TREI GMBH

Unimod Pro

ПУЛЬТ ОПЕРАТОРА M920L

Фирма «ТРЭИ ГМБХ» постоянно совершенствует и развивает свою продукцию. В связи с этим информация, содержащаяся в данном документе, может изменяться без дополнительного предупреждения пользователей.

ООО «ТРЭИ ГМБХ»
Россия,
440028, Пенза, ул. Титова, 1Г
Телефон (fax): +7 (8412) 55-58-90, 49-95-39
fax: +7 (8412) 49-85-13
e-mail: trei@trei-gmbh.ru

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Терминал M920L предназначен для организации интерфейса между оператором и интеллектуальным модулем M900. Интерфейс программируется с помощью технологического приложения в среде **Unimod Pro.** Основным языком написания приложений для терминала M920L в системе Unimod Pro является язык ST (структурированный текст).

Документ предназначен для разработчиков программного обеспечения, а также для проектировщиков систем контроля и управления.

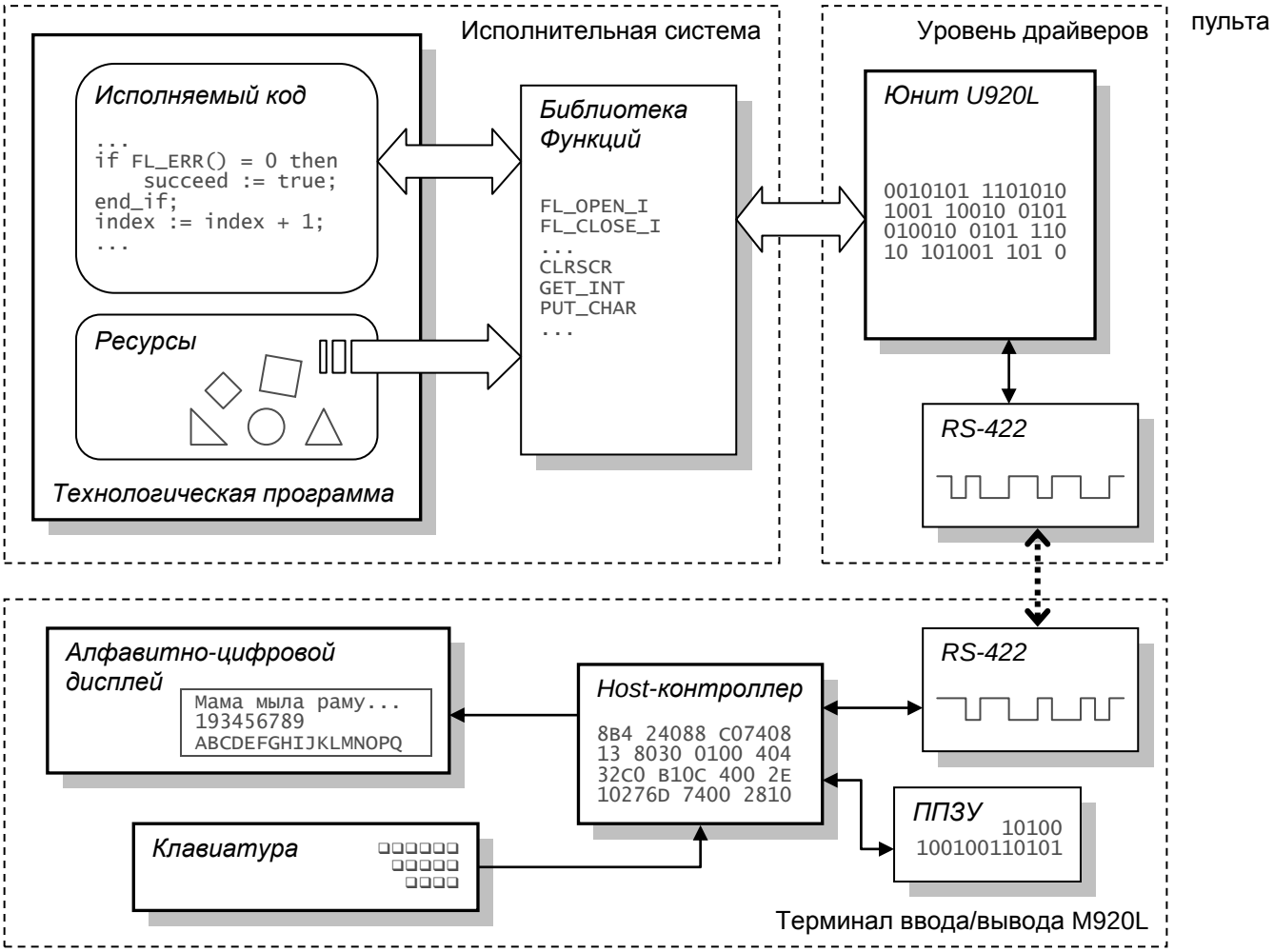
При работе используйте документацию следующих фирм:

TREI GmbH

- «Unimod Pro. Руководство пользователя»
- «Unimod Pro. Руководство по программированию»

1. АРХИТЕКТУРА

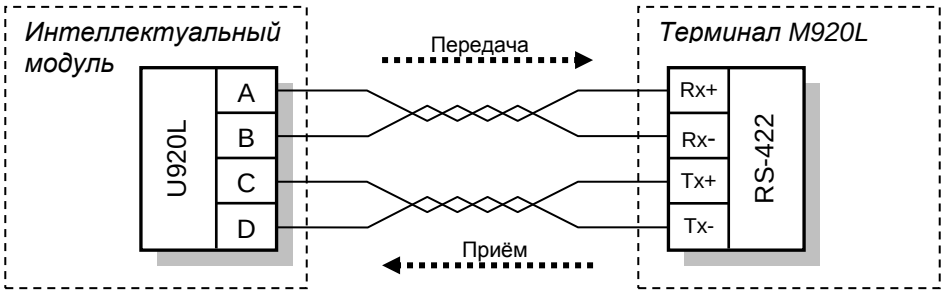
Технологическая программа для модуля M900 разрабатывается в среде Unimod Pro. Обмен данными между технологической программой и каналом юнита U920L происходит через файловые операции из библиотеки функций исполнительной системы модуля M900.



Пульт оператора M920L принимает данные из последовательного канала и производит вывод на экран, а также анализирует состояние клавиатуры и передаёт интеллектуальному модулю M900 информацию о нажатых клавишах.

2. ПОРЯДОК РАБОТЫ КАНАЛОВ ЮНИТА U920L

Пульт оператора M920L подключается к интеллектуальному модулю M900 через последовательный канал RS-422, реализуемый юнитом U920L.



Предупреждение. На интеллектуальном модуле M900 может быть установлен только один юнит U920L. Пульт оператора M920L **нельзя** подключать через юнит U920L к модулям расширения серии W900.

Обмен данными между интеллектуальным модулем и пультом оператора происходит в полнодуплексном режиме (Full Duplex) по отдельным дифференциальным линиям для приёма и передачи. По умолчанию интеллектуальный модуль M900 устанавливает скорость обмена с пультом оператора 115200 бит/с. Изменение скорости обмена возможно через вызов функции OPERATE (код команды 011h).

Юнит U920L реализует четыре дискретных канала ввода:

KBREADY	Буфер клавиатуры содержит код нажатой клавиши.
TXBUSY	Происходит асинхронная передача данных пульту оператора.
RCBUSY	Пульт оператора занят вводом с клавиатуры.
RCREADY	Ввод с клавиатуры завершен.

Примечание. Для корректной работы пульта оператора M920L ко всем четырем каналам в Unimod Pro должны быть привязаны переменные ввода/вывода.

Как передача данных пульта оператора, так и приём от него происходит асинхронно. Принятый код нажатой клавиши попадает в четырехуровневый буфер FIFO и устанавливает KBREADY. Переполнение буфера клавиатуры расценивается как ошибка, и если установлен флаг конфигурации ввода SOUND, ошибка подтверждается звуковым сигналом.

Так как переменные, привязанные к каналам ввода/вывода, обновляются один раз за цикл приложения, а приём кодов нажатых клавиш происходит асинхронно, возможна ситуация, когда при сброшенном состоянии KBREADY буфер клавиатуры будет содержать код нажатой клавиши.

Код нажатой клавиши читается из буфера FIFO вызовом одной из файловых операций или специализированной функцией GET_CHAR. Прочитанный код удаляется из буфера FIFO. Чтение последнего кода сбрасывает состояние канала KBREADY.

Первая запись в поток ввода/вывода CONIO устанавливает TXBUSY и начинает асинхронную передачу данных пульта оператора. Запись может быть выполнена как через низкоуровневые функции работы с файлами FL_WR, так и через высокоуровневые PUT_INT или PUT_REAL.

Время, необходимое для завершения асинхронной операции вывода зависит от скорости передачи и объёма передаваемых данных. Поскольку асинхронная передача данных пульту оператора производится в фоновом режиме, она не увеличивает длительность цикла технологической программы. По окончании передачи переменная TXBUSY принимает значение FALSE.

Следующая запись в поток ввода/вывода, произошедшая в то время, когда в фоновом режиме происходит вывод, приведет к приостановке выполнения технологической программы. После окончания текущего вывода инициируется следующая асинхронная передача данных пульту оператора, а технологическая программа возобновляет свою работу. В этом случае цикл технологической программы увеличится на время, необходимое для завершения предыдущей операции вывода.

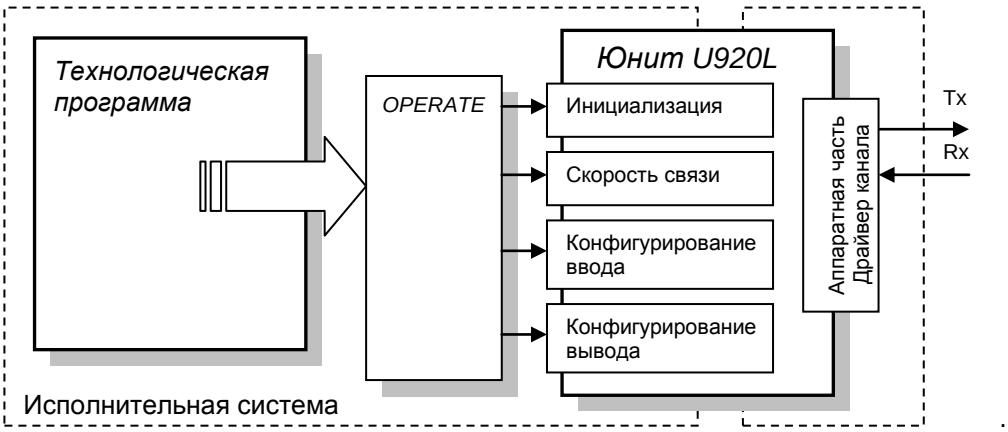
В стандартную библиотеку исполнительной системы включены высокоуровневые функции GET_INT и GET_REAL, реализующие асинхронный ввод целых или вещественных чисел с клавиатуры пульта оператора (см. описание соответствующих функций).

Каналы RCBUSY и RCREADY доставляют дополнительную информацию о текущей операции ввода. Так значение TRUE канала RCBUSY сигнализирует о том, что в данный момент происходит ввод с клавиатуры. Установка RCREADY происходит в том случае, если введенное оператором значение доступно для чтения технологической программе.

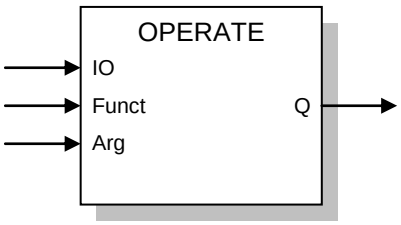
RCBUSY	*RCREADY*	*Состояние*
FALSE	FALSE	Неактивное состояние. Ввод не был начат или уже завершен.
TRUE	FALSE	Пульт оператора занят вводом с клавиатуры.
FALSE	TRUE	Такого состояния в принципе не может быть.
TRUE	TRUE	Оператор завершил ввод нажатие ‘Esc’ или ‘Enter’. Результат преобразования доступен для чтения.

3. КОНФИГУРИРОВАНИЕ ЮНИТА U920L

Для конфигурирования интерфейса связи, установки параметров форматирования вывода и режимов ввода служит функция или функциональный блок OPERATE. Упрощенная блок-схема конфигурации юнита U920L представлена ниже:



3.1 OPERATE, код команды 01h



Параметры функции:

IO:	PTR	Переменная, привязанная к каналу ввода/вывода
Funct:	INT	Код команды
Arg:	INT	Дополнительный параметр функции
Q:	INT	Результат вызова

Описание функции:

Функция OPERATE, вызванная кодом команды 01h, служит для переинициализации интерфейса связи с пультом оператора. Все текущие асинхронные операции ввода и вывода прерываются, буфер клавиатуры очищается.

Ни содержимое экрана пульта оператора, ни его конфигурация не меняются. Следует иметь в виду, что обрыв асинхронной передачи может привести к потере части передаваемого пакета и как следствие – к потере синхронизации обмена между модулем и пультом оператора.

Вызов этой функции не влияет на код последней ошибки. Функция OPERATE возвращает ненулевое значение, если операция завершена успешно, или ноль в случае возникновения ошибки.

Примеры использования:

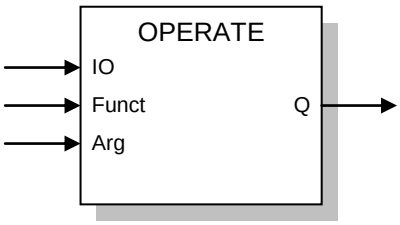
Для вызова функции или функционального блока OPERATE необходима переменная, привязанная к одному из дискретных каналов юнита U920L.

```
Result := OPERATE (U920LInput, 1, 0); (* переинициализация *)
if Result <> 0 then
      (* успешное завершение операции *)
      ...
end_if;
```

Неиспользованный аргумент OPERATE должен быть равен нулю.

Команда прекращает любой активный вывод. Если во время вызова функции происходила передача байта от модуля к пульту оператора, передаваемый байт будет оборван, а приёмное устройство примет ошибочные данные. Чтобы этого избежать, перед вызовом OPERATE следует анализировать состояние канала TXBUSY.

3.2 OPERATE, код команды 11h



Параметры функции:

IO:	PTR	Переменная, привязанная к каналу ввода/вывода
Funct:	INT	Код команды
Arg:	INT	Дополнительный параметр функции
Q:	INT	Результат вызова

Описание функции:

Функция или функциональный блок OPERATE вызванный с кодом команды 11h служит для конфигурирования скорости связи с пультом оператора M920L из технологической программы модуля. Устанавливаемый индекс скорости передаётся как аргумент OPERATE.

Перед изменением скорости последовательного интерфейса функция производит инициализацию пульта оператора так же, как функция OPERATE с кодом команды 01h: очищает буфер клавиатуры и прерывает все текущие асинхронные операции ввода и вывода.

Вступительная инициализация последовательного интерфейса происходит во время запуска технологической программы модуля и устанавливает скорость обмена 115200 бод.

Возможные скорости обмена для пульта оператора M920L:

0	Задает скорость обмена 1200bps.
1	Задает скорость обмена 2400bps.
2	Задает скорость обмена 4800bps.
3	Задает скорость обмена 7200bps.
4	Задает скорость обмена 9600bps.
5	Задает скорость обмена 1200bps.
6	Задает скорость обмена 14400bps.
7	Задает скорость обмена 16800bps.
8	Задает скорость обмена 19200bps.
9	Задает скорость обмена 21600bps.
10	Задает скорость обмена 24000bps.
11	Задает скорость обмена 26400bps.
12	Задает скорость обмена 28800bps.
13	Задает скорость обмена 38400bps.
14	Задает скорость обмена 57600bps.
15	Задает скорость обмена 115200bps.
16	Задает скорость обмена 230400bps.

Вызов этой функции не влияет на код последней ошибки. При успешном завершении функция OPERATE возвращает ненулевое значение. В случае возникновения ошибки функция OPERATE возвращает ноль.

Примечание. Данная функция предназначена для установки скорости обмена интеллектуального модуля, и не влияет на скорость обмена самого пульта оператора. Изменение скорости обмена, как и других конфигурационных параметров, следует производить через встроенное сервисное меню пульта оператора.

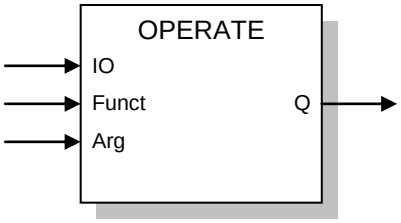
Примеры использования:

Основная область применения данной функции – изменение скорости обмена. Для вызова функции или функционального блока OPERATE необходима переменная, привязанная к одному из дискретных каналов юнита U920L. Установленная скорость сразу вступает в силу.

```
(* инициализация интерфейса для работы на скорости 9600 bps *)
Result := OPERATE (U920LInput, 16#11, 4);
if Result <> 0 then
      (* успешное завершение операции *)
```

```
        ...
    end_if;
```

3.3 OPERATE, код команды 64h



Параметры функции:

IO:	PTR	Переменная, привязанная к каналу ввода/вывода.
Funct:	INT	Код команды.
Arg:	INT	Дополнительный параметр функции.
Q:	INT	Результат вызова

Описание функции:

Конфигурация интерфейса ввода задаётся вызовом функции или функционального блока OPERATE с кодом команды 100 (64h). Слово конфигурации передаётся как параметр OPERATE и имеет следующий вид:

MSB SND SIG X X X X X X X X X X X X X X X L7 L6 L5 L4 L3 L2 L1 L0 **LSB**

L7…L0 LIMIT. Ограничение длины ввода. Если количество вводимых символов превышает значение параметра LIMIT, последующие нажатия клавиатуры не выводятся на индикатор и обрабатываются как ошибка ввода. Для отключения ограничения необходимо установить значение LIMIT равное нулю.

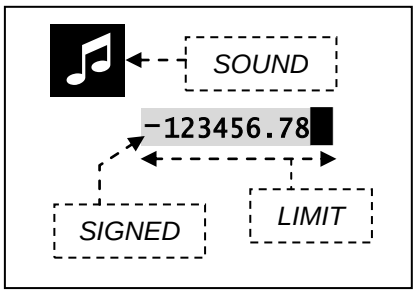
X…X Не используются. Зарезервированы на будущее.

SIG SIGNED. Режим ввода целых и вещественных чисел со знаком/без знака. При установленном флаге SIGNED разрешается ввод отрицательных значений. В противном случае нажатие клавиши ‘*’ (по-нашему – знак минус) обрабатывается как ошибка ввода.

SND SOUND. Включение звукового сигнала подтверждения ошибки ввода.

Вступительная инициализация интерфейса ввода происходит время запуска приложения и устанавливает следующие конфигурационные параметры: LIMIT=0, SIGNED=1, SOUND=1.

Функция OPERATE возвращает ненулевое значение, если операция завершена успешно, или ноль в случае возникновения ошибки



Конфигурация интерфейса ввода

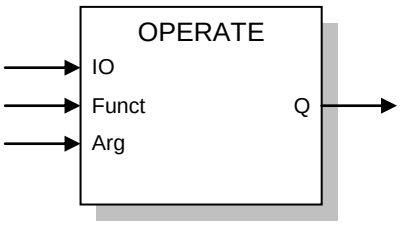
Примеры использования:

Конфигурирование ввода следует производить непосредственно перед вызовом функций GET_INT или GET_REAL.

```
    (* ограничение до 8 знаков, звук, ввод отрицательных чисел *)
    Result := OPERATE (U920LInput, 100, 16#3000008);
    Value := GET_REAL (1, -1); (* начать ввод вещественного числа *)
    ...
    Value := GET REAL (1, -1); (* повторный вызов *)
    case FL_ERR () of
        (* произвести обработку ошибок... *)
        ...
```

```
    end_case;
```

3.4 OPERATE, код команды 65h



Параметры функции:

IO:	PTR	Переменная, привязанная к каналу ввода/вывода.
Funct:	INT	Код команды.
Arg:	INT	Дополнительный параметр функции.
Q:	INT	Результат вызова

Описание функции:

Конфигурация стандартного вывода задаётся вызовом функции или функционального блока OPERATE с кодом команды 101 (65h). Слово конфигурации передаётся как параметр OPERATE и имеет следующий вид:

MSB TRI PLS SIG X X X X X X X P7 P6 P5 P4 P3 P2 P1 P0 L7 L6 L5 L4 L3 L2 L1 L0 **LSB**

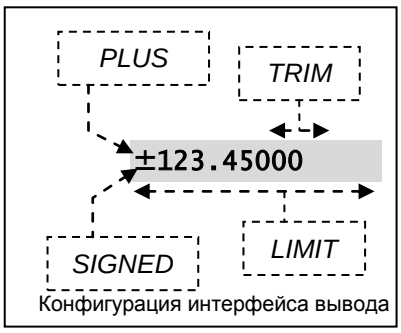
L7…L0 LIMIT. Ограничение длины вывода. Устанавливает номинальную ширину выводимой строки. Для вещественных чисел устанавливает общее количество знаков для целой и вещественной частей строки, включая плюс, минус и точку. Все знаки, выходящие за максимальную длину строки, не выводятся на индикатор. Если количество знаков в отформатированной строке меньше указанного значения, строка дополняется необходимым количеством пробелов. Ограничение длины вывода отключается записью нуля в LIMIT.

P7…P0 PRECISION. Количество знаков после запятой. Устанавливает точность преобразования вещественного числа в строку. При параметре PRECISION равном нулю преобразование происходит с максимальной точностью. Иначе после запятой будет выведено необходимое количество знаков.

X…X Не используются (Зарезервировано).

SIG SIGNED. Режим вывода целых чисел со знаком/без знака. При установленном флаге целые рассматриваются как числа со знаком, иначе – как числа без знака. Флаг SIGNED не оказывает влияния на форматирование вещественных чисел, которые всегда обрабатываются как числа со знаком.

PLS PLUS. Разрешает вставку ‘+’ для положительных чисел. Для целых чисел имеет значение только в случае, если установлен флаг SIGNED.



TRI TRIM. Влияет на форматирование вещественных чисел. Установленный флаг TRIM разрешает удаление завершающих нулей в вещественной части выводимого числа.

Вступительная инициализация интерфейса вывода происходит во время запуска приложения и устанавливает следующие конфигурационные параметры: LIMIT=0, PRECISION=6, SIGNED=1, PLUS=0, TRIM=1.

Изменение режима вывода не влияет на текущую асинхронную операцию.

Примечание. Параметры PRECISION, SIGNED, TRIM и PLUS используются только для форматирования целых и вещественных чисел и не влияют на вывод строк и данных из ресурсов технологической программы. Параметр LIMIT устанавливает фиксированную длину для вывода строковых переменных.

При успешном завершении функция OPERATE возвращает ненулевое значение. В случае возникновения ошибки функция OPERATE возвращает ноль.

Примеры использования:

Конфигурирование вывода следует производить непосредственно перед вызовом функции PUT_INT или PUT_REAL.

```
(* фиксированная длина 16 знаков, вывод чисел со знаком *)
Status := OPERATE (U920LInput, 100, 16#1000010);
if Status <> 0 then (* режим вывода установлен? *)
      Status := PUT_INT (1, Value); (* начать ввод целого числа *)
      if Status <> 0 then
            (* успешное завершение операции *)
            ...
      end_if;
end_if;
```

4. ИСПОЛЬЗОВАНИЕ РЕСУРСОВ

При программировании пульта оператора M920L, в технологической программе Unimod Pro могут быть подключены ресурсы – статические данные, которые не содержат исполняемый код, но могут косвенно использоваться алгоритмом программы как константы для произведения сложных вычислений, для задачи параметров работы модуля и т.п.

Ресурсы делятся на 6 базовых типов: строка, массив байт, массив целых значений, массив вещественных значений, двоичный файл и текстовый файл. Каждому из этих типов присущи собственные свойства и ограничения.

Строки могут содержать как символы ASCII, так и непечатные знаки, для представления которых используется знак ‘$’:

$$ – знак доллара $ **$t** или **$T** – табуляция
$l или **$L** – строка вверх **$n** или **$N** – новая строка
$p или **$P** – пропуск страницы **$r** или **$R** – возврат каретки
$'** – одиночная кавычка **$" – двойные кавычки
$XX – где XX шестнадцатеричные цифры, код ASCII символа

Максимальная длина строки – 128 знаков. Массивы содержат набор однотипных данных. Двоичные и текстовые файлы – это внешние файлы, подключаемые к технологической программе. Размер массивов и файлов ограничен только доступной памятью.

Чтение ресурсов происходит через файловые операции.

5. БИБЛИОТЕКА ФУНКЦИЙ

Программный интерфейс пульта оператора M920L реализован в виде стандартного потока ввода/вывода. Для этого потока зарезервирован идентификатор файла – константа со значением 01h, названная условно CONIO: консоль ввода/вывода.

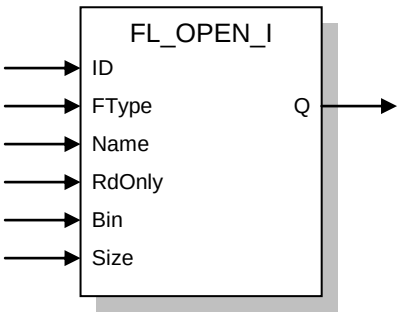
Все операции ввода с клавиатуры и вывода на экран пульта оператора реализованы через низкоуровневые и высокоуровневые файловые функции. Ниже приведено назначение функций библиотеки Unimod Pro Solution, используемых в работе с пультом оператора M920L:

FL_OPEN_I,	
FL_CLOSE, FL_GPOS	В работе с пультом оператора не используется.
FL_SPOS	Устанавливает позицию курсора.
FL_ERR	Возвращает код последней ошибки работы с пультом оператора.
FL_RD_B,	

FL_RD_I, FL_RD_R	Чтение кода нажатой клавиши из буфера клавиатуры.
FL_WR_B, FL_WR_I, FL_WR_R	Передача неформатированного байта модулю индикатора.
CLRSCR	Очистка экрана пульта оператора.
GOTOXY	Устанавливает позицию курсора.
PUT_CHAR	Запись байта в пульт оператора.
PUT_INT	Вывод целого значения с текущей позиции экрана.
PUT_MSG	Вывод строки с текущей позиции экрана
PUT_REAL	Вывод вещественного значения с текущей позиции экрана.
PUT_RSRC	Вывод содержимого ресурса/строки на пульт оператора.
GET_CHAR	Чтение кода нажатой клавиши из буфера клавиатуры.
GET_INT	Вывод целого значения с клавиатуры.
GET_MSG	Вывод строки с клавиатуры
GET_REAL	Вывод вещественного значения с клавиатуры.
OPERATE	Конфигурирование пульта оператора, установка параметров ввода/вывода.

Подробное описание каждой функций представлено ниже.

5.1 FL_OPEN_I



Параметры функции:

ID:	INT	Идентификатор файла.
FType:	INT	Тип используемой памяти (FRAM/SRAM/FLASH и т.п.)
Name:	INT	Имя файла в системе.
RdOnly:	BOOL	Открыть только для чтения.
Bin:	BOOL	Открыть в двоичном режиме.
Size:	INT	Установить максимальный размер файла, байт.
Q:	INT	Идентификатор файла.

Описание функции:
Функция FL_OPEN_I используется для открытия файла или ресурса.

При значении входа ID равном нулю – открывается файл с уникальным идентификатором. На интеллектуальных модулях M953C и M832C уникальные идентификаторы файлов выбираются псевдослучайно в диапазоне от 128 до 255.

При ненулевом входе ID проверяется наличие открытого файла с таким идентификатором. Если такой файл уже открыт, функция FL_OPEN_I не производит проверку параметров и независимо от того, соответствуют ли имя файла, тип памяти и т.п., возвращает идентификатор ID и устанавливает код ошибки “4 – объект уже существует”.

На интеллектуальных модулях M953C и M832C максимальное количество одновременно открытых файлов ограничено размером доступной памяти и равно 5-ти:

_BUFFERS	0x05	Максимальное количество открытых файлов

При превышении этого числа функция FL_OPEN_I возвращает ноль и устанавливает код ошибки “1 – ошибка при работе с памятью”.

В системах с установленным пультом оператора M920L интеллектуальный модуль M953C резервирует идентификатор стандартного потока ввода/вывода, условно называемый CONIO:

CONIO	0x01	Поток ввода/вывода пульта оператора M920L

Идентификатор файла CONIO становится доступным автоматически, если на интеллектуальном модуле установлен юнит U920L, т.е. открывать CONIO функцией FL_OPEN_I ненужно. Использование пульта оператора ввода/вывода не влияет на максимальное количество одновременно открытых файлов, определённое константой _BUFFERS.

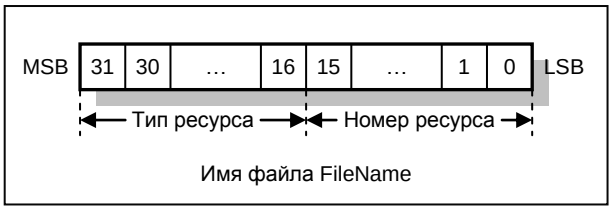
Примечание: CONIO – это **условное** название потока ввода/вывода. Исполнительная система Unimod Pro Solution не объявляет переменную с именем CONIO.

Следует отметить, что файл потока ввода/вывода не может быть закрыт, а идентификатор CONIO имеет фиксированное значение, поэтому задаваемые технологической программой идентификаторы файлов не должны пересекаться со значением CONIO. Основная работа с пультом оператора происходит через файловые функции.

Для того чтобы открыть ресурс технологической программы, функции FL_OPEN_I необходимо задать соответствующий тип памяти. На сегодняшний момент определены следующие типы памяти:

_SYSTEMFAT	0x0000	Стандартная файловая система (только мастер модуль)
_RESOURCE	0x0001	Функция FL_OPEN_I открывает ресурс
_FRAMFAT	0x0002	Резерв, файловая система в памяти FRAM
_FLASHFAT	0x0003	Резерв, файловая система во внешней памяти FLASH

Каждый ресурс технологической программы имеет свой собственный уникальный номер, который передаётся функции FL_OPEN_I как имя файла Name. Опционально, в старших 16 разрядах имени файла может быть указан тип запрашиваемого ресурса (см. рисунок). Если тип ресурса с заданным именем не совпадёт с запрошенным значением, функция FL_OPEN_I возвращает ноль и устанавливает код ошибки “22 - объект с заданным именем не найден”.



Исполнительная система Unimod Pro Solution поддерживает следующие типы ресурсов:

RT_NONE	0x0000	Неизвестный или любой тип ресурса
RT_ASCIISTRING	0x0001	Строка, массив знаков ASCII
RT_BYTEARRAY	0x0002	Массив байт
RT_INTARRAY	0x0003	Массив целых констант
RT_FLOATARRAY	0x0004	Массив вещественных констант
RT_BINFILE	0x0005	Двоичный файл
RT_TEXTFILE	0x0006	Текстовый файл

Открытие ресурсов технологической программы возможно исключительно в режиме “только для чтения”. При попытке открытия ресурса в режиме для записи (т.е. RdOnly = false), функция FL_OPEN_I возвращает ошибку “2 - несоответствие атрибутов объекта”.

Файл ресурса открывается в двоичном режиме независимо от значения аргумента Bin. Аргумент Size не используется и во время работы с ресурсами должен быть равен нулю.

Функция FL_OPEN_I устанавливает следующие коды ошибок:

0	Операция выполнена успешно.
1	Нехватка памяти для открытия файла. Превышено количество открытых файлов.
2	Несоответствие атрибутов объекта. Попытка открытия ресурса для записи.
4	Объект уже существует. Файл с данным идентификатором уже открыт.
21	Указано неверное значение типа памяти.
22	Объект не найден. Не удалось найти файл или ресурс с заданным именем.

Примеры использования:

В приведенном примере технологическая программа открывает массив байт с номером 90 (5Ah), производит проверку на наличие ошибок и читает первые четыре байта ресурса как целое значение.

```
Handle := FL_OPEN_I (0, 1, 16#2005A, true, false, 0);
if Handle <> 0 then
        Value := FL_RD_I (Handle, -1);
        if (Value <> -1) or (FL_ERR () = 0) then
                (* анализ прочитанного значения *)
        end_if;
        Error := FL_CLOSE (Handle);
end_if;
```

Имя ресурса 16#2005A можно условно разделить на две части. Младшие 16 разрядов задают номер ресурса. Старшие 16 разрядов содержат тип запрашиваемого ресурса: 2 для массива байт.

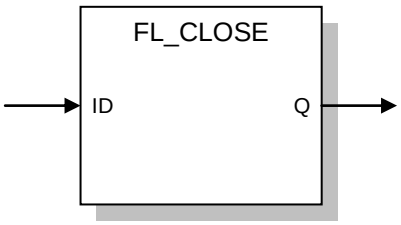
Для открытия ресурса не обязательно указывать его тип, достаточно оставить старшие 16 разрядов имени файла равными нулю, или иначе – задать тип ресурса RT_NONE. Например:

```
Handle := FL_OPEN_I (0, 1, 90, true, false, 0);
```

Программа открывает ресурс любого (или неопределенного) типа с номером 90.

В некоторых случаях бывает полезно задавать фиксированные идентификаторы файлов. Для этого служит аргумент ID. При нулевом значении ID идентификатор файла выделяется автоматически.

5.2 FL_CLOSE



Параметры функции:

ID:	INT	Идентификатор файла.
Q:	INT	Код ошибки.

Описание функции:

Функция FL_CLOSE используется для закрытия файла, открытого функцией FL_OPEN_I. Если идентификатор не найден в списке открытых файлов FL_CLOSE возвращает и устанавливает код ошибки “12 - файл не открыт”. В противном случае исполнительная система производит закрытие файла: освобождает выделенные ресурсы и т.п.

При попытке закрыть поток ввода/вывода (CONIO), функция FL_CLOSE возвращает код ошибки “2 - несоответствие атрибутов объекта”.

Функция FL_CLOSE возвращает и устанавливает следующие коды ошибок:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Доступ запрещен, файл не может быть закрыт.
12	Файл не открыт. Аргумент ID не является идентификатором файла.

Примеры использования:

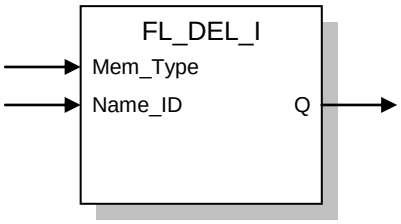
Пример открывает ресурс с номер 100, задавая фиксированный идентификатор файла 21. После проверки программа выполняет определенные действия и закрывает файл.

```
Handle := FL_OPEN_I (21, 1, 100, true, false, 0);
if Handle <> 0 then
        (* выполнить необходимые манипуляции с файлом *)
        ...

        (* в конечном итоге - закрыть файл *)
        Error := FL_CLOSE (Handle);
end_if;
```

Использование идентификатора после закрытия файла не допустимо. Особое внимание нужно уделять файлам, открытым с постоянным, заведомо известным идентификатором.

5.3 FL_DEL_I



Параметры функции:

`Mem_Type:`	`INT`	Тип используемой памяти (FRAM/SRAM/FLASH и т.п.)
`Name_ID:`	`INT`	Имя файла в системе.
`Q:`	`INT`	Код ошибки.

Описание функции:

Функция FL_DEL_I переназначена для удаления файла. Тип памяти и файловой системы, в которой расположен файл, задается параметром Mem_Type, а имя файла – параметром Name_ID.

Удаляемый файл не должен быть открыт для чтения или записи. Проба удаления файла, открытого для записи или чтения, возвращает код ошибки “2 - несоответствие атрибутов объекта”.

Файловая система находится в стадии разработки. Реализация этой функции на интеллектуальном модуле M900 временно отсутствует.

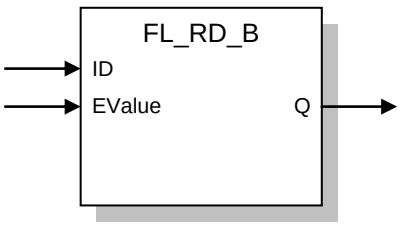
Функция FL_DEL_I возвращает и устанавливает следующие коды ошибок:

20 Вызываемая функция не поддерживается исполнительной системой.

Примеры использования:

Поддержка файловой системы на интеллектуальном модуле находится в стадии разработки. Реализация этой функции временно отсутствует.

5.4 FL_RD_B



Параметры функции:

`ID:`	`INT`	Идентификатор файла.
`EValue:`	`INT`	Результат вызова в случае ошибки.
`Q:`	`INT`	EValue или прочитанный байт.

Описание функции:

Функция FL_RD_B используется для чтения из файла одного байта. Файл ID должен быть потоком ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I. Прочитанный байт возвращается в младших 8-ми разрядах Result, образуя целое значение в диапазоне от нуля до 255.

На вход EValue подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

Чтение из файла производится с текущей позиции для чтения. По завершению операции текущая позиция для чтения увеличивается на размер прочитанного блока данных (1 байт). При обнаружении ошибки позиция указателя не меняется.

Функция FL_RD_B, вызванная с идентификатором файла CONIO, производит чтение кода нажатой клавиши из буфера клавиатуры. Если буфер клавиатуры пуст, FL_RD_B возвращает EValue и устанавливает код ошибки “16 - достигнут конец файла”.

Функция FL_RD_B устанавливает следующие коды ошибок:

0 Операция выполнена успешно.
2 Несоответствие атрибутов объекта. Файл открыт только для записи или нет доступа на побайтное чтение файла.
12 Файл не открыт. Аргумент ID не является идентификатором файла.
16 Достигнут конец файла. Недостаточно данных для прочтения байта.

Таблица соответствия клавиш и возвращаемых кодов для пульта оператора M920L:

Клавиша	Шестнадцатеричный	Десятичный	Знак ASCII
‘.’	02Eh	46	‘.’
‘*’	02Ah	42	‘*’
‘0’	030h	48	‘0’
‘1’	031h	49	‘1’
‘2’	032h	50	‘2’
‘3’	033h	51	‘3’
‘4’	034h	52	‘4’
‘5’	035h	53	‘5’
‘6’	036h	54	‘6’
‘7’	037h	55	‘7’
‘8’	038h	56	‘8’
‘9’	039h	57	‘9’
F1	041h	65	‘A’
F2	042h	66	‘B’
F3	043h	67	‘C’
F4	044h	68	‘D’
←	008h	8	-
↑	00Bh	11	-
→	009h	9	-
↓	00Ah	10	-
Esc	01Bh	27	-
Del	07Fh	127	-
Enter	00Dh	13	-

Примеры использования.

Программа производит чтение одного байта с текущей позиции файла Handle с последующим преобразованием байта в целое число со знаком:

```
Value := FL_RD_B (Handle, -1);
if Value >= 0 then
      (* преобразование в число со знаком *)
      if and_mask (Value, 16#80) <> 0 then
            Value := or_mask (Value, 16#FFFFFF00);
      end_if;

      (* произвести анализ значения Value *)
      ...
else
      (* ошибка. прочитать код ошибки *)
      Error := FL_ERR ();
end_if;
```

Прочитанный байт возвращается в младших 8-ми разрядах Value, образуя число в диапазоне от нуля до 255 включительно. Если код ошибки не столь важен, подавая на вход EValue значение, выходящее за предел допустимых значений байта, можно несколько упростить анализ ошибок.

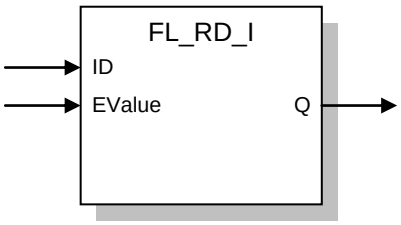
Аналогично происходит чтение кода нажатой клавиши из буфера клавиатуры. Если буфер клавиатуры пуст, функция FL_RD_B возвращает -1 и устанавливает соответствующий код ошибки.

```
KeyCode := FL_RD_B (1, -1);
if KeyCode >= 0 then
```

```
        (* анализ нажатия *)
        ...
end_if;
```

Примечание: Для работы с пультом оператора вместо FL_RD_B следует использовать специализированную функцию GET_CHAR.

5.5 FL_RD_I



Параметры функции:

ID:	INT	Идентификатор файла.
EValue:	INT	Результат вызова в случае ошибки.
Q:	INT	EValue или прочитанное значение.

Описание функции:

Функция FL_RD_I используется для чтения из файла переменной целого типа. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

На вход EValue подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

Чтение производится с текущей позиции указателя файла. Если разница между указателем и концом файла менее 4 байт, функция FL_RD_I возвращает EValue и устанавливает код ошибки “12 - достигнут конец файла”. После успешного завершения операции текущая позиция указателя увеличивается на размер прочитанного блока данных. При обнаружении ошибки позиция указателя не меняется.

Функция FL_RD_I, вызванная идентификатором файла CONIO, производит чтение кода нажатой клавиши из буфера клавиатуры. Если буфер клавиатуры пуст, FL_RD_I возвращает EValue и устанавливает код ошибки “16 - достигнут конец файла”.

Функция FL_RD_I устанавливает следующие коды ошибок:

0 Операция выполнена успешно.
2 Несоответствие атрибутов объекта. Файл открыт только для записи или чтение переменной типа Integer не поддерживается.
12 Файл не открыт. Аргумент ID не является идентификатором файла.
16 Достигнут конец файла. Недостаточно данных для прочтения переменной типа Integer.

Примеры использования:

В приведенном примере технологическая программа производит чтение целого значения из открытого файла Handle:

```
Value := FL_RD_I (Handle, 0);
if FL_ERR () = 0 then
        (* произвести анализ значения Value *)
        ...
end_if;
```

Поскольку Value может принимать любые значения, для обнаружения ошибок следует использовать дополнительный вызов функции FL_ERR.

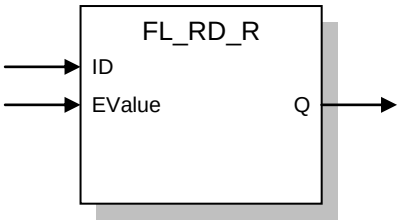
Так как код клавиатуры занимает один байт, при работе с пультом оператора M920L можно упростить алгоритм обработки ошибок и опустить вызов функции FL_ERR:

```
KeyCode := FL_RD_I (1, -1);
if KeyCode >= 0 then
        (* анализ нажатой клавиши *)
        ...
```

```
    end_if;
```

Примечание: Во время работы с пультом оператора M920L, вместо FL_RD_I следует использовать специализированную функцию GET_CHAR.

5.6 FL_RD_R



Параметры функции:

ID:	INT	Идентификатор файла.
EValue:	REAL	Результат вызова в случае ошибки.
Q:	REAL	EValue или прочитанная переменная.

Описание функции:

Функция FL_RD_R используется для чтения из файла переменной вещественного типа. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

На вход EValue подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

Чтение производится с текущей позиции указателя файла. Если разница между указателем и концом файла менее 4 байт, функция FL_RD_R возвращает EValue и устанавливает код ошибки “12 - достигнут конец файла”. После успешного завершения операции текущая позиция указателя увеличивается на размер прочитанного блока данных. При обнаружении ошибки позиция указателя не меняется.

Функция FL_RD_R, вызванная идентификатором файла CONIO, производит чтение кода нажатой клавиши из буфера клавиатуры. Если буфер клавиатуры пуст, FL_RD_R возвращает EValue и устанавливает код ошибки “16 - достигнут конец файла”.

Функция FL_RD_I устанавливает следующие коды ошибок:

0 Операция выполнена успешно.
2 Несоответствие атрибутов объекта. Файл открыт только для записи или нет доступа на чтение вещественных значений.
12 Файл не открыт. Аргумент ID не является идентификатором файла.
16 Достигнут конец файла. Недостаточно данных для чтения значения типа Real.

Примеры использования:

В приведенном примере технологическая программа производит чтение вещественного значения из открытого файла Handle:

```
    Value := FL_RD_R (Handle, 0.0);
    if FL_ERR () = 0 then
        (* произвести анализ значения Value *)
        ...
    end_if;
```

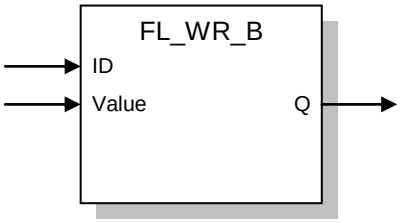
Поскольку Value может принимать любые значения, для обнаружения ошибок следует использовать дополнительный вызов функции FL_ERR.

Так как код клавиатуры занимает один байт, при работе с пультом оператора M920L можно упростить алгоритм обработки ошибок и опустить вызов функции FL_ERR:

```
    KeyCode := FL_RD_R (1, -1.0);
    if KeyCode >= 0.0 then
        (* анализ кода нажатой клавиши *)
        ...
    end_if;
```

В результате выполнения программы в переменную KeyCode копируется код нажатой клавиши в вещественном формате, или константа -1.0, если буфер клавиатуры пуст.

5.7 FL_WR_B



Параметры функции:

ID:	INT	Идентификатор файла
Value:	INT	Значение переменной
Q:	INT	Код ошибки

Описание функции:

Функция FL_WR_B используется для записи одного байта в файл. Старшие 24 разряда значения Value игнорируются. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I. В результате вызова Result функция FL_WR_B возвращает код выполнения операции.

Запись производится с текущей позиции указателя файла. По завершению операции текущая позиция указателя увеличивается на размер записанного блока данных. При обнаружении ошибки позиция указателя не меняется.

Функция FL_WR_B, вызванная идентификатором файла CONIO, производит вывод одного байта на пульт оператора M920L. Выводимое значение может быть как знаком ASCII, так и непечатным знаком или кодом специальной команды.

Так как ресурсы технологической программы доступны только для чтения, попытка записи файла ресурса устанавливает код ошибки “2 - несоответствие атрибутов объекта”.

Коды ошибок, возвращаемые функцией FL_WR_B:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Файл открыт только для чтения.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи байта.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры терминала.

Примеры использования:

Запись в файл Handle младшего байта переменной Value:

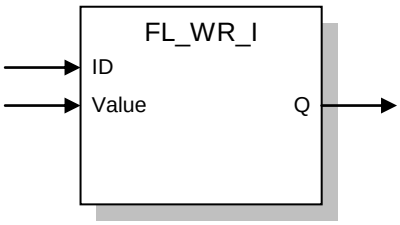
```
if FL_WR_B (Handle, Value) = 0 then
      (* операция выполнена успешно *)
      ...
else
      (* ошибка. прочитать код ошибки *)
      Error := FL_ERR ();
end_if;
```

Передача одного байта в пульт оператора M920L выглядит следующим образом:

```
if FL_WR_B (1, 16#0C) = 0 then
      (* очистка экрана завершена *)
      ...
end_if;
```

Примечание. Для работы с пультом оператора M920L вместо FL_WR_B следует использовать специализированную функцию PUT_CHAR.

5.8 FL_WR_I



Параметры функции:

ID:	INT	Идентификатор файла.
Value:	INT	Значение переменной.
Q:	INT	Код ошибки.

Описание функции:

Функция FL_WR_I используется для записи в файл переменной целого типа. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I. В результате вызова функция FL_WR_I возвращает код выполнения операции.

Запись производится с текущей позиции указателя файла. По завершению операции текущая позиция указателя увеличивается на размер записанного блока данных. При обнаружении ошибки позиция указателя не меняется.

Функция FL_WR_I, вызванная идентификатором файла CONIO, производит вывод одного байта на пульт оператора M920L. Выводимое значение может быть как знаком ASCII, так и непечатным знаком или кодом специальной команды пульта оператора.

Так как ресурсы технологической программы доступны только для чтения, попытка записи файла ресурса устанавливает код ошибки “2 - несоответствие атрибутов объекта”.

Коды ошибок, возвращаемые функцией FL_WR_I:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Файл открыт только для чтения.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи переменной типа Integer.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

Запись в файл Handle значения 32-х разрядной переменной Value:

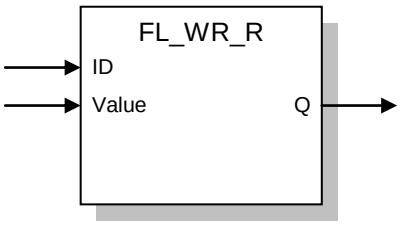
```
if FL_WR_I (Handle, Value) = 0 then
      (* операция выполнена успешно *)
      ...
else
      (* ошибка. прочитать код ошибки *)
      Error := FL_ERR ();
end_if;
```

Во время работы с пультом оператора M920L, функция FL_WR_R служит для вывода одного байта или кода специальной команды:

```
if FL_WR_I (1, 88) = 0 then
      (* вывод знака 'X' успешно завершен *)
      ...
end_if;
```

Примечание. Для работы с пультом оператора M920L вместо FL_WR_I следует использовать специализированную функцию PUT_CHAR.

5.9 FL_WR_R



Параметры функции:

ID:	INT	Идентификатор файла.
Value:	REAL	Значение переменной.
Q:	INT	Код ошибки.

Описание функции:

Функция FL_WR_R используется для записи в файл переменной вещественного типа. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I. В результате вызова функция FL_WR_R возвращает код выполнения операции.

Запись производится с текущей позиции указателя файла. По завершению операции текущая позиция указателя увеличивается на размер записанного блока данных. При обнаружении ошибки позиция указателя не меняется.

Функция FL_WR_R, вызванная идентификатором файла CONIO, производит преобразование вещественного аргумента в целое число, а после этого вывод младшего байта результата преобразования на пульт оператора M920L. При обнаружении переполнения функция возвращает ошибку “23 - ошибка преобразования данных”. Выводимое значение может быть как знаком ASCII, так и непечатным знаком или кодом специальной команды пульта оператора.

Так как ресурсы технологической программы доступны только для чтения, попытка записи файла ресурса устанавливает код ошибки “2 - несоответствие атрибутов объекта”.

Коды ошибок, возвращаемые функцией FL_WR_R:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Файл открыт только для чтения.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи переменной типа Real.
23	Ошибка преобразования данных. Неверное значение Value.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

В приведенном примере технологическая программа производит запись значения вещественной переменной Value в файл Handle:

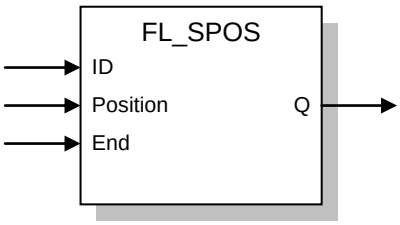
```
if FL_WR_R (Handle, Value) = 0 then
      (* операция выполнена успешно *)
      ...
else
      (* ошибка. прочитать код ошибки *)
      Error := FL ERR ();
end_if;
```

Во время работы с пультом оператора M920L, функция FL_WR_R служит для вывода одного байта или кода специальной команды:

```
if FL_WR_R (1, 12.0) = 0 then
      (* очистка экрана успешно завершена *)
      ...
end_if;
```

Примечание. Для работы с пультом оператора M920L вместо FL_WR_R следует использовать специализированную функцию PUT_CHAR.

5.10 FL_SPOS

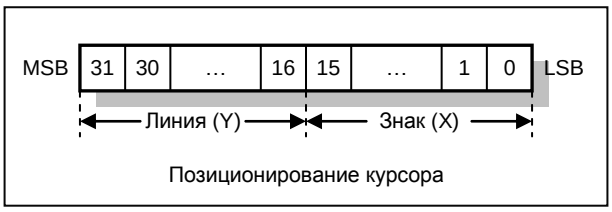


Параметры функции:

ID:	INT	Идентификатор файла.
Position:	INT	Новая позиция указателя в файле.
End:	BOOL	Признак установки текущей позиции в конец файла.
Q:	INT	Код ошибки.

Описание функции:
Функция FL_SPOS используется для установки текущей позиции в файле, который является идентификатором потока ввода/вывода пульта оператора или был открыт функцией FL_OPEN_I.

Если позиция устанавливается в конец файла (End = true), к аргументу Position добавляется размер файла или ресурса. Вместо отрицательных значений аргумента Position подставляется ноль. Функция позволяет устанавливать позицию за границу окончания файла.



При работе с пультом оператора, функция FL_SPOS служит для позиционирования курсора. Аргумент End не используется и может принимать любое значение. В младших 16-ти разрядах аргумента Position передаётся номер знака, в старших – номер линии экрана пульта оператора. Нулевые значения соответствуют левому верхнему углу области вывода. Операция позиционирования курсора кодируется и передаётся в пульт оператора как последовательность трёх байт:

<1Bh> <XX> <YY>

Где XX – координата X курсора, YY – координата Y курсора.

Коды ошибок, возвращаемые функцией FL_SPOS:

0 Операция выполнена успешно.
12 Файл не открыт. Аргумент ID не является идентификатором файла.
100 Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:
Предположим, что файл Handle – это ресурс, содержащий следующий массив байт:

```
20h 21h 22h 23h 24h 25h 26h 27h 28h 29h
```

Установим позицию относительно начала файла и прочитаем один байт. Для упрощения опустим алгоритмы обработки ошибок:

```
Error := FL_SPOS (Handle, 2, false); (* установить позицию *)
Value := FL_RD_B (Handle, -1); (* читать байт *)
```

В результате функция FL_RD_B прочитает байт, начиная с ранее установленной позиции 2, и вернёт значение 22h. Несколько иначе происходит установка позиции относительно конца файла:

```
Error := FL_SPOS (Handle, -2, true);      (* установить позицию *)
Value := FL_RD_B (Handle, -1);            (* читать байт *)
```

В этом случае к смещению Position будет прибавлена длина файла (-2 + 10 = 8), а функция FL_RD_B вернёт значение 28h. После прочтения байта значение позиции будет равно 9-ти.

Следует учитывать, что новая позиция в файле будет установлена, даже если её значение выходит за границы (размер) файла. Рассмотрим следующий пример:

```
Error := FL_SPOS (Handle, 100, false);     (* установ. неверную позицию *)
Value := FL_RD_B (Handle, -1);             (* читать байт *)
Position := FL_GPOS (Handle, -1);          (* прочитать позицию файла *)
```

На этот раз функция FL_RD_B вернёт -1 и установит код ошибки “16 - достигнут конец файла”. Чтение позиции вернёт значение 100.

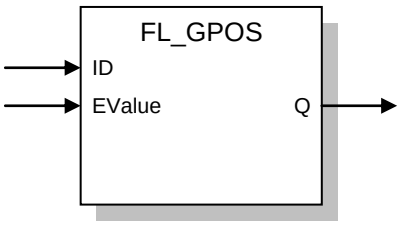
Функция FL_SPOS также может быть использования для позиционирования курсора на экране пульта оператора. Например, следующая программа:

```
if FL_SPOS (1, 16#20001, false) = 0 then
      (* операция выполнена успешно *)
      ...
else
      (* произошла ошибка, прочитать код ошибки *)
      Error := FL_ERR ();
end_if;
```

устанавливает курсор после первого знака второй линии индикатора (X = 1, Y = 2).

Примечание. Для работы с пультом оператора M920L, вместо FL_SPOS следует использовать специализированную функцию GOTOXY.

5.11 FL_GPOS



Параметры функции:

ID:	INT	Идентификатор файла.
EValue:	INT	Результат вызова в случае ошибки.
Q:	INT	Текущая позиция указателя или EValue.

Описание функции:

Функция FL_GPOS используется для получения текущей позиции в файле. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

На вход EValue подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR. Примечание: так как функция FL_SPOS не позволяет устанавливать отрицательное значение позиции в файле,

При работе с пультом оператора M920L функция FL_GPOS не используется. Вызов функции FL_GPOS с идентификатором файла CONIO возвращает EValue и устанавливает код ошибки “20 – реализация метода отсутствует”.

Коды ошибок, устанавливаемые функцией FL_GPOS:

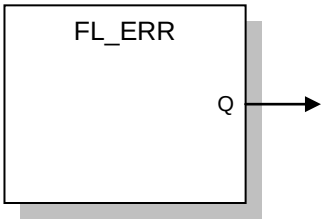
0	Операция выполнена успешно.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
20	Реализация метода отсутствует.

Примеры использования:
Если не обнаружены ошибки, функция возвращает текущую позицию файла. Для простоты примера анализ ошибочных ситуаций опущен.

```
Error := FL_SPOS (Handle, 2, false);        (* установить позицию *)
Value := FL_RD_B (Handle, -1);              (* читать байт *)
Position := FL_GPOS (Handle, -1);           (* прочитать позицию *)
```

Сначала программа устанавливает позицию указатель и читает из файла один байт. После этого в переменную Position копируется текущее значение указателя в файле, которое будет равно: 2 + 1 = 3.

5.12 FL_ERR



Параметры функции:

Q:	INT	Код ошибки последней операции.

Описание функции:
Функция FL_ERR служит для чтения значения внутренней переменной исполнительной системы, содержащей код ошибки последней операции с массивом или файлом.

Примеры использования:
Каждая функция, напрямую (например, FL_RD_B) или косвенно (PUT_CHAR) работающая файлами или массивами, обновляет код ошибки.

В большинстве случаев нецелесообразно использовать FL_ERR для получения состояния функций, результатом которых является код ошибки (таких как FL_SPOS или FL_CLOSE).

```
if FL_SPOS (Handle, 2, false) = 0 then (* установить позицию *)
      (* ошибок нет. продолжить... *)
      ...
end_if;
```

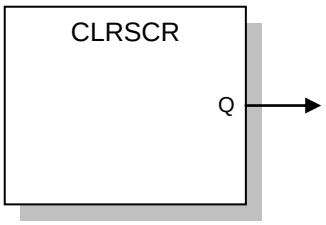
Во всех остальных случаях вызываем FL_ERR сразу после файловой операции:

```
Value := FL_RD_B (Handle, -1); (* читать байт *)
if FL_ERR () = 0 then (* выполнить проверку *)
      (* ok. продолжение работы с файлом *)
      ...
end_if;
```

Пульт оператора M920L является пассивным устройством и предназначен для визуализации технологического процесса, в некоторых случаях для упрощения программы анализ ошибок может быть опущен.

Каждая функция, объявленная исполнительной системой Unimod Pro Solution, обязана возвращать результат. Даже если результат функции не используется программой, под возвращаемое значение должна быть зарезервирована временная переменная.

5.13 CLRSCR



Параметры функции:

Q:	INT	Код ошибки.

Описание функции:

Функция CLRSCR очищает экран пульта оператора M920L и устанавливает позицию курсора в верхний левый угол. Если в списке юнитов интеллектуального модуля не числится юнит U920L, функция CLRSCR устанавливает и возвращает код ошибки “22 - файл не найден”.

Для очистки экрана в пульт оператора передаётся байт 0Ch. Полное писание набора команд можно найти в технической документации пульта оператора M920L.

Вызов функции CLRSCR во время выполнения асинхронного ввода возвращает ошибку с кодом “100 – файл занят”.

Коды ошибок, возвращаемые функцией CLRSCR:

0	Операция выполнена успешно.
22	Файл не найден. На модуле не установлен пульт оператора M920L.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

Там где необходимо произвести очистку экрана:

```
if CLRSCR () = 0 then (* очищаем экран *)
      (* ok. продолжаем вывод... *)
      ...
else
      (* обнаружена ошибка вывода *)
      ...
end_if;
```

Несмотря на интуитивную простоту использования CLRSCR, в большинстве случаев очистка экрана, как и любая другая последовательность команд, может быть закодирована как часть текстового вывода в ресурсах технологической программы.

Например, строковый ресурс с номером 100 содержит:

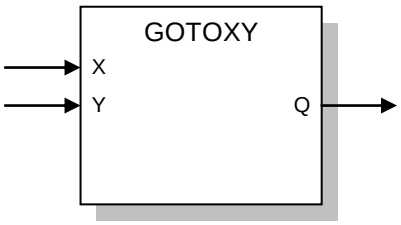
```
$0CЗначение переменной:$1B$05$01
```

Следующий код очищает экран и выводит значение переменной Value и устанавливает позицию курсора перед шестым знаком второй линии:

```
if PUT_RSRC (1, 100) = 0 then (* выводим ресурс *)
      if PUT_INT (1, Value) = 0 then
            (* ошибок не обнаружено. продолжить... *)
            ...
      end_if;
end_if;
```

Примечание. Полное писание набора команд находится в технической документации пульта оператора M920L.

5.14 GOTOXY



Параметры функции:

X:	INT	Номер знака в линии
Y:	INT	Номер линии
Q:	INT	Код ошибки

Описание функции:

Функция CLRSCR производит установку курсора пульта оператора M920L в произвольную позицию. Новая позиция курсора задаётся аргументами X и Y. Если в списке юнитов интеллектуального модуля не числится юнит U920L, функция GOTOXY устанавливает и возвращает код ошибки “22 - файл не найден”.

Операция позиционирования курсора кодируется и передаётся в пульт оператора как последовательность трёх байт:

1Bh XX YY

Где XX – координата X курсора, YY – координата Y курсора.

Полное писание набора команд можно найти в технической документации пульта оператора M920L.

Коды ошибок, возвращаемые функцией GOTOXY:

0 Операция выполнена успешно.
22 Файл не найден. На модуле не установлен пульт оператора M920L.
100 Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

Функция GOTOXY устанавливает позицию курсора, с которой в последствии будет происходить вывод. Основная область использования функции GOTOXY – вывод переменных, значение которых изменяется динамически.

Например, следующая программа устанавливает позицию курсора на вторую линию после 5-го знака и производит вывод значения переменной Value:

```
if GOTOXY (5, 1) = 0 then (* позиционирование курсора *)
      if PUT_INT (1, Value) = 0 then (* вывод значения *)
            (* ошибок не обнаружено. продолжить... *)
            ...
      end_if;
end_if;
```

Поскольку результатом преобразования динамически меняющейся переменной Value является строка переменной длины, перед вызовом функции PUT_INT следует установить номинальную ширину вывода (см. описание конфигурирования интерфейса вывода, функция OPERATE).

Следует отметить, что в большинстве случаев позиционирование курсора, как и любая другая последовательность команд, может быть закодирована как часть текстового вывода в ресурсах технологической программы.

Например, строковый ресурс с номером 100 содержит:

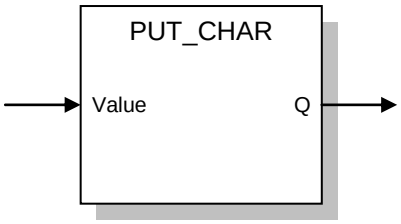
$0CЗначение переменной Value:$1B$05$01

Следующая программа очищает экран, печатает определённый текст, устанавливает позицию курсора и выводит значение переменной Value:

```
if PUT_RSRC (1, 100) = 0 then (* выводим ресурс *)
      if PUT_INT (1, Value) = 0 then (* выводим значение переменной *)
            (* ошибок не обнаружено. продолжить... *)
            ...
      end_if;
end_if;
```

Как видно из примера, три из четырёх выполненных операций закодированы данными ресурса технологической программы.

5.15 PUT_CHAR



Параметры функции:

Value:	INT	Значение.
Q:	INT	Код ошибки.

Описание функции:
Функция PUT_CHAR передаёт байт в пульт оператора M920L. Старшие 24 разряда значения Value игнорируются. Если в списке юнитов интеллектуального модуля не числится юнит U920L, функция PUT_CHAR устанавливает и возвращает код ошибки “22 - файл не найден”.

Коды ошибок, возвращаемые функцией PUT_CHAR:

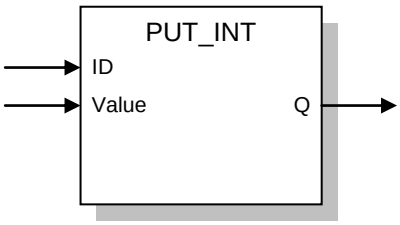
0	Операция выполнена успешно.
22	Файл не найден. На модуле не установлен пульта оператора M920L.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:
PUT_CHAR предназначена работы с пультом оператора ввода/вывода и является функциональным эквивалентом функции FL_WR_B. Функция может быть использована для передачи знака ASCII или кода команды на пульт оператора.

```
if PUT_CHAR (Value) = 0 then (* выводим байт *)
      (* ошибок не обнаружено. продолжить... *)
      ...
end_if;
```

В большинстве случаев вывод байта, как и любая другая последовательность команд, может быть закодирована как часть текстового вывода в ресурсах технологической программы.

5.16 PUT_INT



Параметры функции:

ID:	INT	Идентификатор файла или консоли.
Value:	INT	Выводимое значение.
Q:	INT	Код ошибки.

Описание функции:

Функция PUT_INT производит преобразование целого значения в строку с последующей записью в файл. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

Так как ресурсы технологической программы доступны только для чтения, попытка записи файла ресурса устанавливает код ошибки “2 - несоответствие атрибутов объекта”.

Вызванная с идентификатором файла CONIO, функция PUT_INT служит для асинхронного вывода целого числа в десятичном формате на дисплей пульта оператора.

Вывод числа происходит с текущей позиции курсора. В зависимости от конфигурации интерфейса вывода, возможен вывод числа со знаком или без знака. Если конфигурацией разрешена работа с числами со знаком и аргумент имеет отрицательное значение, первым знаком, выводимым на индикатор, будет ‘-‘ (минус). Количество выводимых знаков ограничено конфигурацией интерфейса вывода (см. описание соответствующей функции OPERATE).

Диапазон выводимых значений:

Со знаком	-2147483648..2147483647
Без знака	0..4294967295

Вызов PUT_INT преобразует полученное значение в строку, начинает передачу строки на пульт оператора и возвращает код ошибки “0 - ошибок не обнаружено”. Асинхронная передача может быть прервана переинициализацией пульта оператора M920L через вызов соответствующей функции OPERATE.

Функция PUT_INT использует следующие параметры конфигурации вывода:

LIMIT	Номинальная ширина выводимой строки.
SIGNED	Режим преобразования числа: со знаком/без знака.
PLUS	Разрешение вставки символа ‘+’ (плюс) перед положительными значениями.

Полный список флагов конфигурации интерфейса вывода приведен в описании функции OPERATE.

Коды ошибок, возвращаемые функцией PUT_INT:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Ошибка записи в файл.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

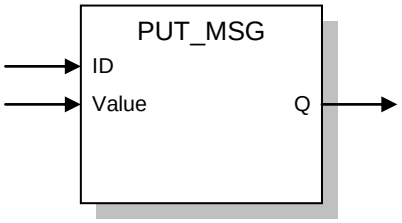
Примеры использования:

Функция производит преобразование целой переменной в строку и производит запись в файл или терминал ввода-вывода.

```
Value := 100;
if PUT_INT (Handle, Value) = 0 then (* пишем строку '100' *)
      (* ошибок не обнаружено *)
      ...
end_if;
```

Во время работы с пульта оператора функция PUT_INT производит вывод целого числа на экран. Дополнительно интерфейс вывода может быть сконфигурирован функцией OPERATE.

5.17 PUT_MSG



Параметры функции:

ID:	INT	Идентификатор файла или консоли
Value:	MSG	Выводимое значение
Q:	INT	Код ошибки

Описание функции:

Функция PUT_MSG производит запись строковой переменной в файл. Файл ID должен быть идентификатором потока ввода/вывода терминала или предварительно открыт функцией FL_OPEN_I.

Так как ресурсы технологической программы доступны только для чтения, попытка записи файла ресурса устанавливает код ошибки “2 - несоответствие атрибутов объекта”.

Вызванная с идентификатором файла CONIO, функция PUT_MSG служит для асинхронного вывода последовательности знаков на дисплей терминала. Также при помощи функции PUT_MSG в терминал может быть передана последовательность управляющих команд.

Вывод строки происходит с текущей позиции курсора. Количество выводимых знаков ограничено конфигурацией интерфейса вывода (см. описание соответствующей функции OPERATE) Асинхронная передача может быть прервана переинициализацией терминала M920L через вызов соответствующей функции OPERATE.

Функция PUT_MSG использует следующие параметры конфигурации вывода:

LIMIT	Номинальная ширина выводимой строки.

Полный список флагов конфигурации интерфейса вывода приведен в описании функции OPERATE.

Коды ошибок, возвращаемые функцией PUT_MSG:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Ошибка записи в файл.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры терминала.

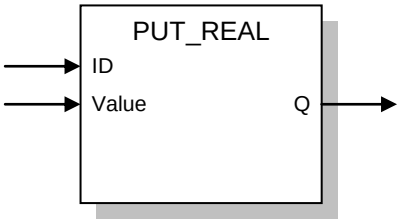
Примеры использования:

Функция производит запись строковой переменной в файл или терминал ввода-вывода M920L.

```
Value := 'this the string';
if PUT MSG (Handle, Value) = 0 then (* пишем строку *)
      (* ошибок не обнаружено *)
      ...
end_if;
```

Во время работы с терминалом функция PUT_MSG производит вывод сообщения на экран. Дополнительно интерфейс вывода может быть сконфигурирован функцией OPERATE.

5.18 PUT_REAL



Параметры функции:

ID:	INT	Идентификатор файла или консоли
Value:	REAL	Выводимое значение
Q:	INT	Код ошибки

Описание функции:

Функция PUT_REAL производит преобразование вещественного значения в строку с последующей записью в файл. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

Вызванная с идентификатором файла CONIO, функция PUT_REAL служит для асинхронного вывода числа с плавающей запятой на дисплей пульта оператора M920L.

Вывод числа происходит с текущей позиции курсора. При отрицательном значении аргумента PUT_REAL выводит число, начинающееся со знака '-' (минус). В противном случае, если разрешена вставка плюса и аргумент не равен нулю, первым выведенным символом будет '+' (плюс). Количество выводимых знаков ограничено конфигурацией интерфейса вывода (см. описание функции OPERATE).

Числа меньше чем установленная точность преобразования, или выходящие за пределы области вывода, выводятся на экран пульта оператора в экспоненциальном формате.

"Неправильные" вещественные числа выводятся на экран в виде строк:

±1.#INF	плюс или минус бесконечность
±1.#NAN	нечисловое выражение
1.#INDF	неопределенное значение

Функция PUT_REAL использует следующие флаги конфигурации вывода:

LIMIT	Номинальная ширина выводимой строки.
PRECISION	Точность преобразования/количество знаков после запятой.
PLUS	Разрешение вставки символа '+' (плюс) перед положительными значениями.
TRIM	Удаление завершающих нулей в вещественной части выводимого числа.

Примечание. Полный список флагов находится в описании функции OPERATE.

Коды ошибок, возвращаемые функцией PUT_REAL:

1	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Ошибка записи в файл.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

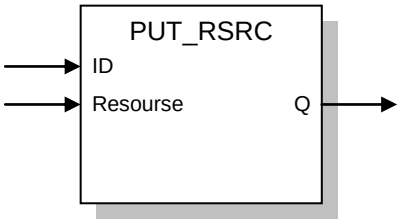
Примеры использования:

Функция производит преобразование вещественной переменной в строку и производит запись в файл или терминал ввода-вывода.

```
Value := 100.0;
if PUT_REAL (Handle, Value) = 0 then (* пишем строку '100.0' *)
      (* ошибок не обнаружено *)
      ...
end_if;
```

Во время работы с пультом оператора функция PUT_REAL производит вывод вещественного значения на экран. Дополнительно, интерфейс вывода может быть сконфигурирован функцией OPERATE.

5.19 PUT_RSRC



Параметры функции:

ID:	INT	Идентификатор файла или консоли
Resourse:	INT	Идентификатор ресурса
Q:	INT	Код ошибки

Описание функции:

Функция PUT_RSRC предназначена для записи данных ресурса в поток ввода/вывода пульта оператора или файл, предварительно открытый функцией FL_OPEN_I.

Уникальный номер (имя) ресурса передаётся в младшем слове RsrcId. Опционально, в старших 16 разрядах RsrcId может быть указан тип запрашиваемого ресурса. Если тип ресурса с заданным именем не совпадёт с запрошенным значением, функция PUT_RSRC возвращает ошибку “22 - файл не найден”. Полный список поддерживаемых типов ресурсов представлен в описании функции FL_OPEN_I.

Вызванная с идентификатором файла CONIO, функция PUT_RSRC служит для асинхронной передачи данных из ресурса в M920L. При этом не накладывается никаких ограничений ни на тип, ни длину ресурса. Вывод данных ресурса происходит с текущей позиции курсора.

Вывод ресурсов на пульт оператора может быть использован для:

1) Вывода строк и графики на экран.
2) Передачи специальных команд.
3) Загрузки символов пользователя.

Коды ошибок, возвращаемые функцией PUT_RSRC:

0	Операция выполнена успешно.
2	Несоответствие атрибутов объекта. Ошибка записи в файл.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно места для записи.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

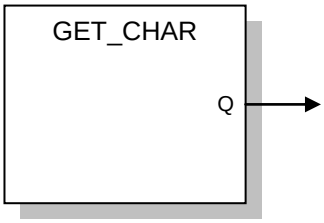
Функция предназначена для копирования данных из ресурса технологической программы в файл или для вывода текста и графики на экран терминала ввода-вывода.

Программа производит запись массива байт с номером 90 (5Ah) в предварительно открытый файл FileHandle:

```
if PUT_RSCR (FileHandle, 16#2005A) = 0 then
```

```
            (* ошибок не обнаружено *)
            ...
      end_if;
```

5.20 GET_CHAR



Параметры функции:

Q:	INT	Код нажатой клавиши или -1.

Описание функции:

Функция GET_CHAR предназначена для чтения кода нажатой клавиши из буфера клавиатуры. Если буфер клавиатуры пуст, GET_CHAR возвращает -1 и устанавливает код ошибки “16 - достигнут конец файла”.

Если в списке юнитов интеллектуального модуля не числится юнит U920L, функция GET_CHAR выходит с кодом ошибки “22 - файл не найден”.

Коды ошибок, устанавливаемые функцией GET_CHAR:

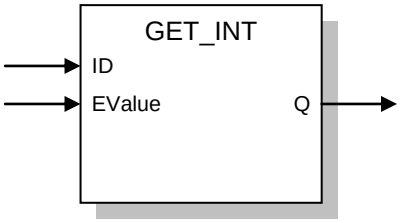
0 Операция выполнена успешно.
2 Несоответствие атрибутов объекта. Ошибка записи в файл.
16 Достигнут конец файла. Буфер клавиатуры пульта оператора пуст.
22 Файл не найден. На модуле не установлен пульт оператора M920L.
100 Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

Функция GET_CHAR предназначена работы с терминалом ввода/вывода и является функциональным аналогом функции FL_RD_B. Функция производит чтение кода нажатой клавиши из буфера клавиатуры:

```
      Value := GET_CHAR ();(* читаем буфер клавиатуры *)
      if Value >= 0 then
            (* Value содержит код клавиши... *)
            ...
      end_if;
```

5.21 GET_INT



Параметры функции:

ID:	INT	Идентификатор файла или консоли.
EValue:	INT	Результат вызова в случае ошибки.
Q:	INT	Прочитанное значение или EValue.

Описание функции:

Функция GET_INT предназначена для чтения значения целого значения из ресурса или файла. Чтение значения выполняется в текстовом режиме. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

Вызванная с идентификатором файла CONIO, функция GET_INT служит для асинхронного ввода целого числа с клавиатуры пульта оператора. В зависимости от конфигурации, возможен ввод числа со знаком или без знака.

Первый вызов функции GET_INT переводит пульт в режим ввода переменной. Модуль перехватывает нажатия клавиатуры и блокирует вызов остальных функций ввода с клавиатуры и вывода на индикатор. До завершения ввода, вызовы заблокированных функций возвращают код ошибки “100 – файл занят”. Ввод заканчивается нажатием клавиши ‘Esc’ или ‘Enter’.

Числовое выражение вводится с помощью клавиш ‘0’…’9’, ‘*’, ‘Del’ и ‘Enter’. Клавиша ‘*’ заменяет знак ‘-‘ (минус) и служит для ввода отрицательных значений. Кроме этого, для того чтобы обеспечить совместимость алгоритма ввода с GET_REAL, функция GET_INT отфильтровывает нажатия на клавишу ‘.’ (точка).

Подробное назначение клавиш в режиме ввода:

‘.’	Совместимость с FL_RD_R. Нажатие на эту клавишу расценивается как ошибка ввода.
‘*’	Заменяет знак ‘-‘ (минус). Для ввода отрицательного значения необходимо, чтобы клавиша ‘*’ была нажата первой. Если нажатие на клавишу ‘*’ произошло в процессе ввода числового выражения, или если используется режим ввода чисел без знака, нажатие расценивается как ошибка ввода.
‘0’…’9’	Собственно цифры. Служат для ввода числового выражения.
‘F1’…’F3’	Не используются. Нажатие на эти клавиши расценивается как ошибка ввода.
←,↑,→,↓	Нажатие на эти клавиши расценивается как ошибка ввода.
‘Esc’	Прерывает процесс ввода. Функция GET_INT возвращает EVALUE и устанавливает код ошибки “99 - операция прервана пользователем”.
‘Del’	Удаляет последний введённый символ.
‘Enter’	Завершает процесс ввода. Функция GET_INT возвращает введенное число и устанавливает код ошибки “0 - операция завершена успешно”.

Если в конфигурации режима ввода установлен флаг SOUND, ошибка ввода сигнализируется звуковым сигналом. Т.е. если в конфигурации пульта оператора включено подтверждение нажатия клавиш, ошибка ввода выдаст двойной сигнал.

Клавиши ‘0’…’9’, ‘*’ выводятся на экран, причем ‘*’ заменяется знаком ‘-’ (минус). Вывод происходит с текущей позиции курсора. Если вводимое число вышло за диапазон допустимых значений (32 разряда), последующие нажатия клавишей ‘0’…’9’ не выводятся на индикатор и расцениваются как ошибка ввода.

Для удаления последнего введенного символа служит клавиша ‘Del’.

Превышение максимально допустимого количества вводимых символов, установленное конфигурацией пульта оператора, сигнализируется как ошибка. Завершение операции ввода нажатием ‘Enter’ возвращает введенное значение. Подробная информация о конфигурации ввода находится в описании функций OPERATE.

Процесс ввода может быть прерван пользователем, нажатием клавиши ‘Esc’, или программно, переинициализацией пульта оператора функцией OPERATE. Следует обратить внимание на то, что вызов GET_INT после OPERATE начинает новый цикл ввода.

Функция GET_INT использует следующие флаги конфигурации ввода:

LIMIT	Максимальная ширина вводимой строки.
SIGNED	Режим ввода числа: со знаком/без знака.
SOUND	Включение звукового подтверждения ошибки ввода.

Полный список флагов конфигурации интерфейса ввода находится в описании функций OPERATE.

Функция GET_INT устанавливает следующие коды ошибок:

0	Операция успешно завершена.
2	Несоответствие атрибутов объекта. Ошибка чтения.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно данных для чтения.
23	Ошибка преобразования строки в целое значение.
99	Операция прервана. Асинхронный ввод прерван пользователем.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

При работе с ресурсом или файлом, функции GET_INT производит последовательное чтение файла (потока) в текстовом режиме и возвращает результат преобразования ‘строка’→’целое’.

Например, ресурс с номером 100 содержит следующую строку: “2300 -169 56 +999”
Чтение этих констант с помощью функции GET_INT :

```
Handle := FL_OPEN_I (0, 1, 100, true, false, 0);
if Handle <> 0 then
      Value1 := GET_INT (Handle, -1);
      Value2 := GET_INT (Handle, -1);
      Value3 := GET_INT (Handle, -1);
      Value4 := GET_INT (Handle, -1);
      Error := FL_CLOSE (Handle);
end_if;
```

После выполнения программы переменные Value1, Value2, Value3 и Value4 будут заполнены прочитанными из ресурса данными и примут значения 2300, -169, 56 и 99 соответственно.

Во время использования совместно с терминалом ввода-вывода, первый вызов функции GET_INT переводит коммуникационный интерфейс пульта оператора в режим асинхронного ввода числа с клавиатуры. Переменная, привязанная к каналу RCBUSY, принимает значение TRUE. В этом режиме модуль перехватывает и обрабатывает нажатия на клавиатуру и производит необходимый вывод на экран. Работа всех функций вывода временно блокируется, а функции, предназначенные для чтения буфера клавиатуры, сигнализируют, что ни одна клавиша не была нажата.

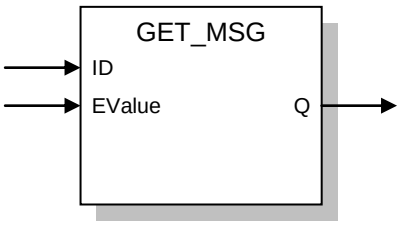
```
Temp := GET_INT (1, -1); (* инициируем ввод *)
```

Завершение ввода оператором происходит после нажатия на клавишу ‘Esc’ (отменить) или клавишу ‘Enter’ (принять). В этот момент устанавливается канал RCREADY, сигнализируя о том, что ввод завершен, и технологическая программа может прочитать его результат.

```
Temp := GET_INT (1, -1); (* чтение результата *)
case FL_ERR () of
(* успешное завершение ввода *)
      0:    Result := Temp;
      ...
      (* операция прервана оператором *)
      99:   ...
else:
      (* обработка остальных ошибок *)
      ...
end_case;
```

Следующий после установки RCREAY вызов функции GET_INT устанавливает соответствующий код ошибки и возвращает результат ввода. При этом сбрасываются каналы RCBUSY и RCREADY. Результат может содержать как введенное значение, так и ‘EValue’, если операция была прервана нажатием клавиши ‘Esc’.

5.22 GET_MSG



Параметры функции:

ID:	INT	Идентификатор файла или консоли
EValue:	MSG	Результат вызова в случае ошибки
Q:	INT	Прочитанное значение или EValue

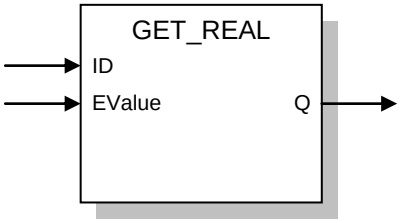
Описание функции:

Функция GET_MSG предназначена для чтения строки из ресурса или файла. Чтение выполняется в текстовом режиме. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

Функция GET_MSG устанавливает следующие коды ошибок:

0 Операция успешно завершена.
2 Несоответствие атрибутов объекта. Ошибка чтения.
12 Файл не открыт. Аргумент ID не является идентификатором файла.
16 Достигнут конец файла. Недостаточно данных для чтения.
23 Ошибка преобразования строки в целое значение.
99 Операция прервана. Асинхронный ввод прерван пользователем.
100 Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

5.23 GET_REAL



Параметры функции:

ID:	INT	Идентификатор файла или консоли.
EValue:	REAL	Результат вызова в случае ошибки.
Q:	REAL	Прочитанное значение или EValue.

Описание функции:

Функция GET_REAL предназначена для чтения значения вещественного типа из файла. Чтение значения выполняется в текстовом режиме. Файл ID должен быть идентификатором потока ввода/вывода пульта оператора или предварительно открыт функцией FL_OPEN_I.

Функция, вызванная идентификатором файла CONIO, производит асинхронный ввод вещественного числа с клавиатуры пульта оператора.

Первый вызов функции GET_REAL переводит пульт в режим ввода переменной. Модуль перехватывает нажатия клавиатуры и блокирует вызов остальных функций ввода с клавиатуры и вывода на индикатор. До завершения ввода, вызовы заблокированных функций возвращают код ошибки “100 – файл занят”. Ввод заканчивается нажатием клавиши ‘Esc’ или ‘Enter’.

Числовое выражение вводится с помощью клавиш ‘0’…’9’, ’.’ и ‘*’, ‘Del’ и ‘Enter’. Клавиша ‘*’ заменяет знак ‘-‘ (минус) и служит для ввода отрицательных значений. Ввод чисел в экспоненциальном формате (вида 1,2345E+10) не поддерживается.

Подробное назначение клавиш в режиме ввода:

‘.’	Отделяет целую и вещественную часть вводимого значения.
‘*’	Заменяет знак ‘-‘ (минус). Для ввода отрицательного значения необходимо, чтобы клавиша ‘*’ была нажата первой. Если нажатие на клавишу ‘*’ произошло в процессе ввода числового выражения, или если используется режим ввода чисел без знака, нажатие расценивается как ошибка ввода.
‘0’…’9’	Собственно цифры. Служат для ввода числового выражения.
‘F1’…’F3’	Не используются. Нажатие на эти клавиши расценивается как ошибка ввода.
←,↑,→,↓	Нажатие на эти клавиши расценивается как ошибка ввода.
‘Esc’	Прерывает процесс ввода. Функция GET_REAL возвращает EVALUE и устанавливает код ошибки “99 - операция прервана пользователем”.
‘Del’	Удаляет последний введённый символ.
‘Enter’	Завершает процесс ввода. Функция GET_REAL возвращает введенное число и устанавливает код ошибки “0 - операция завершена успешно”.

Если в конфигурации режима ввода установлен флаг SOUND, ошибка ввода сигнализируется звуковым сигналом. Т.е. если в конфигурации пульта оператора включено подтверждение нажатия клавиш, ошибка ввода выдаст двойной сигнал.

Вводимая строка может содержать только одну запятую, повторное нажатие клавиши '.' (точка) генерирует сигнал об ошибке. Наличие цифры перед запятой не обязательно, строка может начинаться с запятой. Для ввода простых (целых) чисел строка может не содержать ни одной запятой.

Если вводимое число вышло за диапазон допустимых значений (32-х разрядный тип FLOAT), последующие нажатия клавишей '0'…'9' расцениваются как ошибка ввода, а при нажатии на 'Enter' функция GET_REAL возвращает EVALUE и устанавливает код ошибки "неверное значение".

Для удаления последнего введенного символа служит клавиша 'Del'.

Превышение максимально допустимого количества вводимых символов, установленное конфигурацией пульта оператора, сигнализируется как ошибка. Завершение операции ввода нажатием 'Enter' возвращает введенное значение. Подробная информация о конфигурации ввода находится в описании функций OPERATE.

В случае, когда конфигурация ввода запрещает ввод чисел со знаком, нажатие клавиши '*' расценивается и сигнализируется как ошибка ввода.

Функция GET_REAL использует следующие флаги конфигурации ввода:

LIMIT	Максимальная ширина вводимой строки.
SIGNED	Режим ввода числа: со знаком/без знака.
SOUND	Включение звукового подтверждения ошибки ввода.

Полный список флагов конфигурации интерфейса ввода находится в описании функций OPERATE.

Функция GET_REAL устанавливает следующие коды ошибок:

0	Операция успешно завершена.
2	Несоответствие атрибутов объекта. Ошибка чтения.
12	Файл не открыт. Аргумент ID не является идентификатором файла.
16	Достигнут конец файла. Недостаточно данных для чтения.
23	Ошибка преобразования строки в вещественное значение.
99	Операция прервана. Асинхронный ввод прерван пользователем.
100	Файл занят. Выполняется асинхронный ввод с клавиатуры пульта оператора.

Примеры использования:

При работе с ресурсом или файлом, функции GET_REAL производит последовательное чтение файла (потока) в текстовом режиме и возвращает результат преобразования 'строка'→'вещественное'.

Например, ресурс с номером 100 содержит следующую строку: "23.2 -169.88 56.9 +999.1"

Чтение этих констант с помощью функции GET_REAL:

```
Handle := FL_OPEN_I (0, 1, 100, true, false, 0);
if Handle <> 0 then
      Value1 := GET_REAL (Handle, -1);
      Value2 := GET_REAL (Handle, -1);
      Value3 := GET_REAL (Handle, -1);
      Value4 := GET_REAL (Handle, -1);
      Error := FL_CLOSE (Handle);
end_if;
```

После выполнения программы переменные Value1, Value2, Value3 и Value4 будут заполнены прочитанными из ресурса данными и примут значения '23.2', '-169.88', '56.9' и '+999.1' соответственно.

Во время использования совместно с терминалом ввода-вывода, первый вызов функции GET_REAL переводит коммуникационный интерфейс пульта оператора в режим асинхронного ввода числа с клавиатуры. Переменная, привязанная к каналу RCBUSY, принимает значение TRUE. В этом режиме модуль перехватывает и обрабатывает нажатия на клавиатуру и производит необходимый вывод на экран. Работа всех функций вывода временно блокируется, а функции, предназначенные для чтения буфера клавиатуры, сигнализируют, что ни одна клавиша не была нажата.

```
Temp := GET_REAL (1, -1); (* инициируем ввод *)
```

Завершение ввода оператором происходит после нажатия на клавишу 'Esc' (отменить) или клавишу 'Enter' (принять). В этот момент устанавливается канал RCREADY, сигнализируя о том, что ввод завершен, и технологическая программа может прочитать его результат.

```
Temp := GET_REAL (1, -1); (* чтение результата *)
case FL_ERR () of
      (* успешное завершение ввода *)
      0:    Result := Temp;
      ...

      (* операция прервана оператором *)
      99:   ...
else:
      (* обработка остальных ошибок *)
      ...
end_case;
```

Следующий после установки RCREAY вызов функции GET_REAL устанавливает соответствующий код ошибки и возвращает результат ввода. При этом сбрасываются каналы RCBUSY и RCREADY. Результат может содержать как введенное значение, так и ‘EValue’, если операция была прервана нажатием клавиши ‘Esc’.

6. КОДЫ ОШИБОК

Для хранения кода ошибок исполнительная система модуля резервирует внутреннюю переменную, доступную для чтения через вызов функции FL_ERR. Выполнение любой библиотечной функции, работающей с файлами или пультом оператора, изменяет значение этой переменной.

Код ошибки – это значение-константа целого типа. Ненулевой код ошибки устанавливается, когда в силу тех или иных обстоятельств операция не могла быть завершена. Нулевой код ошибки сигнализирует об успешном завершении операции.

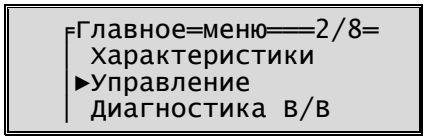
Перечень кодов ошибок интеллектуального модуля см. в документе «Unimod Pro. Руководство по программированию» п.6.4.7.15 *FL_ERR - Код ошибки операции с файлом.*

7. ТИПОВЫЕ ПРИЕМЫ ПРОГРАММИРОВАНИЯ ПУЛЬТА ОПЕРАТОРА

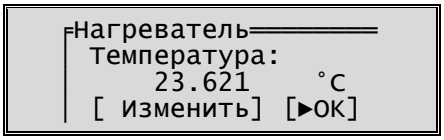
Условно обмен данными с пультом оператора можно разделить на несколько составляющих:

- Реакция технологической программы на нажатия клавиш.
- Вывод статического значения на экран пульта оператора.
- Вывод динамически изменяющегося значения переменной.
- Ввод технологического параметра с клавиатуры.
- Интерактивные/анимированные элементы экрана.

Строение графического интерфейса, реализуемого технологической программой, зависит от архитектуры и функциональности конкретной системы управления. Как правило, графический интерфейс состоит из нескольких позиций меню или нескольких экранов. На экране также могут присутствовать графические объекты, не несущие информационного содержания, такие как рамки и различного рода разделители, индикатор хода процесса (например, постепенно закрашиваемый в ходе копирования прямоугольник), графический курсор и т.п.



Меню



Экран

Программа должна анализировать нажатия на клавиши и осуществлять переход между позициями меню или экранами. Для этого необходимо считывать код нажатой клавиши из буфера клавиатуры и производить определённые действия.

Для чтения кода нажатой клавиши можно использовать функции FL_RD_B, FL_RD_I или FL_RD_R, но рациональней – функцию GET_CHAR, предназначенную для работы с пультом оператора. Функция GET_CHAR не требует аргументов и возвращает код нажатой клавиши или -1, если буфер клавиатуры пуст (см. описание функции GET_CHAR).

```
1
2   case GET_CHAR () of
3         16#0B: (* нажатие на кнопку '↑' *)
4         ...
5
6         16#0A: (* нажатие на кнопку '↓' *)
7         ...
8
9         -1: (* попадаем сюда, если буфер клавиатуры пуст *)
10        ...
11  else:
12        (* этого мы не ожидали... ошибка:
13           звуковой сигнал или 'message beep' *)
14        Status := PUT_CHAR (7);
15  end_case;
16
```

В приведенном выше примере реализовано чтение кода клавиши из буфера клавиатуры и простейший анализ с использованием оператора CASE.

Примечание. Следует отметить, что программирование алгоритмов работы с пультом оператора на языках FBD и LD затруднительно. В конечном итоге блок-диаграмма может разрастись до гигантских размеров. Поэтому программы рекомендуется разрабатывать на языке ST.

Далее рассмотрим процесс вывода. Выводимые переменные могут быть двух типов: статические и динамические. К статическим переменным относятся параметры, жёстко заданные пользователем и не меняющиеся в процессе работы устройства. Измеренные значения, технологические параметры, задаваемые сверху (например, с мастер-модуля), а также любые переменные, меняющиеся со временем, относятся к динамическим переменным.

Между выводом статической и динамической информации существует одно принципиальное отличие: если вывод статической информации происходит только раз, то для визуализации изменений, вывод динамической информации должен повторяться в каждом цикле программы или с определённым интервалом.

Алгоритм технологической программы составляется таким образом, чтобы в определённой последовательности произвести вывод графического и текстового содержания экрана и значение одной или нескольких переменных. После этого, как правило, программа переходит в состояния ожидания вмешательства со стороны пользователя, истечения определенного времени или какого-либо другого события. Рассмотрим пример, реализующий статический вывод:

```
1
2   Status := CLRSCR (); (* очистка экрана, установка X = 0, Y = 0 *)
3   Status := PUT_RSRC (1, 100); (* вывод на экран текста и графики *)
4   Status := GOTOXY (6, 2); (* установить позицию перед выводом значения *)
5   Status := OPERATE (ConsoleIO, 101, 16#5000009); (* задать параметры *)
6   Status := PUT_REAL (1, HeaterTemp); (* ...и вывести значение *)
7
```

Так как вывод динамической составляющей экрана информации должен периодически обновляться, выводимые данные должны быть разбиты на пассивную (графическую) и динамическую (информативную) составляющие. Сначала происходит прорисовка текста и графики и установка параметров форматирования вывода. В последствии, по таймеру или в каждом цикле приложения - установка позиции курсора и вывод значения одной или нескольких динамически меняющихся переменных.

Рассмотрим пример, реализующий асинхронную прорисовку экрана и вывод одной статической StatVar1 и двух динамических переменных DynVar1 и DynVar2. Текущее состояние экрана описывается локальной переменной ScreenState, а переменная ConTxBusy привязана к каналу TXBUSY.

```
1
2   if ConTxBusy then
3         return; (* выйти, если передатчик занят *)
4   end_if;
5
6   case ScreenState of
7         0:    Status := PUT_RSRC (1, 100); (* выводим графику и текст *)
8         1:    Status := OPERATE (ConTxBusy, 101, 16#5000008);
9               Status := PUT_REAL (1, StatVar1); (* вывод `StatVar1' *)
10
11        2:    Status := GOTOXY (10, 1); (* позиция для `DynVar1' *)
12        3:    Status := OPERATE (ConTxBusy, 101, 16#5000006);
13              Status := PUT_INT (1, DynVar1); (* вывод `DynVar1' *)
14
15        4:    Status := GOTOXY (10, 2); (* позиция для `DynVar2' *)
16        5:    Status := OPERATE (ConTxBusy, 101, 16#500000A);
17              Status := PUT_INT (1, DynVar2); (* вывод `DynVar2' *)
18              ScreenState := 1; (* вернутся к выводу `DynVar1' *)
19  end_case;
20
21  ScreenState := ScreenState + 1;
22
```

Пример ввода с клавиатуры. Для упрощения программа не использует каналы юнита U920L.

```
1
2   Status := CLRSCR (); (* очищаем экран *)
3   TempVar := GET_REAL (1, 0.0); (* запрашиваем вещественное число *)
4   if FL_ERR () = 0 then (* ошибки есть? *)
5         Result := TempVar; (* нет. буферизируем полученное значение *)
6   end_if;
7
```

ПРИЛОЖЕНИЕ 1. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ПУЛЬТА ОПЕРАТОРА

Таблицы наборов символов

Набор символов, соответствующий кодовой странице CP-866

Код символа		Старший полубайт													
		2	3	4	5	6	7	8	9	A	B	C	D	E	F
Младший полубайт	0		0	@	P	`	p	А	Р	а	U0			р	Ё
	1	!	1	A	Q	a	q	Б	С	б	U1			с	ё
	2	"	2	B	R	b	r	В	Т	в	U2			т	
	3	#	3	C	S	c	s	Г	У	г	U3			у	
	4	$	4	D	T	d	t	Д	Ф	д	U4			ф	
	5	%	5	E	U	e	u	Е	Х	е	U5			х	
	6	&	6	F	V	f	v	Ж	Ц	ж	U6			ц	
	7	'	7	G	W	g	w	З	Ч	з	U7			ч	
	8	(	8	H	X	h	x	И	Ш	и				ш	
	9	)	9	I	Y	i	y	Й	Щ	й				щ	
	A	*	:	J	Z	j	z	К	Ъ	к				ъ	
	B	+	;	K	[	k	10	Л	Ы	л				ы	
	C	,	<	L	¢	l	12	М	Ь	м				ь	
	D	-	=	M	]	m	15	Н	Э	н				э	
	E	.	>	N	^	n	↵	О	Ю	о				ю	
	F	/	?	O	_	o		П	Я	п				я	

Набор символов, соответствующий кодовой странице WIN-1251

Код символа		Старший полубайт													
		2	3	4	5	6	7	8	9	A	B	C	D	E	F
Младший полубайт	0		0	@	P	`	p				U0	А	Р	а	р
	1	!	1	A	Q	a	q				U1	Б	С	б	с
	2	"	2	B	R	b	r				U2	В	Т	в	т
	3	#	3	C	S	c	s				U3	Г	У	г	у
	4	$	4	D	T	d	t				U4	Д	Ф	д	ф
	5	%	5	E	U	e	u				U5	Е	Х	е	х
	6	&	6	F	V	f	v				U6	Ж	Ц	ж	ц
	7	'	7	G	W	g	w				U7	З	Ч	з	ч
	8	(	8	H	X	h	x			Ё	ё	И	Ш	и	ш
	9	)	9	I	Y	i	y					Й	Щ	й	щ
	A	*	:	J	Z	j	z					К	Ъ	к	ъ
	B	+	;	K	[	k	10					Л	Ы	л	ы
	C	,	<	L	¢	l	12					М	Ь	м	ь
	D	-	=	M	]	m	15					Н	Э	н	э
	E	.	>	N	^	n	↵					О	Ю	о	ю
	F	/	?	O	_	o						П	Я	п	я

Набор символов, соответствующий кодовой странице KOI8-r

Код символа		Старший полубайт													
		2	3	4	5	6	7	8	9	A	B	C	D	E	F
Младший полубайт	0		0	@	P	`	p					ю	п	Ю	П
	1	!	1	A	Q	a	q					а	я	А	Я
	2	"	2	B	R	b	r					б	р	Б	Р
	3	#	3	C	S	c	s			ё	Ё	ц	с	Ц	С
	4	$	4	D	T	d	t					д	т	Д	Т
	5	%	5	E	U	e	u					е	у	Е	У
	6	&	6	F	V	f	v					ф	ж	Ф	Ж
	7	'	7	G	W	g	w					г	в	Г	В
	8	(	8	H	X	h	x				U_0	х	ь	Х	Ь
	9	)	9	I	Y	i	y				U_1	и	ы	И	Ы
	A	*	:	J	Z	j	z				U_2	й	з	Й	З
	B	+	;	K	[	k	10				U_3	к	ш	К	Ш
	C	,	<	L	¢	l	12				U_4	л	э	Л	Э
	D	-	=	M	]	m	15				U_5	м	щ	М	Щ
	E	.	>	N	^	n	↵				U_6	н	ч	Н	Ч
	F	/	?	O	_	o					U_7	о	ъ	О	Ъ

Примечание. Символы U_0 - U_7 являются пользовательскими, их вид программируется отдельно.

Таблицы соответствия клавиш и возвращаемых кодов

Клавиша	Возвращаемый символ
.	'.'
*	'*'
0	'0'
1	'1'
2	'2'
3	'3'
4	'4'
5	'5'
6	'6'
7	'7'
8	'8'
9	'9'
F1	'A'
F2	'B'
F3	'C'
F4	'D'
←	08h
↑	0Bh
→	09h
↓	0Ah
Esc	1Bh
Del	7Fh
Enter	0Dh

ПРИЛОЖЕНИЕ 2. НАБОР ВСТРОЕННЫХ КОМАНД ПУЛЬТА ОПЕРАТОРА

Часть кодов, которые терминал M920L принимает от интеллектуального модуля M900, интерпретируется как управляющие команды. Полный список встроенных команд пульта оператора представлен в таблице:

Код	Команда
02h	Выключить подсветку
03h	Включить подсветку
04h	Включить звуковое подтверждение нажатия клавиш
05h	Выключить звуковое подтверждение нажатия клавиш
07h	Звуковой сигнал
08h	«Забой» (Backspace)
09h	Сдвинуть курсор вправо
0Ah	Переход на следующую строку
0Bh	Сдвинуть курсор вверх
0Ch	Очистка экрана и установка курсора в левый верхний угол
0Dh	Возврат курсора в начало строки
0Eh	Запретить прокрутку экрана
0Fh	Разрешить прокрутку экрана
10h	Запрограммировать символ пользователя
11h	Сделать курсор невидимым
12h	Курсор в виде мигающего прямоугольника
13h	Курсор в виде подчеркивания
14h	Курсор в виде подчеркнутого мигающего прямоугольника
15h	Сдвинуть курсор влево
1Bh	Установить курсор в произвольную позицию
1Eh	Префикс расширенного набора функций

Префикс расширенного набора функций

Префикс 1Eh предоставляет доступ к следующим функциям:

- 1Eh 10h – Задает тип курсора. Описание этой команды приводится в разделе “Вид и позиционирование курсора”;
- 1Eh 11h – Задает яркость подсветки. Описание этой команды приводится в разделе “Подсветка”;
- 1Eh 12h – Задает границы области вывода. Описание этой команды приводится в разделе “Область вывода”;
- 1Eh 20h – Прокручивает область вывода вверх на одну строку;
- 1Eh 21h – Прокручивает область вывода вниз на одну строку;
- 1Eh 22h – Прокручивает область вывода вправо на одну колонку;
- 1Eh 23h – Прокручивает область вывода влево на одну колонку;

Вид и позиционирование курсора

Абсолютное позиционирование курсора

Курсор может позиционироваться в произвольную позицию области вывода командой 1Bh. Команда <1Bh> имеет длину 3 байта:

<1Bh><XX><YY>

Где, <XX> – координата X курсора, <YY> – координата Y курсора.

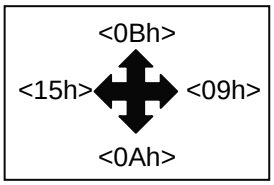
Максимальные значения параметров XX и YY определяются типом подключенного LCD или VFD дисплея, а так же границами области вывода, заданными командой 1Eh 12h. Нулевые значения соответствуют левому верхнему углу области вывода.

Пример использования:

Требуется установить курсор в 14-ю колонку 2-й строки области вывода. Подставляем соответствующие значения в команду и получаем следующую последовательность байт:

<1Bh><0Eh><02h>

Относительное позиционирование курсора



Кроме абсолютного позиционирования курсора, его можно перемещать на одну позицию влево (команда <15h>), вправо (команда <09h>), вверх (команда <0Bh>) и вниз (команда <0Ah>). Команда <0Dh> перемещает курсор в начало строки, а команда <08h> стирает символ перед курсором и перемещает курсор на одну позицию влево. Фактически команда <08h> может быть заменена последовательностью <15h><20h><15h>.

При перемещении курсора на одну позицию влево из крайней левой позиции, курсор перемещается в крайнюю правую позицию предыдущей строки. Если курсор находится в крайней левой позиции верхней строки, то он остается на месте.

При перемещении курсора на одну позицию вправо из крайней правой позиции, курсор перемещается в крайнюю левую позицию следующей строки. Если курсор находится в крайней правой позиции нижней строки, то если разрешена автоматическая прокрутка экрана – курсор перемещается в начало строки, а содержимое области вывода смещается на одну позицию вверх, в противном случае – курсор остается на месте.

При перемещении курсора на одну позицию вниз из последней строки, курсор остается на месте. При этом если разрешена автоматическая прокрутка экрана, то содержимое области вывода смещается на одну позицию вверх.

Вид курсора

Вид курсора может быть задан командами 11h, 12h, 13h, 14h или командой <1Eh><10h>.

Команда 1Eh 10h имеет длину 3 байта:

<1Eh><10h><CT>

Где, <CT> – вид курсора.

Параметр CT может принимать следующие значения:

1. Курсор выключен (невидимый курсор);
2. Курсор выглядит, как мигающий прямоугольник;
3. Курсор выглядит, как подчерк;
4. Курсор выглядит, как подчеркнутым мигающий прямоугольник;
5. На VFD дисплее курсор выглядит, как инвертированный блок, на LCD дисплее – аналогично типу 1;
6. На VFD дисплее курсор выглядит, как мигающий инвертированный блок, на LCD дисплее – аналогично типу 1;
7. Курсор выглядит, как мигающий подчерк;
8. На VFD дисплее курсор выглядит, как подчеркнутый мигающий инвертированный блок, на LCD дисплее – аналогично типу 3;

Команды 11h, 12h, 13h и 14h оставлены для совместимости с предыдущими версиями прошивки и являются краткими формами команды <1Eh><10h> [00h..03h].

Область вывода

Начиная с версии 2.10, в M920L появилось понятие “область вывода”. Область вывода представляет собой прямоугольное пространство, заданное левым верхним и правым нижним углом. Все операции с дисплеем, в том числе вывод информации и позиционирование курсора ограничены областью вывода. Все пространство за пределами области вывода остается неизменным. Исключение составляют символы пользователя, которые едины и для области вывода и для остального пространства дисплея. По умолчанию левый верхний и правый нижний углы области вывода совпадают с левым верхним и правым нижним углами дисплея. Размер и координаты области вывода можно изменять с помощью команды <1Eh><12h>.

Команда 1Eh 12h имеет длину 6 байт:

<1Eh><12h><XL><YT><XR><YB>

XL – координата X левого верхнего угла области вывода. Параметр XL должен принадлежать диапазону 0 >= XL >= XR.

YT – координата Y левого верхнего угла области вывода. Параметр YT должен принадлежать диапазону 0 >= YT >= YB.

XR – координата X правого нижнего угла области вывода. Параметр XR должен принадлежать диапазону XL >= XR >= 19.

YB – координата Y правого нижнего угла области вывода. Параметр YB должен принадлежать диапазону YT >= YB >= 3.

Минимальная ширина области вывода равна одной колонке, минимальная высота – одной строке.

Подсветка

Применяемые в пульте оператора M920L LCD дисплеи, имеют светодиодную подсветку. Для управления подсветкой применяется широтно-импульсная модуляция (ШИМ), что позволяет программно изменять ее яркость.

Подсветка включается командой 03h и выключается командой 02h. При этом команда 03h устанавливает яркость подсветки, заданную командой <1Eh><11h>.

Яркость включенной подсветки задается командой <1Eh><11h>. Эта команда имеет длину 3 байта:

*<1Eh><11h>
*

Параметр BR задает яркость и может принимать значения от 0 (подсветка полностью погашена) до 255 (максимальная яркость подсветки).

VFD дисплей не имеет подсветки, однако яркость свечения VFD дисплея так же можно изменять программно. Яркость свечения VFD дисплея имеет только 8 градаций.

Команда 02h для VFD дисплея устанавливает минимальную яркость свечения (12.5%), а команда 03h, как и для LCD дисплея – устанавливает яркость, заданную командой *<1Eh><11h>
*.

Соотношение параметра BR, команды 1Eh 11h и яркости свечения VFD дисплея								
BR	0..31	32..63	64..95	96..127	128..156	160..191	192..223	224..255
Яркость (%)	12.5	25	37.5	50	62.5	75	87.5	100

Символы пользователя

Пульт оператора M920L поддерживает восемь символов пользователя (U_0–U_7), т.е. символов, для которых пользователь может задать произвольный внешний вид. Символы пользователя имеют размер 5x8 или 7x8 пикселей (задается через МЕНЮ НАСТРОЕК). Размер 5x8 рекомендуется при использовании с LCD дисплеем, а 7x8 – с VFD. дисплеем. При этом следует учесть, что VFD дисплей нормально поддерживает размер 5x8, а LCD дисплей, при использовании размера 7x8, будет обрезать по одному пикселю слева и справа.

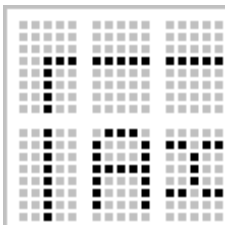
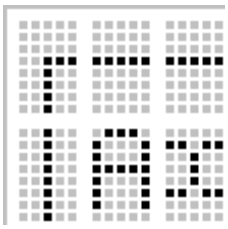
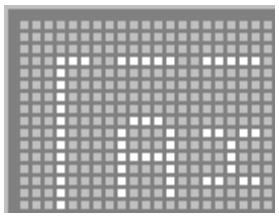
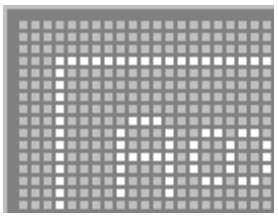
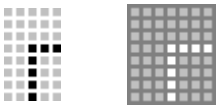
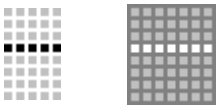
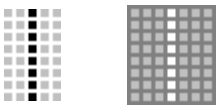
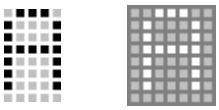
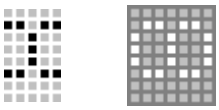
Внешний вид символов пользователя задается с помощью команды 10h.

Команда 10h имеет длину 10 байт:

<10h><SS><XX><XX><XX><XX><XX><XX><XX><XX>

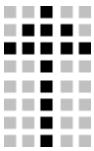
Где, <SS> – номер символа пользователя, <XX> – битовый образ строки.

Номер символа пользователя может принимать значения от 00h до 07h. Первый байт битовых образов соответствует верхней строке символа, последний – нижней. Младший бит соответствует крайнему правому пикселю в строке.

Примеры отображения символов пользователя			
		Размер	
		5x8	7x8
Дисплей	LCD		
	VFD		
Битовые образы используемых символов		00000000b 00000000b 00000000b 00000111b 00000100b 00000100b 00000100b 00000100b	00000000b 00000000b 00000000b 00001111b 00001000b 00001000b 00001000b 00001000b
		00000000b 00000000b 00000000b 00011111b 00000000b 00000000b 00000000b 00000000b	00000000b 00000000b 00000000b 01111111b 00000000b 00000000b 00000000b 00000000b
		00000100b 00000100b 00000100b 00000100b 00000100b 00000100b 00000100b 00000100b	00001000b 00001000b 00001000b 00001000b 00001000b 00001000b 00001000b 00001000b
		00001110b 00010001b 00010001b 00011111b 00010001b 00010001b 00010001b 00000000b	00011100b 00100010b 00100010b 00111110b 00100010b 00100010b 00100010b 00000000b
		00000000b 00011011b 00000100b 00000100b 00000100b 00011011b 00000000b 00000000b	00000000b 00110110b 01001001b 01001001b 01001001b 00110110b 00000000b 00000000b

Пример использования:

Требуется задать для символа U3 на пульте оператора следующий вид:



Шаг 1. Заменяем светлые пиксели нулями, а черные - единицами:

00100
01110
11111
00100
00100
00100
00100
00100

Шаг 2. Добавляем ведущие нули, и переводим полученные 8-битные двоичные значения в шестнадцатеричные, считая левые биты – старшими, а правые – младшими.

00000100 = 04h
00001110 = 0Eh
00011111 = 1Fh
00000100 = 04h
00000100 = 04h
00000100 = 04h
00000100 = 04h
00000100 = 04h

Шаг 3. Подставляем полученные значения в команду:

<10h><03h><04h><0Eh><1Fh><04h><04h><04h><04h><04h>