – При запуске приложения (загрузке dll) создается временный буфер измерительных данных – два файла (по числу каналов) с уникальными именами, размером по 100 Мб, в папке временных файлов текущего пользователя (%USERPROFILE%\Local Settings\Temp\). Это означает, что размер данных в ходе одного измерения по одному каналу не может превышать 100 Мб и нужно рассчитывать время запуска измерений исходя из этого ограничения с учетом реальной интенсивности событий. Каждое событие в буфере занимает 4 байта (DWORD).

Входные аналоговые тракты и измерительные каналы имеют полностью независимые настройки и управление. Запуск измерений в канале(-ах) осуществляется командой StartCh(). Останов – по StopCh(), а также, при необходимости, по выполнению следующих условий:
– (dwEventsLimit > 0) достижению нужного числа событий в канале;
– (dwTimeLimit > 0) истечении заданного максимального времени измерения;
– (dwRetriesLimit > 0) превышении установленного порога количества ошибок передачи.

Нулевое значение означает неактивное (выключенное) условие.

Измеритель также имеет полностью независимо программно-управляемые Калибратор и Генератор выходных синхроимпульсов.

1.1 НАЧАЛО И ЗАВЕРШЕНИЕ РАБОТЫ

```
extern "C" int Initialize(unsigned short pid, unsigned short vid, unsigned short ser);
```

Команда выполняет поиск и инициализацию найденного измерителя.

Параметры:
pid – 0x0471 – идентификатор устройства;
vid – 0x6871 – идентификатор производителя;
ser – серийный номер измерителя или 0xFFFF – первый свободный; если к компьютеру подключено более одного измерителя, указывается серийный (заводской) номер измерителя, с которым предполагается производить измерение.

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE, ERR_FAILED, ERR_TMPFILE0…ERR_TMPFILE4, «+».

```
extern "C" int Reset();
```

Сброс измерителя в начальное состояние ("do nothing"). Сброс производится автоматически в ходе инициализации. Используйте функцию в ходе работы только при необходимости (если состояние измерителя неизвестно).

Возвращаемые значения:
NO_ERRORS, ERR_FAILED.

```
extern "C" int Deinitialize();
```

Производит деинициализацию измерителя и временного буфера данных.

Возвращаемые значения:
NO_ERRORS;

1.1.1 НАЧАЛЬНОЕ СОСТОЯНИЕ ИЗМЕРИТЕЛЯ

После инициализации измеритель находится в начальном состоянии ("do nothing"):
– нулевая отстройка частоты встроенного опорного генератора;
– оба аналоговых тракта подключены к калибратору, аттенюаторы в положении 1:1, уровень дискриминации на границе входного амплитудного диапазона, что предотвращает его срабатывание;
– измерительные каналы 1 и 2 подключены к соотв. аналоговым трактам I и II.
– калибратор выключен, оба уровня 0 В, режим индикатора lmRISE;
– выходной синхроимпульс выключен, на выходе низкий уровень.

1.2 ЗАПУСК И ОСТАНОВКА ИЗМЕРЕНИЙ

```
extern "C" int StartCh( int ch );
extern "C" int StopCh( int ch );
```

Запуск и останов измерения по одному или обоим каналам.

Параметры:
ch – номер канала (0/1/-1 – канал 1 / канал 2 / все каналы одновременно);

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE, ERR_CH;
«+»: 1, 2, 3 – канал 1 / канал 2 / оба канала уже запущен(ы)/остановлен(ы).

1.3 ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ХОДЕ ИЗМЕРЕНИЙ

```
extern "C" int GetStatusCh( int ch, B471CntChanStatus* pStatusCh );
```

Позволяет получить информацию о ходе измерения в указанном измерительном канале.

Параметры:
ch – номер измерительного канала (0/1 – канал 1 / канал 2);
pStatusCh – указатель на структуру B471CntChanStatus (см. приложение Б), которая будет содержать текущее состояние измерительного канала;

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE.

```
extern "C" int GetStartTime( int ch, char* czDateStart, char* czTimeStart );
extern "C" int GetStopTime( int ch, char* czDateStop, char* czTimeStop );
```

Команды позволяют получить дату и время последнего запуска и останова измерительного канала в текущем сеансе работы с измерителем (например, для формирования файла отчета). До первого запуска и останова измерения результат выполнения команд неопределенный.

Параметры:
ch – номер измерительного канала (0/1 – канал 1 / канал 2);
czDateStart – указатель на строку, в которую надо поместить дату запуска (9 символов);
czTimeStart – указатель на строку, в которую надо поместить время запуска (9 символов);
czDateStop – указатель на строку, в которую надо поместить дату останова (9 символов);
czTimeStop – указатель на строку, в которую надо поместить время останова (9 символов);

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE.

1.4 ПОЛУЧЕНИЕ ИЗМЕРИТЕЛЬНЫХ ДАННЫХ

```
extern "C" int GetDataCh(int ch, unsigned long dwIndex, unsigned long** ppData );
```

Получение указателя на измерительные данные указанного измерительного канала непосредственно во временном буфере измерительных данных (наиболее быстрый способ получения данных, но работает не во всех приложениях, например, в MatLab 7.0.1 R14).

Параметры:
ch – номер измерительного канала (0/1 – канал 1 / канал 2);
dwIndex – требуемый номер отсчета, не должен быть больше dwDataSize (0 – первый отсчет)
ppData – указатель на инициализируемый указатель на требуемый отсчет измерительных данных канала.
Примечание – При dwIndex=0 (*ppData) есть указатель на начало всего массива измерительных данных.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE.

```
extern "C" int FillDataCh(int ch, unsigned long* pData );
```

Получение измерительных данных указанного измерительного канала. Данная команда производит копирование измерительных данных в указанный существующий массив (медленный способ, но надежный). Размер массива должен быть не менее текущего значения dwDataSize (в B471CntChanStatus) и/или dwEventsLimit (в B471CntChanSettings).

Параметры:
ch – номер измерительного канала (0/1 – канал 1 / канал 2);
pData – указатель на начало массива, который должен быть заполнен измерительными данными канала.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE.

```
extern "C" int GetTimeBase( double* pfTimeBase );
```

Служит для определения значения единицы временной шкалы в ходе измерения (зависит от аппаратной конфигурации измерителя, может измениться в последующих версиях программного обеспечения). Единица временной шкалы соответствует единице счета измерителя.

pfTimeBase – указатель на переменную, которая будет содержать значение единицы временной шкалы.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE.

2 НАСТРОЙКА РЕЖИМА РАБОТЫ (УСТАНОВКА ПАРАМЕТРОВ) ИЗМЕРИТЕЛЯ

2.1 УПРАВЛЕНИЕ ИЗМЕРИТЕЛЬНЫМИ КАНАЛАМИ

После инициализации измерителя входные аналоговые тракты и измерительные каналы находятся в начальном состоянии. Перед началом работы необходимо произвести установку их параметров в соответствии с условиями измерений.

```
extern "C" int GetChanSettings( int ch, B471CntChanSettings* pChSet );
extern "C" int SetChanSettings( int ch, B471CntChanSettings ChSet );
```

Получение текущего и установка нужного состояния измерительного канала.

GetChanSettings() позволяет определить текущее состояние измерительного канала. При вызове команды сразу после инициализации измерителя – это начальное состояние канала.

Установка состояния производится путем задания параметров канала в соответствующих полях структуры B471CntChanSettings (см. приложение Б) и выполнением команды SetChanSettings().

Параметры:
ch – номер канала (0/1 – канал 1 / канал 2);
pChSet – указатель на структуру B471CntChanSettings, которая будет содержать текущие установки канала;
ChSet –структура B471CntChanSettings, содержащая нужные установки канала.

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE, ERR_CH, «+».

```
extern "C" int GetDiskrLevel( int ch, float* pfValue);
extern "C" int SetDiskrLevel( int ch, float fValue);
```

Получение текущего и установка нужного напряжения срабатывания (уровня запуска) дискриминатора канала. Установка может производиться в любое время, в т.ч. и непосредственно во время измерения, при этом следует учитывать возможность ложного срабатывания дискриминатора из-за изменении уровня дискриминации (при пересечении уровнем дискриминации уровня входного сигнала).

Параметры:
ch – номер канала (0/1 – канал 1 / канал 2);
pfValue – указатель на переменную, которая будет содержать текущий уровень дискриминации;
fValue – устанавливаемое значение уровня дискриминации, В. Диапазон допустимых значений – не менее ±4 В при положении аттенюатора 1:1 (bDivider=0) и не менее ±40 В при положении аттенюатора 1:10 (bDivider=1).

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE, ERR_CH, ERR_FAILED,

2.2 УПРАВЛЕНИЕ ВСТРОЕННЫМ КАЛИБРАТОРОМ

При работе с калибратором следует иметь в виду, что в ходе измерения рекомендуется держать калибратор в выключенном состоянии для предотвращения влияния наводок импульсного сигнала калибратора на измеряемые сигналы (особенно актуально при работе от источника с высоким выходным сопротивлением). В выключенном состоянии калибратор может быть использован в качестве источника постоянного напряжения (текущего активного логического уровня).

```
extern "C" int GetClbrSettings( B471ClbrSettings* pClbrSet );
extern "C" int SetClbrSettings( B471ClbrSettings ClbrSet );
```

Получение текущего и установка нужного состояния калибратора.

GetClbrSettings() позволяет определить текущее состояние калибратора. При вызове команды сразу после инициализации измерителя – это начальное состояние калибратора.

Установка состояния производится путем задания параметров калибратора в соответствующих полях структуры B471ClbrSettings (см. приложение Б) и выполнением команды SetClbrSettings().

В случае, если указанные значения периода и длительности импульсов калибратора выходят за пределы возможных/реализуемых значений (например, длительность импульсов больше периода следования), в измеритель записываются ближайшие возможные значения.

Параметры:

pClbrSet – указатель на структуру B471ClbrSettings, которая будет содержать текущие установки калибратора;

ClbrSet –структура B471ClbrSettings, содержащая нужные установки калибратора.

Возвращаемые значения:

NO_ERRORS, ERR_NODEVICE, «+».

```
extern "C" int GetClbrLevel( long bHiLevel, float* pfValue );
extern "C" int SetClbrLevel( long bHiLevel, float fValue );
```

Получение текущего и установка нужного значения напряжения указанного логического уровня калибратора. Установка может производиться в любое время, в т.ч. и непосредственно во время измерения, при этом следует учитывать возможность ложного срабатывания дискриминатора из-за изменении уровня дискриминации (при пересечении уровнем дискриминации уровня входного сигнала).

Параметры:

bHiLevel – 1/0: верхний/нижний логический уровень.

Примечание – это весьма условно, т.к. напряжения уровней могут независимо принимать любые значения из всего диапазона. Тем не менее, при выключенном калибраторе на выходе будет уровень напряжения, соответствующий установленной полярности импульсов;

pfValue – указатель на переменную, которая будет содержать текущее значение напряжения выбранного логического уровня;

fValue – устанавливаемое значение напряжения выбранного логического уровня, В. Диапазон допустимых значений – не менее ±4 В.

Возвращаемые значения:

NO_ERRORS, ERR_NODEVICE, ERR_PARAM, ERR_FAILED.

2.3 УПРАВЛЕНИЕ ГЕНЕРАТОРОМ СИНХРОИМПУЛЬСОВ

Управление генератором синхроимпульсов аналогично управлению калибратора, за исключением отсутствия установки значений напряжения логических уровней (всегда ТТЛ).

```
extern "C" int GetTrigSettings( B471TrigSettings* pTrigSet );
extern "C" int SetTrigSettings( B471TrigSettings TrigSet );
```

Получение текущего и установка нужного состояния генератора синхроимпульсов.

GetTrigSettings() позволяет определить текущее состояние генератора синхроимпульсов. При вызове команды сразу после инициализации измерителя – это начальное состояние.

Установка состояния производится путем задания параметров генератора синхроимпульсов в соответствующих полях структуры B471TrigSettings (см. приложение Б) и выполнением команды SetTrigSettings().

В случае, если указанные значения периода и длительности импульсов выходят за пределы возможных/реализуемых значений (например, длительность импульсов больше периода следования), в измеритель записываются ближайшие возможные значения.

Параметры:
pTrigSet – указатель на структуру B471TrigSettings, которая будет содержать текущие установки генератора синхроимпульсов;
TrigSet –структура B471TrigSettings, содержащая нужные установки генератора синхроимпульсов.

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE, «+».

---

3 СЕРВИСНЫЕ И СПЕЦИАЛЬНЫЕ ФУНКЦИИ

Команды, перечисленные в данном разделе, предназначены для использования в целях отладки создаваемого программного обеспечения, контроля качества связи измерителя с компьютером, получения дополнительной информации об измерителе, (например, дискретности установки уровней), а также подстройки измерителя под условия эксперимента (например, для компенсации систематической погрешности источника измеряемых временных интервалов).

```
extern "C" int Test();
```

Команда предназначена для проверки инициализации dll.

Возвращаемое значение: 2 (всегда).

3.1 УПРАВЛЕНИЕ ЦАП В ДИСКРИМИНАТОРАХ КАНАЛОВ

```
extern "C" int SetDiskrCode( int ch, unsigned long dwCode );
extern "C" int GetDiskrCode( int ch, unsigned long* pdwCode );
```

Установка/чтение кода ЦАПа, управляющего уровнем дискриминатора указанного канала. Диапазон допустимых значений 0..4096.

Примечание – значение погрешности установки уровня дискриминатора в несколько раз превышает величину его дискрета (минимального изменения), т.е. данную функцию можно использовать для точной подстройки уровня дискриминатора.

Параметры:
pdwCode – указатель на переменную, которая должна содержать текущее значение кода ЦАП;
dwCode – устанавливаемsq код ЦАП.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE, ERR_CH, ERR_RANGE, ERR_FAILED.

```
extern "C" int GetDiskrCodeByValue( int ch, float fValue, unsigned long* pdwCode );
extern "C" int GetDiskrLevelByCode( int ch, unsigned long dwCode, float* pfValue );
```

Определение соответствия значений кода ЦАПа и номинального значения напряжения срабатывания дискриминатора (уровня запуска).

Параметры:
ch – номер канала (0/1 – канал 1 / канал 2);
fValue – значение уровня запуска, для которого нужно найти ближайший код ЦАП;
pdwCode – указатель на переменную, в которую будет записан соответствующий код ЦАП;
dwCode – код ЦАП, для которого нужно узнать соответствующее значение уровня запуска;
pfValue – указатель на переменную, в которую будет записано соответствующее значение уровня запуска.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE, ERR_CH, ERR_RANGE.

3.2 УПРАВЛЕНИЕ ЦАП КАЛИБРАТОРА

```
extern "C" int GetClbrCode( long bHiLevel, unsigned long* pdwCode );
extern "C" int SetClbrCode( long bHiLevel, unsigned long dwCode );
```

Получение текущего / установка нового кода ЦАП, управляющего значением напряжения заданного логического уровня калибратора. Диапазон допустимых значений 0..4096.
Примечание – значение погрешности установки значения напряжения логических уровней калибратора в несколько раз превышает величину его дискрета (минимального изменения), т.е. данную функцию можно использовать для точной подстройки напряжения (по внешнему вольтметру при выключенном калибраторе).

Параметры:
bHiLevel – 1/0: верхний/нижний логический уровень (условно, см. примечание п.1.3);
pdwCode – указатель на переменную, которая будет содержать текущий код ЦАП;
dwCode – устанавливаемый код ЦАП.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE, ERR_RANGE, ERR_PARAM, ERR_FAILED.

```
extern "C" int GetClbrCodeByValue( long bHiLevel, float fValue, unsigned long* pdwCode );
extern "C" int GetClbrLevelByCode( long bHiLevel, unsigned long dwCode, float* pfValue );
```

Определение соответствия значений кода ЦАПа и номинального значения напряжения указанного логического уровня калибратора.

Параметры:
bHiLevel – 1/0: верхний/нижний логический уровень (условно, см. примечание п.1.3);
fValue – значение напряжения, для которого нужно найти ближайший код ЦАП;
pdwCode – указатель на переменную, в которую будет записан соответствующий код ЦАП;
dwCode – код ЦАП, для которого нужно узнать соответствующее значение напряжения;
pfValue – указатель на переменную, в которую будет записано соответствующее значение уровня запуска.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE, ERR_RANGE, ERR_PARAM.

3.3 УПРАВЛЕНИЕ ЧАСТОТОЙ ВСТРОЕННОГО ОПОРНОГО ГЕНЕРАТОРА

```
extern "C" int GetClkPpm( float* pfValue );
extern "C" int SetClkPpm( float fValue );
```

Установка/чтение значения относительной отстройки частоты встроенного опорного генератора. Значение – в ppm (10^{-6}, 0.0001%) относительно номинального значения частоты. Диапазон допустимых значений при выпуске измерителя не менее ±10 ppm.

Команда может использоваться в целях установки действительного значения частоты опорного генератора в текущих рабочих условиях с погрешностью менее 0,1 ppm (10^{-7}) по внешнему эталону частоты (измерением эталонных временных интервалов) или эталонному частотомеру (измерением частоты выходного синхроимпульса измерителя).

Параметры:
pfValue – указатель на переменную, которая будет содержать текущее значение отстройки;
fValue – устанавливаемое значение отстройки.

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE, ERR_FAILED.

```
extern "C" int SetClkCode( unsigned long dwCode );
extern "C" int GetClkCode( unsigned long* pdwCode );
```

Установка/чтение кода ЦАПа, корректирующего частоту опорного генератора. Диапазон допустимых значений 0..1023.

Параметры:
pdwCode – указатель на переменную, которая должна содержать текущее значение кода ЦАП;
dwCode – устанавливаемsq код ЦАП.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE, ERR_RANGE, ERR_FAILED.

```
extern "C" int GetClkCodeByValue( float fValue, unsigned long* pdwCode );
extern "C" int GetClkPpmByCode( unsigned long dwCode, float* pfValue );
```

Определение соответствующих значений (кода ЦАПа / значения отстройки частоты).

3.4 КОНТРОЛЬ КАЧЕСТВА СВЯЗИ ИЗМЕРИТЕЛЯ С КОМПЬЮТЕРОМ

```
extern "C" int GetRetriesLimit( unsigned long* pdwRetriesLimit );
extern "C" int SetRetriesLimit( unsigned long dwRetriesLimit );
```

Получить текущее / установить новое предельное значение числа ошибок передачи данных.

Параметры:
pdwRetriesLimit – указатель на переменную, в которой будет содержаться текущее предельное значение числа ошибок;
dwRetriesLimit – устанавливаемое предельное значение числа ошибок.

Возвращаемое значение:
NO_ERRORS, ERR_NODEVICE.

---

4 ИНФОРМАЦИОННЫЕ КОМАНДЫ

```
extern "C" int GetDevId( int* piSerNo );
```

Позволяет узнать серийный номер найденного и инициализированного измерителя.

Параметры:
piSerNo – указатель на переменную, которая будет содержать серийный номер.

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE;

```
extern "C" int GetChanName( int ch, char* pChName );
```

Позволяет узнать название канала в том виде, в каком оно указано на передней панели измерителя.

Параметры:
ch – номер канала (0/1 – канал 1 / канал 2);
pChName – указатель на строку, которая будет содержать название канала (16 символов).

Возвращаемые значения:
NO_ERRORS, ERR_NODEVICE;

ПРИЛОЖЕНИЕ А

КОДЫ ОШИБОК

const int NO_ERRORS = 0; // OK
Означает, что в ходе выполнения команды ошибок не возникло (нормальное функционирование).

const int ERR_NODEVICE = -1; // device not connected / not initialized
При инициализации – измеритель не найден (не подключен, не установлены драйверы).
В ходе работы – измеритель недоступен: предпринята попытка выполнения команды до инициализации измерителя или после сбоя в работе интерфейса измерителя с компьютером вследствие отсоединения кабеля, статического разряда или других причин.

const int ERR_CH = -2; // invalid channel index
При выполнении команды задан неверный номер канала. Команда не выполнена.
Допустимые значения:
0 – аналоговый тракт I / измерительный канал 1;
1 – аналоговый тракт II / измерительный канал 2;
-1 – все тракты / каналы одновременно.

const int ERR_RANGE = -3; // requested dwCode is out of range
При выполнении команды передан код, выходящий за пределы допустимого диапазона. Команда не выполнена.

const int ERR_PARAM = -4; // one of other func params is out of range
При выполнении команды, по крайней мере, один из параметров выходит за пределы допустимого диапазона. Команда не выполнена.

const int ERR_FAILED = -5; // device control/data transfer failed
При выполнении команды возникла ошибка передачи данных. Статус выполнения команды неопределенный (с малой степенью вероятности команда могла быть выполнена).

const int ERR_TMPFILE0 = -10; // TMP file create error (file already open)
Повторный вызов Initialize() в ходе одного сеанса работы с измерителем.
Предыдущий сеанс работы с измерителем был завершен некорректно (без выполнения команды DeInitialize()).

const int ERR_TMPFILE1 = -11; // TMP file create error (disk full)
Для создания временных файлов недостаточно места на диске, содержащем профиль пользователя (файлы не созданы). Работа невозможна.

const int ERR_TMPFILE2 = -13; // TMP file size error (not enough disk space)
Для создания временных файлов недостаточно места на диске, содержащем профиль пользователя (файлы имеют меньший размер). Работа невозможна.

const int ERR_TMPFILE3 = -13; // TMP file mapping creation failed
const int ERR_TMPFILE4 = -14; // TMP file map view failed
Ошибка при инициализации временного буфера измерительных данных. Работа невозможна.

СТРУКТУРЫ УПРАВЛЯЮЩИХ ДАННЫХ

```
typedef struct {
        unsigned long   lmLedMode;      // (lmOFF, lmLO, lmHI, lmON, lmFALL, lmRISE, lmBOTH) not used (no ext LED)
        long            bEnabled;       // 0/1 - OFF/ON
        unsigned long   dwEnMaskCh;     // reserved (=0)
        long            bActiveHi;      // fLength defines 0/1 - active lo/hi pulse length
        double          fPeriod;        // pulse period, s
        double          fLength;        // pulse length, s
} B471TrigSettings;


typedef struct {
        unsigned long   lmLedMode;      // (lmOFF, lmLO, lmHI, lmON, lmFALL, lmRISE, lmBOTH)
        long            bEnabled;       // 0/1 - OFF/ON
        unsigned long   dwEnMaskCh;     // reserved (=0)
        long            bActiveHi;      // fLength defines 0/1 - active lo/hi pulse length
        double          fPeriod;        // pulse period, s
        double          fLength;        // pulse length, s
} B471ClbrSettings;


typedef struct {
        unsigned long   lmLedMode;      // (lmOFF, lmLO, lmHI, lmON, lmFALL, lmRISE, lmBOTH)
        long            bDivider;       // attenuation ratio (0/1 - 1:1/1:10)
        long            bInput;         // measuring channel signal source (0/1 - Clbr/Input)
        unsigned long   ssStartCh;      // Start signal source (0/1/2/3 - ssCHI/ssCHII/ssTI/ssTO)
        unsigned long   ssStartSlope;   // Start signal slope (0/1 - ssFALL/ssRISE)
        unsigned long   ssEventCh;      // Events signal source (0/1/2/3 - ssCHI/ssCHII/ssTI/ssTO)
        unsigned long   ssEventSlope;   // Event signal slope (0/1 - ssFALL/ssRISE)
        unsigned long   dwEventsLimit;  // Events Count - end of measurement condition
        unsigned long   dwTimeLimit;    // in milliseconds (by computer clock) - end of measurement condition
} B471CntChanSettings;


typedef struct {
        long            bStopped;       // software flag (for threads syncronization).
        long            bMeasEn;        // device flag (channel is waiting for start event or is in measuring cycle)
        long            bBusy;          // device flag (channel in measuring cycle)
        unsigned long   dwDataSize;     // channel's events count, which are already transmitted from device
        unsigned long   dwFIFOSize;     // channel FIFO size in EVENTS (depends on PLD configuration)
        unsigned long   dwFIFOEvents;   // channel FIFO status (measured events, ready for read)
        long            bFIFOError;     // device flag, indicates FIFO FULL, channel stopped to prevent event loss
        long            bReadFail;      // software flag, indicates there was at least one error during data transfer
        unsigned long   dwTime;         // software controlled (indicates time in ms from channel measurement start, not
from start event(!) )
        long            bTimeOut;       // software flag (indicates channel was stopped on timeout)
        long            bLedOn;         // device flag, indicates whether channel LED is on/off (1/0)
} B471CntChanStatus;


// B471LedMode
const unsigned long     lmOFF   = 0;    // led is always off
const unsigned long     lmLO    = 1;    // led is on when signal under discriminator threshold level
const unsigned long     lmHI    = 2;    // led is on when signal above discriminator threshold level
const unsigned long     lmON    = 3;    // led is always on (for ex., led test)
const unsigned long     lmFALL  = 5;    // led sparks when signal falls under discriminator threshold level
const unsigned long     lmRISE  = 6;    // led sparks when signal rises above discriminator threshold level
const unsigned long     lmBOTH  = 7;    // led sparks on both signal edges (lmFALL + lmRISE)


// B471SignalSource
const unsigned long     ssCH0   = 0;    // Signal input I is connected to measuring channel
const unsigned long     ssCH1   = 1;    // Signal input II is connected to measuring channel
const unsigned long     ssTI    = 2;    // Trigger input is connected to measuring channel (reserved)
const unsigned long     ssTO    = 3;    // Trigger output is connected to measuring channel (reserved)


// B471SignalSlope
const unsigned long     ssFALL  = 0;    // Falling signal slope is used as "start" or "event"
const unsigned long     ssRISE  = 1;    // Rising signal slope is used as "start" or "event"
```