Данный проект демонстрирует пример использования библиотеки SysSocket для реализации TCP-сокетов на ПЛК NLScon-RSB. В примере рассматривается возможность реализации сокета клиента и сокета сервера на ПЛК NLScon-RSB для обмена данными с виртуальным ПЛК Codesys Control Win V3 по сети Ethernet

Для разработки проекта использована среда Codesys 3.5.16 patch 4 и установочные пакеты компонентов для модулей RealLab! CoDeSys Linux package.

При подготовке примера были использованы следующие материалы:

1. Программирование сокетов в CODESYS V3 [Электронный ресурс] // Режим доступа : https://www.youtube.com/watch?v=ThVLXygHnnU

2. Н. Кузьмина. Реализация TCP- и UDP-сокетов на контроллере FASTWEL CPM723-01 в среде разработки CODESYS V3 // Современные технологии автоматизации. — 2018. — №3. С. 80–90.

Ссылка на электронную версию статьи: https://www.cta.ru/articles/spravochnik/v-zapisnuyu-knizhku-inzhenera/124410/?sphrase_id=5258813

Для работы с сокетами понадобятся следующие библиотеки Codesys:

- SysSocket;
- CmpErrors.

Для описания совместной работы серверного и клиентского TCP-сокетов удобно использовать схему рис. 1. Указанная схема предполагает закрытие клиентского сокета каждый раз после получения ответа. Как отмечено в [2], такая реализация работы сокетов не является эффективной, но в нашем случае она подходит для демонстрации работы TCP-сокетов сервера и одного клиента.

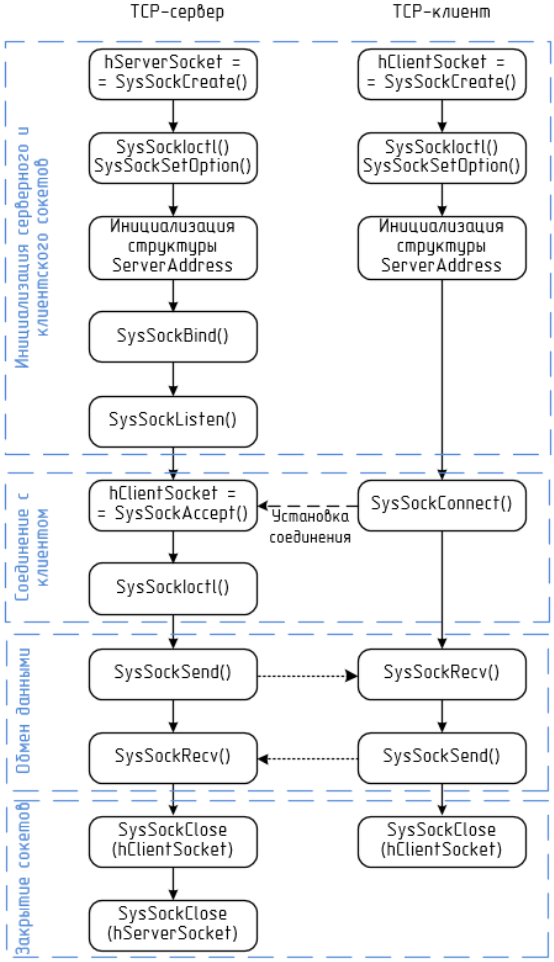

Рис. 1. Схема работы серверного и клиентского TCP-сокетов

Реализация TCP-сокета сервера

Работу с сокетом проще всего описать последовательностью шагов: создание сокета сервера, настройка опций, привязка к IP-адресу и порту, включение режима прослушивания, прием запроса на соединение («звонка») от клиента, обмен данными, закрытие сокета клиента, закрытие сокета сервера.

Первые несколько шагов можно объединить в блок, отвечающий за инициализацию серверного сокета. Он будет включать использование следующих функций библиотеки SysSocket:

SysSockCreate — функция создания нового сокета. Возвращает дескриптор сокета

Scope	**Name**	**Type**	**Comment**
Return	SysSockCreate	RTS_IEC_HANDLE	Дескриптор нового сокета.
Input	iAddressFamily	INT	Семейство адресов сокета (SOCKET_AF_INET)
	diType	DINT	Тип сокета (SOCKET_STREAM)
	diProtocol	DINT	Протокол сокета (SOCKET_IPPROTO_TCP)
	pResult	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки выполнения функции (см. CmpErrors.library)

SysSockIoctl -- функция, контролирующая режим работы сокета.

Scope	**Name**	**Type**	**Comment**
Return	SysSockIoctl	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	hSocket	RTS_IEC_HANDLE	Дескриптор сокета
	diCommand	DINT	Команда (SOCKET_FIONBIO — это команда управления блокирующим режимом*)
	pdiParameter	POINTER TO DINT	Значение параметра команды (16#1)

* Блокирующий (синхронный) режим работы сокета позволяет функциям типа SysSockAccept(), SysSockSend(), SysSockRecv() и некоторым другим выполняться, не возвращая управление программному коду, пока не будет получен результат функции. Например, при случайном обрыве связи по Ethernet блокирующий режим приведет к «зависанию» программного кода на выполнении одной из перечисленных функций. Чтобы исключить такую ситуацию, сокет переводят в неблокирующий режим (как в данном примере).

SysSockSetOption — настройка сокета

Scope	**Name**	**Type**	**Comment**
Return	SysSockSetOption	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	hSocket	RTS_IEC_HANDLE	Дескриптор сокета
	diLevel	DINT	Уровень сокета (SOCKET_SOL)
	diOption	DINT	Опция (SOCKET_SO_REUSEADDR*)
	pdiOptionValue	POINTER TO DINT	Указатель на значение опции (значение 16#1)
	diOptionLen	DINT	Длина значения в байтах

* SOCKET_SO_REUSEADDR — опция, отвечающая за повторное использование локального адреса, уже используемого другим открытым сокетом. Эта опция позволяет исключить появление ошибки ERR_SOCK_ADDRINUSE = 16#207.

SysSockInetAddr — конвертирует IP-адрес, заданный в строковой переменной, в 4 байтовый формат.

Scope	Name	Type	Comment
Return	SysSockInetAddr	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	szIPAddress	REFERENCE TO STRING	Строковая переменная с адресом
	pInAddr	POINTER TO UDINT	Указатель на поле структуры SOCKADDRESS.sin_addr.ulAddr

SysSockHtons — конвертирует номер порта, возвращая номер с сетевым порядком байтов.

Scope	Name	Type	Comment
Return	SysSockHtons	WORD	Результат конвертирования
Input	usHost	WORD	Значение номера порта

SysSockBind — функция выполняет привязку сокета к адресу и порту, которые были предварительно заданы в структуре SOCKADDRESS.

Scope	Name	Type	Comment
Return	SysSockBind	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	hSocket	RTS_IEC_HANDLE	Дескриптор сокета
	pSockAddr	POINTER TO SOCKADDRESS	Адрес сокета (указатель на предварительно сформированную структуру SOCKADDRESS)
	diSockAddrSize	DINT	Размер структуры адреса сокета

SysSockListen — функция «включает» прослушивание входящих клиентских соединений.

Scope	Name	Type	Comment
Return	SysSockListen	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	hSocket	RTS_IEC_HANDLE	Дескриптор сокета
	diMaxConnections	DINT	Максимальное количество клиентских подключений

Примеры использования перечисленных функций приведены на рис. 2.

```
hServerSocket := SysSockCreate(SysSocket.GVL.SOCKET_AF_INET, SysSocket.GVL.SOCKET_STREAM, SysSocket.GVL.SOCKET_IPPROTO_TCP, ADR(Result));

Result := SysSockIoctl(hServerSocket, SysSocket.GVL.SOCKET_FIONBIO, ADR(mode));

Result := SysSockSetOption(hServerSocket, SOCKET_SOL, SOCKET_SO_REUSEADDR, ADR(mode), SIZEOF(mode));

ServerAddress.sin_family := SysSocket.GVL.SOCKET_AF_INET;

Result := SysSockInetAddr(sIPaddr, ADR(ServerAddress.sin_addr.ulAddr));

ServerAddress.sin_port := SysSockHtons(wHostPort);

Result:= SysSockBind(hServerSocket, ADR(ServerAddress), SIZEOF(ServerAddress));

Result:= SysSockListen(hServerSocket, 10);
```

Рис. 2. Функции для инициализации серверного сокета

Следующий шаг алгоритма — прием запроса на соединение от клиента. На этом этапе требуется одна основная функция:

SysSockAccept — функция принимает запрос на следующее входное клиентское соединение, формируя дескриптор для нового клиентского соединения

Scope	Name	Type	Comment
Return	SysSockAccept	RTS_IEC_HANDLE	Дескриптор «принятого» клиента или значение 16#FFFFFFFF при отсутствии принятого клиентского соединения.
Input	hSocket	RTS_IEC_HANDLE	Дескриптор **серверного сокета**
	pSockAddr	POINTER TO SOCKADDRESS	Указатель на структуру адреса подлкюченного клиента
	pdiSockAddrSize	POINTER TO DINT	Указатель на переменную,
	pResult	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки выполнения функции (см. CmpErrors.library)

Если функция SysSockAccept() возвращает значение, отличное от 16#FFFFFFFF, значит клиентское соединение успешно принято для обмена данными. Тогда следует включить неблокирующий режим для этого клиента с помощью функции SysSockIoctl(), указав в ее параметрах дескриптор клиента.

Следующий шаг выполняет непосредственно обмен данными. Для этого используются функции SysSockSend() и SysSockRecv().

SysSockSend — функция отправляет данные в TCP сокет

Scope	Name	Type	Comment
Return	SysSockSend	__XINT	Количество переданных байт. 0 в случае ошибки передачи
Input	hSocket	RTS_IEC_HANDLE	Дескриптор **клиентского сокета**
	pbyBuffer	POINTER TO BYTE	Буфер данных, подготовленных для отправки
	diBufferSize	__XINT	Максимальный размер буфера
	diFlags	DINT	Параметр, позволяющий задать дополнительные опции для функции (SOCKET_MSG_NONE — без доп. опций)
	pResult	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки выполнения функции (см. CmpErrors.library)

SysSockRecv — функция принимает данные из TCP сокета:

Scope	Name	Type	Comment
Return	SysSockRecv	__XINT	Количество принятых байт. 0 в случае ошибки передачи
Input	hSocket	RTS_IEC_HANDLE	Дескриптор **клиентского сокета**
	pbyBuffer	POINTER TO BYTE	Буфер для приема данных
	diBufferSize	__XINT	Максимальный размер буфера
	diFlags	DINT	Параметр, позволяющий задать дополнительные опции для функции (SOCKET_MSG_NONE — без доп. опций)
	pResult	POINTER TO RTS_IEC_RESULT	Указатель на код ошибки выполнения функции (см. CmpErrors.library)

Если дальнейший обмен данными с клиентом не предусмотрен, то следует закрыть его сокет. На последнем шаге функция SysSockClose выполняет закрытие и может быть использована для закрытия как клиентского, так и серверного сокета.

Scope	Name	Type	Comment
Return	SysSockClose	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	hSocket	RTS_IEC_HANDLE	Дескриптор сокета

Реализация TCP-сокета клиента

Работа клиентского сокета также может быть описана последовательностью шагов: инициализация сокета, подключение к серверу, обмен данными и закрытие сокета.

Настройка клиента в основном предполагает использование тех же функций, что и для сервера, но с декстриптором сокета клиента (рис. 3).

```
    ClientAddress.sin_family    := SysSocket.GVL.SOCKET_AF_INET;
// номер порта сервера
    ClientAddress.sin_port      := SysSockHtons(usHost := wHostPort);
// конвертировать IP-адрес сервера в байтовый формат
    Result := SysSockInetAddr(sIPaddr, ADR(ClientAddress.sin_addr.ulAddr));
// создать клиентский сокет
    hClientSocket := SysSockCreate(SysSocket.GVL.SOCKET_AF_INET,
                                   SysSocket.GVL.SOCKET_STREAM,
                                   SysSocket.GVL.SOCKET_IPPROTO_TCP,
                                   ADR(Result));
// включить для клиентского сокета неблокирующий режим
    Result := SysSockIoctl(hClientSocket, SysSocket.GVL.SOCKET_FIONBIO, ADR(mode));
```

Рис. 3. Настройка клиентского сокета

На втором шаге, подключении к серверу, со стороны клиента используется функция SysSockConnect(): подключиться к TCP-серверу как клиент. Функция отправляет запрос на подключение, который на сервере принимается функцией SysSockAccept().

SysSockConnect()

Scope	Name	Type	Comment
Return	SysSockConnect	RTS_IEC_RESULT	Код ошибки выполнения функции (см. CmpErrors.library)
Input	hSocket	RTS_IEC_HANDLE	Дескриптор клиентского сокета
	pSockAddr	POINTER TO SOCKADDRESS	Указатель на структуру SOCKADDRESS, в которой содержится адрес сервера
	diSockAddrSize	DINT	Размер структуры SOCKADDRESS

При обмене данными и закрытии клиентского сокета в программе TCP-клиента используются те же функции, что и у сервера, подобным образом.

В данном примере можно использовать ПЛК NLScon-RSB как TCP-сервер и как TCP-клиент. Если NLScon-RSB выполняет серверные функции, то клиентом может быть запущенный на ПК Codesys Control Win V3, и наоборот, если NLScon-RSB – клиент, то сервер — виртуальный контроллер. При этом следует не забывать в серверной и клиентской программах менять IP-адрес на адрес сервера (см. рис. 4).

PRG_server

```
6       // структура для хранения адреса и порта сервера...
7       ServerAddress : SysSocket.SOCKADDRESS;
8       // ...и клиента
9       ClientAddress : SysSocket.SOCKADDRESS;
10
11      // результат выполнения функции
12      Result : SysSocket.SysSocket_Interfaces.RTS_IEC_RESULT;
13
14      //строковый IP-адрес сервера в сети Ethernet
15      sIPaddr : STRING := '192.168.0.91';
16      // номер порта, к которому будет обращаться клиент
17      wHostPort : WORD := 504;
```

PRG_client

```
1   PROGRAM PRG_client
2   VAR
3       // обработчик TCP-сокета клиента
4       hClientSocket : SysSocket.SysSocket_Interfaces.RTS_IEC_HANDLE;
5       // структура для хранения адреса и порта сервера
6       ServerAddress : SysSocket.SOCKADDRESS;
7
8       //строковый IP-адрес сервера в сети Ethernet
9       sIPaddr : STRING(16) := '192.168.0.91';
10      // номер порта сервера, к которому будет обращаться клиент
11      wHostPort : WORD := 504;
12
```

Рис. 4. Настройка IP-адреса сервера в серверном и клиентском приложениях