www.e-core.ru

Программирование источников питания E-core

PRG 1.3

HybridPowerCoder v1.2

1 ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

1 Общее описание

Интегрированная среда разработки HybridPowerCoder (далее по тексту IDE) предназначена для написания программы (алгоритма) работы источников питания E-core, имеющих соответствующую функцию. А также компиляции и записи этих программ в источник питания, чтения программ и представления их в текстовом виде.

Внешний вид основного окна IDE представлен на рисунке 1.

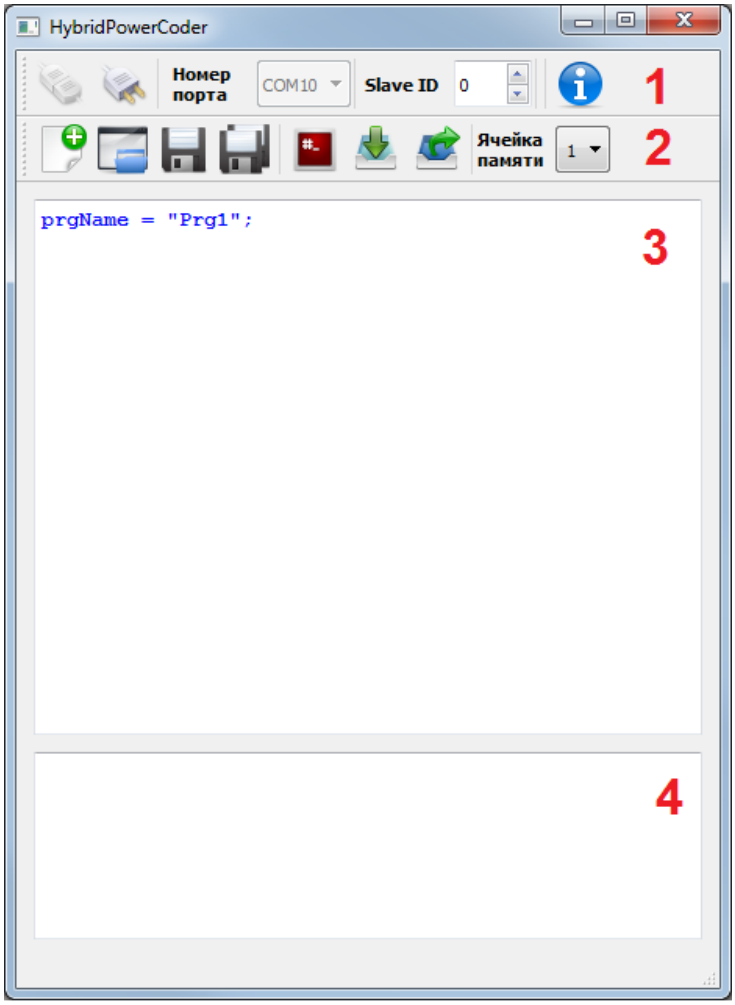


Рисунок 1 - основное окно IDE

На рисунке 1 цифрами обозначены:

1 и 2 – верхняя и нижняя панель инструментов соответственно;

3 – текстовое поле для написания программы;

4 – информационное текстовое поле.

Органы управления верхней панели инструментов:

 – кнопка подключения устройства;

 – кнопка отключения устройства;

– панель выбора COM порта адаптера;

– панель настройки сетевого идентификатора источника питания;

 – кнопка информации о программе.

Органы управления нижней панели инструментов:

 – кнопка создать новый файл программы;

 – кнопка открыть файл программы;

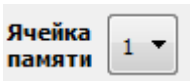
 – кнопка сохранить файл программы;

 – кнопка сохранить как файл программы;

 – кнопка скомпилировать файл программы;

 – кнопка записать программу в устройство;

 – кнопка считать программу из устройства;



– панель настройки ячейки памяти устройства.

2 Файл программы

Файл программы сохраняется в текстовом виде в формате *.prge . Создание нового файла, запись или открытие файла осуществляется соответствующими кнопками на нижней панели задач.

3 Компиляция/запись/чтение программы

При компиляции программы текстовый код интерпретируется в машинные коды источника питания.

Компиляция программы осуществляется при нажатии кнопки  , а также автоматически перед записью программы в источник питания.

Для записи (или чтения) программы в источник питания необходимо сначала подключить адаптер к источнику питания и к ПК, выбрать COM порт на который установился адаптер и нажатием кнопки  подключить источник питания. После чего необходимо выбрать ячейку памяти.

Подключение адаптера к ПК необходимо выполнять до запуска IDE.

Для записи программы в источник питания необходимо нажать кнопку  , после чего будет автоматически запущена компиляция и при отсутствии ошибок результат компиляции будет записан в источник питания.

Для чтения программы из источника питания необходимо нажать кнопку  , при успешном чтении данных они будут переведены из машинных кодов в текстовый код программы.

Информация о результатах компиляции/записи/чтении программы отображается в информационном текстовом поле.

4 Подсветка

В IDE реализована цветовая подсветка синтаксиса.

Если строка названия программы написана корректно, то она выделяется жирным шрифтом, синим цветом.

Если строка инструкции написано корректно, то она выделяется синим цветом.

Если объявление ссылки написано корректно, то она выделяется розовым цветом.

Комментарии выделяются зеленым цветом.

2 ЯЗЫК ПРОГРАММИРОВАНИЯ

2.1 Синтаксис

При написании программ используются переменные, константы, инструкции и ссылки.

Переменные могут являться входными/выходными сигналами источника питания или просто участками памяти для хранения данных. Допустимые имена переменных описаны в разделе 2.3.

Запись в переменную осуществляется когда она используется в инструкциях присваивания слева от знака присваивания.

Чтение переменной осуществляется когда она используется в инструкциях присваивания справа от знака присваивания, а также при использовании в инструкции if слева или справа от знака сравнения.

Константы записываются в десятичном виде, знак минус не должен быть отделен пробелом.

Каждая инструкция занимает отдельную строку и оканчивается знаком ; . Описание инструкций в разделе 2.4.

Ссылки представляют собой метки программы для перехода к ним инструкций условного if и безусловного jmp перехода. Ссылка занимает отдельную строку и оканчивается знаком ; .

Время выполнения любой инструкции 100мс, ссылки обрабатываются без задержки.

2.2 Название программы

При написании программы обязательно необходимо задать ее краткое название. По этому названию программа будет выбираться (активироваться) в пользовательском меню источника питания.

Название программы задается в первой строке.

Синтаксис

```
prgName="name";
```

name – имя программы длинной не более 5 символов. В имени программы могут использоваться буквы латинского алфавита, цифры и знак _ .

2.3 Переменные

Out

Переменная управления включением/выключение выхода.

Запись в Out значения 1 включает выход (источник питания переходит в режим CC или CV), запись в Out любого другого значения переводит источник питания в режим Off.

При чтении Out возвращает следующие значения:

0 – источник питания в режиме Off;

1 – источник питания в режиме CV;

2 – источник питания в режиме CC.

ExOut

Переменная управления линей внешнего выхода.

Запись в ExOut значения 1 устанавливает активный уровень линии, запись любого другого значения устанавливает неактивный уровень.

При чтении ExOut возвращает записанное значение.

При включении прибора ExOut устанавливается в 0.

ExIn

Переменная линии внешнего входа.

Возвращаемое значение зависит от выбранного режима работы.

Когда параметр меню External input в значении Off при чтении ExIn возвращает следующие значения:

0 - нет сигнала;

1 - есть сигнал;

2 - положительный (нарастающий) фронт сигнала;

3 - отрицательный (спадающий) фронт сигнала.

Когда параметр меню External input в значении PWM при чтении ExIn возвращает значение заполнения ШИМ.

Запись в переменную ExIn возможен, но не имеет практического смысла.

Us

Is

Переменные установки выходного напряжения Us и тока Is.

Значения, записанные в эти переменные, устанавливаются в качестве выходных параметров.

Напряжение задается в мВ, ток в мА.

При чтении Us и Is возвращают текущие установленные значения напряжения и тока.

Uo

Io

Переменные выходного напряжения Uo и тока Io.

При чтении Uo и Io возвращают измеренные АЦП напряжение и ток. Напряжение в мВ, ток в мА.

Запись в переменные возможна, но имеет практического смысла т.к. при ближайшем преобразовании АЦП в них будут записаны результаты АЦП.

Частота обновления примерно 3 Гц.

M1 ... M8

Переменные для хранения пользовательских параметров.

При запуске программы автоматически устанавливаются в 0.

Переменные могут принимать положительные и отрицательные значения, размерность 32 бит.

T1 ... T4

Переменные таймеры.

При запуске программы автоматически устанавливаются в 0. При выполнении каждой инструкции увеличиваются на 1.

Доступны для чтения и записи.

Переменные могут принимать положительные и отрицательные значения, размерность 32 бит.

2.4 Инструкции

=

Инструкция присваивания.

Синтаксис

```
Переменная=Аргумент;
```

В качестве переменной может использоваться переменная из раздела 2.3.

В качестве аргумента может использоваться переменная из раздела 2.3 или константа.

При выполнении инструкции в переменную записывается значение поля аргумент.

Пример использования

```
....
Us = 25000;
....
```

Этот участок программы устанавливает выходное напряжение 25В.

+=

Инструкция сложения и присваивания.

Синтаксис

```
Переменная+=Аргумент;
```

Инструкция эквивалентна выражению

```
Переменная=Переменная+Аргумент;
```

В качестве переменной может использоваться переменная из раздела 2.3.

В качестве аргумента может использоваться переменная из раздела 2.3 или константа.

При выполнении инструкции происходит чтение переменной, суммирование прочитанного значения с аргументом и запись результата в переменную.

Пример использования

```
....
Us += 100;
....
```

Этот участок программы увеличивает выходное напряжение на 100мВ.

-=

Инструкция вычитания и присваивания.

Синтаксис

```
Переменная-=Аргумент;
```

Инструкция эквивалентна выражению

```
Переменная=Переменная-Аргумент;
```

В качестве переменной может использоваться переменная из раздела 2.3.

В качестве аргумента может использоваться переменная из раздела 2.3 или константа.

При выполнении инструкции происходит чтение переменной, вычитание из прочитанного значения аргумента и запись результата в переменную.

Пример использования

```
....
M1 = 200;
....
Is -= M1;
....
```

Этот участок программы уменьшает ограничение выходного тока на 200мА.

***=**

Инструкция умножения и присваивания.

Синтаксис

```
Переменная*=Аргумент;
```

Инструкция эквивалентна выражению

```
Переменная=Переменная*Аргумент;
```

В качестве переменной может использоваться переменная из раздела 2.3.

В качестве аргумента может использоваться переменная из раздела 2.3 или константа.

При выполнении инструкции происходит чтение переменной, умножение прочитанного значения на значение аргумента и запись результата в переменную.

Пример использования

```
....
M1 *= -1;
....
```

Этот участок программы меняет знак значения M1.

/=

Инструкция деления и присваивания.

Синтаксис

```
Переменная/=Аргумент;
```

Инструкция эквивалентна выражению

```
Переменная=Переменная/Аргумент;
```

В качестве переменной может использоваться переменная из раздела 2.3.

В качестве аргумента может использоваться переменная из раздела 2.3 или константа.

При выполнении инструкции происходит чтение переменной, деление прочитанного значения на значение аргумента и запись результата в переменную.

При делении на ноль в переменную записывается ноль.

Пример использования

```
....
Us /= 2;
....
```

Этот участок программы в два раза уменьшает выходное напряжение.

if

Инструкция условного перехода.

Синтаксис

```
if•Условие•#ссылка1•else•#ссылка2;
```

или

```
if•Условие•#ссылка1;
```

Знаком • обозначены обязательные пробелы.

При выполнении инструкции если `Условие` истинно, то осуществляется переход к строке программы, следующей за `#ссылка1` . Если `Условие` ложно, то при наличии `else` осуществляется переход к строке программы, следующей за `#ссылка2`, иначе осуществляется переход к следующей строке программы.

Синтаксис поля `Условие`

```
Переменная ОператорСравнения Аргумент
```

В качестве переменной может использоваться переменная из раздела 2.3.

Операторы сравнения:

== равно

!= не равно

< меньше

<= меньше или равно

> больше

>= больше или равно

В качестве аргумента может использоваться переменная из раздела 2.3 или константа.

Пример использования

```
....
#1;
Us += 10;
if Us > 14500 #2;
if Io < 3000 #1;
#2;
....
```

Этот участок программы увеличивает выходное напряжение на 10мВ пока выходной ток не будет больше или равен 3А или напряжение не будет больше 14,5В.

nop

Пустая команда. Может использоваться для формирования задержки на один такт выполнения программы.

jmp

Инструкция безусловного перехода.

Синтаксис

`jmp•#ссылка;`

Знаком • обозначены обязательные пробелы.

При выполнении этой инструкции происходит переход к строке программы следующей за `#ссылка`.

delay_ms

Инструкция задержки

Синтаксис

`delay_ms•X;`

Знаком • обозначены обязательные пробелы.

`X` значение задержки в миллисекундах, которое задается константой.

При выполнении этой инструкции происходит задержка выполнения следующей строки кода заданной длительности.

При формировании задержки учитывается задержка выполнения самой инструкции. Задержка не может быть меньше 100мс.

Пример использования

```
....
delay_ms 1500;
....
```

Этот участок программы формирует задержку длительностью 1500мс.

delay_s

Инструкция задержки

Синтаксис

```
delay_s•X;
```

Знаком • обозначены обязательные пробелы.

`X` значение задержки в секундах.

При выполнении этой инструкции происходит задержка выполнения следующей строки кода заданной длительности.

При формировании задержки учитывается задержка выполнения самой инструкции.

Пример использования

```
....
delay_s 5;
....
```

Этот участок программы формирует задержку длительностью 5с.

2.4 Ссылки

Синтаксис объявления ссылки

```
#ссылка;
```

Синтаксис перехода по ссылке

```
jmp #ссылка;
if Io > 1000 #ссылка;
```

Поле `ссылка` должно быть числом от 1 до 99.

При компиляции программы все ссылки, используемые в инструкциях условного и безусловного перехода, должны быть объявлены.

Объявляемые ссылки должны быть уникальны, номер ссылки не должен повторяться.

2.5 Комментарии

Комментарии предназначены для пояснения кода, при компиляции они игнорируются.

Каждый комментарий начинается с новой строки, в начале комментария ставятся без пробелов два знака / .

Комментарии могут быть на русском языке.

Пример использования

```
....
// задержка 5 секунд
delay_s 5;
....
```