Описание приборов 4L и 4M, а также протоколов для взаимодействия с ними.

Для нашей работы прибор можно мыслить как коллекцию элементов со специфическими операциями чтения и записи, т.е. как специфическую структуру данных. Рассмотрим общие принципы ее организации:

Вся информация внутри прибора хранится в виде тэгов (пока мыслите их как переменные примитивных типов) и архивов. Если коротко, то тэги хранят текущую информацию о состоянии прибора, а архивы – состояние прибора в определенные моменты прошлого (например есть архив, сохраняющий ‘снимок’ значений всех тэгов каждый час).

1) Организация тэгов

Все тэги внутри прибора лежат в специальных контейнерах, именуемых каналами. Из канала тэг можно получить, зная его номер (т.е. канал можно мыслить как хэш-таблицу, ключи которой числа). В свою очередь каждый канал также имеет свой уникальный номер. Таким образом, каждый тэг однозначно идентифицируется двумя числами: его номером в канале и номером самого канала в приборе.

В памяти прибора тэг представляет из себя просто какое-либо значение примитивного типа (int,float,string ,int[] и т.д.), а также иногда информацию о том, является ли это значение изменяемым.

Допустим мы хотим считать все тэги какого-нибудь прибора. Как нам это сделать? Мы ведь не знаем ни номеров каналов, ни номеров тэгов в каждом из них. Также у нас возникнут проблемы если мы захотим понять смысл значения тэга, который мы собрались считать или записать.

Для решения этих проблем в программе используется база данных(файл logikaEx открывать любым mdb opener), которая для каждого прибора хранит описание всех его тэгов. Вид этого описания представлен на рисунке 1.

Device	Channel	Ordinal	Kind	Basic	DataType	dbType	UpdateRate	Name	DisplayFormat	Description	VarT	Units	DescriptionEx	Range
SPT941_20	ОБЩ	1038	Realtime	False	Single		10	t4	0.00	Температу...		°C	Температура, из...	
SPT941_20	ОБЩ	1039	Realtime	False	Single		10	P1	0.000	Давление т...		[P]	Значения не ис...	
SPT941_20	ОБЩ	1040	Realtime	False	Single		10	P2	0.000	Давление т...		[P]	Значения не ис...	
SPT941_20	ОБЩ	1041	Realtime	False	Single		10	P3	0.000	Давление т...		[P]	Значения не ис...	
SPT941_20	ОБЩ	1042	Realtime	False	Single		10	Px	0.000	Давление х...		[P]	Значение не ис...	
SPT941_20	ОБЩ	1043	Realtime	False	Single		10	P4	0.000	Давление п...		[P]	Значение этого ...	
SPT941_20	ОБЩ	1044	Realtime	False	Int32[]	binary(8)	10	ДС		Сборка фл...			Номера активн...	
SPT941_20	ОБЩ	1045	Realtime	False	Int32[]	binary(8)	10	НС		Сборка фл...			Номера активн...	
SPT941_20	ОБЩ	2048	TotalCtr	False	Double		10	V1	0.00	объем тепл...	V	м³	Измеряется с н...	
SPT941_20	ОБЩ	2049	TotalCtr	False	Double		10	V2	0.00	объем тепл...	V	м³	Измеряется с н...	
SPT941_20	ОБЩ	2050	TotalCtr	False	Double		10	V3	0.00	объем тепл...	V	м³	Измеряется с н...	
SPT941_20	ОБЩ	2051	TotalCtr	False	Double		10	M1	0.00	масса тепл...	M	т	Вычисляется с ...	
SPT941_20	ОБЩ	2052	TotalCtr	False	Double		10	M2	0.00	масса тепл...	M	т	Вычисляется с ...	
SPT941_20	ОБЩ	2053	TotalCtr	False	Double		10	M3	0.00	масса тепл...	M	т	Вычисляется с ...	
SPT944	ОБЩ	1027	Realtime	False	Single		10	tx	0.00	Температу...	T	°C	Температура хо...	
SPT944	ОБЩ	1028	Realtime	False	Single		10	tв	0.00	Температу...	T	°C	Температура во...	
SPT944	ОБЩ	1029	Realtime	False	Single		10	t4	0.00	Температу...	T	°C	Температура, из...	
SPT944	ОБЩ	1030	Realtime	False	Single		10	t5	0.00	Температу...	T	°C	Температура, из...	
SPT944	ОБЩ	1031	Realtime	False	Single		10	t6	0.00	Температу...	T	°C	Температура, из...	
SPT944	ОБЩ	1032	Realtime	False	Single		10	Px	0.000	Давление х...	P	[P]	Давление холод...	
SPT944	ОБЩ	1033	Realtime	False	Single		10	P4	0.000	Давление P4	P	[P]	Давление, изме...	
SPT944	ОБЩ	1034	Realtime	False	Single		10	P5	0.000	Давление P5	P	[P]	Давление, изме...	
SPT944	ОБЩ	1035	Realtime	False	Single		10	P6	0.000	Давление P6	P	[P]	Давление, изме...	
SPT944	ОБЩ	1036	Realtime	False	Int32[]	binary(...	10	НС		Сборка фл...			Номера активн...	
SPT944	ОБЩ	1037	Realtime	False	Int32[]	binary(...	10	ДС		Сборка фл...			Номера активн...	
SPT944	ОБЩ	2048	TotalCtr	False	Double		10	Qc	0.00	Суммарная...	W	[Q]	Суммарное кол...	

Рисунок 1.

Разберем основные колонки этой таблицы:

Device: модель прибора, которому принадлежит тэг.

Channel: тип канала, на котором лежит тэг. Тут стоит сделать небольшое отступление и рассказать немного о каналах: У каждого канала есть свой тип. Возможные варианты: ОБЩ(также обозначается как 0), г(группа), п(потребитель),т(трубопровод), к(канал). Для нас не важно, что эти типы значат на функциональном уровне, можно мыслить их просто как некую дополнительную характеристику. Информация о том, сколько каналов содержит прибор и каковы их типы также содержится в базе данных (Рисунок 2).

Device	Key	Description	Start	Count
SPG762_1	к	доп.канал	1	16
SPG762_1	п	потребитель	1	6
SPG762_1	т	трубопровод	1	12
SPG763	0	системный канал	0	1
SPG763	п	потребитель	1	2
SPG763	т	трубопровод	1	3
SPG763_1	0	системный канал	0	1
SPG763_1	к	доп.канал	1	16
SPG763_1	п	потребитель	1	6
SPG763_1	т	трубопровод	1	12
SPT940	ОБЩ	системный канал	0	1
SPT941	ОБЩ	общий канал	0	1
SPT941_10	ОБЩ	общий канал	0	1
SPT941_20	ОБЩ	системный канал	0	1
SPT942	ОБЩ	общий канал	0	1
SPT942	ТВ	тепловой ввод	1	2
SPT943	ОБЩ	общий канал	0	1
SPT943	ТВ	тепловой ввод	1	2
SPT943rev3	ОБЩ	общий канал	0	1
SPT943rev3	ТВ	тепловой ввод	1	2
SPT944	ОБЩ	общий канал	0	1
SPT944	ТВ	тепловой ввод	1	2
SPT961	0	системный	0	1
SPT961	п	магистраль	1	2
SPT961	т	трубопровод	1	5
SPT961_1	0	системный канал	0	1

Рисунок 2.

Возьмем для примера прибор SPT942. Как видно из таблицы, у него 1 канал типа ОБЩ(канал этого типа должен быть у всех приборов и при том в единственном экземпляре) и 2 канала типа ТВ(start- начало нумерации для каналов данного типа, count-количество каналов данного типа. Эти два поля как раз и обеспечивают каждому каналу уникальный номер).

Важная особенность: два канала с одинаковым типом содержат одни и те же тэги в логическом смысле(например если в канале ТВ с номером 1(ТВ1) тэг с номером n содержит значение текущей температуры, то в ТВ2 тэг с номером n также будет хранить значение температуры, только уже по другому(второму) трубопроводу).

Поэтому на рисунке 1 в колонке Channel используется не номер канала, а его тип: по типу можно будет получить номера всех каналов, которые ему соответствуют, и на каждом таком канале будет лежать описываемый тэг.

Ordinal: номер тэга в канале.

Kind: тип тэга. Возможные варианты: Настроечный(parameter), текущий(Realtime),тотальный(TotalCtr) и информационный(info). Настроечные тэги хранят в себе информацию о настройках прибора, текущие- данные, измеряемые прибором, а информационные содержат данные о самом приборе(например его серийный номер). Смысл тотальных параметров я не понял, но скорее всего они являются неким специальным подтипом текущих параметров. Только настроечные тэги в памяти прибора дополнительно хранят информацию о своей изменяемости. Другие типы тэгов неизменяемы.

DataType: тип данных для значения тэга.

Name: имя тэга. Можно считать символьной альтернативной номеру тэга в канале (внутри одного канала имена уникальны).

Description: описание смысла тэга.

Units: Физические единицы измерения значения тэга(например секунды, если тэг хранит в своем значении какое-либо время).

Остальные характеристики тэгов не так важны (Например есть метаданные о формате отображения тэга на экране прибора, скорость обновления его значения, возможный диапазон принимаемых значений и т.п.). Не все характеристики тэга могут быть определены (например, если тэг не хранит значение физической величины, его характеристика Units равна null).

Набор характеристик, описывающих тэг, отличается в зависимости от семейства прибора, который этот тэг содержит. На рисунке 1 приведены характеристики тэгов внутри приборов 4M(4 modern). В 4L(4 legacy) в описание тэга также вшиваются дополнительные данные, необходимые для считывания(что за данные неважно, т.к. вы все равно будете использовать высокоуровневый API). В семействе 6 также есть свои особенности, но здесь прибора этого типа вообще не будут рассмотрены.

Основные характеристики, описанные выше, есть у всех тэгов как из 4M, так и 4L.

Программной аналогией строки в базе данных выступают объекты классов TagDef4M(для тэгов из семейства 4M) и TagDef4L(для тэгов из семейства 4L): именно по строкам базы данных эти объекты и строятся.

Из вышеописанного видно, что данные объекты не являются тэгами (они даже не содержат поле для хранения значения тэга). Они лишь хранят информацию, необходимую для его считывания и интерпретации значения. Точнее, они бы делали это, если бы каждая строка в бд описывала только один тэг. Но это не так (как уже упоминалось ранее, значение поля channel может определять не один, а сразу несколько каналов. Получается, что один из двух индексов, идентифицирующих тэг, оказывается массивом). Каждая строка определяет сразу множество тэгов.

Программной аналогией понятия тэга выступает класс **DataTag.** Он имеет следующие поля:

1) TagDef def и Channel Channel (Tagdef-общий класс для TagDef4M и TagDef4L, Channel – класс для описания канала, содержит тип канала и его номер.) Эти два поля уже задают описание для конкретного тэга.

2) object Value - Поле, в которое будет записано значение тэга при его считывании из памяти прибора

3) bool oper – поле в которое запишется информация об изменяемости данного тэга.

4) TimeStamp – время последнего обновления значения данного тэга.

Вообще объекты DataTag используются только при считывании тэгов прибора. В метод передаются пустые/устаревшие экземпляры (с пустыми/устаревшими полями value и oper), а возвращаются уже

заполненные/обновленные. Конечно, и пустые DataTag уже содержат в себе кучу информации, и возникает резонный вопрос: откуда их взять? Не вручную же по базе данных составлять. Ответ на это будет дан при описании методов для считывания тэгов.

> С основами разобрались, теперь рассмотрим какие действия необходимо совершить для выполнения считывания/записи значений тэгов. Для этих целей будет использоваться класс M4Protocol.
>
> *Справка: Сетево́й протоко́л — набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами.*

Для того, чтобы считать/записать тэг, надо сначала подключиться к прибору. Это подключение состоит из 2 логических уровней. Первый уровень определяет способ, по которому приборы объединены в одну сеть. Это может быть TCP, UDP протоколы и т.д. Второй уровень уже определяет формат общения приборов поверх первого уровня (например в построенной первым уровнем сетью каждому прибору присваивается адрес(это адрес именно 2 уровня, не путать с каким-нибудь ip адресом!)). Протокол первого уровня, вместе с требуемым от него интерфейсом, определяет абстрактный класс Connection (TCPConnection, UDPConnection уже являются его реализациями). Если грубо, то этот интерфейс можно мыслить, как стандартный поток чтения/записи. Протоколы 2 уровня используют для отправки сообщений именно класс Connection, т.к. им не важно, как именно мы организовали связь с приборами, важно лишь чтобы поддерживался определенный интерфейс (каждый протокол 2 уровня имеет поле Connection cn, которое он использует для отправки и получения своих запросов). Протоколов 2 уровня три штуки: M4, СПСсеть и протокол для связи с Legacy

приборами. В библиотеке, правда, протокол для связи с Legacy интегрирован в M4 и в итоге класс M4Protocol содержит методы их обоих.

Т.к. в данном документе рассматриваются только приборы 4L и 4M, то здесь будет рассмотрен только класс M4Protocol(в библиотеке он поделен на 2 файла: M4Protocol и M4Protocol_API.). Описания протоколов здесь приведены не будут, т.к. вы будете пользоваться ими не напрямую, а через специальные методы, которые представляют более удобный интерфейс и мощную функциональность.

Чтобы пользоваться методами, описанными далее, необходимо проделать следующие шаги (тут могут быть неточности т.к. я сам эти методы ни разу не запускал.):

1) Создать объект класса, определяющий протокол первого уровня (например, TCPConnection). (возможно с ним затем придется выполнить еще какие-нибудь операции, как, например, открытие соединения, но я это еще не разбирал).
2) Создать объект класса M4Protocol (есть конструктор без параметров) и затем инициализировать его поле Connection объектом, созданным на предыдущем шаге.

1) Запись тэгов

Производится методами WriteParameterL4 и WriteParamsM4, расположенными в файле M4Protocol.cs

1.1) WriteParameterL4

Служит для записи тэга в прибор из семейства 4L. В качестве параметров принимает класс-модель прибора из семейства 4L(mtr), адрес прибора(nt), номер канала(channel), номер тэга(nParam), строку, представляющую значение тэга(у 4L приборов все тэги имеют значение

string) value , и bool operflag- информацию, сделать ли записываемый параметр изменяемым или нет.

Немного об классах-моделях: для каждой модели прибора в программе есть класс, ей соответствующий (папка Meters). Иерархия наследования для этих классов представлена на рисунке 3.

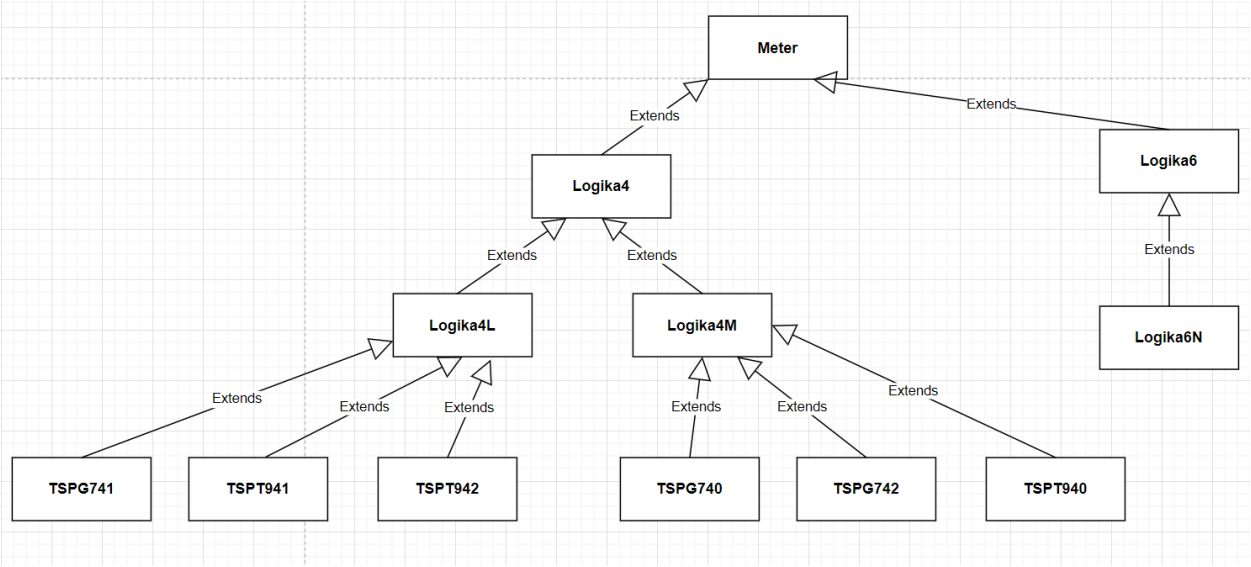


Рисунок 3.

Данная схема является упрощенной: приборы, наследуемые от Logika6N на ней вообще не указаны, а для Logika4L и Logika4M их список был сокращен.

Создание класса прибора: пусть мы хотим создать объект, соответствующий прибору СПТ 940. В библиотеке есть класс TSPT940, поэтому можно сделать так: new TSPT940(). Однако этот конструктор доступен только внутри сборки библиотеки, поэтому лучше воспользоваться другим способом: в классе Meter есть статический метод, который по имени прибора возвращает соотв. ему объект: Meter.FromTypeString(‘TSPT940’).

nt- адрес прибора в сети. Понятно, что информации о модели прибора недостаточно для однозначной его идентификации: в сети вполне может быть, скажем, три СПТ 940. Поэтому каждому прибору дополнительно присваивается спец. адрес (он используется протоколами 2 уровня).

Значения оставшихся 4 параметров уже были описаны ранее.

Стоит заметить, что записывать можно только значения настроечных тэгов (т.к. остальные типы тэгов неизменяемы).

Данный метод возвращает ошибку записи в качестве объекта перечисления ErrorCode, если таковая возникла.

1.2) WriteParamsM4

Служит для записи массива настроечных тэгов в прибор семейства 4M.В качестве параметров принимает класс-модель прибора из семейства 4M(m), адрес прибора(nt) и массив объектов TagWriteData, каждый элемент которого однозначно определяет тэг и значение, которое мы хотим в него записать. Возвращает массив байт, каждый элемент которого либо равен null, если при записи тэга с соответствующим номером проблем не возникло, либо содержит однобайтовый код ошибки (описание этих кодов есть в файле описания протокола M4).

Прежде чем закончить с записью, подробнее рассмотрим понятие оперативности(изменяемости) параметра.

Как уже упоминалось ранее, можно изменять значения только для настроечных тэгов. В приборе есть два режима работы: защищенный и незащищенный. В незащищённом можно менять значения тэгов, в независимости от того, являются ли они оперативными или нет. Если же режим защищенный, то можно изменять только оперативные параметры (то есть, по идее, если записать с помощью WriteParameterL4 какой-нибудь оперативный тэг с operFlag = false, то этот тэг перестанет быть оперативным, и мы не сможем изменять его значение, пока не выключим защищенный режим).

Две вышеописанные функции можно использовать чтобы изменять оперативность тэга, без изменения его значения. Можно ли этим способом в

защищенном режиме сделать из неоперативного тэга оперативный мне неизвестно, надо будет попробовать.

2) Считывание тэгов

Производится методом UpdateTags, расположенном в файле M4Protocol_API.cs . В качестве параметров он принимает src(в этом протоколе это не нужно, просто передайте null), dst(то же самое,что и nt), и массив определений тэгов DataTag, значения которых необходимо обновить.

Вернемся здесь к вопросу получения пустых DataTag для конкретного прибора. Пусть у нас есть прибор СПТ943rev3, и мы хотим получить его настроечный параметр с именем СП (за имя тэга отвечает колонка бд name), который находится на канале ТВ2. Для получения соответствующего объекта DataTag нужно:

1) Создать объект класса TSPT943rev3(TSPT943rev3 device = new TSPT943rev3())

2) Сделать вызов device.Tags.find(“ТВ”, “СП”). Первым параметром передается имя типа канала, а вторым – имя тэга. Этот метод возвращает объект типа DataTagDef.
3) По полученному DataTagDef построить объект DataTag:

DataTag res = new DataTag(device.Tags.find(“ТВ”, “СП”), 2).

Первым аргументом в конструктор передается DataTagDef, а вторым номер канала.

После получения DataTag зная адрес прибора, уже можно воспользоваться UpdateTags.

3) Организация Архивов

Каждый архив привязан к одному каналу и содержит информацию о состоянии этого канала в определенные моменты прошлого. Все архивы делятся на 2 основных подтипа: интервальные и сервисные.

1) Интервальные архивы

Сохраняют значения ключевых тэгов в канале каждый определенный интервал времени. По этому интервалу интервальные архивы делятся на часовые, суточные, декадные(каждые 10 дней) и месячные. Также еще есть контрольный архив(только для М4 приборов): он генерируется каждые сутки и отличается от суточного большим количеством сохраняемых тэгов. Структуру интервального архива можно мыслить как таблицу.

Пример:

Таблица 1.

t	V1	V2	V3
02.04.2023. 13:00	25	40	5
02.04.2023. 14:00	29	17	6
02.04.2023. 15:00	31	2	6
02.04.2023. 16:00	33	47	8

В таблице 1 приведен пример часового архива для какого-нибудь канала (например, для ОБЩ). V1, V2, V3- имена для тэгов в этом канале, которые подвергаются архивированию.

Одному каналу может соответствовать несколько интервальных архивов с разным шагом архивирования.

2) Сервисные архивы

Эти архивы используются для регистрации изменений значений

настроечных тэгов и нештатных ситуаций по каналу. Записи в этом типе архивов производятся при возникновении соответствующего изменения/ошибки, а не через определенные интервалы времени. Это семейство делится на ErrorsLog(ошибки) и ParamsLog(изменения в настроечной бд).

Если предыдущий тип архивов представлялся как таблица, то тут будет уместнее сравнение со списком.

Информация о том, какие архивы содержат каналы прибора, также есть в базе данных.

Device	ArchiveType	Channel	RecordType	Name	Description	Capacity
SPG742	Month	ОБЩ	Object[]	арх_М	месячный архив	100
SPG742	ParamsLog	ОБЩ	String	ИЗМ	архив изменений БД пр...	500
SPT940	Control	ОБЩ	Object[]	арх_К	контрольный архив	400
SPT940	Day	ОБЩ	Object[]	арх_С	суточный архив	400
SPT940	ErrorsLog	ОБЩ	String	НСа	архив нештатных ситуац...	2000
SPT940	Hour	ОБЩ	Object[]	арх_Ч	часовой архив	2000
SPT940	Month	ОБЩ	Object[]	арх_М	месячный архив	100
SPT940	ParamsLog	ОБЩ	String	ИЗМ	архив изменений БД пр...	2000
SPT941_20	Control	ОБЩ	Object[]	арх_К	контрольный архив	400
SPT941_20	Day	ОБЩ	Object[]	арх_С	суточный архив	400
SPT941_20	ErrorsLog	ОБЩ	String	НСа	архив нештатных ситуац...	2000
SPT941_20	Hour	ОБЩ	Object[]	арх_Ч	часовой архив	2000
SPT941_20	Month	ОБЩ	Object[]	арх_М	месячный архив	100
SPT941_20	ParamsLog	ОБЩ	String	ИЗМ	архив изменений БД пр...	2000
SPT943rev3	Control	ТВ	Object[]	арх_К	контрольный архив	400
SPT943rev3	Day	ТВ	Object[]	арх_С	суточный архив	400
SPT943rev3	ErrorsLog	ТВ	String	НСа	архив нештатных ситуац...	1024
SPT943rev3	Hour	ТВ	Object[]	арх_Ч	часовой архив	2000
SPT943rev3	Month	ТВ	Object[]	арх_М	месячный архив	100
SPT943rev3	ParamsLog	ТВ	String	ИЗМ	архив изменений БД пр...	1024
SPT944	Control	ОБЩ	Object[]	арх_К	контрольный архив	400
SPT944	Day	ОБЩ	Object[]	арх_С	суточный архив	400
SPT944	ErrorsLog	ОБЩ	String	НСа	архив нештатных ситуац...	4000
SPT944	Hour	ОБЩ	Object[]	арх_Ч	часовой архив	2000
SPT944	Month	ОБЩ	Object[]	арх_М	месячный архив	100
SPT944	ParamsLog	ОБЩ	String	ИЗМ	Архив изменений БД пр...	2000

Рисунок 4.

Как уже было сказано ранее, логическая структура у всех каналов одного типа одинаковая. Т.е. если у канала ТВ1 есть часовой, суточный и месячные архивы, то точно такой же набор есть и у ТВ2.Поэтому архив указывается не для конкретного канала, а для целого типа.

Рассмотрим, как библиотека хранит интервальные и сервисные архивы.

Интервальный архив представлен классом IntervalArchive. Все его данные хранятся в поле DataTable table(тип DataTable является встроенным в c#, и вам придется прочитать по нему документацию, т.к. я не буду здесь писать то, что и так уже неплохо описано).Если коротко, то это таблица, каждый столбец которой, во первых, имеет имя, и во вторых задает тип данных для всех элементов, которые будут в этом столбце лежать.

Формат хранения архивных записей в таблице DataTable table почти такой же как и в таблице 1: первый столбец используется для времени записи, а другие используются для хранения значений одного тэга в разные моменты времени. Однако есть 2 небольших различия:

1) В библиотеке все архивы для каналов одного типа объединены в один архив. Например, два часовых архива для двух ТВ каналов объединятся в один часовой архив для каналов типа ТВ, два дневных архива, в один дневной и т.д. (Т.е. архив описывает уже не конкретный канал, а тип каналов в приборе).

2) Каждый столбец в таблице, хранящий значения одного тэга в разные моменты времени, дополнительно имеет ссылку на объект типа ArchiveField, который хранит подробную информацию об этом тэге.(ссылка реализуется через ExtendedProperties столбца).

t	ТВ1_V1	ТВ1_V2	ТВ1_V3	ТВ2_V1	ТВ2_V2	ТВ3_V3
02.04.2023. 13:00	23	221	235	90	1	6
02.04.2023. 14:00	67	356	67	15	2	8

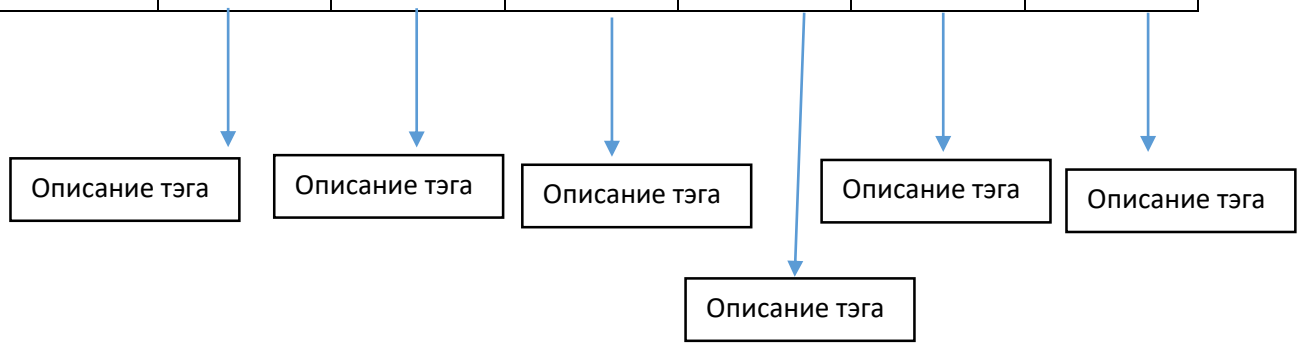


Рисунок 5 – часовой архив для каналов типа ТВ.

Строка этой таблицы называется записью архива.

Описание тэга содержит практически ту же информацию, что и TagDef4M/TagDef4L (в зависимости от семейства прибора, которому принадлежит архив): имя, тип данных, описание, физический тип данных и т.д.

Как можно заметить, имена столбцов поменялись: теперь они строятся из типа канала, номера канала и имени тэга в канале. Это нужно из-за объединения нескольких архивов. Однако если объединения не произошло (в приборе канал данного типа только один), имена столбцов будут состоять только из имени тэга, как в таблице 1.

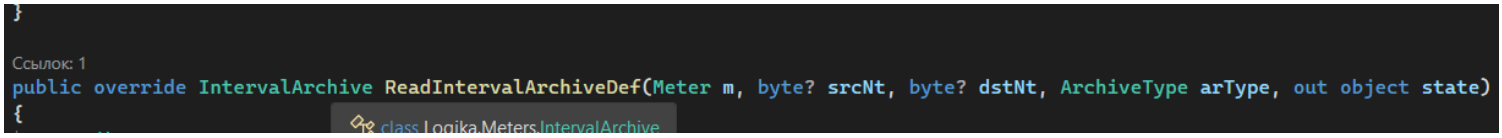
Сервисные архивы в библиотеке представляются классом ServiceArchive. Он хранит свои записи в списке List<ServiceRecord> Records. ServiceRecord определяет структуру каждой записи. Он содержит следующие поля: DateTime tm –время создания записи, string event- произошедшее событие, string description –описание события.

Рассмотрим методы для считывания архивов:

1) Интервальные архивы.

   Для того, чтобы считать интервальный архив, нужно сначала создать пустую таблицу, в которую будут записаны данные из прибора.

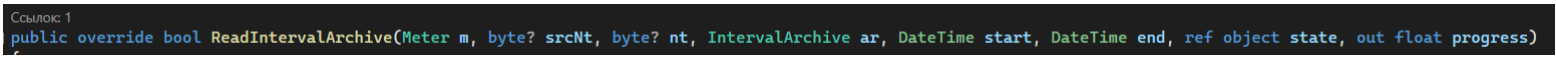
   Этим занимается метод ReadIntervalArchiveDef:



Здесь m – класс-модель прибора, из которого мы хотим считать архивы, srcNt ничего не значит (передайте null), dstNt –адрес прибора в протоколе 2 уровня, arType- тип интервального архива, который мы хотим считать (часовой, суточный и т.п), state – можно передать пустой object экземпляр(он понадобится на других этапах считывания). Данный метод возвращает объект IntervalArchive, поле table которого будет содержать построенную пустую таблицу.

Ранее было сказано, что чтение архивов производится для определенных типов каналов. Почему тут нет параметра, который этот тип бы задавал? Все дело в особенностях 4L и 4M приборов: у них только 2 возможных типа каналов (ОБЩ и ТВ) и архивы могут быть одновременно только у одного типа. Информация о том, у какого именно типа они есть содержится в классе-модели прибора.

После получения пустой таблицы мы можем запросить архивные данные у прибора. Для этого предназначен метод ReadIntervalArchive.



Здесь m- класс-модель прибора, srcNt=null , nt- адрес прибора, ar-объект интервального архива, полученный при вызове предыдущего метода, start и finsh – задают временной диапазон для считываемых записей(нам может понадобится не весь архив, а лишь его часть), state-объект модифицированный предыдущим методом ReadIntervalArchiveDef,progress-процент считанных данных(нужен если мы хотим отображать пользователю прогресс считывания).Вызывать данный метод необходимо пока он не вернет false(считывание архива производится в несколько итераций). В результате его работы таблица внутри IntervalArchive ar будет заполнена.

2) Сервисные архивы.

Считывание производится методом ReadServiceArchive.

Параметры этого метода идентичны таковым у ReadIntervalArchive, единственное что для сервисного архива таблица не требуется таблица и в качестве параметра ar можно передать ServiceArchive, созданный с помощью конструктора ServiceArchive(Meter m, ArchiveType art). Вызов этого метода, опять же, надо повторять, пока она не вернет false.

В результате список records для ServiceArchive ar будет обновлен.