6 – Протокол связи

6.1 Отказ от обязательств

В этой главе приводится информация, требуемая для управления устройством PMM EP600/EP601/EP602/EP603 через волоконно-оптический кабель, подключенный к ПК, с помощью собственных программных приложений пользователя, установленных на ПК. Компания Narda STS S.r.l только предоставляет достоверную информацию и не несет никаких обязательств за любые последствия, наступившие в результате использования этой информации. Пользователь всецело отвечает за возможные риски и несет ответственность за включение функций протокола связи, разработанного компанией Narda, в пользовательское или стороннее программное обеспечение. Никоим образом компания Narda STS S.r.l не несет ответственности за неисправности любого вида, связанные с использованием информации, представленной в этой главе.

ПРИМЕЧАНИЕ

Все приведенные ниже примеры в одинаковой степени относятся к устройствам PMM EP600, EP601, EP602 и EP603.

6.2 Протокол

Последовательная связь между ПК и PMM EP600/EP601/EP602/EP603 осуществляется в соответствии со стандартом RS232 или USB (через преобразователь USB-RS232). Технические характеристики:

- Скорость: 9600 Бод
- 1 стартовый бит
- 1 стоповый бит
- Отсутствие контроля по четности

Команды записываются в виде ASCII-строки, ограниченной символами **“#”** (0x23) и “*” (0x2A)
В версиях до FW 1.02 включительно каждая команда должна начинаться с адреса, который формируется в виде строки “00”.
Начиная с версии FW 1.10 каждая команда начинается с передачи адреса, который формируется в виде двухсимвольной строки в диапазоне от “00” до “99”.
“00” представляет собой специальный адрес, поскольку он рассматривает как адрес широковещательной передачи, в то время как другие адреса должны соответствовать адресу, сохраненному в блоке (см. команду “**I**”).
Другими словами, EP600 всегда предоставляет ресурсы всем командам, начинающимся с символов “#00”, вне зависимости от собственного сохраненного адреса.
Режим широковещательной передачи предназначен для использования EP600 в отсутствие шины (NON-BUS), обычно в том случае, если ПК непосредственно связан с EP600, или для установки нового адреса (см. команду “**I**”). В этом режиме адрес может изменяться даже в при отсутствии информации о текущем адресе.
Однако необходимо соблюдать осторожность, если EP600 работает с использованием шины (BUS), например через SB10, поскольку при использовании широковещательного адреса одновременный ответ от всех устройств, подключенных к шине, может привести к конфликту.
Далее все примеры приводятся с указанием широковещательного адреса, однако, очевидно, что они также справедливы при использовании другого адреса. Единственное ограничение заключается в том, что адрес должен формироваться в виде двух символов, заключенных в диапазоне от “00” до “99”.

В соответствии с переданной командой ответ может пересылаться в виде ASCII-строки или в двоичном виде. Первый символ всегда совпадает с переданным символом и может использоваться как управляющий маркер или символ синхронизации для ответа.

Существуют команды трех категорий:

- **Команды запроса**
- **Команды установки значений параметров**
- **Команды эксплуатации**

Команды вводятся в следующем формате:
#00Qкоманда(параметры)*, где

= символ начала командной строки
00 = эти символы всегда вводятся в строку команды
Q = **?** – указывает на команды запроса
S – указывает на команды установки значений параметров
команда = командная строка
(параметры) = установленные значения параметров (там где они необходимы)
* = символ завершения командной строки

ПРИМЕЧАНИЕ

При подаче питания EP600/EP601/EP602/EP603 находится в режиме ведущего устройства для взаимодействия с переносным измерителем 8053B; в этом случае EP600/EP601/EP602/EP603 продолжает передавать результаты измерений вне зависимости от приема команд. Такой режим может оказаться неприемлемым для взаимодействия с другим программным обеспечением , передайте команду #00?v* для перевода EP600/EP601/EP602/EP603 в подчиненный режим с целью получения ответа только после приема запроса.
С целью экономии энергопотребления от батареи устройство EP600/EP601/EP602/EP603 автоматически отключается через 180 секунд после приема команды; используйте команду эксплуатации #00e n* (см. таблицу 6-2) для установки времени функционирования EP600, которое должно истечь перед автоматическим отключением устройства.

Таблица 6-1 Команды запроса	
Команда:	Описание
?v	По команде запроса #00?v* возвращается строка, содержащая информацию о модели, версии и дате микропрограммного обеспечения. Пример отклика на команду **#00?v***: **"vEP600:1.02 10/05;"**
?p	По команде запроса **#00?p*** возвращается строка, содержащая информацию о дате калибровки. Пример отклика на команду **#00?p***: **"10/05;"**
?b Батарея	По команде запроса **#00?b*** возвращаются 3 байта, содержащие информацию о напряжении батареи EP600. Массив состоит из 3 байтов, в первом из которых закодирован символ 'b', а два последующих за ним байта содержат 16-битовое целое число без знака (**nn**) в формате с обратным порядком байтов. Для получения напряжения батареи используйте следующую формулу: V_battery= **3 * (nn / 1024 * 1.6)**
?t Температура	По команде запроса **#00?t*** возвращаются 3 байта, содержащие информацию о температуре датчика EP600. Массив состоит из 3 байтов, в первом из которых закодирован символ 't', а два последующих за ним байта содержат 16-битовое целое число без знака (**nn**) в формате с обратным порядком байтов. Для получения температуры в градусах используйте следующую формулу: T_ep600= **((nn / 1024 * 1.6) – 0.986) * 1000 / 3.55**
?s Серийный номер	По команде запроса **#00?s*** возвращается строка, содержащая серийный номер устройства. Пример отклика на команду **#00?s***: **"s123456789AAAA"**
?T Суммарное значение напряженности поля	По команде запроса **#00?T*** возвращаются 5 байтов, содержащие информацию о **суммарном** значении напряженности поля, измеренном датчиком EP600. Массив состоит из 5 байтов, в первом из которых закодирован символ 'T', а четыре последующих за ним байта содержат 32-битовое число с плавающей точкой в формате IEEE без знака (**ff**) и с прямым порядком байтов. Число **(ff)** представляет квадрат суммарной напряженности поля (изотропное измерение). Для получения напряженности поля необходимо извлечь квадратный корень из этого значения: **V/m=√ff**
?A Все компоненты напряженности поля	По команде запроса **#00?A*** возвращаются 13 байтов, содержащие информацию о значении напряженности поля, измеренном датчиком EP600 по каждой **отдельной оси**. Массив состоит из 13 байтов, в первом из которых закодирован символ 'A', а двенадцать последующих за ним байтов содержат значение 3 координат (X,Y,Z) в виде 32-битового числа с плавающей точкой в формате IEEE (**ff**) и с прямым порядком байтов. 3 числа **(fx,fy,fz)** непосредственно представляют напряженность поля по соответствующей оси и выражены в В/м.

Таблица 6-2 Команды установки значений параметров	
Команда:	Описание
k fr	По команде **#00k frq*** устанавливается частота (**frq**), с которой связан поправочный коэффициент. Число **fr** представляет собой ASCII-строку, задающую **целое** значение частоты, умноженное на 100, в результате чего устанавливается разрешение, равное 10 кГц. После приема и подтверждения этой команды датчиком EP600 все измерения корректируются с использованием коэффициента, сохраненного в заводских условиях, относительно этой частоты. Передача частоты, выходящей за пределы диапазона EP600, приводит к отмене выполнения функции применения коэффициента частотной коррекции. Отклик представляет собой массив из 5 байтов, в первом из которых закодирован символ 'k', а четыре последующих за ним байта содержат 32-битовое число с плавающей точкой в формате IEEE (**ff**) и с прямым порядком байтов. Число **(ff)** представляет частоту, используемую EP600. Пример команды: **#00k 10000***: (Установка внутренней частоты 100 МГц.)
f n	По команде **#00f n*** устанавливается фильтр обработки (**n**), используемый для измерений. Индекс **n** должен находиться в диапазоне от 0 до 7. Дополнительная информация о фильтрах содержится в отдельном документе. Пример команды:**#00f 2***
e n	По команде **#00e n*** устанавливается время, по истечении которого EP600 автоматически отключается после приема распознаваемой команды. Следует отметить, что это установленное значение **не является постоянным** и сохраняется только при включенном датчике EP600. Каждый раз после отключения EP600 по умолчанию устанавливается интервал в 180 секунд. Аргумент **n** задается в секундах и не должен превышать 10800 (3 часа). В ответ на эту команду возвращается символ 'e', если команда воспринята, или символ 'x', если аргумент выходит за пределы допустимого диапазона (по умолчанию принимается значение 180). Пример команды: **#00e 600*** (установка 10-минутного периода до момента отключения).

Таблица 6-3 Команды эксплуатации	
Команда:	Описание
@c	По команде **#00@c*** датчик EP600 временно переводится в режим сохранения (“Storing Mode”), в котором допускается сохранять новый адрес. Поскольку это разрешение действительно только в течение 1 секунды, непосредственно за этой командой необходимо передать команду “I”. Отклик на эту команду не возвращается. Пример команды: **#00@c***
@Iaddr	По команде **#00@Iaddr*** устанавливается адрес, используемый для протокола связи. Адрес формируется с помощью двухсимвольной строки, содержащей целое число в диапазоне от “00” до “99”. Пробелы и знаки пунктуации между “#@00” и addr вводить не допускается. Эта команда установки выполняется только в том случае, если она передается в течение 1 секунды с момента выполнения команды “c”. В ответ на эту команду возвращается непосредственно адрес, если эта команда была воспринята, в противном случае возвращается строка “ERR”, если EP600 не находится в режиме сохранения (Storing Mode). Пример команды: **#00@0I53***. По этой команде устанавливается адрес "53". Таким образом, в этом случае будут выполняться все команды, начинающиеся с адреса **#53……*,** также как и с адреса **#00….***.

Эта страница преднамеренно оставлена пустой.

7 – Справочник по функциям, содержащимся в DLL

ПРИМЕЧАНИЕ

Все приведенные ниже примеры в одинаковой степени относятся к устройствам PMM EP600, EP601, EP602 и EP603.

ПРИМЕЧАНИЕ

Функции библиотеки DLL можно использовать только для работы с широковещательным адресом (“00”).

7.1 Язык C

7.1.1 PMM_CreateProbe()

int PMM_CreateProbe(const char *name, HANDLE *probeHandle, const char *commPort);

Назначение:
Установление связи с указанным датчиком.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
const char *name: название модели PMM EP60X, например: EP601
const char *commPort: имя последовательного коммуникационного порта, например: COM1, COM3…COM99

Выходные параметры:
HANDLE * Handle

ПРИМЕЧАНИЕ

Используйте директиву #include<windows.h> для определения данных типа HANDLE.
Это специальное значение, которое после его создания используется для ссылки на данный датчик с целью последующих вызовов функций.

7.1.2 PMM_RemoveProbe()

int PMM_RemoveProbe(const HANDLE probeHandle);

Назначение:
Закрытие коммуникационного порта и освобождение памяти для последующего использования системой.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle; значение, возвращаемое функцией *CreateProbe*

Выходные параметры:
Отсутствуют

7.1.3 PMM_Firmware()

int PMM_Firmware(const HANDLE probeHandle, char *firmware, int *arraySize);

Назначение:
Получение версии микропрограммного обеспечения датчика.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle; значение, возвращаемое функцией *CreateProbe*

Выходные параметры:
Передаваемая по ссылке символьная строка: указывается длина строки

7.1.4 PMM_ProbeName()

int PMM_ProbeName(const HANDLE probeHandle, char *name, int *arraySize);

Назначение:
Возврат идентификационной информации об имени датчика.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle; значение, возвращаемое функцией *CreateProbe*

Выходные параметры:
Имя размещается в выделенном пользователем буфере для строки.

7.1.5 PMM_Model()

int PMM_Model(const HANDLE probeHandle, char *model, int *arraySize);

Назначение:
Возврат идентификационной информации о модели датчика.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle; значение, возвращаемое функцией *CreateProbe*

Выходные параметры:
Информация о модели размещается в выделенном пользователем буфере для строки.

7.1.6 PMM_CalibrationDate()

int PMM_CalibrationDate(const HANDLE probeHandle, char *calibrationDate, int *arraySize);

Назначение:
Возврат даты последней калибровки датчика. Эта функция недоступна в более ранних моделях датчиков.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle; значение, возвращаемое функцией *CreateProbe*

Выходные параметры:
Передаваемая по ссылке символьная строка: calibrationDate: дата калибровки датчика.
arraySize: длина строки.

7.1.7 PMM_ReadBattery()

int PMM_ReadBattery(HANDLE probeHandle, float *battery);

Назначение:
Эта функция считывает состояние батареи датчика.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle

Выходные параметры:
Передаваемая по ссылке символьная строка: состояние батареи в виде напряжения в вольтах.

7.1.8 PMM_ReadTemperature()

int PMM_ReadTemperature(HANDLE probeHandle, float *temperature);

Назначение:
Эта функция считывает внутреннюю температуру датчика.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle

Выходные параметры:
Передаваемая по ссылке температура в виде числа с плавающей точкой: числовое значение внутренней температуры датчика в градусах по шкале Цельсия.

7.1.9 PMM_SerialNumber()

int PMM_SerialNumber(const HANDLE probeHandle, char *serialNumber, int *arraySize);

Назначение:
Возврат серийного номера датчика.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle

Выходные параметры:
Передаваемая по ссылке символьная строка serialNumber: серийный номер датчика. arraySize: длина строки.

7.1.10 PMM_SetFrequency()

int PMM_SetFrequency(const HANDLE probeHandle, int Frequency);

Назначение:
Установка частоты, к которой относится поправочный коэффициент. После приема и подтверждения этой команды датчиком EP600 все измерения корректируются с использованием коэффициента, сохраненного в заводских условиях, относительно этой частоты.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle, int Frequency: целое значение частоты, умноженное на 100, в результате чего устанавливается разрешение, равное 10 кГц.
Передача частоты, выходящей за пределы диапазона EP600, приводит к отмене выполнения функции применения коэффициента частотной коррекции.

Выходные параметры:
Отсутствуют

7.1.11 PMM_SetFilter()

int PMM_SetFilter(const HANDLE probeHandle, int FILTER);

Назначение:
Установка фильтра обработки (n), используемого для измерений.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle, int range. Допустимые значения: 0 – 7

Выходные параметры:
Отсутствуют

7.1.12 PMM_SetTimeout()

int PMM_SetTimeout(int tout);

Назначение:
Установка тайм-аута в процессе связи с датчиками серий PMM EP600.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
int tout, целое значение в миллисекундах. Значение по умолчанию – 500 мс.

Выходные параметры:
Отсутствуют

7.1.13 PMM_SetAutoOffTime()

int PMM_SetAutoOffTime(const HANDLE probeHandle, int Time);

Назначение:
Установка времени, по истечении которого EP600 автоматически отключается после приема распознаваемой команды.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle, int time – время в секундах. Допустимые значения: 180 (3 минуты) – 10800 (3 часа)

Выходные параметры:
Отсутствуют

7.1.14 PMM_ReadTotalField()

int PMM_ReadTotalField (const HANDLE probeHandle, float &XYZField);

Назначение:
Возвращение суммарного значения напряженности поля по осям X, Y и Z.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle; значение, возвращаемое функцией *CreateProbe*

Выходные параметры:
Передаваемое по ссылке число с плавающей точкой. Суммарное значение напряженности поля по осям X, Y и Z.

7.1.15 PMM_ReadAxisField

PMM_ReadAxisField (const HANDLE probeHandle, float *xField, float *yField, float *zField);

Назначение:
Считывание значения напряженности поля по осям X, Y и Z.

Возвращаемое значение:
Возвращается код состояния в виде целого числа. Числовое значение 0 указывает на отсутствие ошибок. Описание кодов ошибок содержится в главе "Код состояния".

Входные параметры:
HANDLE probeHandle

Выходные параметры:
Передаваемое по ссылке число с плавающей точкой. Возвращаются значение напряженности поля по осям X, Y и Z.

7.2 Язык Visual Basic

В меню Project (проект) выберите пункт References (ссылки) для открытия диалогового окна References, затем нажмите кнопку Browse (просмотр) для поиска новой библиотеки типов (PMM_EP60X.tlb). После выбора этой библиотеки нажмите кнопку OK. После того как библиотека указана в первый раз, эта библиотек автоматически регистрируется в среде Visual Basic. Убедитесь, что требуемая библиотека ("PMM_EP60X") отмечена в списке References, затем закройте диалоговое окно.

7.3 Код состояния

Таблица 7-1 Код состояния

0	OK
1	Некорректный дескриптор (Handle)
2	Невозможно открыть порт
3	Устройство не подключено
4	Некорректный отклик
5	Отсутствие отклика
6	Недействительный параметр
7	Последовательный порт занят
8	Тайм-аут
9	Ошибка в последовательном порте
10	Проблема при записи в последовательный порт
11	Ошибка при чтении данных из последовательного порта
12	Некорректная строка подключения
13	Значение не может быть установлено
14	Датчик не поддерживается
15	Датчик выходит за пределы диапазона
16	Датчик вне пределов диапазона
17	Ошибка при закрытии последовательного порта
18	Ошибка при очистке последовательного порта

ПРИМЕЧАНИЕ

Программа WinEP600 Setup.exe автоматически записывает файлы PMM_EP60X.DLL и PMM_EP60X.TLB в системную папку C:\Windows\System32\.