MOVICOM

Контроллер MRC120

Руководство пользователя

Мовиком, 2007
Версия 1.21
23.01.2007

Инновационное предприятие МГУ им. М.В. Ломоносова
119192, Москва, Мичуринский пр-т, д. 1
тел/факс: +7 (495) 939-5289
тел: +7 (495) 410-0691
www.movicom.ru

Содержание

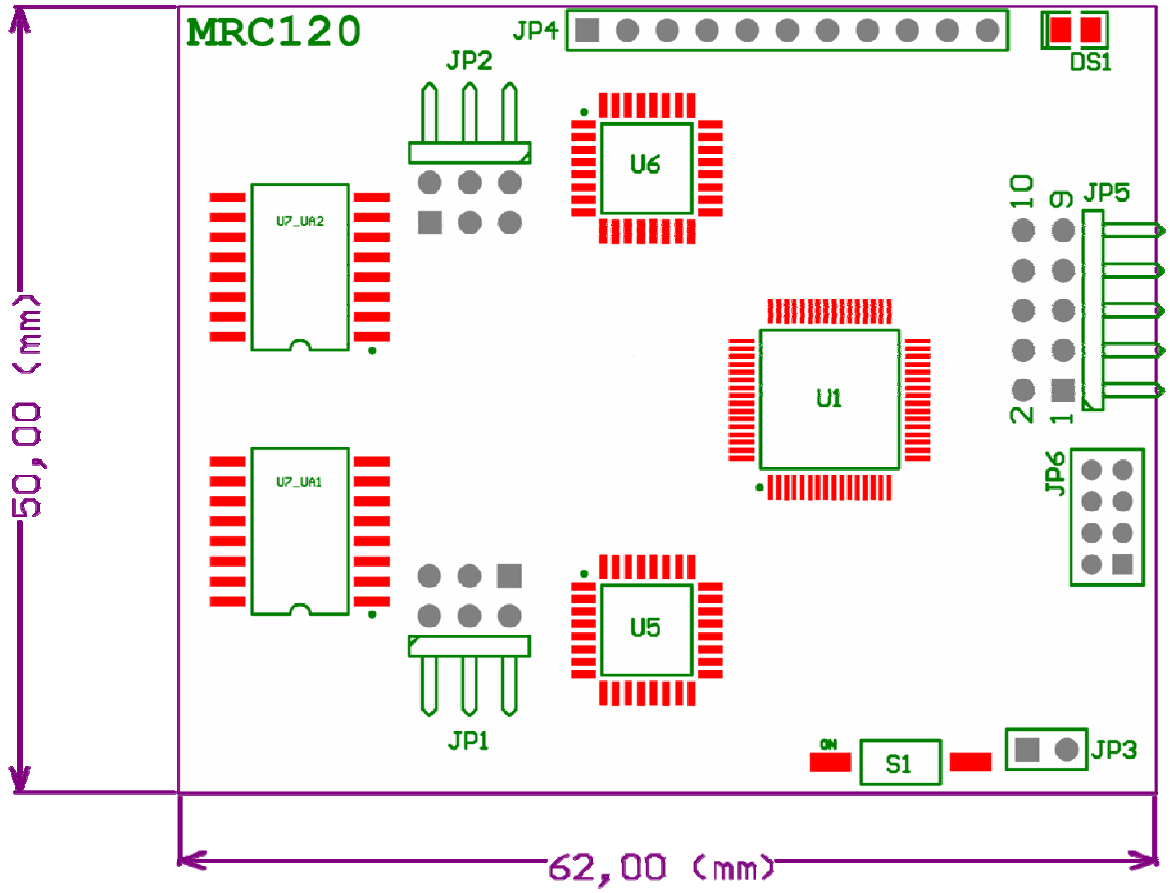
Инновационное предприятие МГУ им. М.В. Ломоносова
119192, Москва, Мичуринский пр-т, д. 1
тел/факс: +7 (495) 939-5289
тел: +7 (495) 410-0691
www.movicom.ru

1. Общее описание контроллера

Контроллер MRC120 предназначен для использования в двухколесных микроробототехнических системах, а также в различных других малоразмерных роботах.

2. Функциональный состав контроллера

Схема **контроллера MRC120**



Контроллер включает в себя следующие узлы:

1) 32-х битный микроконтроллер NXP LPC2131 (U1) с ядром ARM7 TDMI-S
   - Тактовая частота ядра ~ 59 МГц
   - 32 КБ flash памяти
   - 8 КБ оперативной памяти
2) два 8-битных микроконтроллера Atmel ATMega48 (U5, U6), обрабатывающие сигналы от импульсных датчиков (энкодеров). Осуществляют подсчет скорости и координат по сигналам от датчиков.
3) два драйвера двигателя постоянного тока Allegro A3949 (U7_UA1, U7_UA2) с подсоединенными каналами ШИМ от микроконтроллера.

Описание разъемов контроллера:

1) Разъем питания (JP3).
   - [1] «+» питания
   - [2] «-» питания

2) Разъемы для подключения двигателей со встроенными энкодерами (JP1, JP2). Спроектированы для использования с малоразмерными двигателями производства Faulhaber со встроенным энкодером. Могут быть использованы и с другими двигателями.

3) Последовательный порт UART (JP4) для соединения с внешним радиомодулем.
   - [1] UART RX (т.е. ножка приема процессора)
   - [2] UART TX (т.е. ножка посылки процессора)
   - [3] – [10] GND
   - [11] +3.3 V

4) Отладочный порт JTAG (JP5) для микроконтроллера LPC2131.
   - [1] TMS
   - [2] nTRST
   - [3] TCK
   - [4] TDI
   - [5] TDO
   - [6] GND
   - [7] XRST
   - [8] + 3.3 V
   - [9], [10] NC (не соединены)

5) Дополнительный порт (P5), включающий 3 аналоговых входа, 1 цифровой, а также дублирование ШИМ и питание 3.3 В.
   - [1] GND
   - [2] +3.3 V
   - [3] PWM1 (канал ШИМ)
   - [4] DIN1 (цифровой вход)
   - [5] AIN2 (аналоговый вход)
   - [6] AIN3 (аналоговый вход)
   - [7] PWM2 (канал ШИМ)
   - [8] AIN4 (аналоговый вход)

Контроллер допускает напряжения питания от 5.5 до 15 вольт. На входе питания стоит защита от переполюсовки и выключатель (S1).

2.1. Обмен данными по UART

С помощью последовательного порта UART контроллер может обмениваться данными с PC и другими устройствами. Разъем порта (JP4) позволяет подключить радиомодули R-Cat производства Movicom, а также другие совместимые радиомодули или модули Bluetooth.

2.2. Аналого-цифровой преобразователь

АЦП служит для преобразования сигнала 0 ÷ 3.3 V в цифровой код с разрешением 10 бит. Время одного преобразования составляет около 3 нс. После этого результат преобразования можно считать из внутренних регистров микроконтроллера. Входы микроконтроллера рассчитаны на напряжения 0 ÷ 5 V. Тем не менее, не рекомендуется превышать уровень 3.3 V, так как это может привести к искажениям результатов аналого-цифрового преобразователя.

2.3. Широтно-импульсный модулятор

Контроллер содержит два канала ШИМ, в стандартной библиотеке работающие с частотой 20 кГц и разрешением 10 бит, они выведены на дополнительный разъем, а также подключены к усилителям мощности, расположенным на плате.

Для подключения двигателей можно использовать специальные разъемы на плате. Напряжение на них подается от встроенных усилителей мощности. Тем не менее, есть возможность подключить двигатель через внешний усилитель мощности. Для этого необходимо подключить усилитель к каналам ШИМ на дополнительном разъеме.

2.4. Обработка импульсных датчиков

Контроллер содержит 2 идентичных канала обработки датчиков. Обработка происходит с помощью микроконтроллеров ATMega48. Рабочая программа записывается при изготовлении контроллеров, существует возможность замены этой программы, но стандартная библиотека не поддерживает эту функцию.

3. Библиотека нижнего уровня

Все нижеследующее верно для библиотеки версии 1.1

3.1. Комплект поставки и пример использования

Библиотека предназначена для компилятора IAR for ARM.
Состоит из 2 файлов

- **fblib.h** содержит прототипы функций библиотеки.
- **fblib.r79** собственно файл библиотеки для IAR.

Для использования необходимо присоединить к проекту IAR (Project->Add Files) файл библиотеки **fblib.r79** и прописать `#include "fblib.h"` в программе**.**

В комплект поставки входит проект в среде IAR Embedded Workbench, иллюстрирующий использование функций библиотеки, а также реализующий простейшее движение робота с помощью PID-регуляторов на каждое колесо.

Также в комплект поставки входит файл **32bit_types.h,** который производит переопределение типов данных для удобства программирования.

3.2. Инициализация

Для инициализации контроллера MRC120 с использованием библиотеки необходимо вызвать функцию

```
void InitSystem(void (*servo_addr)(), UI_16 freq)
```

Функция инициализирует базовую функциональность контроллера, датчиков, радио, драйверов двигателей, устанавливает периодическое прерывание на вызов серво функции с адресом `servo_addr` и частотой `freq` Гц.

Рекомендуется использовать эту функции в первую очередь при начале работы программы.

Функция, которая должна находиться в начале функции прерывания таймера (`servo_addr` в `InitSystem` указывает на эту функцию)

```
void ServoPreamble(void)
```

Добавление этой функции в начало функции таймера является важным для корерктной работы

3.3. Импульсные датчики

Инициализация датчиков происходит автоматически при вызове функции `InitSystem`.

Для получения информации от энкодеров необходимо вызвать функцию

```
void ReadEncoders(SI_32 *distance, SI_32 *velocity)
```

В качестве параметров `distance` и `velocity` необходимо задать предварительно созданные массивы `SI_32[2]`.

- `distance` – положение вала датчика, измеряется в метках датчика и может лежать в пределах от 0 до 65535 (16-bit unsigned integer). Вычисление положения происходит за счет подсчета количества меток пройденных двигателем и не сбрасывается при полных оборотах. При подсчете пройденного

пути необходимо учесть, что при переполнении значение позиции перескакивает с максимума на минимум и с минимума на максимум. Число меток на один оборот зависит от датчика. Для вычисления пути, пройденного между двумя вызовами функции `ReadEncoders`, необходимо разницу позиций нормировать с учетом возможных переполнений, и затем перевести в единицы длины по формуле[1]

```
distance*[ПИ*ДИАМЕТР_КОЛЕСА / (ЧИСЛО_ МЕТОК*РЕДУКЦИЯ)]
```

- `velocity` – знаковое время между двумя метками, для перевода в скорость необходимо

  ```
  14745600*[ПИ*ДИАМЕТР_КОЛЕСА / (ЧИСЛО_ МЕТОК*РЕДУКЦИЯ)]/velocity
  ```

  В случае, если двигатель не крутится и новая метка не была замечена, возвращается значение ±0xFFFF соответствующее нулевой скорости[1].

[1] Смотри пример PID

3.4. Широтно-импульсный модулятор

Инициализация ШИМ происходит автоматически в `InitSystem`.
Для задания необходимых значений ШИМ существует функция

```
void SetMotorsVoltage (SI_16 right_val, SI_16 left_val)
```

Задаваемые значения ШИМа могут меняться в пределах от -1024 до 1024, значения вне этих пределов будут обрезаться до границ.

3.5. Функции работы с радио (UART)

Инициализация UART происходит при вызове функции `InitSystem`.
Для посылки сообщения через радиоканал следует пользоваться функцией

```
void SendRadioMessage(UI_8 robo_num, UI_8 msg_num, UI_8 *data)
```

- `robo_num` – номер робота. Должен быть от 0 до 7 включительно.
- `msg_num` – номер сообщения. Должен быть от 0 до 31 включительно.

В этом случае длина сообщения (`data`) не может превышать 10 байт. Для приема сообщений существует функция

```
UI_8 ReceiveRadioMessage(UI_8 *robo_num, UI_8 *msg_num, UI_8 *data)
```

которая считывает из буфера сообщение длинной 10 байт и возвращает 1 в случае успеха и 0, если сообщение еще не было получено.

3.6. Аналого-цифровой преобразователь (АЦП)

Каждое аналого-цифровое преобразование нужно инициировать функцией

```
void InitAnalogConv(UI_8 chan_num)
```

Где `chan_num` – номер канала АЦП. Допустимые номера 0, 1, 2, 3. Канал № 0 зарезервирован для системы определения заряда батареи.

Для чтения требуемого канала нужно воспользоваться функцией

```
UI_8 AnalogConvCompleteCheck(UI_8 chan_num, UI_16 *value)
```

Функция проверяет, есть ли уже полученное значение и в случае успеха возвращает 1 и записывает значение преобразования по адресу `value`, иначе возвращает 0. Возвращаемое значение меняется в пределах от 0 до 1023. Нулевому значению соответствует напряжение 0 вольт, а максимальному значению соответствует напряжение +3.3 V.

3.7. Прочие функции

1) Включение и выключение LED-индикатора:

```
void PwrLEDOn(void)
void PwrLEDOff(void)
```

Функция для индикации напряжения частотой мигания лампочки

```
UI_8 PwrLEDIndicator(void)
```

моргает тем чаще, чем меньше зарядка аккумулятора и возвращает заряд батареи в процентах. Если моргания редко, значит все в порядке, если постоянно, то надо заряжать аккумулятор (подразумевается, что используется аккумулятор на 7.4 V).

Перед использованием надо проинициализировать АЦП функцией `InitAnalogConv(0)`.

2) Функция

```
UI_8 ReadDIN1(void)
```

возвращает состояние цифрового входа DIN1. Если на нем присутствует логический 0, то функция возвращает 0. В случае положительного сигнала функция возвращает 1.