РИФТЭК

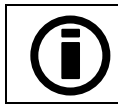
БИБЛИОТЕКА ФУНКЦИЙ ДЛЯ ТРИАНГУЛЯЦИОННЫХ ЛАЗЕРНЫХ ДАТЧИКОВ СЕРИИ РФ60Х

RF60x SDK (Набор разработчика ПО) 3
1. Функции для работы с датчиками, подключенными к последовательному порту. 3
1.1. Подключение к COM-порту (RF60x_OpenPort) 3
1.2. Отключение от COM-порта (RF60x_ClosePort) 4
1.3. Идентификация устройства (RF60x_HelloCmd) 4
1.4. Чтение параметров (RF60x_ReadParameter) 5
1.5. Чтение неопределённых параметров (RF60x_ReadCustomParameter) 6
1.6. Запись неопределённых параметров (RF60x_WriteCustomParameter) 7
1.7. Запись параметров (RF60x_WriteParameter) 7
1.8. Сохранение текущих параметров во FLASH-памяти (RF60x_FlushToFlash) 8
1.9. Восстановление во FLASH-памяти значений параметров по умолчанию (RF60x_RestoreFromFlash) 8
1.10. Защелкивание текущего результата (RF60x_LockResult) 9
1.11. Получение результата измерения (RF60x_Measure) 9
1.12. Запуск потока измерений (RF60x_StartStream) 10
1.13. Останов потока измерений (RF60x_StopStream) 10
1.14. Получение результатов измерений из потока (RF60x_GetStreamMeasure) 11
1.15. Передача пользовательских данных (RF60x_CustomCmd) 12
2. Функции для работы с датчиками, подключенными к USB с помощью FTDI. 13

RF60x SDK (Набор разработчика ПО)

Библиотека RF60x предназначена для создания пользователем своего собственного ПО для работы с триангуляционными лазерными датчиками серии РФ60Х, для этого предоставлены:

- ✓ Заголовочный файл **RF60x.h** с константами и прототипами функций;
- ✓ Подключаемая библиотека **RF60x.lib**, необходимая для работы с VC++/VS2003/VS2005/Borland C++;
- ✓ Исполняемая библиотека **RF60x.dll** с набором готовых функций;
- ✓ Примеры для работы с библиотекой **RF60x**;
- ✓ Настоящее описание.

	*Библиотека RF60x предназначена для ОС Windows 98/Me/2000/XP*

1. Функции для работы с датчиками, подключенными к последовательному порту.

1.1. Подключение к COM-порту (RF60x_OpenPort)

Функция **RF60x_OpenPort** открывает COM-порт с заданным символьным именем, заполняет указатель на дескриптор устройства и возвращает результат операции:

```
BOOL  RF60x_OpenPort(
            LPCSTR      lpPort_Name,
            DWORD       dwSpeed,
            HANDLE *    lpHandle
);
```

Параметры:

lpPort_Name – название COM-порта (например “COM1:”), полный синтаксис при задании имени COM-порта можно посмотреть в MSDN, функция CreateFile;

dwSpeed – скорость работы через COM-порт. Параметр идентичен полю BaudRate в структуре DCB подробно описанной в MSDN;

lpHandle – указатель на дескриптор устройства;

Возвращаемое значение:

Если COM-порт не открыт, и/или настроить его не удалось, функция вернёт FALSE иначе, если COM-порт открыт и настроен успешно – функция вернёт TRUE. Более детальные сведения об возвращаемых ошибках можно получить с помощью API функции GetLastError, описанной в MSDN.

1.2. Отключение от COM-порта (RF60x_ClosePort)

Функция **RF60x_ClosePort** закрывает COM-порт и возвращает результат операции:

```
BOOL  RF60x_ClosePort(
            HANDLE              hHandle
);
```

Параметры:

hHandle – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

Возвращаемое значение:

Если COM-порт закрыть не удалось, функция вернёт FALSE, иначе, если COM-порт был успешно закрыт – функция вернёт TRUE.

1.3. Идентификация устройства (RF60x_HelloCmd)

Функция **RF60x_HelloCmd** производит идентификацию устройства RF60x сети по сетевому адресу и заполняет структуру ***RF60xHELLOANSWER***:

```
typedef struct _RF60x_HELLO_ANSWER_ {
      BYTE        bDeviceType;
      BYTE        bcDeviceModificaton;
      WORD        wDeviceSerial;
      WORD        wDeviceMaxDistance;
      WORD        wDeviceRange;
} RF60xHELLOANSWER, *LPRF60xHELLOANSWER;
```

Где:

bDeviceType	–	однобайтная величина показывающая тип устройства (для RF60x данная величина равна 60) (тип **BYTE**);
bDeviceModificaton	–	однобайтная величина показывающая модификацию устройства (тип **BYTE**);
wDeviceSerial	–	двубайтная величина содержащая серийный номер устройства устройства (тип **WORD**);
wDeviceMaxDistance	–	двубайтная величина содержащая значение базового расстояния для датчика РФ60Х (тип **WORD**);
wDeviceRange	–	двубайтная величина содержащая значение диапазона для датчика РФ60Х (тип **WORD**).

```
BOOL RF60x_HelloCmd (
            HANDLE                  hCOM,
            BYTE                    bAddress,
            LPRF60xHELLOANSWER      lprfHelloAnswer
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства;

lprfHelloAnswer – указатель на структуру RF60xHELLOANSWER.

Возвращаемое значение:

Если устройство не ответило на запрос идентификации, функция возвращает FALSE, иначе функция возвращает TRUE и заполняет структуру **RF60xHELLOANSWER**.

1.4. Чтение параметров (RF60x_ReadParameter)

Функция **RF60x_ReadParameter** читает внутренние параметры датчика РФ60X и возвращает текущее значение по адресу параметра:

```
BOOL RF60x_ReadParameter (
            HANDLE                  hCOM,
            BYTE                    bAddress,
            WORD                    wParameter,
            DWORD *                 lpdwValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства;

wParameter – номер параметра, см. табл. 1, детальные значение параметров смотрите в технической документации на датчик РФ60X;

Табл. 1

Параметр	Описание
RF60x_PARAMETER_POWER_STATE	Питание датчика
RF60x_PARAMETER_ANALOG_OUT	Питание аналогового выхода
RF60x_PARAMETER_SAMPLE_AND_SYNC	Управление выборкой и синхронизацией
RF60x_PARAMETER_NETWORK_ADDRESS	Сетевой адрес
RF60x_PARAMETER_BAUDRATE	Скорость передачи данных через последовательный порт
RF60x_PARAMETER_LASER_BRIGHT	Уровень яркости лазера
RF60x_PARAMETER_AVERAGE_COUNT	Количество усредняемых значений

RF60x_PARAMETER_SAMPLING_PERIOD	Период выборки
RF60x_PARAMETER_ACCUMULATION_TIME	Максимальное время накопления
RF60x_PARAMETER_BEGIN_ANALOG_RANGE	Начало диапазона аналогового выхода
RF60x_PARAMETER_END_ANALOG_RANGE	Конец диапазона аналогового выхода
RF60x_PARAMETER_RESULT_DELAY_TIME	Время задержки результата
RF60x_PARAMETER_ZERO_POINT_VALUE	Точка нуля
RF60x_PARAMETER_CAN_SPEED	Скорость передачи данных по CAN интерфейсу
RF60x_PARAMETER_CAN_STANDARD_ID	Стандартный идентификатор CAN
RF60x_PARAMETER_CAN_EXTENDED_ID	Задает расширенный идентификатор CAN
RF60x_PARAMETER_CAN_ID	Идентификатор CAN интерфейса

lpdwValue – указатель на переменную типа DWORD, в которую будет сохранено текущее значение параметра.

Возвращаемое значение:

Если устройство не ответило на запрос чтения параметра, функция возвращает FALSE, иначе функция возвращает TRUE и заполняет переменную *lpdwValue*.

1.5. Чтение неопределённых параметров (RF60x_ReadCustomParameter)

Функция **RF60x_ReadCustomParameter** читает внутренние параметры датчика РФ60X и возвращает текущее значение по адресу параметра:

```
BOOL  RF60x_ReadCustomParameter(
            HANDLE                  hCOM,
            BYTE                    bAddress,
            BYTE                    bParameterAddress,
            BYTE                    bParameterSize,
            void *                  lpValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства;

bParameterAddress – адрес внутреннего параметра в датчике РФ60X;

bParameterSize – размер внутреннего параметра в датчике РФ60X;

lpValue – указатель на массив, в который будут сохранены прочитанные данные;

Возвращаемое значение:

Если устройство не ответило на запрос чтения параметра, функция возвращает FALSE, иначе функция возвращает TRUE и заполняет массив переданный через *lpValue*.

1.6. Запись неопределённых параметров (RF60x_WriteCustomParameter)

Функция **RF60x_WriteCustomParameter** записывает внутренние параметры датчика РФ60Х:

```
BOOL  RF60x_WriteCustomParameter(
            HANDLE                  hCOM,
            BYTE                    bAddress,
            BYTE                    bParameterAddress,
            BYTE                    bParameterSize,
            void *                  lpValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства;

bParameterAddress – адрес внутреннего параметра в датчике РФ60Х;

bParameterSize – размер внутреннего параметра в датчике РФ60Х;

lpValue – указатель на массив, который будет сохранён начиная с адреса *bParameterAddress*;

Возвращаемое значение:

Если запись параметра не произведена, функция возвращает FALSE, иначе функция возвращает TRUE.

1.7. Запись параметров (RF60x_WriteParameter)

Функция **RF60x_WriteParameter** записывает внутренние параметры датчика РФ60Х:

```
BOOL RF60x_WriteParameter (
             HANDLE                 hCOM,
             BYTE                   bAddress,
             WORD                   wParameter,
             DWORD                  dwValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства;

wParameter – номер параметра, см. табл. 1, детальные значение параметров смотрите в технической документации на датчик РФ60X;

dwValue – значение параметра, которое будет сохранено в датчик РФ60X.

Возвращаемое значение:

Если запись параметра не произведена, функция возвращает FALSE, иначе функция возвращает TRUE.

1.8. Сохранение текущих параметров во FLASH-памяти (RF60x_FlushToFlash)

Функция **RF60x_FlushToFlash** сохраняет все параметры во FLASH-память датчика РФ60X:

```
BOOL  RF60x_FlushToFlash(
            HANDLE      hCOM,
            BYTE        bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства.

Возвращаемое значение:

Если устройство не ответило на запрос сохранения всех параметров во FLASH-память, функция возвращает FALSE, иначе, если от датчика получено подтверждение о записи, функция возвращает TRUE.

1.9. Восстановление во FLASH-памяти значений параметров по умолчанию (RF60x_RestoreFromFlash)

Функция **RF60x_RestoreFromFlash** восстанавливает значения всех параметров во FLASH по умолчанию:

```
BOOL  RF60x_RestoreFromFlash(
                  HANDLE            hCOM,
                  BYTE              bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства.

Возвращаемое значение:

Если устройство не ответило на запрос восстановления всех параметров во FLASH-памяти, функция возвращает FALSE, иначе, если от датчика получено подтверждение о восстановлении, функция возвращает TRUE.

1.10. Защелкивание текущего результата (RF60x_LockResult)

Функция **RF60x_LockResult** защелкивает текущее измеренное значение до следующего вызова функции **RF60x_Measure**:

```
BOOL  RF60x_LockResult(
            HANDLE          hCOM,
            BYTE            bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства.

Возвращаемое значение:

Если устройство не ответило на запрос защелкивания результата, функция возвращает FALSE, иначе функция возвращает TRUE.

1.11. Получение результата измерения (RF60x_Measure)

Функция **RF60x_Measure** читает из датчика РФ60X текущее измеренное значение. Значение передаваемого датчиком результата (D) нормировано таким образом, чтобы полному диапазону датчика (S в мм) соответствовала величина 4000h (16384), поэтому результат в миллиметрах получают по следующей формуле: X=D*S/4000h (мм):

```
BOOL  RF60x_Measure(
            HANDLE          hCOM,
            BYTE            bAddress,
            USHORT *        lpusValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства.

lpusValue – указатель на переменную типа USHORT/WORD, содержащую результат D.

Возвращаемое значение:

Если устройство не ответило на запрос результата, функция возвращает FALSE, иначе, если от датчика получено подтверждение о восстановлении, функция возвращает TRUE.

1.12. Запуск потока измерений (RF60x_StartStream)

Функция **RF60x_StartStream** переводит датчик РФ60Х в режим непрерывной передачи результатов измерений:

```
BOOL  RF60x_StartStream(
            HANDLE          hCOM,
            BYTE            bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства.

Возвращаемое значение:

Если устройство не удалось перевести в режим непрерывной передачи результатов измерений, функция возвращает FALSE, иначе функция возвращает TRUE.

1.13. Останов потока измерений (RF60x_StopStream)

Функция **RF60x_StopStream** переводит датчик из режима непрерывной передачи результатов измерений в режим «запрос-ответ»:

```
BOOL  RF60x_StartStream(
            HANDLE          hCOM,
            BYTE            bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

bAddress – адрес устройства.

Возвращаемое значение:

Если устройство не удалось остановить непрерывную передачу данных, функция возвращает FALSE, иначе функция возвращает TRUE.

1.14. Получение результатов измерений из потока (RF60x_GetStreamMeasure)

Функция **RF60x_GetStreamMeasure** читает из входного буфера COM-порта данные, полученные от датчика РФ60Х, после успешного выполнения функции RF60x_StartStream. В буфер данные приходят со скоростью, установленной в параметрах датчика РФ60Х, т.к. глубина входного буфера ограничена 1024 байтами, то желательно вычитывать данные с периодичностью, равной установленной в параметрах датчика РФ60Х. Параметр *lpusValue* идентичен параметру *lpusValue* в функции RF60x_Measure.

```
BOOL  RF60x_GetStreamMeasure(
            HANDLE                  hCOM,
            USHORT *                lpusValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

lpusValue – указатель на переменную типа USHORT/WORD, содержащую результат D.

Возвращаемое значение:

Если в буфере данные отсутствуют, то функция возвращает FALSE, иначе функция возвращает TRUE и заполняет значение *lpusValue*.

	*Для стабильной работы функции **RF60x_GetStreamMeasure** необходимо использовать её в отдельном потоке, с приоритетом, не ниже THREAD_PRIORITY_NORMAL, иначе происходит переполнение входного буфера последовательного порта, что приводит к непредсказуемым результатам.*

1.15. Передача пользовательских данных (RF60x_CustomCmd)

Функция **RF60x_CustomCmd** используется для передачи и/или приёма данных от датчика РФ60X.

```
BOOL  RF60x_CustomCmd(
            HANDLE          hCOM,
            char *          pcInData,
            DWORD           dwInSize,
            char *          pcOutData,
            DWORD *         pdwOutSize
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF60x_OpenPort либо CreateFile;

pcInData – указатель на массив данных, который будет передан в датчик РФ60X. Если передавать данные не требуется, *pcInData* должен быть NULL и *dwInSize* должен быть 0.

dwInSize – размер передаваемых данных. Если данные передавать не требуется, данный параметр должен быть 0.

pcOutData – указатель на массив данных, в который будет сохранены данные полученные от датчика РФ60X. Если получать данные не требуется, *pcOutData* должен быть NULL.

pdwOutSize – указатель на переменную содержащую размер получаемых данных. Если данные принимать не требуется, данный параметр должен быть NULL. После успешного получения данных в переменную, на которую указывает данный параметр, будет записано количество прочитанных байт.

Возвращаемое значение:

Если передача либо приём данных не удался, то функция возвращает FALSE, иначе функция возвращает TRUE.

Пример:

```
HANDLE            hRF60x      = INVALID_HANDLE_VALUE;
DWORD             dwValue;
USHORT            usMeasured;
RF60xHELLOANSWER  hlans;

// Чистим структуру RF60xHELLOANSWER
memset(&hlans, 0x00, sizeof(RF60xHELLOANSWER));

// Открываем COM-порт
if (!RF60x_OpenPort("COM2:", CBR_9600, &hRF60x)
      return (FALSE);

// Опрашиваем устройство
if (RF60x_HelloCmd( hRF60x, 1, &hlans ))
{

      ////////////////////////////////////////////////
      //                                            //
      // После успешного выполнения RF60x_HelloCmd  //
      // в структуре hlans содержится информация о  //
      // датчике РФ60Х, ответившем на запрос        //
      //                                            //
      ////////////////////////////////////////////////

      //Читаем параметр: Яркость лазера
      RF60x_ReadParameter(
                              hRF60x,
                              1,
                              RF60x_PARAMETER_LASER_BRIGHT,
                              &dwValue
                        );

      /* В dwValue содержится значения яркости лазера */

      //Получаем значения расстояния из датчика РФ60Х
      RF60x_Measure( hRF60x, 1, &usMeasured );

      /* В usMeasured содержится измеренный результат */

}

RF60x_ClosePort( hRF60x );
```

2. Функции для работы с датчиками, подключенными к USB с помощью FTDI.

При работы с USB устройствами на FTDI в данной библиотеке реализована поддержка функций работающих через D2XX библиотеку FTDI. Работа функций идентична функциям для работы с последовательным портом, основное отличие это присутствие префикса **FTDI_** в имени функции, например:
Функция получения результата **RF60x_Measure** для последовательного порта и **RF60x_FTDI_Measure** для устройств с FTDI USB.