РИФТЭК

БИБЛИОТЕКА ФУНКЦИЙ

ДЛЯ

КОНТАКТНЫХ ДАТЧИКОВ

ПОЛОЖЕНИЯ

СЕРИИ РФ25х

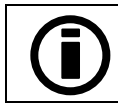
Содержание

Содержание 2
RF25x SDK (Набор разработчика ПО) 3
1. Функции для работы с датчиками, подключенными к последовательному порту. 3
1.1. Подключение к COM-порту (RF25x_OpenPort) 3
1.2. Отключение от COM-порта (RF25x_ClosePort) 4
1.3. Идентификация устройства (RF25x_HelloCmd) 4
1.4. Чтение параметров (RF25x_ReadParameter) 5
1.5. Чтение неопределённых параметров (RF25x_ReadCustomParameter) 6
1.6. Запись неопределённых параметров (RF25x_WriteCustomParameter) 6
1.7. Запись параметров (RF25x_WriteParameter) 7
1.8. Сохранение текущих параметров во FLASH-памяти (RF25x_FlushToFlash) 8
1.9. Восстановление во FLASH-памяти значений параметров по умолчанию (RF25x_RestoreFromFlash) 8
1.10. Защелкивание текущего результата (RF25x_LockResult) 9
1.11. Получение результата измерения (RF25x_Measure) 9
1.12. Запуск потока измерений (RF25x_StartStream) 10
1.13. Останов потока измерений (RF25x_StopStream) 10
1.14. Получение результатов измерений из потока (RF25x_GetStreamMeasure) 11
1.15. Задание начала отсчетов системы координат (RF25x_SetDatumPoint) 12
1.16. Передача пользовательских данных (RF25x_CustomCmd) 12
2. Функции для работы с датчиками, подключенных к USB с помощью FTDI. 14
2.1. Подключение к USB через FTDI (RF25x_FTDI_OpenPort) 14

RF25x SDK (Набор разработчика ПО)

Библиотека RF25x предназначена для создания своего ПО, управляющего контактным датчиком положения серии РФ25х, для этого предоставлены:

- ✓ Заголовочный файл **RF25x.h** с константами и прототипами функций;
- ✓ Подключаемая библиотека **RF25x.lib**, необходимая для работы с VC++/VS2003/VS2005/Borland C++;
- ✓ Исполняемая библиотека **RF25x.dll** с набором готовых функций;
- ✓ Примеры для работы с библиотекой **RF25x**;
- ✓ Настоящее описание.

	*Библиотека RF25x предназначена для ОС Windows 98/Me/2000/XP*

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

1.1. Подключение к COM-порту (RF25x_OpenPort)

Функция **RF25x_OpenPort** открывает COM-порт с заданным символьным именем, заполняет указатель на дескриптор устройства и возвращает результат операции:

```
BOOL  RF25x_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 (RF25x_ClosePort)

Функция **RF25x_ClosePort** закрывает COM-порт и возвращает результат операции:

```
BOOL  RF25x_ClosePort(
            HANDLE              hHandle
);
```

Параметры:

hHandle – дескриптор устройства, полученного от функции RF25x_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 (RF25x_HelloCmd)

Функция **RF25x_HelloCmd** производит идентификацию устройства RF25x сети по сетевому адресу и заполняет структуру ***RF25xHELLOANSWER***:

```
typedef struct _RF25x_HELLO_ANSWER_ {
      BYTE        bDeviceType;
      BYTE        bcDeviceModificaton;
      WORD        wDeviceSerial;
      WORD        wDeviceMaxDistance;
      WORD        wDeviceRange;
} RF25xHELLOANSWER, *LPRF25xHELLOANSWER;
```

Где:

bDeviceType	–	однобайтная величина показывающая тип устройства (для RF25x данная величина равна 65) (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 максимального расстояния для датчика РФ25х (тип **WORD**);

wDeviceRange – двубайтная величина содержащая значение диапазона для датчика РФ25х (тип **WORD**).

```
BOOL RF25x_HelloCmd (
            HANDLE                  hCOM,
            BYTE                    bAddress,
            LPRF25xHELLOANSWER      lprfHelloAnswer
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства;

lprfHelloAnswer – указатель на структуру RF25xHELLOANSWER.

Возвращаемое значение:

Если устройство не ответило на запрос идентификации, функция возвращает FALSE, иначе функция возвращает TRUE и заполняет структуру **RF25xHELLOANSWER**.

1.4. Чтение параметров (RF25x_ReadParameter)

Функция **RF25x_ReadParameter** читает внутренние параметры датчика РФ25х и возвращает текущее значение по адресу параметра:

```
BOOL RF25x_ReadParameter (
            HANDLE                  hCOM,
            BYTE                    bAddress,
            WORD                    wParameter,
            DWORD *                 lpdwValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства;

wParameter – номер параметра, см. табл. 1, детальные значение параметров смотрите в технической документации на датчик РФ25х (стр. 4);

Табл. 1

Параметр	Описание
RF25x_PARAMETER_POWER_STATE	Питание датчика
RF25x_PARAMETER_PRIORITY_AND_SYNC	Приоритет и взаимная синхронизация
RF25x_PARAMETER_NETWORK_ADDRESS	Сетевой адрес
RF25x_PARAMETER_BAUDRATE	Скорость передачи данных через

	последовательный порт
RF25x_PARAMETER_PIXEL_SIZE	Размер пикселя
RF25x_PARAMETER_DATUM_POINT	Начало системы координат
RF25x_PARAMETER_SAMPLING_PERIOD	Период выборки
RF25x_PARAMETER_BEGIN_ANALOG_RANGE	Начало диапазона аналогового выхода
RF25x_PARAMETER_END_ANALOG_RANGE	Конец диапазона аналогового выхода

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 (RF25x_ReadCustomParameter)

Функция **RF25x_ReadCustomParameter** читает внутренние параметры датчика РФ25х и возвращает текущее значение по адресу параметра:

```
BOOL  RF25x_ReadCustomParameter(
            HANDLE                  hCOM,
            BYTE                    bAddress,
            BYTE                    bParameterAddress,
            BYTE                    bParameterSize,
            void *                  lpValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства;

bParameterAddress – адрес внутреннего параметра в датчике РФ25х;

bParameterSize – размер внутреннего параметра в датчике РФ25х;

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 (RF25x_WriteCustomParameter)

Функция **RF25x_WriteCustomParameter** записывает внутренние параметры датчика РФ25х:

```
BOOL  RF25x_WriteCustomParameter(
            HANDLE                  hCOM,
            BYTE                    bAddress,
            BYTE                    bParameterAddress,
            BYTE                    bParameterSize,
            void *                  lpValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства;

bParameterAddress – адрес внутреннего параметра в датчике РФ25х;

bParameterSize – размер внутреннего параметра в датчике РФ25х;

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 (RF25x_WriteParameter)

Функция **RF25x_WriteParameter** записывает внутренние параметры датчика РФ25х:

```
BOOL RF25x_WriteParameter (
            HANDLE                  hCOM,
            BYTE                    bAddress,
            WORD                    wParameter,
            DWORD                   dwValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства;

wParameter – номер параметра, см. табл. 1, детальные значение параметров смотрите в технической документации на датчик РФ25х;

dwValue – значение параметра, которое будет сохранено в датчик РФ25х.

Возвращаемое значение:

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

1.8. Сохранение текущих параметров во FLASH-памяти (RF25x_FlushToFlash)

Функция **RF25x_FlushToFlash** сохраняет все параметры во FLASH-память датчика РФ25х:

```
BOOL  RF25x_FlushToFlash(
            HANDLE      hCOM,
            BYTE        bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_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 (RF25x_RestoreFromFlash)

Функция **RF25x_RestoreFromFlash** восстанавливает значения всех параметров во FLASH по умолчанию:

```
BOOL  RF25x_RestoreFromFlash(
                  HANDLE            hCOM,
                  BYTE              bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_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 (RF25x_LockResult)

Функция **RF25x_LockResult** защелкивает текущее измеренное значение до следующего вызова функции **RF25x_Measure**:

```
BOOL  RF25x_LockResult(
            HANDLE              hCOM,
            BYTE                bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_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 (RF25x_Measure)

Функция **RF25x_Measure** читает из датчика РФ25х текущее измеренное значение. Значение *lpdwValue* в отсчётах по 0.1мкм:

```
BOOL  RF25x_Measure(
            HANDLE              hCOM,
            BYTE                bAddress,
            DWORD *             lpdwValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства.

lpdwValue – указатель на переменную типа ULONG/DWORD, 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 (RF25x_StartStream)

Функция **RF25x_StartStream** переводит датчик РФ25х в режим непрерывной передачи результатов измерений:

```
BOOL  RF25x_StartStream(
            HANDLE          hCOM,
            BYTE            bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_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 (RF25x_StopStream)

Функция **RF25x_StopStream** переводит датчик из режима непрерывной передачи результатов измерений в режим «запрос-ответ»:

```
BOOL  RF25x_StartStream(
            HANDLE          hCOM,
            BYTE            bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_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 (RF25x_GetStreamMeasure)

Функция **RF25x_GetStreamMeasure** читает из входного буфера COM-порта данные, полученные от датчика РФ25х, после успешного выполнения функции RF25x_StartStream. В буфер данные приходят со скоростью, установленной в параметрах датчика РФ25х, т.к. глубина входного буфера ограничена 1024 байтами, то желательно вычитывать данные с периодичностью, равной установленной в параметрах датчика РФ25х Параметр *lpdwValue* идентичен параметру *lpdwValue* в функции RF25x_Measure.

```
BOOL  RF25x_GetStreamMeasure(
            HANDLE                  hCOM,
            DWORD *                 lpdwValue
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

lpdwValue – указатель на переменную типа ULONG/DWORD, содержащую результат D.

Возвращаемое значение:

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

	*Для стабильной работы функции **RF25x_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. Задание начала отсчетов системы координат (RF25x_SetDatumPoint)

Функция **RF25x_SetDatumPoint** задаёт записанное в параметрах значение начала отсчётов системы координат:

```
BOOL  RF25x_SetDatumPoint (
            HANDLE              hCOM,
            BYTE                bAddress
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

bAddress – адрес устройства.

Возвращаемое значение:

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

1.16. Передача пользовательских данных (RF25x_CustomCmd)

Функция **RF25x_CustomCmd** используется для передачи и/или приёма данных от датчика РФ25х.

```
BOOL  RF25x_CustomCmd(
            HANDLE              hCOM,
            char *              pcInData,
            DWORD               dwInSize,
            char *              pcOutData,
            DWORD *             pdwOutSize
);
```

Параметры:

hCOM – дескриптор устройства, полученного от функции RF25x_OpenPort либо CreateFile;

pcInData – указатель на массив данных, который будет передан в датчик РФ25х. 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 РФ25х. 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            hRF25x      = INVALID_HANDLE_VALUE;
DWORD             dwValue;
DWORD             dwMeasured;
RF25xHELLOANSWER  hlans;

// Чистим структуру RF25xHELLOANSWER
memset(&hlans, 0x00, sizeof(RF25xHELLOANSWER));

// Открываем COM-порт
if (!RF25x_OpenPort("COM1:", CBR_115200, &hRF25x)
      return (FALSE);

// Опрашиваем устройство
if (RF25x_HelloCmd( hRF25x, 1, &hlans ))
{

      ////////////////////////////////////////////////
      //                                            //
      // После успешного выполнения RF25x_HelloCmd  //
      // в структуре hlans содержится информация о  //
      // датчике РФ25х, ответившем на запрос        //
      //                                            //
      ////////////////////////////////////////////////

      //Читаем параметр: Период выборки
      RF25x_ReadParameter(
                           hRF25x,
                           1,
                           RF25x_PARAMETER_SAMPLING_PERIOD,
                           &dwValue
                     );

            /* В dwValue содержится значение периода выборки */

      //Получаем измерение из датчика РФ25х
      RF25x_Measure( hRF25x, 1, &dwMeasured );

      /* В dwMeasured содержится измеренный результат */

}

RF25x_ClosePort( hRF25x );
```

	*Для работы с DLL можно также использовать функции **LoadLibrary** для загрузки библиотеки и функцию **GetProcAddress** для получения указателя на функцию.*

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

При работы с USB устройствами на FTDI-микросхемах, 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 **RF25x_Measure** для последовательного порта и **RF25x_FTDI_Measure** для устройств с FTDI USB. Единственное отличие лишь при работе с функцией **RF25x_FTDI_OpenPort** заключается в передаче в качестве имени порта - символьную ссылку.

2.1. Подключение к USB через FTDI (RF25x_FTDI_OpenPort)

Функция **RF25x_FTDI_OpenPort** открывает соединение через USB с заданным символьным именем, заполняет указатель на дескриптор устройства и возвращает результат операции:

```
BOOL  RF25x_FTDI_OpenPort(
            LPCSTR      lpPort_Name,
            DWORD       dwSpeed,
            FT_HANDLE * lpftHandle
);
```

Параметры:

lpPort_Name – название USB-устройства (например “*D2XX Recovery PID for XP*”), полный синтаксис при задании имени USB-устройства можно посмотреть в **D2XX Programmers Guide**, функция FT_W32_CreateFile выполняемая с параметром FT_OPEN_BY_DESCRIPTION;

dwSpeed – скорость работы через USB-устройство. Параметр идентичен полю BaudRate в структуре FTDCB подробно описанной в **D2XX Programmers Guide**;

lpftHandle – указатель на дескриптор устройства;

Возвращаемое значение:

Если соединение через USB-порт не открыто, и/или настроить его не удалось, функция вернёт FALSE иначе, если соединение через USB-порт открыто и настроено успешно – функция вернёт TRUE. Более детальные сведения об возвращаемых ошибках можно получить с помощью функции FT_W32_GetLastError, описанной в **D2XX Programmers Guide**.