Пример работы с библиотекой OwenLibFileAsync.lib и внешним USB-накопителем на ПЛК110 [М02].

1. Описание библиотеки OwenLibFileAsync.lib

Встроенное программное обеспечение контроллера в новой версии предусматривает работу с файловыми системами: поддерживаются функции сохранения в файлы произвольных данных, чтение, удаление, копирование, переименование файлов и др., данные функции реализованы в отдельном модуле – **OwenLibFileAsync.lib .**

Имеется возможность сохранять файлы на три устройства хранения:

- внешний накопитель, подключенный к порту USB-host (например, flash-память, или жесткий диск),
- внутренний flash-накопитель,
- внутренний виртуальный диск – RAM Диск – 64кБайт.

На носителях поддерживаются подкаталоги, а также доступны операции с файлами в директориях и просмотр содержимого директорий.

Устройство, на которое файл будет записываться, задается с помощью префикса к имени файла: “ffs:” для внутренней flash-памяти, “ram:” для виртуального диска и “usb:” для внешнего накопителя, подключенного к порту USB-host. Например, чтобы записать файл с именем “file1.txt” на разные носители, необходимо преобразовать его в “ffs:file1.txt” для записи во внутреннюю flash-память, “ram:file1.txt” – для записи на виртуальный диск и “usb:file.txt” – для записи на внешний накопитель.

На ПЛК поднят TFTP сервер для передачи данных по протоколу TFTP (RFC 1350 - THE TFTP PROTOCOL (REVISION 2) http://tools.ietf.org/html/rfc1350). Особенностью является: только RAM диск доступен для TFTP протокола, а также отсутствие функции разграничения доступа. Сервер доступен на всех интерфейсах ПЛК по соответствующим IP адресам на порту 69.

Внимание! Не рекомендуется использовать встроенную Flash-память для записи часто переписываемых файлов, так как ее ресурс ограничен (≈50 000 циклов записи). В создаваемых программах для контроллера рекомендуется программировать сохранение файла с предпочтением внешнего накопителя при его наличии.

Внимание! Виртуальный диск расположен в оперативном запоминающем устройстве контроллера, поэтому все файлы, записанные под именами с префиксом “ram:”, будут храниться в контроллере до первого его выключения. Чтобы сохранить эти файлы, необходимо их копировать на flash-носители – внутренний или внешний.

Внимание! Допускается работа программы пользователя одновременно не более чем с пятью файлами.

При работе с USB Host следует учесть следующие особенности:

- Поддержка USB Hub-ов. Однако суммарное потребление хабом и подключёнными устройствами не должно превышать 0,5А.

– USB HOST имеет функцию защиты от перегрузки и короткого замыкания. Срабатывание защиты приводит к выключению питания на USB HOST с последующими периодическими попытками восстановления питания.

– К ПЛК могут быть подключены USB MassStorage и USB HID устройства. Общее число MSD и HID устройств не должно превышать 1 для каждого типа. Остальные устройства игнорируются. Инициализация устройств в порядке подключения. Если, к примеру, 2 USB MSD уже подключены и подаётся питание на ПЛК — порядок их инициализации непредсказуем.

– Стек USB HOST поддерживает опрос не менее 1 устройства класса USB HID

– Используя библиотеку OwenLibHidEvent пользователь может получать сообщения от HID устройства, например мыши и/или клавиатуры.

– В ПЛК поддерживается класс USB MSD. Поддерживается только файловая система FAT (12, 16, 32). Ограничения на размер накопителя налагаются только ограничениями файловой системы FAT.

Рекомендуется использование файловой системы FAT 32!

Запись на USB MSD происходит без кэширования, т.о. для безопасного удаления накопителя необходимо:

- Завершить все процедуры записи (остановить запись из программы через библиотеку OwenLibFileAsync и закрыть открытые файлы, дождаться завершения всех системных загрузок файлов, остановить работу модуля(ей) архивации, остановить опрос файлов через 0x20 функцию ModBus slave)
- Дождаться прекращения активности на накопителе (если индикация активности присутствует) или выждать не менее 3 секунд.
- Удалить накопитель

Не рекомендуется подключать USB MSD на базе жёстких дисков и SDD без дополнительного внешнего питания, т.к. это может привести к перегрузке по питанию, и циклическому включению/выключению внешнего диска.

Внимание! Не гарантируется корректная работа ПЛК с USB устройствами, если последние обратно не совместимы с протоколом USB 1.1

В приборе реализован асинхронный механизм доступа к файлам из программы пользователя. Асинхронный доступ гарантирует отсутствие задержек в выполнении программы пользователя при доступе к файлам, в т.ч. расположенным на внешних носителях. Библиотека **OwenLibFileAsync.lib** предоставляет интерфейс для создания и обработки запросов к файловой системе на открытие, чтение, запись, модифицирование и т.п. файлов в асинхронном режиме. Основной особенностью использования данной библиотеки является – выполнение функций в **два этапа**. А именно:

- на первом этапе необходимо подать команду для работы с файлом;
- на втором этапе проверяется завершенность выполнения указанной команды и разрешается давать следующие команды для дальнейших операций с файлом.

Библиотека OwenLibFileAsync.lib рекомендуется для всех новых разработок.

В имеющейся версии библиотеки реализованы функции работы с файлами, перечисленные в таблице 4.2.

Таблица 4.2 – Функции работы с файлами в асинхронном режиме

Название функции	Краткое описание функции
OwenSysFileCloceAllOpenAsync	Данный функциональный блок закрывает все открытые файлы. Не нужно сообщать имена или дескрипторы файлов, поскольку они все уже известны системе.
OwenSysFileCloseAsync	Используется для закрытия файла. После закрытия файл освобождается для других процессов, дескриптор более не имеет значения.
OwenSysFileWriteAsync	Используется для записи данных в файл. Файл должен быть предварительно успешно открыт с помощью SysFileOpenAsync.
OwenSysFileReadAsync	Используется для чтения данных из файла. Файл должен быть предварительно успешно открыт с помощью SysFileOpenAsync
OwenSysFileDeleteAsync	Удаление файла с заданным именем.
OwenSysFileGetPosAsync	Возвращает позицию (смещение от начала файла в байтах) записи и чтения в файл.
OwenSysFileSetPosAsync	Задает позицию записи и чтения в файл.
OwenSysFileEOFAsync	Возвращает TRUE, если текущая позиция чтения-записи находится в конце файла, иначе возвращает FALSE.
OwenSysFileGetSizeAsync	Возвращает размер файла с заданным именем.
OwenSysFileGetTimeAsync	Возвращает время создания, последнего доступа и последнего изменения файла с заданным именем.
OwenSysFileCopyAsync	Копирование файла с заданным именем в файл с другим именем.
OwenSysFileRenameAsync	Переименование (перенос) файла с заданным именем.
OwenFileOpenAsync	Используется для открытия существующего или создания нового файла. Выход hFile (DWORD) сообщает дескриптор файла. Он используется другими функциональными блоками для работы с данным файлом.

Функция SysFileOpenAsync

Функция возвращает значение типа DWORD, используется для открытия существующего или создания нового файла. Возвращаемое значение – дескриптор файла, либо '0' в случае ошибки. Дескриптор файла используется для доступа к открытому файлу другими функциями библиотеки. В некоторых случаях сообщение об ошибке имеет вид 16#FFFFFFFF (в десятичной системе это 4 294 967 295 при интерпретации как беззнакового числа или -1 при интерпретации, как числа со знаком).

Входные переменные:

- **FileName** типаSTRING – имя файла;
- **Mode** типа STRING – режим работы с файлом, может имеет следующие значения:
  “w+”,если требуется открыть файл только для записи, при этом, если файл существовал до начала записи, то он будет стерт и создан пустой файл с заданным именем;
  “r”, если требуется открыть файл только для чтения;

“a” –аналогично “w+”, но если файл существовал до начала операции, данные будут дописываться в конец файла.

Пример открытия файла «a» и разрешение на переход к дальнейшим операциям с файлом на языке ST:

```
0:
res:=OwenFileOpenAsync(filename,'a',ADR(handle));
IF res=ASYNC_WORKING THEN
    state:=1;
END_IF

1:
res:=OwenFileOpenAsync(filename,'a',ADR(handle));
IF res=ASYNC_DONE THEN
    IF handle<>0 THEN
        state:=2;
    ELSE
        state:=0;
    END_IF
ELSIF res<0 THEN
    state:=0;
END_IF
```

Функция SysFileCloseAsync

Функция закрывает файл, открытый ранее функцией SysFileOpenAsync, возвращает значение типа BOOL, которое равно TRUE при успешном закрытии файла, иначе (например, если файл не был открыт) FALSE. Входным параметром является дескриптор закрываемого файла.

Входные переменные:

- **hFile** типа DWORD - дескриптор файла, число, которое возвратила функция SysFileOpenAsync

Пример закрытия файла «a» и разрешение на переход к дальнейшим операциям программы на языке ST:

```
res:=OwenFileCloseAsync(handle,ADR(result));
IF res=ASYNC_WORKING THEN
    state:=7;
ELSE
    state:=0;
END_IF

7:
res:=OwenFileCloseAsync(handle,ADR(result));
IF res=ASYNC_DONE THEN
    IF result=0 THEN
        state:=8;
    ELSE
        state:=8;
    END_IF
ELSIF res<0 THEN
```

```
            ▪   state:=8;
        –   END_IF
```

Функция SysFileWriteAsync

Функция записи данных в файл, открытый с помощью SysFileOpenAsync, возвращает значение типа DWORD – количество записанных байт данных.

Входные переменные:

- **File** – типа DWORD – дескриптор файла, число, которое возвратила функция SysFileOpenAsync;
- **Buffer** – адрес буфера, содержащего данные, которые необходимо записать в файл, число, которое возвратила функция ADR с аргументом – именем переменной-буфера; тип – массив, например, массив байт, или строка.
- **Size** – типа DWORD – размер буфера в байтах, можно использовать функцию SIZEOF с аргументом – именем переменной-буфера.

Пример на рисунке 4.9 языке ST, записывающая данные в файл «a» значение «buffout».

```
– 2:
  res:=OwenFileWriteAsync(handle,ADR(bufout),14,ADR(result));
–       IF res=ASYNC_WORKING THEN
–           state:=3;
–       ELSE
–           state:=6;
–       END_IF
–
–       3:
– res:=OwenFileWriteAsync(handle,ADR(bufout),14,ADR(result));
–       IF res=ASYNC_DONE THEN
–           IF result=14 THEN
–               state:=4;
–           ELSE
–               state:=6;
–           END_IF
–       ELSIF res<0 THEN
–           state:=6;
–       END_IF
```

Рисунок 4.9 – Использование функции записи в файл

Функция SysFileReadAsync

Функция чтения данных из файла, открытого с помощью SysFileOpenAsync, возвращает значение типа DWORD– количество считанных байт данных.

Входные переменные:

- **File** – типа DWORD – дескриптор файла, число, которое возвратила функция SysFileOpenAsync;

- **Buffer** – адрес буфера, содержащего данные, которые необходимо записать в файл, число, которое возвратила функция ADR с аргументом – именем переменной-буфера; тип – массив, например, массив байт или строка.
- **Size** – типа DWORD – размер буфера в байтах, можно использовать функцию SIZEOF с аргументом – именем переменной-буфера.

Использование функции – аналогично SysFileWriteAsync (см. рисунок 4.9).

Функция SysFileDeleteAsync

Функция удаления файла, возвращает значение типа BOOL: TRUE при успешном удалении, FALSE при ошибках.

Входная переменная – **FileName –** типа STRING – имя удаляемого файла.

Функция SysFileGetPosAsync

Функция возвращает число типа DWORD – фактическое смещение в байтах от начала до текущей позиции в открытом файле с заданным дескриптором; число определяет «место» в файле, откуда будет считана или куда будет записана информация если операция чтения или записи будет произведена без дополнительного предварительного смещения позиции.

Входная переменная – **File** – типа DWORD– дескриптор открытого файла.

Функция SysFileSetPosAsync

Функция устанавливает для открытого файла с заданным дескриптором позицию чтения (записи) с помощью заданного смещения; возвращает значение типа BOOL. Если позиция была установлена, то возвращается значение TRUE, иначе возвращается FALSE.

Входные переменные:

- **File** – типа DWORD – дескриптор открытого файла, в котором необходимо задать текущую позицию чтения (записи);
- **Pos** – типа DWORD – позиция чтения (записи), заданная смещением в байтах от начала файла.

Функция SysFileEOFAsync

Функция, определяющая достигнут ли коней файла. Возвращает значение типа BOOL, равное TRUE, если текущим положением позиции чтения (записи) является конец файла, иначе FALSE.

Входной переменной функции является **File** – типа DWORD – дескриптор открытого файла, в котором и проверяется достижение текущей позицией конца.

Функция SysFileGetSizeAsync

Функция возвращает значение типа DWORD – размер файла в байтах; входной переменной функции является **FileName** типа STRING –имя файла.

Функция SysFileGetTimeAsync

Функция определяет значения даты и времени создания, последнего изменения и последнего доступа к файлу. Возвращает значение типа BOOL, которое равно TRUE при успешном завершении выполнения функции и FALSE в случае любой ошибки (например, доступа к файлу). Для хранения трех значений времени в одной переменной используется структура FileTime, которую можно описать так:

```
TYPE FILETIME
    STRUCT
        dtCreation:DT; (* Дата и время создания файла *)
        dtLastAccess:DT;(* Дата и время последнего доступа *)
        dtLastModification:DT; (* Дата и время последнего изменения файла *)
    END_STRUCT
END_TYPE
```

Входные переменные:

- **FileName** – типа STRING – имя файла;
- **ftFileTime** – типа POINTER TO FILE TIME – адрес структуры, в которую будут сохраняться считанные данные о времени создания файла, последнего доступа к нему и его последней модификации. Извлекается с помощью функции ADR.

Пример на рисунке 4.10 – программа на языке ST, считывающая структуру со значениями времени создания файла.

```
VAR
        file_time:POINTER TO FILETIME;
        returnvalue:POINTER TO DWORD;
        w: FILETIME;
END_VAR
```

```
    8:
        OwenFileGetTimeAsync(filename, file_time, returnvalue);
        w:=file_time;

    ELSE
        state:=0;
    END_CASE
```

Рисунок 4.10 – Использование функции возврата значений времени

Функция SysFileCopyAsync

Функция копирования одного файла в другой, файлы задаются именами. Возвращает значение типа UDINT, в котором содержится количество действительно скопированных байт.

Входные переменные:

- **FileDest** – типа STRING – имя копии файла (файл-приемник);
- **FileSource** – типа STRING – имя копируемого файла (файл-источник).

Функция SysFileRenameAsync

Функция переименования файла. Возвращает значение типа BOOL, равное TRUE в случае успешного переименования, или FALSE в случае ошибки. Ошибка может возникать, например, если попытаться переименовать открытый файл.

Входные переменные:

- **FileOldName** – типа STRING – текущее имя файла;
- **FileNewName** – типа STRING – новое имя файла.

2. Описание примера работы с внешним USB-накопителем.

Постановка задач, решение которых приведено в примере:

- проверка состояния подключенного USB-накопителя к порту ПЛК110 M02;
- создание архива на USB с меткой времени по двум переменным вещественного типа по определенному периоду времени;
- создание нового архива с уникальным именем каждые сутки.

Для того, чтобы формировать архив с меткой времени, необходимо предварительно подключить библиотеку **SysLibTime**, идущую на диске ПЛК. В программе предварительно вызывается состояние часов реального времени, а также доступна возможность поменять время и дату из самой программы.

Внимание! Часы реального времени в ПЛК110 M02 зависят от заряда батарейки, встроенной в контроллер. Более подробно см. РЭ на ПЛК110 M02.

```
(*-------------Работа с временем и датой ПЛК------------*)
(* получаем текущее время*)
        TimeAndDate.Day :=0;
        TimeAndDate.DayOfWeek :=0;
        TimeAndDate.dwHighMsec :=0;
        TimeAndDate.dwLowMSecs :=0;
        TimeAndDate.Milliseconds :=0;
        TimeAndDate.Minute :=0;
        TimeAndDate.Second :=0;
        TimeAndDate.Hour :=0;
        TimeAndDate.Year :=0;
        TimeAndDate.Month :=0;
        Sys_time.ulHigh :=0;
        Sys_time.ulLow :=0;
        GetTime (SystemTime:=Sys_Time , TimeDate:= TimeAndDate);
(*Если set_time - истина, то можно менять время*)
IF set_time THEN
        set_time:=0;
        TimeAndDate.Minute:=MM;
        TimeAndDate.Second:=SS;
        TimeAndDate.Hour:=HH;
        GetTime (SystemTime:=Sys_Time , TimeDate:= TimeAndDate);

END_IF
(*Если set_date - истина, то можно менять дату*)
IF set_date THEN
        set_date:=0;
        TimeAndDate.Day:=D;
        TimeAndDate.Month:=M;
        TimeAndDate.Year:=Y;
        GetTime (SystemTime:=Sys_Time , TimeDate:= TimeAndDate);

END_IF
```

Далее идёт проверка подключен или нет USB-накопитель к порту ПЛК. Это происходит при помощи функции **GetUsbSerial** (библиотека **OwenLibUsbSerial**), которая используется для получения серийного номера устройства USB, а также позволяет определить подключено ли устройство. Описание функции **GetUsbSerial** можно посмотреть в проекте Codesys, описание которого Вы читаете.

```
(*проверка готовности USB-устройства. В примере используется только для ознакомления, пользователь может указать необходимые условия обработки событий самостоятельно*)
IF get_usb()=-261  THEN
str1:='устройство USB не подключено!';
ELSIF get_usb()=0  THEN
str1:='USB-устройство готово к использованию';
END_IF
```

Как было сказано, архивирование будет происходить по выбранному промежутку времени. В примере данный период времени равен 3 секундам:

```
(*Условие для выполнения записи файла с периодом 3 секунды*)
timer_archiv(in:=1, pt:=T#3s) ;
IF timer_archiv.Q THEN start_archiv:=1; timer_archiv(in:=0);
ELSIF timer_archiv.Q=0 AND state=0 THEN start_archiv:=0;
END_IF
```

Если таймер сработал, то выставляется команда на начало архивирования. После чего идёт проверка на текущую дату, и определяется необходимость создания нового файла с новым именем:

```
IF start_archiv THEN (*если необходимо архивировать, то*)


IF today<>TimeAndDate.Day THEN (*если наступил новый день, то формируем новый файл с новым именем даты*)
name_of_file:=CONCAT(pref,UINT_TO_STRING(TimeAndDate.Day));
name_of_file:=CONCAT(name_of_file,'.');
name_of_file:=CONCAT(name_of_file,UINT_TO_STRING(TimeAndDate.Month))
name_of_file:=CONCAT(name_of_file,'.');
name_of_file:=CONCAT(name_of_file,UINT_TO_STRING(TimeAndDate.Year));
name_of_file:=CONCAT(name_of_file,'.csv');
today:=TimeAndDate.Day; (*запомнить дату *)
need_new_header:=1; (*далее в новом файле необходимо будет создать "шапку" архива*)
END_IF
```

Далее идет работа с библиотекой **SysLibFileAsync**, которая, как уже было сказано ранее, происходит в два этапа для каждой функции (более подробно представлено в проекте примера).

Для понимания работы примера, стоит отметить, что при необходимости создания нового файла с новым именем выставляется флаг «need_new_header», что говорит о создании нового заголовка в файле формата .csv:

```
(*запись в файл – этап 1*)
IF need_new_header THEN (*если новый день суток, формируем новую «шапку»*)
(*формируем колонку Дата*)
header:=CONCAT('число','.месяц.год'); (*|число.месяц.год*)
header:=CONCAT(header,';'); (*|число.месяц.год|*)
(*формируем колонку Время*)
header:=CONCAT(header,'часы.минуты.секунды');(*|часы.минуты.секунды*)
                ....
```

Здесь и далее символ «|» - используется для того, чтобы показать, где начинается или заканчивается ячейка в файле Excel.

После формирования заголовка таблицы происходит запись текущих показаний времени и переменных. Если же это не первая запись за текущие сутки, то в данном случае действие происходит по оператору ELSE и файл откроется на «дозапись» текущих показаний, определенных для архивации:

```
ELSE (*иначе, если файл открывается на дозапись, то формируем только строку архива в формате |число.месяц.год|часы.минуты.секунды|значение var1|значение var2|*)
header:=CONCAT(header,UINT_TO_STRING(TimeAndDate.Day));(*|число*)
header:=CONCAT(header,'.'); (*|число.*)

                ....

(*формируем значение первой переменной var1*)
header:=CONCAT(header,REAL_TO_STRING(var1));(*|число.месяц.год.|часы.минуты.секунды.|var1*)
header:=CONCAT(header,';');(*|число.месяц.год.|часы.минуты.секунды.|var1|*)
(*формируем значение второй переменной var2*)
header:=CONCAT(header,REAL_TO_STRING(var2));(*|число.месяц.год.|часы.минуты.секунды.|var1|var2*)
header:=CONCAT(header,';');(*|число.месяц.год.|часы.минуты.секунды.|var1|var2|*)
header:=CONCAT(header, '$0A');(*0A =Line Feed - перевод строки, см. ASCII таблицу*)
END_IF
```

После чего можно переходить к записи файла на USB-накопитель:

```
(*Пишем строку в файл*)
res:=OwenFileWriteAsync(handle,ADR(header),LEN(header),ADR(result));(*пишем данные *)
        IF res=ASYNC_WORKING THEN
                state:=3;(*нет ошибок, переход на запись файла этап 2, либо по усмотрению пользователя*)
        ELSE
                state:=4;(*иначе можно перейти на  этап закрытия файла, либо предусмотреть собственное действие самостоятельно *)
        END_IF
```

Стоит отметить, что в случае ошибки на том или ином этапе выполнения функции (например, **OwenFileWriteAsync**) пользователь может самостоятельно инициировать дальнейшие действия.

Более подробное описание программы Вы можете посмотреть в проекте Codesys.