Программа монитор EZ-KIT Lite 7

Обзор

Программа монитор находится в памяти микросхемы EPROM. Она автоматически загружается при перезапуске во встроенную память программ и память данных. После загрузки программа монитор начинает выполняться, производя самотестирование регистров DSP во встроенной памяти, и перезапускает и инициализирует кодек AD1847. После этого программа ждет команд через последовательный порт связи. Подпрограмма «общения» кодека работает и во время исполнения подпрограммы прерывания.

Возможности программы монитора

Программа монитор обладает следующими возможностями:

- Самотестирование при включении
- Инициализация кодека и связь с ним
- Эмуляция UART со скоростью 9600 бод
- Выгрузка и загрузка содержимого памяти через последовательный порт

Ограничения

Чтобы свободного выполнения программы монитора следует учесть некоторые ограничения на все пользовательские программы. Среди них есть строгие ограничения, которые могут привести к нарушению процесса загрузки пользовательских программ:

- Зарезервированная область памяти от 0x3800 до 0x3FFF.
  Здесь располагается программа монитор. Любое нарушение приведет к разрушению программы монитора в результате незаконченной загрузки программы.

- Зарезервированная область памяти от 0x3E00 до 0x3FDF.
  Здесь располагаются переменные программы монитора. Пользовательские программы, использующие эти места памяти будут накладываться на переменные монитора; в результате программа пользователя не загрузится полностью.

- Возврат в монитор
  После завершения загрузки пользовательской программы, программное обеспечение монитора вызовет ее как подпрограмму. Пользовательская программа должна заканчиваться инструкцией RTS, чтобы должным образом вернутся в монитор.

Для пользовательских программ, которые не возвращаются в программу монитор, не существует никаких ограничений после завершения процесса загрузки. Другими словами, в то время как файл образа памяти не должен пытаться инициализировать зарезервированную память, пользовательская программа, тем не менее, свободна использовать эту память после начала своего выполнения.

Создание программ, используемых вместе с монитором

Для создания таких программ можно изменить один из примеров, включенных в комплект программного обеспечения EZ-KIT Lite, или написать собственную программу, используя включенный инструмент разработки. Для получения дополнительной информации обратитесь к главе 5. Существует несколько моментов, которые следует иметь ввиду при разработке программ, используемых с монитором.

Пользовательские программы могут использовать всю незарезервированную память программ и незарезервированную память данных. Это адреса с 0x0000 до 0x37FF для памяти программ и адреса с 0x0000 до 0x3DFF для памяти данных.

- Точка входа в программу
  Пользовательские программы, загружаемые хост программой, должны иметь свою точку входа в начале ядра памяти программ. Это достигается присваиванием точке входа программы первой инструкции в модуле и указанием этого модуля первым в процессе компоновки. В большинстве случаев программа будет стартовать с адреса 0x0000. Для этого можно применить директиву ассемблера `.MODULE/RAM/ABS=0`.

- Разрешение прерываний
  Перед передачей процесса выполнения пользовательской программе, все прерывания маскируются (IMASK=0). Это делается для того, чтобы пользовательская программа могла безопасно выйти из процесса инициализации перед собственным разрешением прерываний.

- Векторы прерываний
  Пользовательская программа должна инициализировать все используемые вектора прерываний. При этом в программе не может использоваться прерывание IRQ1, так как оно предназначено для управления связью через RS-232.

- Команды монитора
  Исходный код программы монитора содержится в каталоге `21XX\EZKITL\MONITOR`. Разрешается изменять этот код в соответствии с вашими потребностями. Если вы приступите к редактированию программы монитора, то вам потребуется доступ к его рабочим командам.

  Монитор может отвечать через последовательный порт приведенным ниже командам. Они могут использоваться при сопряжении EZ-KIT Lite с другой, нежели базовой программой.

0. **Имя команды**:

принимает: обозначение кода и данных, принятых от хост программы. Символы, не заключенные в скобки принимаются непосредственно, заключенные в скобки – в порядке от старшего байта к младшему.

посылает: обозначение кода и данных, отправленных хост программе. Символы, не заключенные в скобки принимаются непосредственно, заключенные в скобки – в порядке от старшего байта к младшему.

1. **Beep**

принимает: $$$
посылает: ok [*led count*][*alive called count*][*selftest hi*][*selftest lo*]

[*led count*] младшие 8 разрядов счетчика, увеличивающегося со скоростью 28800 в секунду. Задержка счета показывает, что прерывание таймера возможно не работает.
[*alive called count*] увеличивается на единицу каждый раз, когда эта вызывается эта команда или команда *alive*.
[*selftest hi*][*selftest lo*] результат самотестирования.

На аудио выход будет послан короткий сигнал.

2. **Alive inquiry**

принимает: $OK
посылает: ok [*led count*][*alive called count*][*selftest hi*][*selftest lo*]

[*led count*] младшие 8 разрядов счетчика, увеличивающегося со скоростью 28800 в секунду. Задержка счета показывает, что прерывание таймера возможно не работает.
[*alive called count*] увеличивается на единицу каждый раз, когда эта вызывается эта команда или команда *alive*.
[*selftest hi*][*selftest lo*] результат самотестирования.

Подобна команде Beep, но сигнал на аудио выходе не генерируется.

3. **upload data memory content (выгрузить содержимое памяти данных)**

принимает: $UD [*hi start*][*lo start*][*hi len*][*lo len*]
посылает: [*hi*][*lo*] ..... [*hi sum*][*lo sum*]$!

[*hi start*][*lo start*] стартовый адрес в памяти.
[*hi len*][*lo len*] количество слов.
[*hi*][*lo*] .... содержимое памяти, начиная со стартового адреса.
[*hi sum*][*lo sum*] 16-разрядное значение контрольной суммы данных.

4. **upload program memory content (выгрузить содержимое памяти программ)**

принимает: $UP [*hi start*][*lo start*][*hi len*][*lo len*]
посылает: [*hi*][*mi*][*lo*] ..... [*hi sum*][*mi sum*][*lo sum*]$!

[*hi start*][*lo start*] стартовый адрес в памяти.
[*hi len*][*lo len*] количество слов.
[*hi*][*mi*][*lo*] .... содержимое памяти, начиная со стартового адреса.
[*hi sum*][*mi sum*][*lo sum*] 24-разрядное значение контрольной суммы данных.

5. **download data memory content (загрузить содержимое памяти данных)**

принимает: $DD [*hi start*][*lo start*][*hi len*][*lo len*][*hi*][*lo*] ……
посылает: [*hi sum*][*lo sum*]

[*hi start*][*lo start*] стартовый адрес в памяти.
[*hi len*][*lo len*] количество слов.
[*hi*][*lo*] .... содержимое памяти, начиная со стартового адреса.
[*hi sum*][*lo sum*] 16-разрядное значение контрольной суммы данных.

6. **download program memory content (загрузить содержимое памяти программ)**

принимает: $DP [*hi start*][*lo start*][*hi len*][*lo len*][*hi*][*mi*][*lo*] ……
посылает: [*hi sum*][*mi sum*][*lo sum*]

[*hi start*][*lo start*] стартовый адрес в памяти.
[*hi len*][*lo len*] количество слов.
[*hi*][*lo*] .... содержимое памяти, начиная со стартового адреса.
[*hi sum*][*mi sum*][*lo sum*] 24-разрядное значение контрольной суммы данных.

7. **Begin user program execution (начать выполнение программы пользователя)**

принимает: $GO [*hi address*][*lo address*]
посылает: [*hi address*][*lo address*]

[*hi address*][*lo address*] точка входа в программу, посланная хост программой программе монитору. Для подтверждения приема адрес посылается программой монитора обратно хост программе.

Отладка

В процессе отладки пользовательских программ могут быть полезны следующие инструкции:

- SET FL1
- RESET FL1
- TOGGLE FL1

Состояние светодиода FL1 непосредственно управляется флагом DSP FL1. Приведенные инструкции могут быть размещены в любом месте пользовательской программы, и использоваться для визуального отображения состояния некоторых переменных, или отображения выполнения какой-либо секции.

Для некритичных к времени программ, возможно получать некоторую информацию, программируя FL1 на мерцание с различной скоростью. Например, следующая программа включает мерцание светодиода с некоторой скоростью.

```
          cntr = 80;
          do lp1 until ce;
               cntr = 50;
               do lp2 until ce;
                    cntr = 12000;
                    do lp3 until ce;
lp3:                     nop;
lp2:                nop;
lp1:          toggle fl1;
```

Память DSP

Пользовательские программы могут хранить некоторые данные в памяти программ или в памяти данных, которые могут быть восстановлены после того, как пользовательские программы вернут управление обратно программе монитору (посредством инструкции RTS).

Для этого необходимо использовать команды выгрузки памяти программ или памяти данных.