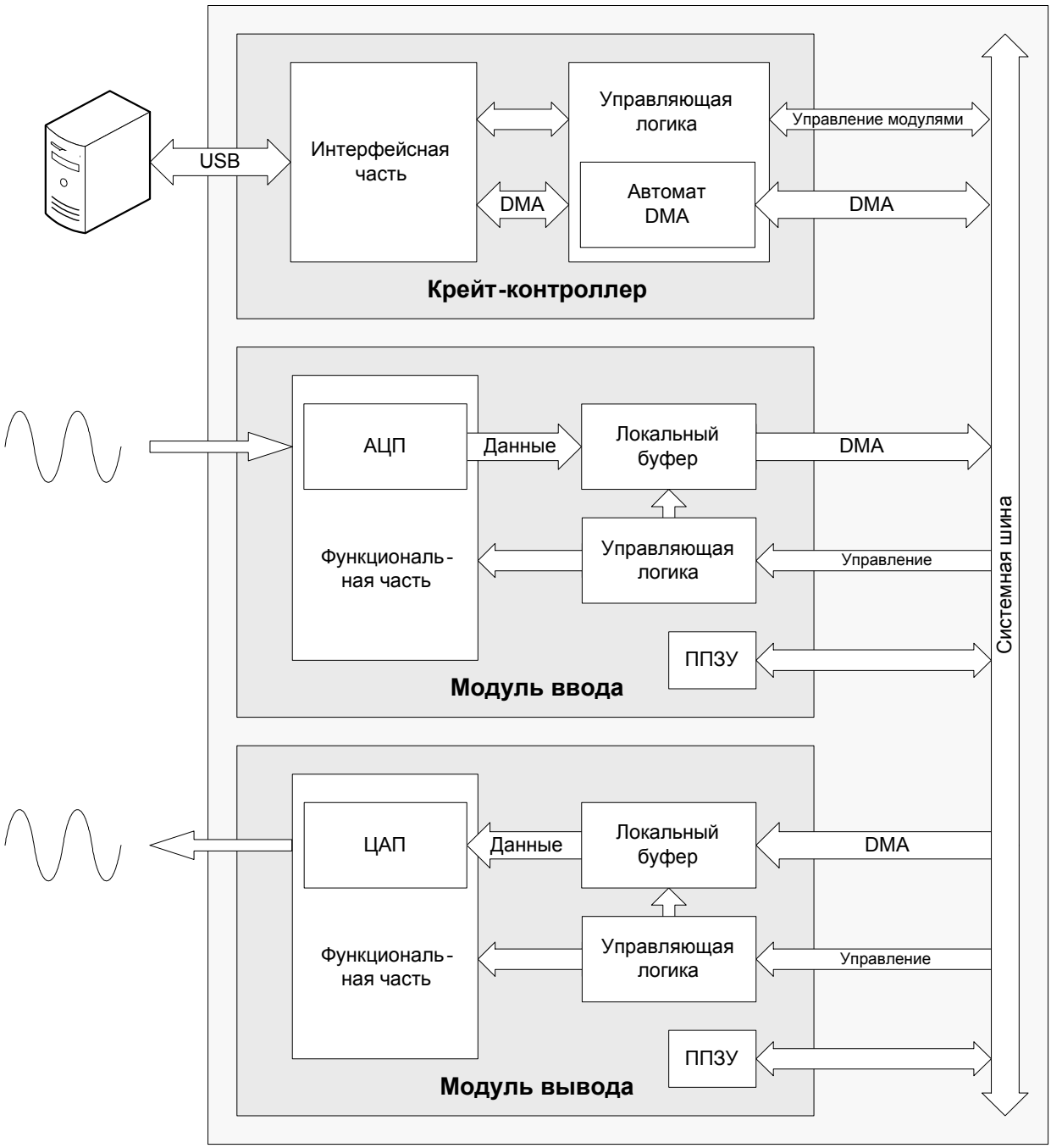
Контакты:
http://www.R-Technology.ru
Info@R-Technology.ru - Общие вопросы
Sales@R-Technology.ru - Отдел продаж
Support@R-Technology.ru - Техническая поддержка

1. Структура устройства QMBox.

Аппаратура устройства QMBox представляет собой систему, состоящую из контроллера, который подключается к USB порту компьютера, набора функциональных модулей (от 1 до 8), подключаемых к объекту, и внутренней информационной шины связывающей контроллер и каждый из модулей.

На рисунке приведена структурная схема системы QMBox, в состав которой входят модуль ввода (АЦП) и модуль вывода (ЦАП).



Устройство QMBox взаимодействует с компьютером пользователя посредством интерфейса USB.

Для обеспечения непрерывного потока данных каждый функциональный модуль имеет внутренний локальный буфер, где данные буферизируются перед обменом с компьютером.

На каждом модуле помимо его функциональной части и управляющей логики имеется ППЗУ (Перепрограммируемое Запоминающее Устройство). ППЗУ содержит служебную информацию, в частности, калибровочные коэффициенты – это даёт возможность заменять модули, на аналогичные, без изменения конфигурационной информации ПО. Также ППЗУ содержит информацию о типе и серийном номере модуля, что позволяет ПО автоматически определять текущую аппаратную конфигурацию системы.

Системное ПО QMBox предназначено для устранения необходимости низкоуровневого программирования и состоит из:

1. драйвера ОС (инсталлируется при первом подключении QMBox к компьютеру);
2. библиотеки функций управления системой (используется в программах пользователя, на этапах компиляции и линкования);
3. программы сервера данных (должна быть запущена при работе с QMBox);

2. Порядок работы с устройством.

Устройство QMBox (крейт-контроллер + набор установленных в крейт модулей) подключается к USB-порту. С программной точки зрения такая система представляет собой некое устройство, имеющее свой идентификатор (далее “дескриптор системы”). Так как к одному компьютеру может быть подключено несколько устройств QMBox, большинство библиотечных вызовов принимает в качестве входного параметра дескриптор системы, с которой будет работать вызываемая функция.

В состав каждого устройства может входить до 8 различных модулей, установленных в физические слоты устройства в различном порядке. Для идентификации модуля библиотечные вызовы принимают в качестве входного параметра номер ЛОГИЧЕСКОГО слота, ассоциированного с требуемым модулем. Номера физических и логических слотов лежат в диапазоне [0..7].

В случае, если количество установленных модулей меньше количества физических слотов устройства, номер логического слота не будет равен номеру физического слота: При инициализации устройства производится последовательная (начиная с ближайшего к крейт-контроллеру) проверка физических слотов на наличие установленных модулей. Если в очередном физическом слоте обнаружен модуль, то с ним ассоциируется текущее значение счетчика логического слота (перед началом проверки счетчик равен нулю), который после этого увеличивается на единицу, т.е. все пустые слоты устройства игнорируются.

Пример1: Устройство состоит из трех модулей: A, B, C

Модуль	Физический слот	Логический слот
A	1	0
B	3	1
C	7	2

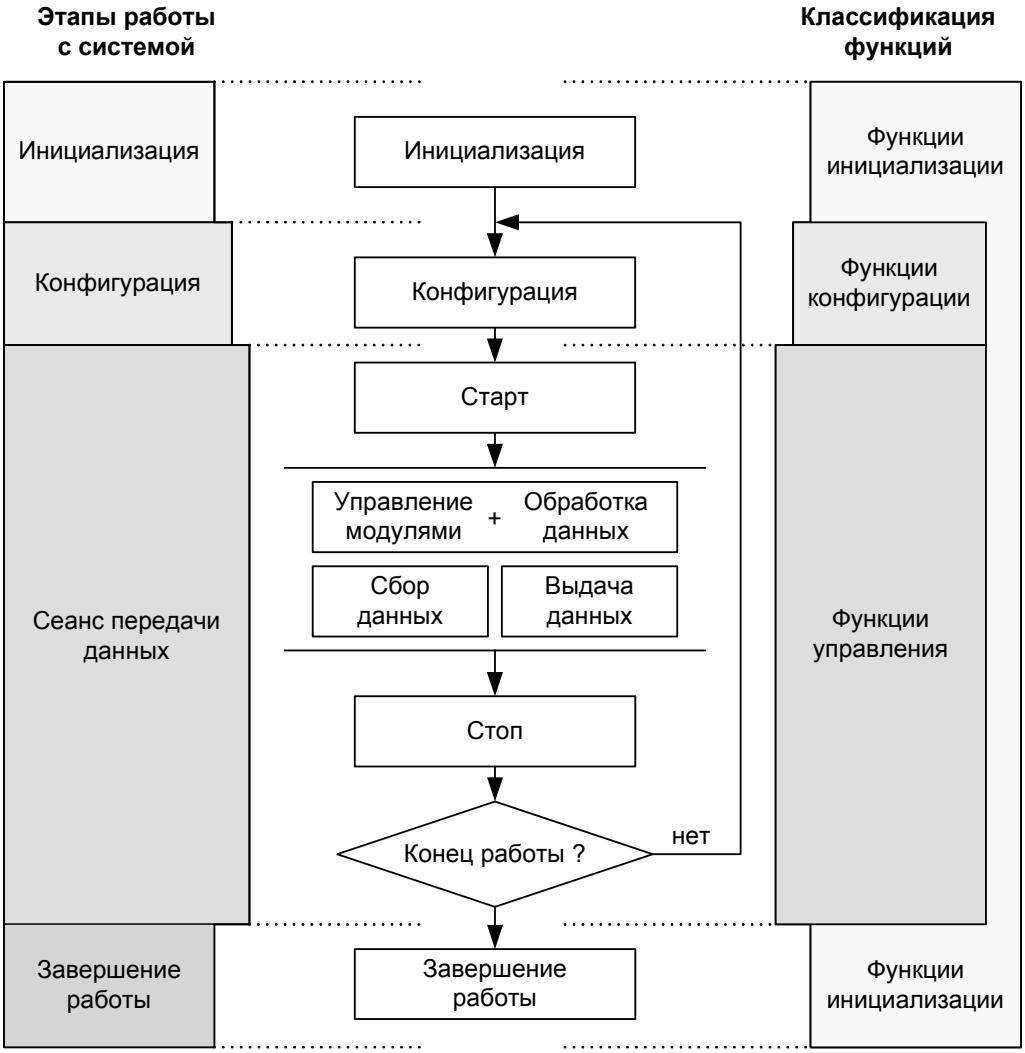
Пример2: Устройство состоит из пяти модулей: A, B, C, D, E

Модуль	Физический слот	Логический слот
A	0	0
B	1	1
C	3	2
D	5	3
E	7	4

Работа с системой состоит из следующих этапов:

- Инициализация.
- Конфигурация.
- Сеанс передачи данных.
- Завершение работы.

Этапы ДОЛЖНЫ выполняться в следующем порядке:



Рассмотрим каждый этап по отдельности:

2.1. Инициализация.

Перед началом работы с аппаратурой приложение должно запустить программу сервера данных (qmbox_ds.exe). Естественно, программа-сервер может быть запущена и «вручную», и при помощи командного файла.

Перед непосредственной работой с системой QMBox программа должна ее обнаружить и получить ее дескриптор. Для этого используется функция базовой библиотеки QMX_CircBufAttach. Если эта функция успешно выполняется, то работу с подключенной системой можно продолжать.

Далее необходимо проинициализировать систему с помощью функции QMX_InitModules.

В случае, когда прикладное ПО не ориентировано на жесткую конфигурацию устройства (т.е. «не знает» заранее, из каких модулей устройство состоит), можно узнать все параметры системы (структура QMX_CONFIG), вызвав фиктивно функцию QMX_Prepare .

2.2. Конфигурация.

Теперь нужно произвести настройку параметров модулей, и произвести подготовку к сеансу передачи данных, во время которого будет происходить обмен данными между модулями и компьютером. Пользователь выбирает активные модули (т.е. модули, которые будут участвовать в предстоящем сеансе передачи данных), скорость каждого модуля, диапазоны измерений и т.п., а также задает требуемое время реакции (время, за которое компьютер может получить доступ к обновленным данным любого модуля).

Например: в системе имеется 4 модуля ввода (АЦП1, АЦП2, АЦП3, АЦП4). Допустим, пользователю необходимо получать в память компьютера данные с АЦП1 (со скоростью 3МГц) и АЦП3 (со скоростью 1кГц). Таким образом, в предстоящем сеансе передачи данных активными будут модули АЦП1, АЦП3.

Во время сеанса передачи данных компьютер получает данные с модулей ввода строго по половинам их локальных буферов. Пока половина локального буфера на модуле не заполнится данными, крейт-контроллер не сможет передать эти данные в компьютер. Таким образом, время реакции системы главным образом определяется скоростью сбора данных самого медленного модуля (в нашем примере это модуль АЦП3) и размером его локального буфера. Если для пользователя критично время реакции системы, он должен задать его величину в явном виде при вызове функции QMX_Prepare(). Эта функция вызывается непосредственно перед началом сеанса передачи данных, и рассчитывает размеры локальных буферов модулей в соответствии с требуемым временем реакции.

Подготовка к сеансу передачи данных происходит в следующем порядке:

1. С помощью функции QMX_MOD_SetActive выбираются активные модули. С помощью функций QMX_XXX_ задается скорость работы (скорость передачи данных) каждого активного модуля, а также все остальные требуемые параметры, например: усиление аналогового тракта модуля, порядок переключения каналов АЦП на модуле и т.д.
2. Вызывается функция QMX_Prepare. Эта функция рассчитывает параметры сбора данных (в т.ч. и размеры локальных буферов модулей[1]) таким образом, чтобы во время работы системы, с одной стороны обеспечивалось отсутствие потерь данных (т.е. исключалась возможность переполнения данными локальных буферов модулей), а с другой стороны обеспечивалось требуемое время реакции системы (если оно задано пользователем). В процессе выполнения функции QMX_Prepare рассчитанные параметры сеанса передачи данных записываются в структуру QMX_CONFIG, указатель на которую передается в функцию.
3. Если необходимо изменить режим запуска модулей, то вызывается функция QMX_MOD_SetStartMode.

[1] На модуле установлена память фиксированного размера. Функция лишь рассчитывает размер части этой памяти, которая будет использоваться.

2.2.1. Режимы синхронного запуска.

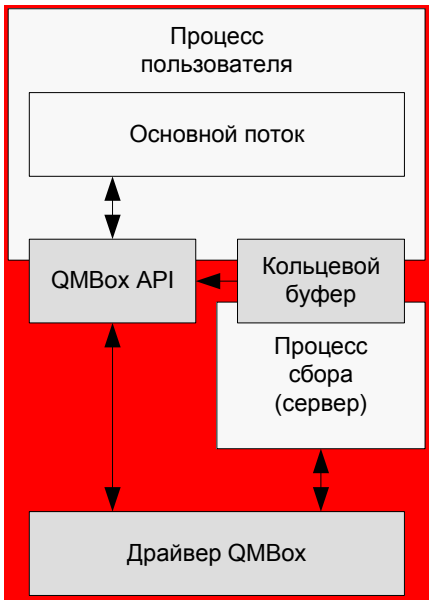
В системе поддерживается следующие режимы синхронного запуска:

- Глобальный “ручной” запуск. Сеанс передачи данных и все модули запускаются сразу после вызова функции QMX_Start. В этом режиме система работает по умолчанию.
- Глобальный синхронный запуск. Сеанс передачи данных и все модули запускаются по внешнему синхросигналу, подаваемому на контакт SYNC внешнего разъема любого из модулей.
- Смешанный запуск. Сеанс передачи данных запускается сразу после вызова функции QMX_Start. По желанию пользователя (программиста) каждый из модулей может быть индивидуально настроен на запуск совместно с процессом передачи данных, или на запуск по внешнему синхросигналу, подаваемому на контакт SYNC соответствующего модуля.

Дополнительную информацию см. в описании функции QMX_MOD_SetStartMode.

2.3. Сеанс передачи данных.

Все действия низкоуровневого управления передачей данных между драйвером и пользовательской областью памяти выполняются в отдельном процессе – сервере, который должен быть запущен до начала работы с системой. Поток данных направляется в кольцевой буфер регулируемого размера. Указатель на начало буфера доступен пользовательскому процессу на этапе инициализации. Также пользовательское приложение в любой момент времени может считать указатель на последний записанный в буфер элемент данных. Формат поступающих в буфер данных описан в гл. 5.



Во время сеанса передачи данных программа имеет асинхронный доступ к модулям (посредством функций набора QMX_XXX_) из основного потока программы. Например, можно считывать состояние цифровых линий модуля, менять коэффициент усиления АЦП и т.д. Все подобные действия можно выполнять, не прерывая сеанса передачи данных. Но необходимо помнить, что при этом обращение к модулям со стороны компьютера может происходить с некоторой задержкой.

Сеанс передачи данных может длиться сколь угодно долго. Тем не менее, если необходимо увеличить количество активных модулей или изменить скорость работы какого-либо модуля, то НЕОБХОДИМО прекратить текущий сеанс передачи данных и снова перейти на этап “Конфигурация”. На этапе “Конфигурация” все параметры передачи данных будут вновь пересчитаны в соответствии с требуемыми изменениями, и сеанс передачи данных можно возобновить.

Останов сеанса передачи данных осуществляется функцией QMX_Stop. Эта функция останавливает крейт-контроллер (прекращает передачу данных между компьютером и устройством) и останавливает активные модули.

2.4. Завершение работы.

Для завершения работы с системой QMBox необходимо освободить ресурсы USB с помощью функции QMX_CircBufDetach и остановить сервер кольцевого буфера qmbox_ds. Перед завершением работы сеанс передачи данных должен быть завершен.

2.5. Работа с модулями вывода.

В состав системы QMBox может входить один или несколько модулей вывода (например, ЦАП). Так же, как и модули ввода, модули вывода имеют свои локальные буферы, где перед выводом буферизируются полученные из компьютера пользователя данные. Локальный буфер каждого модуля вывода имеет жестко заданный размер (как правило, 256 КСлов), который зависит от типа модуля.

Каждый модуль вывода может работать в *динамическом* или *статическом* режиме:

В статическом режиме данные в локальном буфере модуля не обновляются. Т.е. модуль ввода во время сеанса передачи данных выводит данные из своего локального буфера по циклу. Этот режим подходит, например, для вывода простых периодических сигналов (при этом, естественно, в локальный буфер должно помещаться целое количество периодов сигнала).

В динамическом режиме в процессе сеанса передачи данные в локальном буфере модуля обновляются, т.е. происходит постоянная «подкачка» новых порций данных из компьютера пользователя. Такой режим позволяет выводить сигналы произвольной формы, а также любые периодические сигналы с квазинулевым шагом фазы и частоты.

Внимание! Наличие в системе динамического модуля вывода накладывает некоторые ограничения на работу системы:

1. Хотя система QMBox может содержать несколько модулей ввода, только один из них может работать в динамическом режиме.
2. Если в системе имеются модули ввода, то частота работы модуля вывода должна быть меньше суммарной частоты работы модулей ввода. Например, в системе есть модуль ЦАП и 2 модуля АЦП, работающие с частотой 2МГц, и 1МГц соответственно. Значит, ЦАП может работать только на частоте ниже 3 МГц.

2.5.1. Порядок инициализации модулей вывода

На этапе конфигурации перед сеансом сбора данных необходимо предварительно заполнить локальные буферы всех модулей вывода (и статических, и динамических) необходимыми данными. Для этого библиотечные функции вызываются в следующем порядке:

3. С помощью функции QMX_MOD_SetActive первый модуль вывода объявляется активным.
4. Вызывается функция QMX_Prepare.
5. Вызывается функция QMX_MOD_OutputPrestage, которая заполняет весь локальный буфер первого модуля вывода подготовленными заранее данными.
6. С помощью функции QMX_MOD_SetActive первый модуль вывода объявляется НЕактивным.

Если в системе несколько модулей вывода, то указанная последовательность должна быть выполнена для всех модулей вывода. При этом динамический модуль вывода инициализируется **последним.** Точнее говоря, тот модуль, который инициализировался последним, и будет восприниматься системой как динамический.

После инициализации непосредственно перед стартом сеанса передачи данных все модули вывода должны быть объявлены активными с помощью функции QMX_MOD_SetActive

2.5.2. Передача данных в динамический модуль вывода.

В процессе сеанса передачи данных данные в локальном буфере динамического модуля вывода постоянно обновляются новыми данными, приходящими из компьютера пользователя. Алгоритм процесса подкачки должен выглядеть следующим образом:

Приложение следит за специальным счетчиком *CBOutput.buf_ptr* (см. описание структуры QMX_CONFIG), отображающим процесс освобождения локального буфера модуля. Как только счетчик увеличивается, приложение отправляет очередную порцию подготовленных данных с помощью функции QMX_CircBufOutputUpdate.

Буфер, в котором должны готовиться данные, адресуется указателем *CBOutput.buf_start* и имеет размер *CBOutput.buf_size*, соответствующий размеру локального буфера модуля вывода. Данные отправляются в модуль по половинам локального буфера (т.е. счетчик *CBOutput.buf_ptr* инкрементируется, как только освобождается половина локального буфера модуля), поэтому данные должны подготавливаться и отправляться порциями по (*CBOutput.buf_size* / **2**) отсчетов.

Если в процессе сеанса передачи данных не осуществлять подкачку данных, то модуль вывода будет работать в статическом режиме, т.е. выводить по циклу данные, содержащиеся в его локальном буфере.

3. Библиотека функций (QMBox API).

Библиотека функций включает в себя несколько наборов функций:

1. Функции, предназначенные для работы с системой "в целом". К ним относятся функции инициализации системы QMBox, функции управления передачей данных между системой и компьютером, различные служебные функции. Эти функции являются общими для любой конфигурации системы QMBox (т.е. для любого набора модулей).
2. Функции для работы с любыми типами модулей.
3. Специализированные функции, предназначенные для работы с конкретными типами модулей.

Для упрощения идентификации библиотечных функций их названия начинаются одним из префиксов:

QMX_	Функция, предназначенная для работы с системой
QMX_MOD_	Функция, предназначенная для работы с модулями любого типа
QMX_XXX_	Функция, предназначенная для работы с модулем типа "XXX"

В следующей главе этого документа описаны базовые функции библиотеки, т.е. функции серий QMX_ и QMX_MOD_. Функции для работы с конкретными типами модулей (функции QMX_XXX_) описаны в соответствующих документах: «QMS10 Programming Guide», «QMS20 Programming Guide» и т.д.

Применение библиотечных функций и порядок их вызова подробно проиллюстрированы в примерах программирования, находящихся в папке \Examples, а также доступных на сайте http://www.r-technology.ru/products/automation/qmsoft.php

4. Описание базовых функций и структур.

На разных этапах выполнения программы допустимы вызовы не всех библиотечных функций. По назначению библиотечные функции распределяются следующим образом (в описании каждой функции ее назначение указывается отдельно):

- Функции инициализации: Инициализация библиотек, определение конфигурации устройства и т.п. Эти функции можно вызывать только на этапе инициализации.
- Функции конфигурации: Настройка скоростей сбора, выбор активных модулей и т.п. Эти функции можно вызывать только на этапе конфигурации.
- Функции управления: Старт/стоп передачи данных, изменение диапазонов измерений и т.п. Эти функции можно вызывать как на этапе конфигурации, так и в сеансе передачи данных.

Некоторые функции можно вызывать на разных этапах, в этом случае указываются все этапы, на которых вызов корректен.

4.1. Функции

4.1.1.QMX_CircBufAttach()

HANDLE QMX_CircBufAttach (
WORD *VirtualSlot* // номер виртуального слота USB
)

Инициализация	X	Конфигурация		Управление	

Назначение:

Обнаружение системы, получение дескриптора системы, получение доступа к кольцевому буферу, в который сервер кольцевого буфера направляет поток данных.

Параметры:

VirtualSlot

Указывает номер виртуального слота USB, к которому подключена система и для которого запущен сервер кольцевого буфера.

Возвращаемые значения:

В случае успешного завершения функция возвращает дескриптор системы
В случае ошибки функция возвращает NULL.

Примечания:

Перед вызовом функции обязательно должен быть запущен сервер кольцевого буфера.

4.1.2. QMX_CircBufDetach()

int QMX_CircBufDetach (
HANDLE *SD*, // дескриптор системы
)

Инициализация	X	Конфигурация	X	Управление	

Назначение:

Освобождает ресурсы USB перед выходом из программы

Параметры:

SD

Дескриптор системы.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки возвращает QMX_ERR.

Примечания:

Функция вызывается на этапе завершения работы программы непосредственно перед закрытием сервера кольцевого буфера.

4.1.3. QMX_InitModules()

int QMX_InitModules (
HANDLE *SD*, // дескриптор системы
WORD *LogLevel*, // служебная переменная
BYTE **FaulSlot,* // служебная переменная
)

Инициализация	X	Конфигурация		Управление	

Назначение:

Инициализация (загрузка) модулей системы.

Параметры:

SD

Дескриптор системы.

LogLevel

Служебная переменная, всегда должна быть равна QMX_LOG_NONE.

FaulSlot

Служебная переменная, всегда должна быть равна NULL.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки возвращает QMX_ERR.

Примечания:

Для корректного выполнения этой функции необходимо, чтобы файлы прошивок модулей (файлы с расширением .ttf из папки SDK\TTF\) находились в рабочем каталоге.

4.1.4. QMX_Prepare()

int QMX_Prepare(
HANDLE *SD*, // дескриптор системы
WORD *ResponseTime,* // требуемое время реакции в миллисекундах
QMX_CONFIG **Config* // указатель на конфигурацию системы
)

Инициализация		Конфигурация	X	Управление	

Назначение:
Конфигурирует систему в соответствии с заданными параметрами.

Параметры:
SD
Дескриптор системы.
ResponseTime
Требуемое время реакции в миллисекундах. Если время реакции для пользователя не критично, *ResponseTime* должен быть равен 0.
Config
По этому указателю возвращается полное описание системы, включая параметры предстоящего сеанса передачи данных.

Возвращаемые значения:
В случае успешного завершения функция возвращает QMX_OK
В случае ошибки функция возвращает QMX_ERR.

Примечания:
До вызова функции хотя бы один модуль системы должен быть объявлен активным с помощью функции QMX_MOD_SetActive.

Указанное пользователем время реакции не всегда достижимо, поэтому его расчетное значение будет записано функцией в поле *RespTime* структуры QMX_CONFIG, указатель на которую передается в функцию через параметр *Config.* Так как эта функция рассчитывает внутренние параметры системы на основе данных, задаваемых другими конфигурационными функциями, рекомендуется вызывать функцию непосредственно перед вызовом QMX_Start.

В случае, когда прикладное ПО не ориентировано на жесткую конфигурацию устройства (т.е. «не знает» заранее, из каких модулей устройство состоит), можно узнать все параметры системы вызвав фиктивно функцию QMX_Prepare (которая заполняет структуру QMX_CONFIG) еще на этапе инициализации.

4.1.5. QMX_Start()

int QMX_Start(
HANDLE *SD* // дескриптор системы
)

Инициализация		Конфигурация		Управление	X

Назначение:

Запуск сеанса передачи данных.

Параметры:

SD

Дескриптор системы.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки функция возвращает QMX_ERR.

Примечания:

После выполнения этой функции все активные модули системы одновременно начинают работу с заданными пользователем параметрами. Крейт-контроллер системы начинает передачу данных между локальными буферами активных модулей и компьютером. Сеанс передачи данных будет продолжаться до вызова функции QMX_Stop. Режим внешней синхронизации запуска задается на этапе конфигурация функцией QMX_MOD_SetStartMode.

4.1.6. QMX_Stop()

int QMX_Stop(
  HANDLE *SD*, // дескриптор системы
  WORD *Mode* // способ останова
)

Инициализация		Конфигурация		Управление	X

Назначение:

Останов сеанса передачи данных и активных модулей.

Параметры:

SD

Дескриптор системы.

Mode

Способ останова, должен иметь значение QMX_STOP_ALL.

Остальные значения в текущей версии библиотеки зарезервированы.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.

В случае ошибки функция возвращает QMX_ERR.

4.1.7. QMX_CircBufOutputUpdate ()

int QMX_CircBufOutputUpdate(
HANDLE *SD* // дескриптор системы
)

Инициализация		Конфигурация		Управление	X

Назначение:

В процессе сеанса передачи данных отправляет в динамический модуль вывода очередную порцию подготовленных данных для вывода.

Параметры:

SD

Дескриптор системы.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки функция возвращает QMX_ERR.

Примечания:

4.1.8. QMX_GetLastError()

int QMX_GetLastError(
 HANDLE *SD*, // дескриптор системы
 char **str*, // указатель на буфер
 DWORD *len* // размер буфера в байтах
)

Инициализация	X	Конфигурация	X	Управление	X

Назначение:

Получение кода и строкового описания последней ошибки, произошедшей при выполнении библиотечных функций.

Параметры:

SD

Дескриптор системы.

str

Указатель на буфер, в который функция запишет строку, содержащую описание ошибки.

len

Размер буфера в байтах (рекомендуется 128 символов).

Возвращаемые значения:

Функция возвращает код последней ошибки, произошедшей при вызовах библиотечных функций. Если указатель *str* не равен NULL, то по нему функция запишет описание ошибки.

Примечания:

Если в процессе выполнения какой-либо библиотечной функции произошла ошибка, то с помощью данной функции можно получить ее код и краткое толкование произошедшего сбоя.

После вызова этой функции код последней ошибки сбросится, т.е. до возникновения новой ошибки функция будет возвращать код QMX_OK и строку “No error.”.

4.1.9. QMX_MOD_SetActive()

int QMX_MOD_SetActive(
HANDLE *SD*, // дескриптор системы
BYTE *Slot*, // номер логического слота
BYTE *IsActive* // флаг активности модуля
)

Инициализация		Конфигурация	X	Управление	

Назначение:

Указывает, будет ли модуль участвовать в процессе передачи данных (т.е. являться активным).

Параметры:

SD

Дескриптор системы.

Slot

Номер логического слота модуля.

IsActive

Флаг активности модуля, может принимать следующие значения:

- QMX_ON – модуль активен
- QMX_OFF – модуль не активен

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки функция возвращает QMX_ERR.

4.1.10. QMX_MOD_SetStartMode()

int QMX_MOD_SetStartMode(
HANDLE *SD*, // дескриптор системы
BYTE *Slot*, // номер логического слота
BYTE *Mode* // режим запуска
)

Инициализация		Конфигурация	X	Управление	

Назначение:

Задает режим запуска сеанса передачи данных и отдельно взятых модулей.

Параметры:

SD

Дескриптор системы.

Slot

Номер логического слота модуля. Для настройки режима запуска индивидуального модуля этот параметр должен лежать в диапазоне [0..7]. Для одновременной настройки режима запуска сеанса передачи данных и всех модулей этот параметр должен быть равен 0xFF.

Mode

Режим запуска, может принимать следующие значения:

- QMX_START_MODE_MANUAL - Ручной запуск. Сеанс передачи данных или указанный модуль запускаются сразу после выполнения функций QMX_Start.
- QMX_START_MODE_EXTERNAL - Внешний запуск. После выполнения функции QMX_Start сеанс передачи данных или модуль запускаются по переходу внешнего синхросигнала (SYNC) в активное состояние.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки функция возвращает QMX_ERR.

Примечания:

По умолчанию сеанс передачи данных и все модули работают в режиме ручного запуска.

Пример 1 QMX_MOD_SetStartMode(SD, 0xFF, QMX_START_MODE_MANUAL)
И сеанс передачи данных, и все модули запустятся сразу после вызова функции QMX_Start.

Пример 2 QMX_MOD_SetStartMode(SD, 0, QMX_START_MODE_EXTERNAL)
Сеанс передачи данных и все модули, кроме модуля установленного в слот 0, запустятся сразу после вызова функции QMX_Start. Модуль, установленный в слот 0 запустится только после активации сигнала SYNC на внешнем разъеме ТОЛЬКО ЭТОГО модуля.

Пример 3 QMX_MOD_SetStartMode(SD, 0xFF, QMX_START_MODE_EXTERNAL)
После вызова функции QMX_Start, и сеанс передачи данных и все модули СИНХРОННО запустятся только после активации сигнала SYNC на внешнем разъеме ЛЮБОГО модуля.

Если в процессе выполнения программы режимы запуска могут неоднократно изменяться, рекомендуется перед каждым очередным запуском системы вначале вызывать функцию как указано в примере 1, а затем продолжать конфигурацию для отдельных модулей.

4.1.11. QMX_MOD_OutputPrestage()

int QMX_MOD_OutputPrestage(
HANDLE *SD*, // дескриптор системы
BYTE *Slot*, // номер логического слота
)

Инициализация		Конфигурация	X	Управление	

Назначение:

Осуществляет предварительную инициализацию данными локального буфера модуля вывода.

Параметры:

SD

Дескриптор системы.

Slot

Номер логического слота модуля.

Возвращаемые значения:

В случае успешного завершения функция возвращает QMX_OK.
В случае ошибки функция возвращает QMX_ERR.

Примечания:

Данные должны быть подготовлены в массиве по указателю *CBOutput.buf_start,* размер массива данных соответствует полному размеру локального буфера модуля и равен *CBOutput.buf_size.* (см. описание структуры QMX_CONFIG)

4.2. Структуры.

4.2.1. Структура QMX_CONFIG.

Структура применяется для получения следующей информации:

- Конфигурация устройства.
- Параметры сеанса передачи данных.

Структура заполняется при вызове функции QMX_Prepare.

Поля структуры:

BYTE *UsbSpeed*	Ревизия USB-хоста компьютера. Может принимать 2 значения: • 0x00 соответствует USB1.1. (12 Mbit/S) • 0x01 соответствует USB2.0. (480 Mbit/S)
CHAR *CtlName[11]*	Название крейт-контроллера
CHAR *CtlBIOSversion[6]*	Версия прошивки BIOS интерфейсной части крейт-контроллера
CHAR *CtlSerialNumber[10]*	Серийный номер крейт-контроллера
BYTE *ModQuantity*	Количество модулей в системе
CHAR *ModName[8][10]*†	Названия модулей
WORD *ModType[8]*	Типы модулей. (описаны в qmx.h)
BYTE *ModRevision[8]*	Ревизии модулей
CHAR *ModSerialNumber[8][10]*	Серийные номера модулей
HANDLE *DevHandle*	Служебное поле
WORD *RespTime*	Расчетное время реакции (латентность системы) – среднестатистическое время (в мс), за которое компьютер получит доступ к новым данным с самого медленного модуля. Рассчитывается на основании требуемого времени реакции, заданного пользователем в вызове QMX_Prepare.
DWORD *ReadFileSize*	Служебное поле
DWORD *WriteFileSize*	Служебное поле
DWORD *ModLocalBufferSize[8]*	Служебное поле
double *InputRate*	Общая скорость ввода данных (слов) в кГц
double *OutputRate*	Служебное поле
QMX_CIRC_BUF_CONFIG *CBInput*	Конфигурация кольцевого буфера ввода
QMX_CIRC_BUF_CONFIG *CBOutput*	Конфигурация кольцевого буфера вывода

4.2.2. Структура QMX_CIRC_BUF_CONFIG.

Структура используется для работы с сервером кольцевого буфера ввода, а также для работы с модулями вывода. Изменение данных по указателям полей структуры запрещено. Внимание, поля структуры могут изменяться после вызова функции QMX_Prepare.

Поля структуры:

WORD *buf_start	Указатель на начало буфера.
DWORD *buf_ptr	Указатель на счетчик записи в буфер [0..buf_size-1]. (В случае модуля вывода – указатель на счетчик обновленных половинок локального буфера модуля)
DWORD *buf_size	Указатель на размер буфера в словах (WORD) (В случае модуля вывода – размер буфера соответствует размеру локального буфера модуля).
HANDLE event	Дескриптор события, зарезервировано

4.2.3. Структура QMX_CC_F.

Структура используется для работы с функциями, которые манипулируют калибровочными коэффициентами.

Поля структуры:

float *Offset*	Калибровочный коэффициент масштаба.
float *Scale*	Калибровочный коэффициент смещения нуля.
char *Units[6]*	Единицы измерения.
BYTE *UnitsCode*	Числовой код соответствующей единицы измерения.

5. Сервер кольцевого буфера qmbox_ds.

Программа-сервер называется qmbox_ds.exe. Сервер должен быть обязательно запущен перед приложением. Через командную строку серверу можно передавать (в любом порядке) следующие опции:

Опция	Назначение	Знач. по умолчанию
-p	Задает приоритет процесса сервера. Значение может лежать в диапазоне от 0 (IDLE) до 4 (REALTIME) .	4
-u	Задает номер виртуального USB-слота, который должен использовать сервер.	0
-b	Задает размер кольцевого буфера ввода в словах.	4194304
-s	Команда. 1 – запустить сервер. 0 – остановить сервер.	1

Примеры:

qmbox_ds.exe	Запустить сервер, USB-слот 0, размер буфера ввода 4194304 слов, приоритет – REALTIME
qmbox_ds.exe –s0	Остановить сервер, USB-слот 0
qmbox_ds.exe –u1 -b1048576	Запустить сервер, USB-слот 1, размер буфера 1048576 слов.

После запуска, сервер ожидает поступления команд на запуск / останов сеанса передачи данных. Команды вырабатываются автоматически при вызовах функций QMX_Start и QMX_Stop из приложений пользователя.

Для доступа к данным, записываемым в кольцевой буфер сервером, приложения должны использовать структуру CBInput (тип структуры - QMX_CIRC_BUF_CONFIG), инкапсулированную в структуру QMX_CONFIG.

Для управления процессом передачи данных в динамический модуль вывода приложения должны использовать структуру CBOutput (тип структуры - QMX_CIRC_BUF_CONFIG), инкапсулированную в структуру QMX_CONFIG.

5.1. Формат данных кольцевого буфера.

После получения команды запуска сеанса передачи данных, сервер записывает все поступающие от модулей ввода системы QMBox слова данных (в формате `unsigned short int`) в кольцевой буфер. Перед каждым словом данных записывается 16ти разрядный тэг, упрощающий идентификацию этого слова данных. Тэг имеет следующий формат:

MSB LSB

R	R	R	R	R	R	R	R	M2	M1	M0	CH4	CH3	CH2	CH1	CH0

Старший байт тэга зарезервирован (R).

В младшем байте тэга в 3 старших битах (M0-M2) содержится номер логического слота модуля, с которого пришло слово данных.

В 5 младших битах (CH0 – CH4) содержится логический номер канала в этом модуле, с которого пришло слово данных (кол-во активных каналов каждого модуля задается на этапе конфигурации вызовом соответствующих функций). Логический номер канала представляет собой циклический счетчик [0..N-1], где N – кол-во активных каналов модуля. Необходимо помнить, что логические каналы в тэге нумеруются подряд, в соответствии с количеством активных каналов, а не в соответствии с их физическим расположением на модуле. Например, если на каком-то модуле были объявлены активными физические каналы 0, 2, 3, 8 (всего 4 канала), то в тэге они будут соответствовать логическим каналам 0, 1, 2, 3.