

Wakayama. rbボード
Ver. 2
説明資料 ver1. 1

Wakayama. rb
たろサ

Wakayama.rbボード Ver. 2

特 徴

- ・mrubyを実装したRubyボードです。オブジェクト指向スクリプト言語Rubyを用いてプログラミングできます。作成したプログラムはシリアル経由で書き換えることができます。
- ・頭脳にGR-SAKURA搭載のRX63Nを持ち、ピン配置はGR-KURUMI(ほぼ)互換というガジェルスネ大好きな作者の趣向がもろに出た一品です。



ハード仕様

MCU

32ビットCPU RX63N(100ピン)
96MHz
FlashROM : 1Mバイト
RAM : 128Kバイト
データ用Flash : 32Kバイト

ボード機能

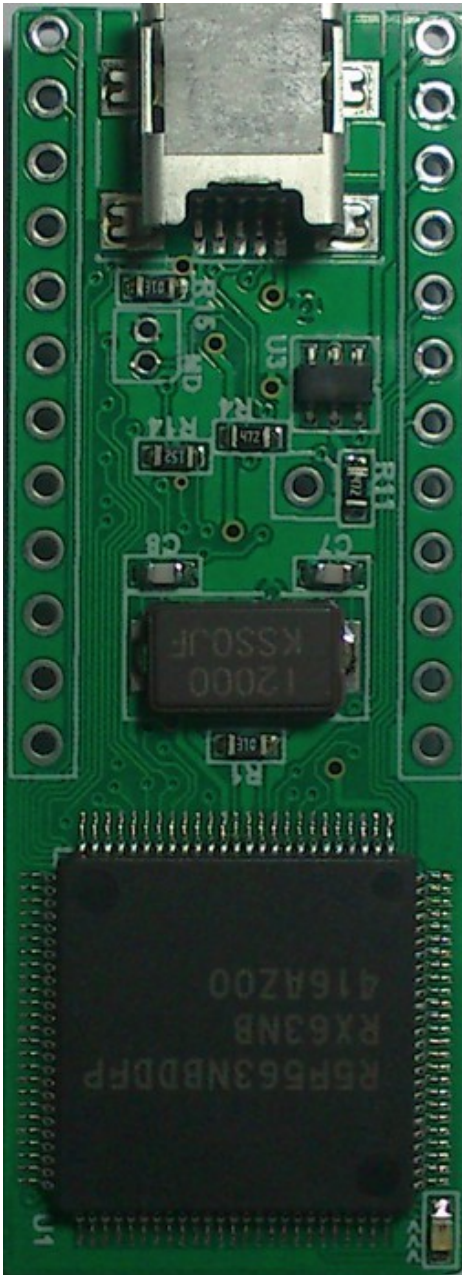
USBファンクション端子 (mini-B)
LED 1個
I/Oピン 20ピン
シリアル 3個(+1個可能)
SPI 1個
A/D 4個
RTC
I2C、PWM、Servoは自由割当てです。

電源

5V (USBバスパワー)

サイズ

50×18mm



ソフト仕様

コマンドはこんな感じです。

カーネルクラス

```
pinMode(pin, mode)
digitalRead(pin)
digitalWrite(pin, value)
analogRead(number)
pwm(pin, value)
pwmHz(value)
analogDac(value)
delay(value)
millis()
micros()
led(sw)
```

システムクラス

```
System.exit()
System.setrun(filename)
System.version(r)
System.push(address, buf, length)
System.pop(address, length)
System.fileload()
```

ファイルクラス

```
MemFile.open(number, filename, mode)
MemFile.close(number)
MemFile.read(number)
MemFile.write(number, buf, len)
MemFile.seek(number, byte)
```

シリアルクラス

```
Serial.begin(number, bps)
Serial.setDefault(number)
Serial.print(number, string)
Serial.println(number, string)
Serial.read(number)
Serial.write(number, buf, len)
Serial.available(number)
Serial.end(number)
```

I2Cクラス

```
I2c.sdasci(sda, scl)
I2c.write(id, address, data)
I2c.read(id, addressL, addressH)
I2c.begin(id)
I2c.lwrite(address)
I2c.end()
I2c.request(id, n)
I2c.lread()
I2c.freq(Hz)
```

サーボクラス

```
Servo.attach(ch, pin[, min, max])
Servo.write(ch, angle)
Servo.us(ch, us)
Servo.read(ch)
Servo.attached(ch)
Servo.detach(ch)
```

ソフト仕様

コマンドはこんな感じです。

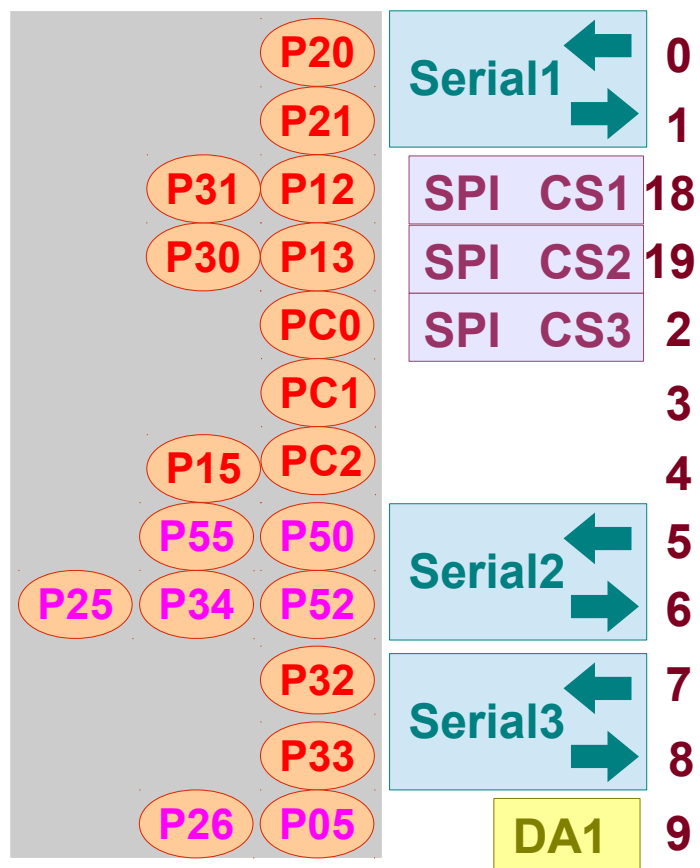
リアルタイムクロッククラス

Rtc. begin()

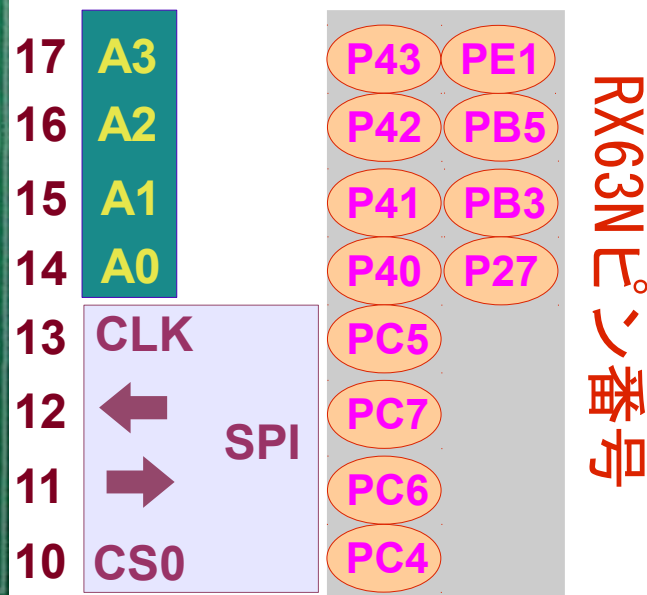
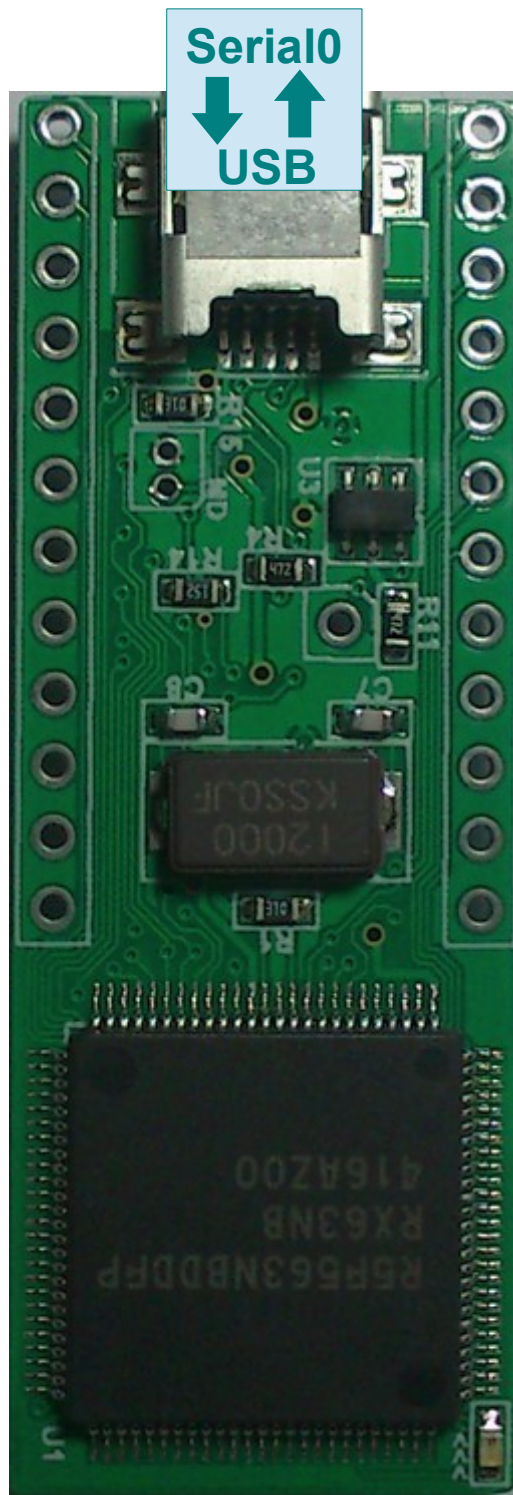
Rtc. setDateTime (Year, Month, Day, Hour, Minute, Second)

Rtc. getDateTime ()

RX63Nピン番号



赤文字ピン番は
5Vトレラント



RX63Nピン番号

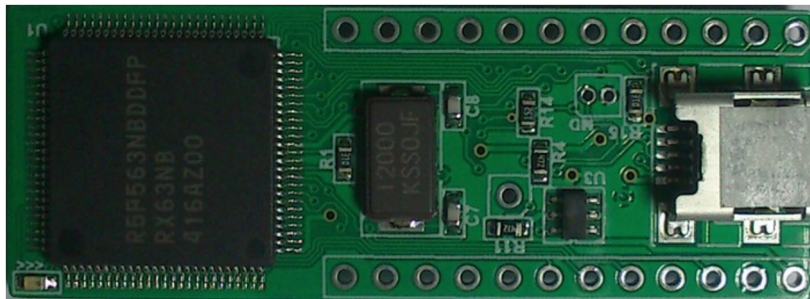
*ruby*プログラムの実行

WRBボードは、内部に*ruby*プログラムを保存できます。ファイル形式はmrbcによりコンパイルしたmrb形式のファイルとなります。コンパイラはWebコンパイルの予定ですがまだできません。

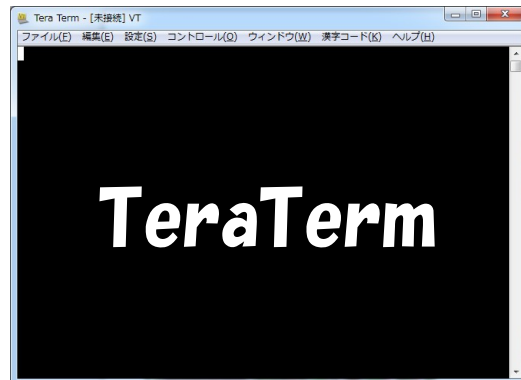
WRBボードは、後述する「電源オンで即実行するモード」に切り替わっていない限り、通常、電源をオンするとコマンドモードとなります。

プログラムの書き込み

WRBボードはPCとUSB経由で接続し、シリアル通信を用いて通信します。
この通信を使って、Rubyのプログラムを書き込んだり、実行したり、WRB
ボードからデータをPCに出力したりします。



シリアル通信

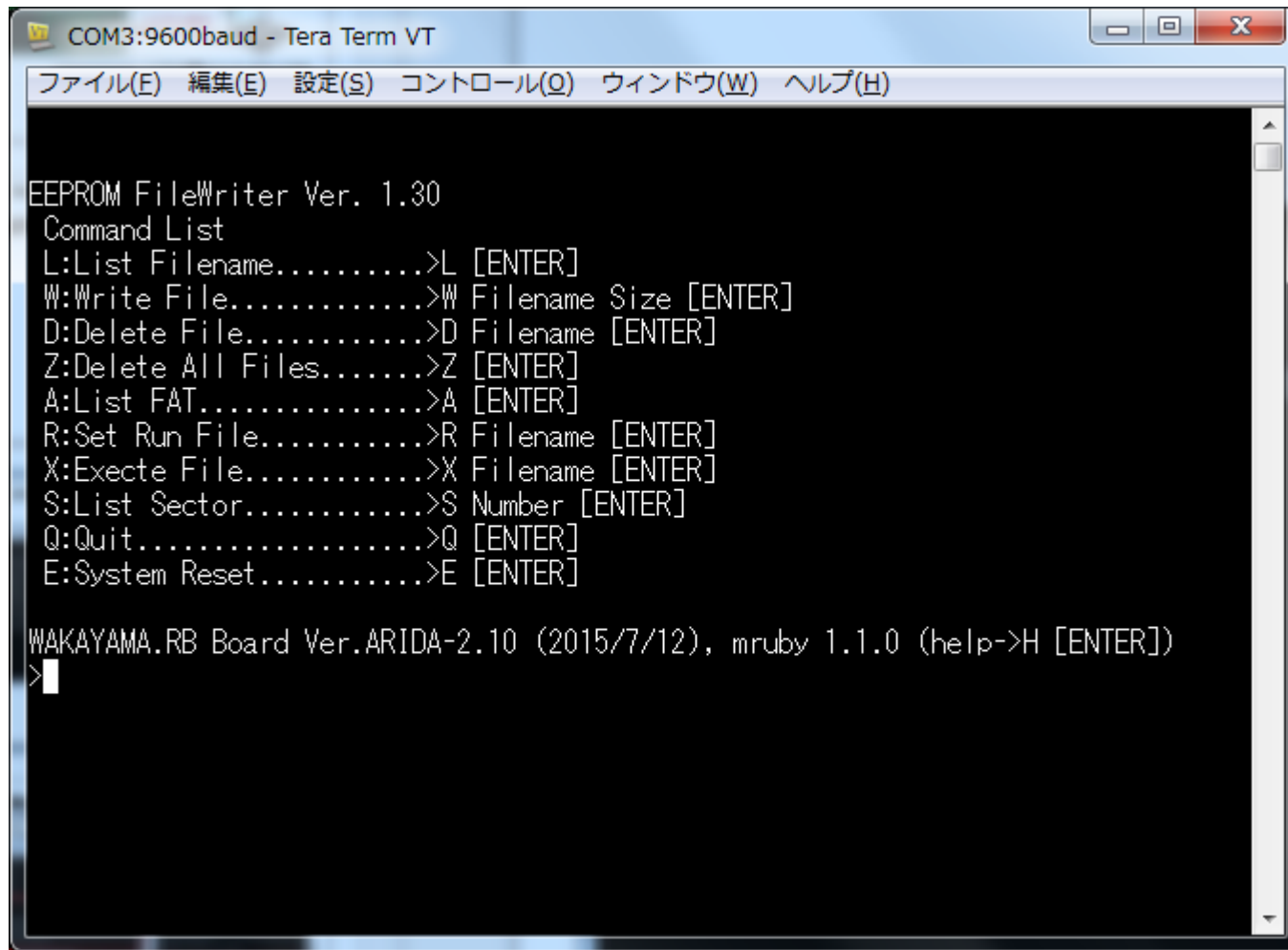


シリアル通信には、ターミナルソフトを使います。
代表的なものにTeraTermがあります。

プログラムの書き込み方法

ターミナルソフトを用いてUSBからシリアル通信をしてプログラムを書き込みます。
ENTERキーで画面にコマンド一覧が表示されます。

アルファベット1文字のコマンドを持っています。



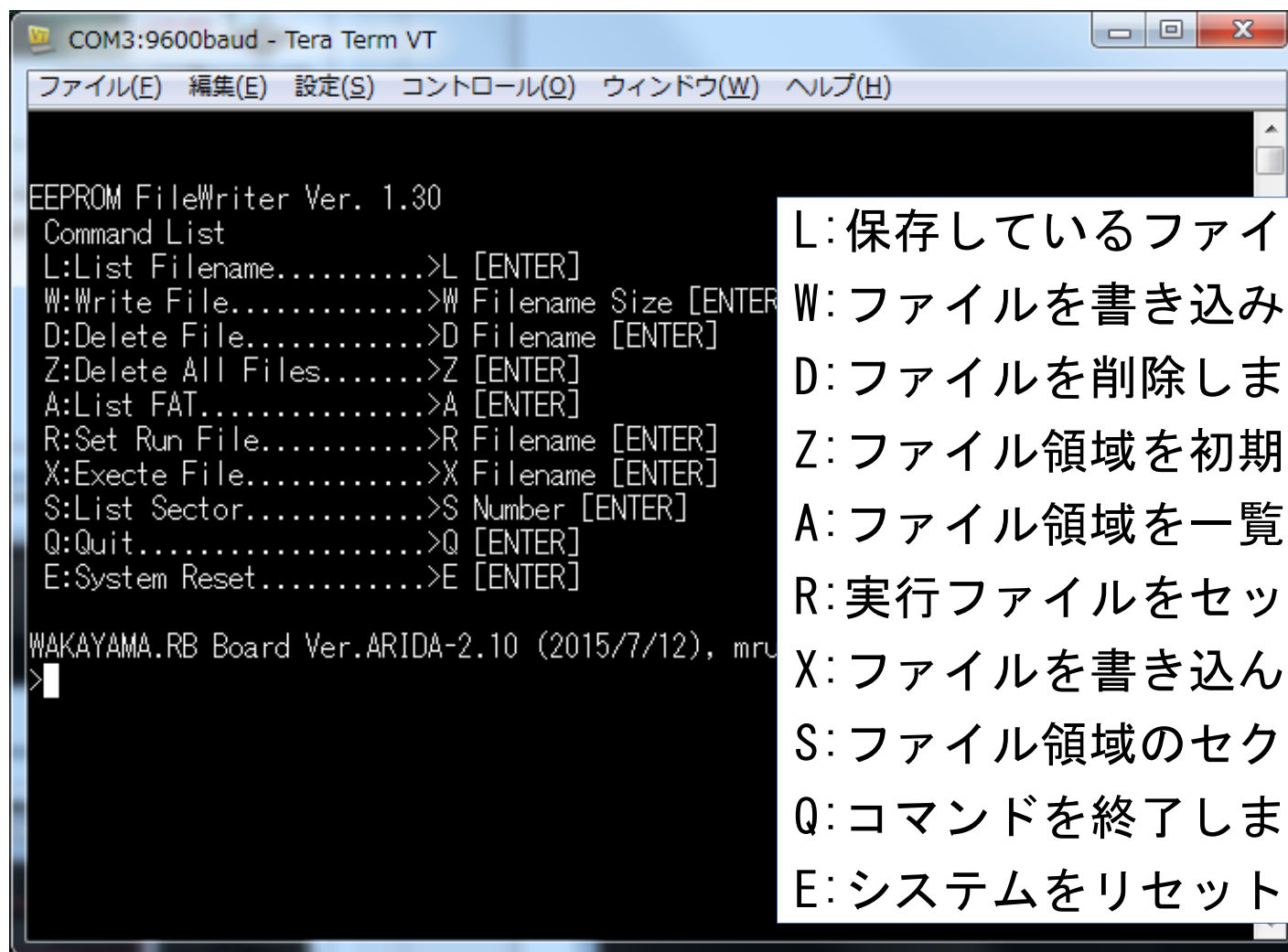
```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)

EEPROM FileWriter Ver. 1.30
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
D:Delete File.....>D Filename [ENTER]
Z:Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>
```

WRBボードの起動画面

コマンドの種類



L:保存しているファイルを一覧します。

W:ファイルを書き込みます。

D:ファイルを削除します。

Z:ファイル領域を初期化します。

A:ファイル領域を一覧します。

R:実行ファイルをセットします。

X:ファイルを書き込んで直ぐ実行します。

S:ファイル領域のセクタ内容を一覧します。

Q:コマンドを終了します。

E:システムをリセットします。

最初に初期化を行う。(Z コマンド)

Aコマンドを用いて、FAT Sector一覧を出した時に、セクタがFFFFで埋めつくされている場合は、初期化を行う必要が有ります。

Zコマンドを用いて、ファイル領域を初期化してください。

```

COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)
W:Write File.....>W Filename Size [ENTER]
D:Delete File.....>D Filename [ENTER]
Z:Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>A

FAT Sector
00-FFFF 01-FFFF 02-FFFF 03-FFFF 04-FFFF 05-FFFF 06-FFFF 07-FFFF 08-FFFF 09-FFFF
0A-FFFF 0B-FFFF 0C-FFFF 0D-FFFF 0E-FFFF 0F-FFFF 10-FFFF 11-FFFF 12-FFFF 13-FFFF
14-FFFF 15-FFFF 16-FFFF 17-FFFF 18-FFFF 19-FFFF 1A-FFFF 1B-FFFF 1C-FFFF 1D-FFFF
1E-FFFF 1F-FFFF 20-FFFF 21-FFFF 22-FFFF 23-FFFF 24-FFFF 25-FFFF 26-FFFF 27-FFFF
28-FFFF 29-FFFF 2A-FFFF 2B-FFFF 2C-FFFF 2D-FFFF 2E-FFFF 2F-FFFF 30-FFFF 31-FFFF
32-FFFF 33-FFFF 34-FFFF 35-FFFF 36-FFFF 37-FFFF 38-FFFF 39-FFFF 3A-FFFF 3B-FFFF
3C-FFFF 3D-FFFF 3E-FFFF 3F-FFFF

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>
  
```

```

COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)
0A-FFFF 0B-FFFF 0C-FFFF 0D-FFFF 0E-FFFF 0F-FFFF 10-FFFF 11-FFFF 12-FFFF 13-FFFF
14-FFFF 15-FFFF 16-FFFF 17-FFFF 18-FFFF 19-FFFF 1A-FFFF 1B-FFFF 1C-FFFF 1D-FFFF
1E-FFFF 1F-FFFF 20-FFFF 21-FFFF 22-FFFF 23-FFFF 24-FFFF 25-FFFF 26-FFFF 27-FFFF
28-FFFF 29-FFFF 2A-FFFF 2B-FFFF 2C-FFFF 2D-FFFF 2E-FFFF 2F-FFFF 30-FFFF 31-FFFF
32-FFFF 33-FFFF 34-FFFF 35-FFFF 36-FFFF 37-FFFF 38-FFFF 39-FFFF 3A-FFFF 3B-FFFF
3C-FFFF 3D-FFFF 3E-FFFF 3F-FFFF

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>Z

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>A

FAT Sector
00-0000 01-0000 02-0000 03-0000 04-0000 05-0000 06-0000 07-0000 08-0000 09-0000
0A-0000 0B-0000 0C-0000 0D-0000 0E-0000 0F-0000 10-0000 11-0000 12-0000 13-0000
14-0000 15-0000 16-0000 17-0000 18-0000 19-0000 1A-0000 1B-0000 1C-0000 1D-0000
1E-0000 1F-0000 20-0000 21-0000 22-0000 23-0000 24-0000 25-0000 26-0000 27-0000
28-0000 29-0000 2A-0000 2B-0000 2C-0000 2D-0000 2E-0000 2F-0000 30-0000 31-0000
32-0000 33-0000 34-0000 35-0000 36-0000 37-0000 38-0000 39-0000 3A-0000 3B-0000
3C-0000 3D-0000 3E-0000 3F-0000

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>
  
```

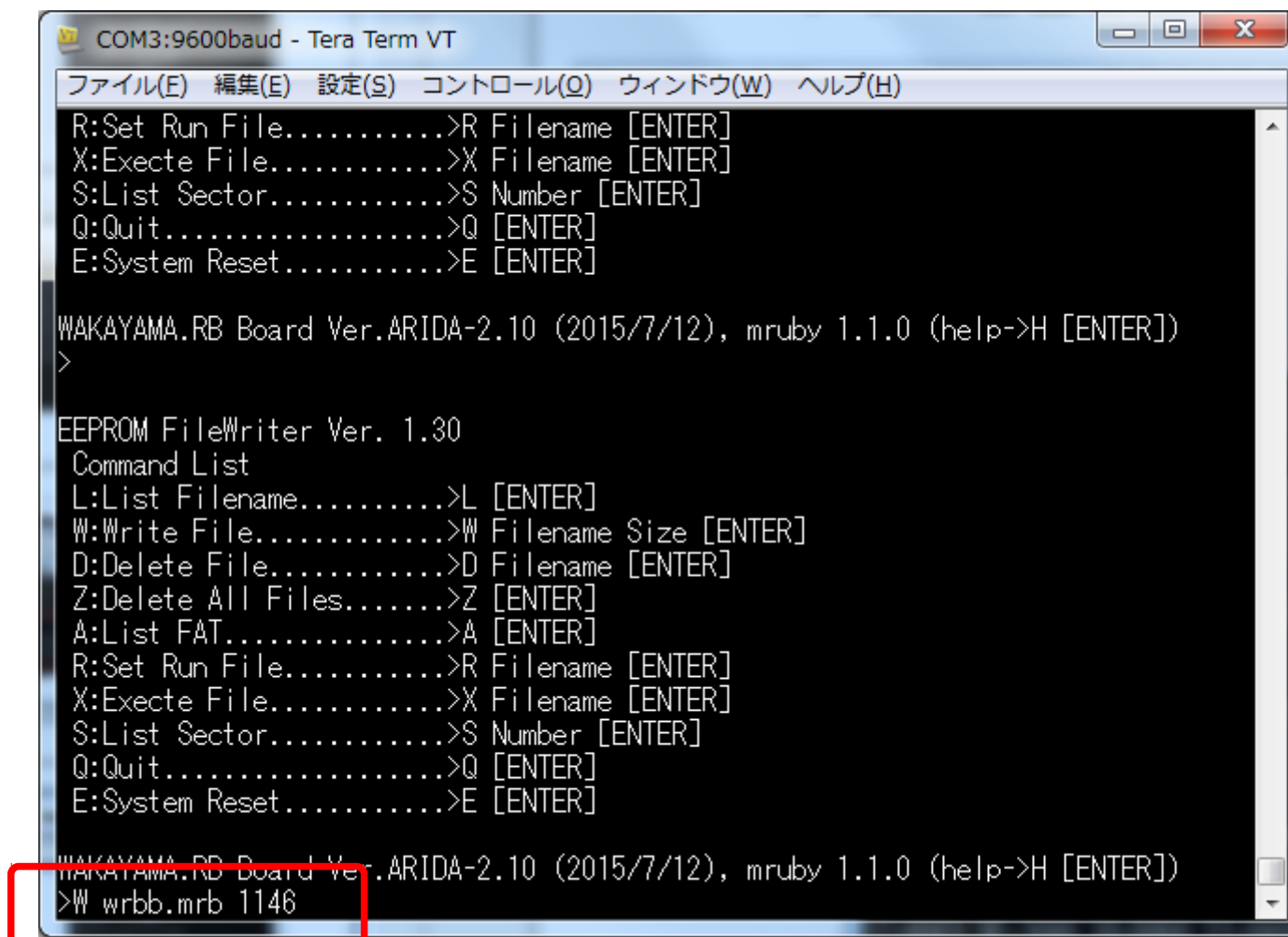
Zコマンドを打って、FFFFが0000に初期化されました。
これで、ファイル領域が使用できるようになります。

プログラムを書き込みます。(W コマンド)

Wコマンドを用いて、mrbファイルを書き込みます。

Wの後にスペースで区切って、ファイル名とファイルサイズを書き、ENTERキーを押します。

>W ファイル名 ファイルサイズ



```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

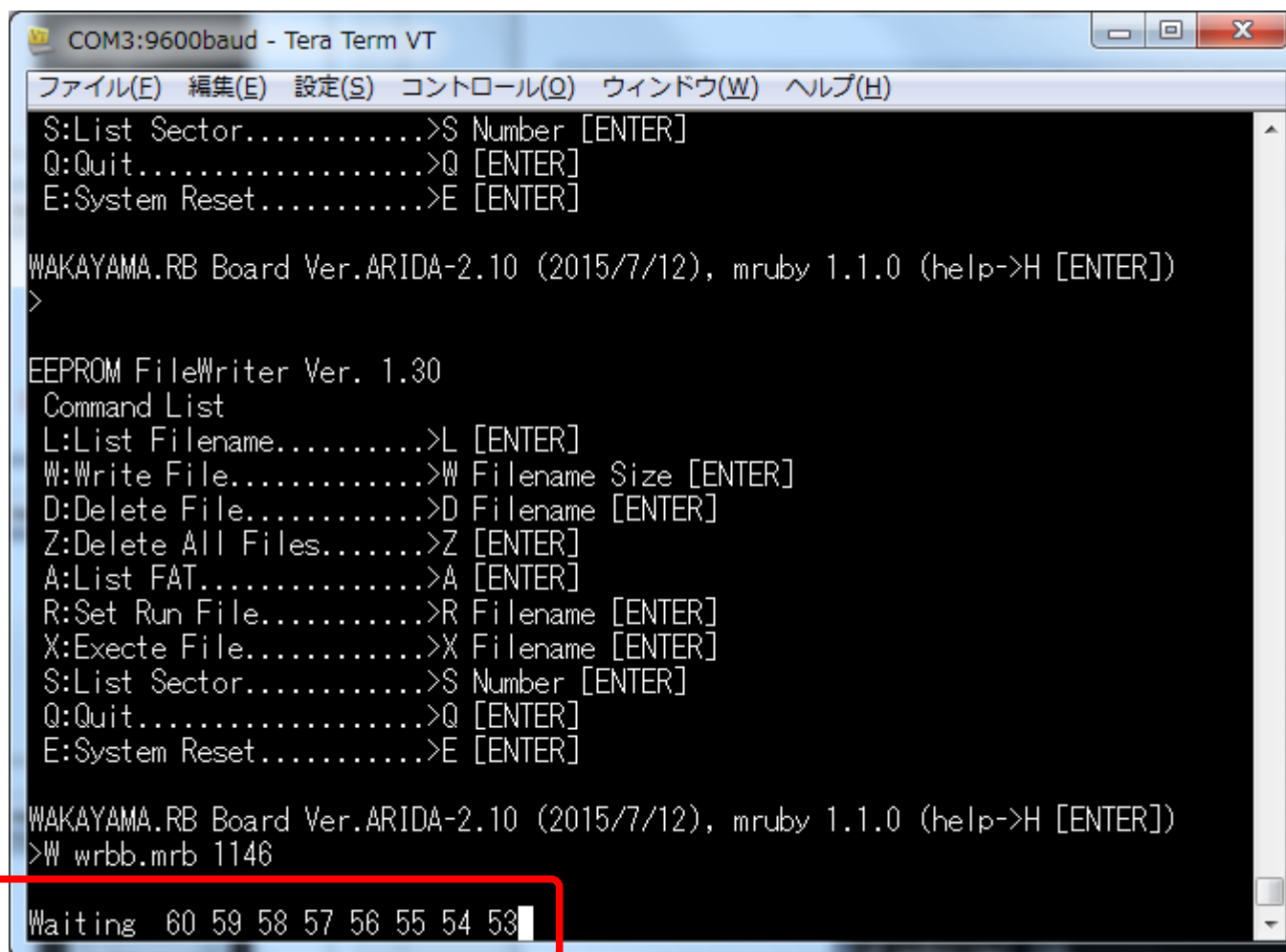
WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>

EEPROM FileWriter Ver. 1.30
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>W wrbb.mrb 1146
```

プログラムを書き込みます。(W コマンド)

ENTERキーを押すと、カウントダウンが始まります。60sec以内にファイルをバイナリ送信してください。



```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>

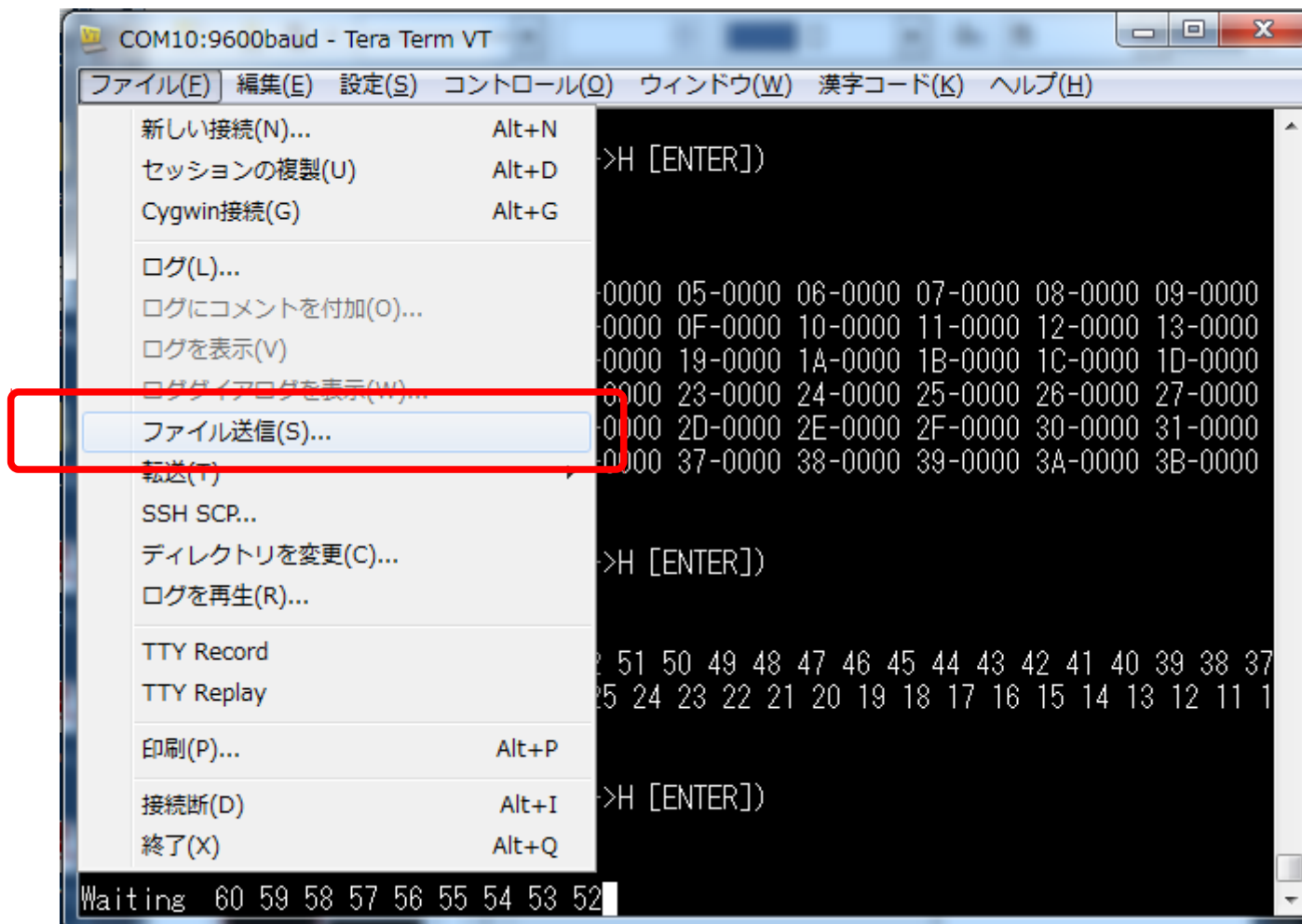
EEPROM FileWriter Ver. 1.30
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>W wrbb.mrb 1146

Waiting 60 59 58 57 56 55 54 53
```

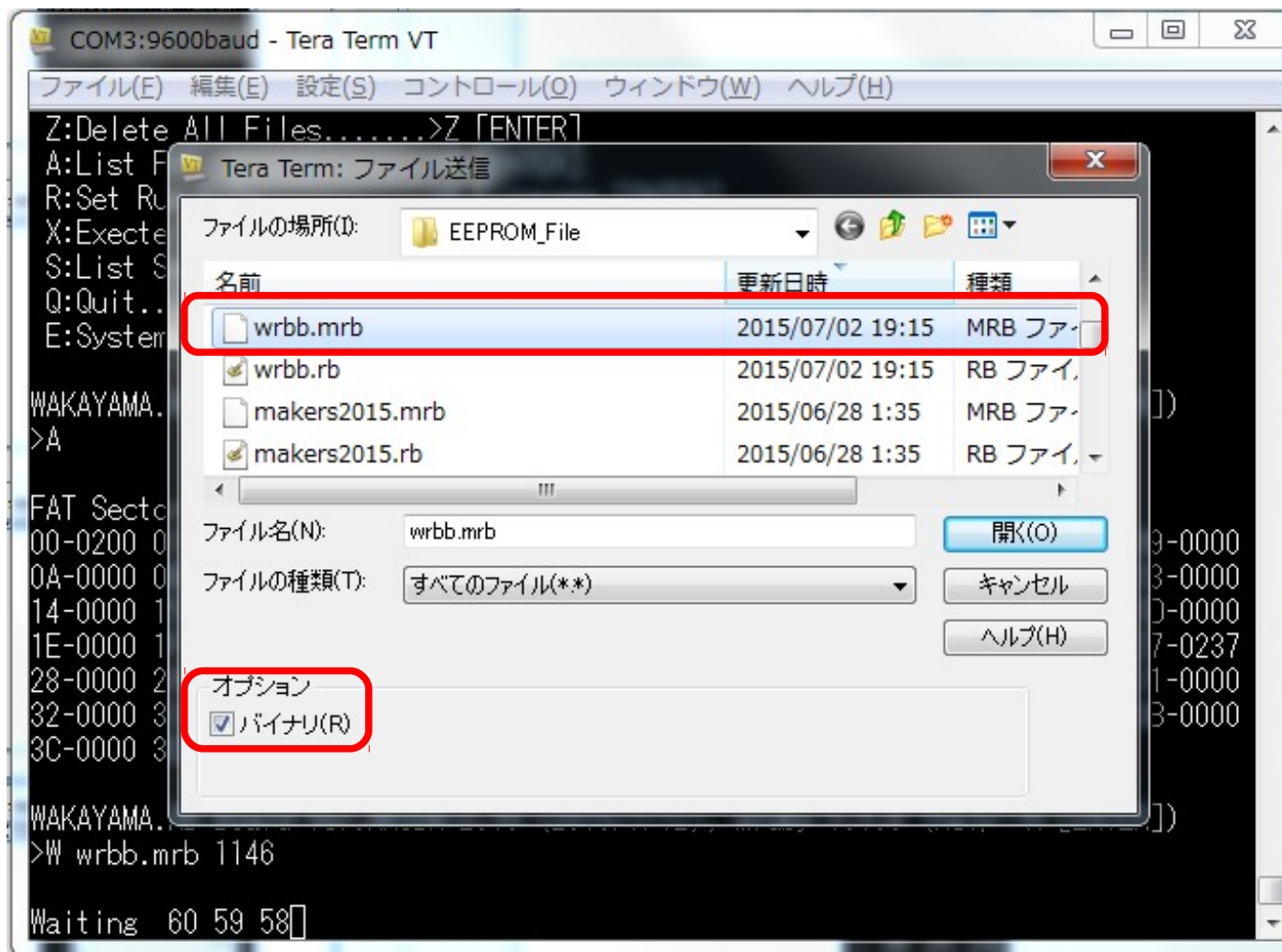
プログラムを書き込みます。(W コマンド)

Tera Termの場合、ファイル→ファイル送信 を選択します。



プログラムを書き込みます。(W コマンド)

Tera Termの場合、オプションのバイナリにチェックを入れます。
その後、送信するファイルを選択して、開く を押します。



プログラムを書き込みます。(W コマンド)

ファイルの書き込みが終了すると、コマンド入力待ちに戻ります。

```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>A

FAT Sector
00-0000 01-0000 02-0000 03-0000 04-0000 05-0000 06-0000 07-0000 08-0000 09-0000
0A-0000 0B-0000 0C-0000 0D-0000 0E-0000 0F-0000 10-0000 11-0000 12-0000 13-0000
14-0000 15-0000 16-0000 17-0000 18-0000 19-0000 1A-0000 1B-0000 1C-0000 1D-0000
1E-0000 1F-0127 20-0000 21-0000 22-0000 23-0000 24-0000 25-0000 26-0000 27-0237
28-0000 29-0000 2A-0000 2B-0000 2C-0000 2D-0000 2E-0000 2F-0000 30-0000 31-0000
32-0000 33-0000 34-0000 35-0000 36-0000 37-0237 38-0000 39-0000 3A-0000 3B-0000
3C-0000 3D-0000 3E-0000 3F-0000

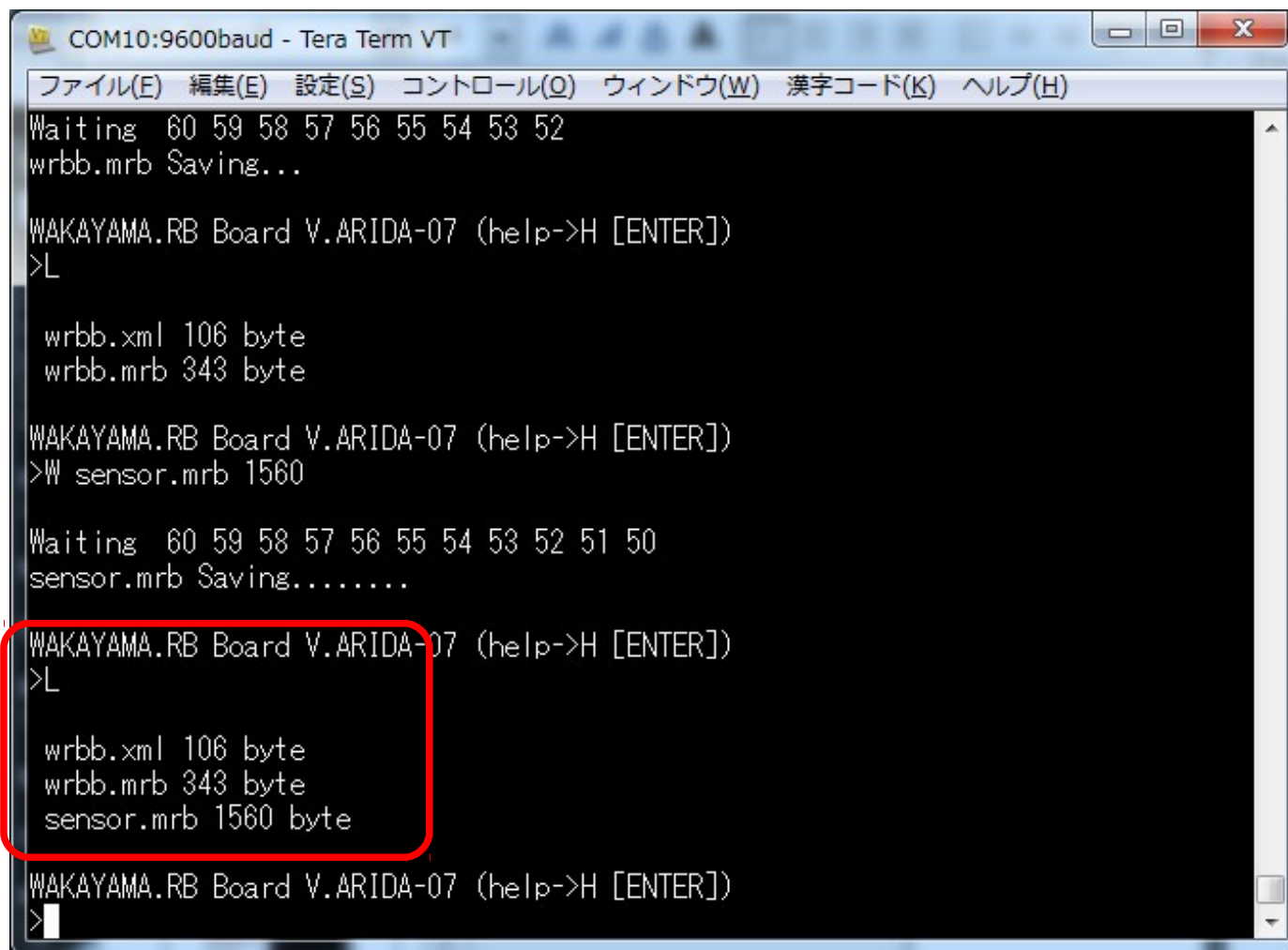
WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>W wrbb.mrb 1146

Waiting 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37
36 35 34 33 32 31 30 29 28
wrbb.mrb Saving.....

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>
```

ファイルを一覧します。(L コマンド)

Lコマンドを入力すると保存されているファイルの一覧が表示されます。



The screenshot shows a Tera Term VT window titled "COM10:9600baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(O)", "ウィンドウ(W)", "漢字コード(K)", and "ヘルプ(H)". The terminal text is as follows:

```
Waiting 60 59 58 57 56 55 54 53 52
wrbb.mrb Saving...

WAKAYAMA.RB Board V.ARIDA-07 (help->H [ENTER])
>L

wrbb.xml 106 byte
wrbb.mrb 343 byte

WAKAYAMA.RB Board V.ARIDA-07 (help->H [ENTER])
>W sensor.mrb 1560

Waiting 60 59 58 57 56 55 54 53 52 51 50
sensor.mrb Saving.....

WAKAYAMA.RB Board V.ARIDA-07 (help->H [ENTER])
>L

wrbb.xml 106 byte
wrbb.mrb 343 byte
sensor.mrb 1560 byte

WAKAYAMA.RB Board V.ARIDA-07 (help->H [ENTER])
>
```

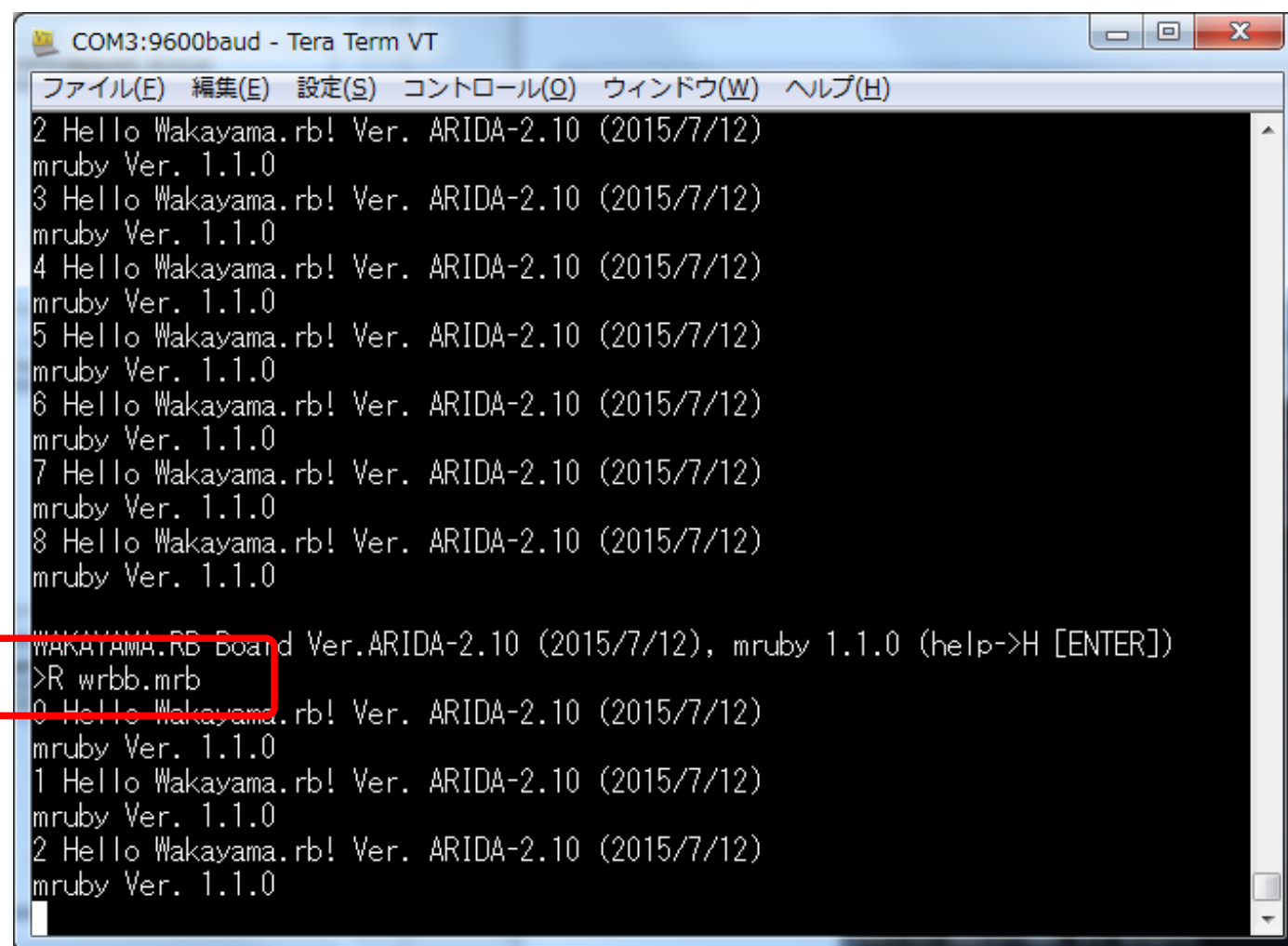
The output of the L command, listing the files wrbb.xml (106 byte), wrbb.mrb (343 byte), and sensor.mrb (1560 byte), is highlighted with a red rectangle.

mrbbファイルを実行します。(R コマンド)

Rコマンドは、mrbbファイルを実行することができます。

Rの後にスペースで区切って、実行させたいファイル名を書き、ENTERを押します。

>R ファイル名.mrb



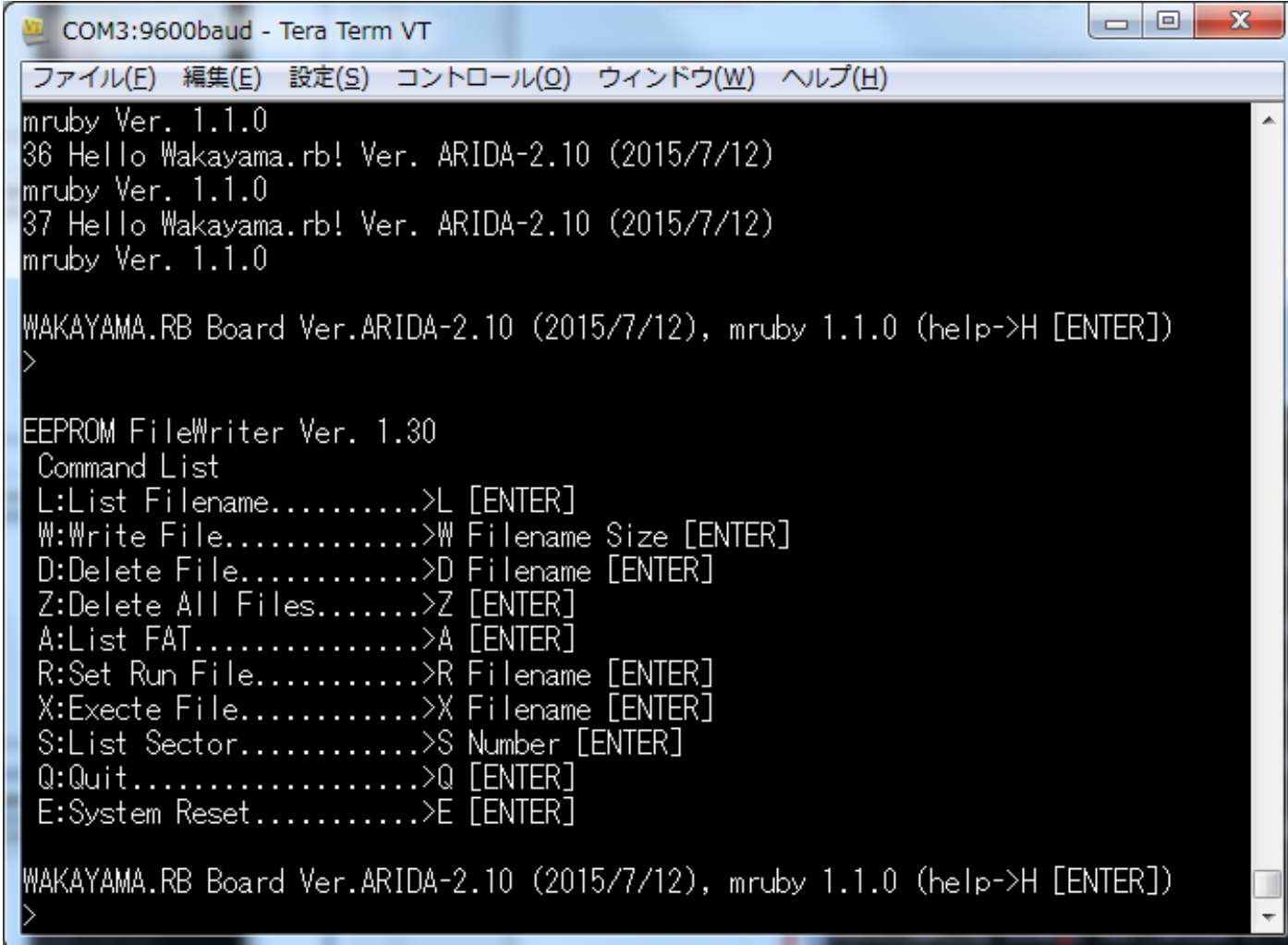
The screenshot shows a Tera Term VT window titled "COM3:9600baud - Tera Term VT". The window has a menu bar with options: ファイル(F), 編集(E), 設定(S), コントロール(C), ウィンドウ(W), ヘルプ(H). The main text area displays a series of messages from a program named "Wakayama.rb". The messages are: "2 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "3 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "4 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "5 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "6 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "7 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "8 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0". Below these, a message is highlighted with a red box: "WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])". Below this, the command ">R wrbb.mrb" is entered. The final messages are: "0 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "1 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0", "2 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)", "mruby Ver. 1.1.0".

```
COM3:9600baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
2 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
3 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
4 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
5 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
6 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
7 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
8 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>R wrbb.mrb
0 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
1 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
2 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
```

mrubyファイルを実行します。(R コマンド)

実行が終了すると、後述する「電源オンで即実行するモード」と同じ条件で次に実行するプログラムを検索して、mrubyファイルを実行します。

もし、条件が見つからなければ、コマンドモードに戻ります。



```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
mruby Ver. 1.1.0
36 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
37 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>

EEPROM FileWriter Ver. 1.30
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>
```

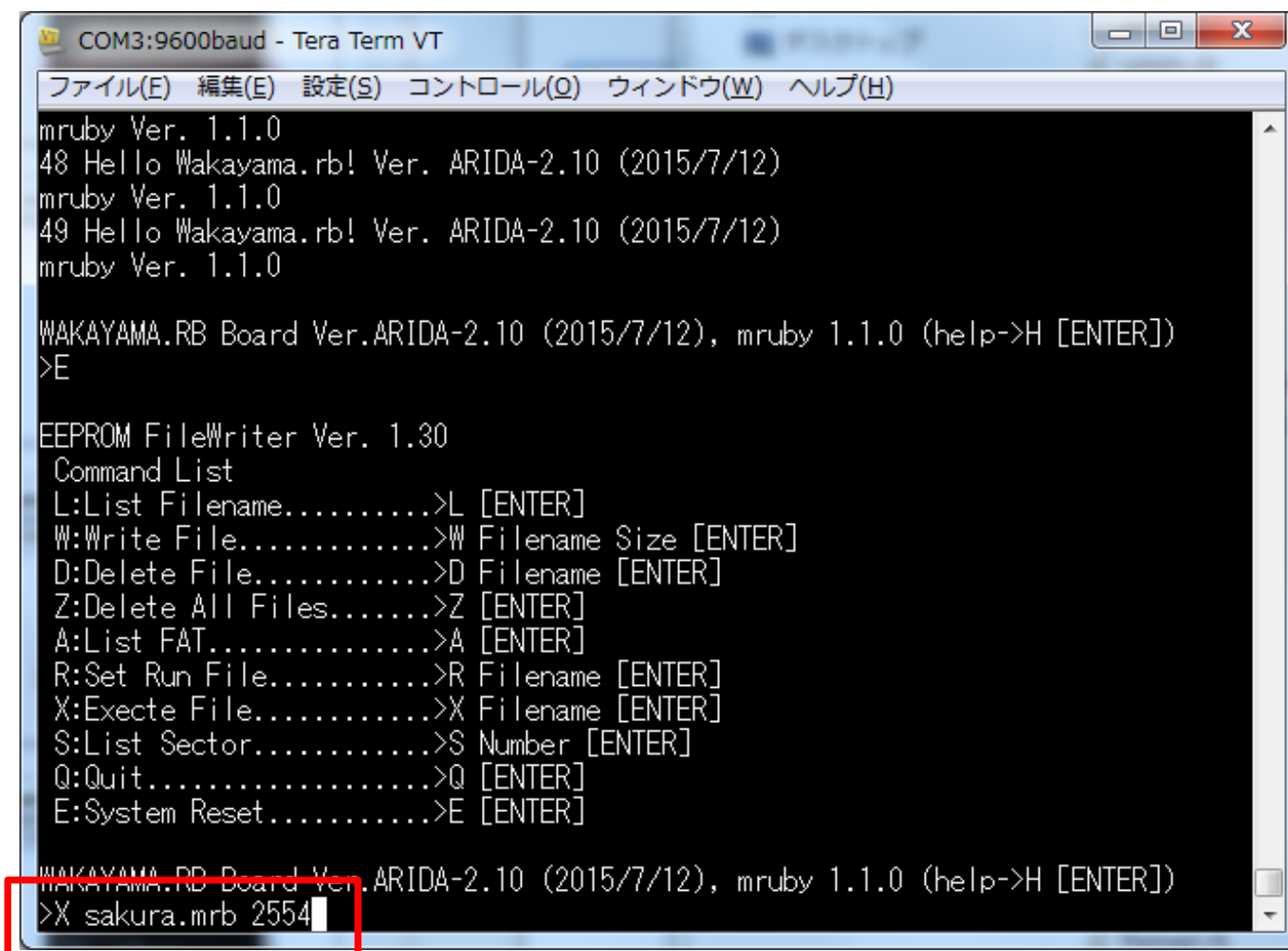
プログラムを書き込み実行します。(X コマンド)

Xコマンドを用いて、mrbファイルを書き込み後直ぐ実行します。

Xの後にスペースで区切って、ファイル名とファイルサイズを書き、ENTERキーを押します。

>X ファイル名 ファイルサイズ

あとは、Wコマンドと同様です。プログラムの書き込みが終了後、直ぐに実行されます。



The screenshot shows a Tera Term VT window titled "COM3:9600baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(Q)", "ウィンドウ(W)", and "ヘルプ(H)". The terminal output shows the following sequence of commands and responses:

```
mruby Ver. 1.1.0
48 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
49 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>E

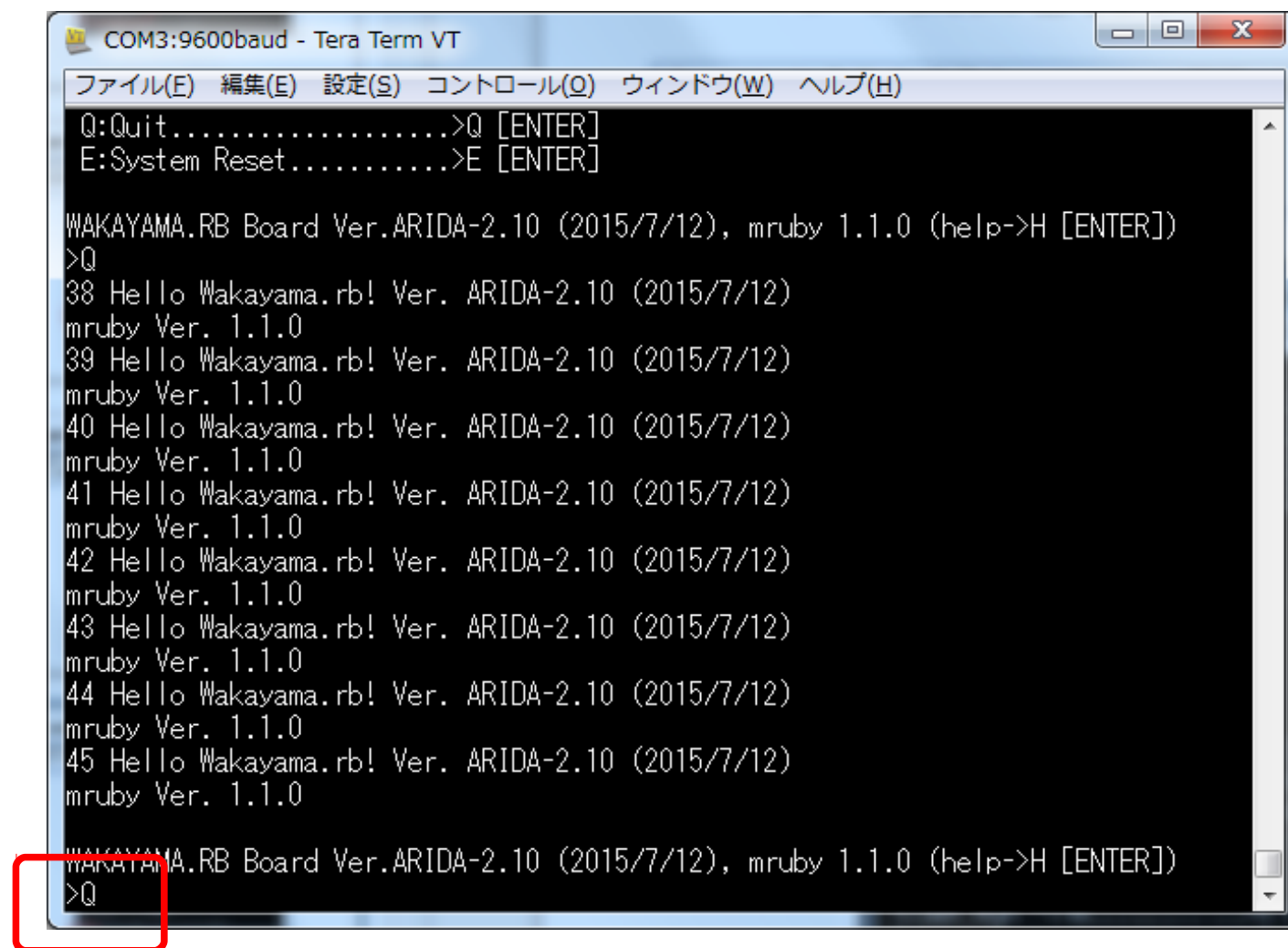
EEPROM FileWriter Ver. 1.30
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
D:Delete File.....>D Filename [ENTER]
Z:Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Set Run File.....>R Filename [ENTER]
X:Execte File.....>X Filename [ENTER]
S:List Sector.....>S Number [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>X sakura.mrb 2554
```

The last command, ">X sakura.mrb 2554", is highlighted with a red rectangle.

コマンド画面を終了する。(Q コマンド)

Qコマンドを入力すると、コマンド画面が終了します。プログラムの途中で呼び出されている場合は、元のプログラムに戻ります。

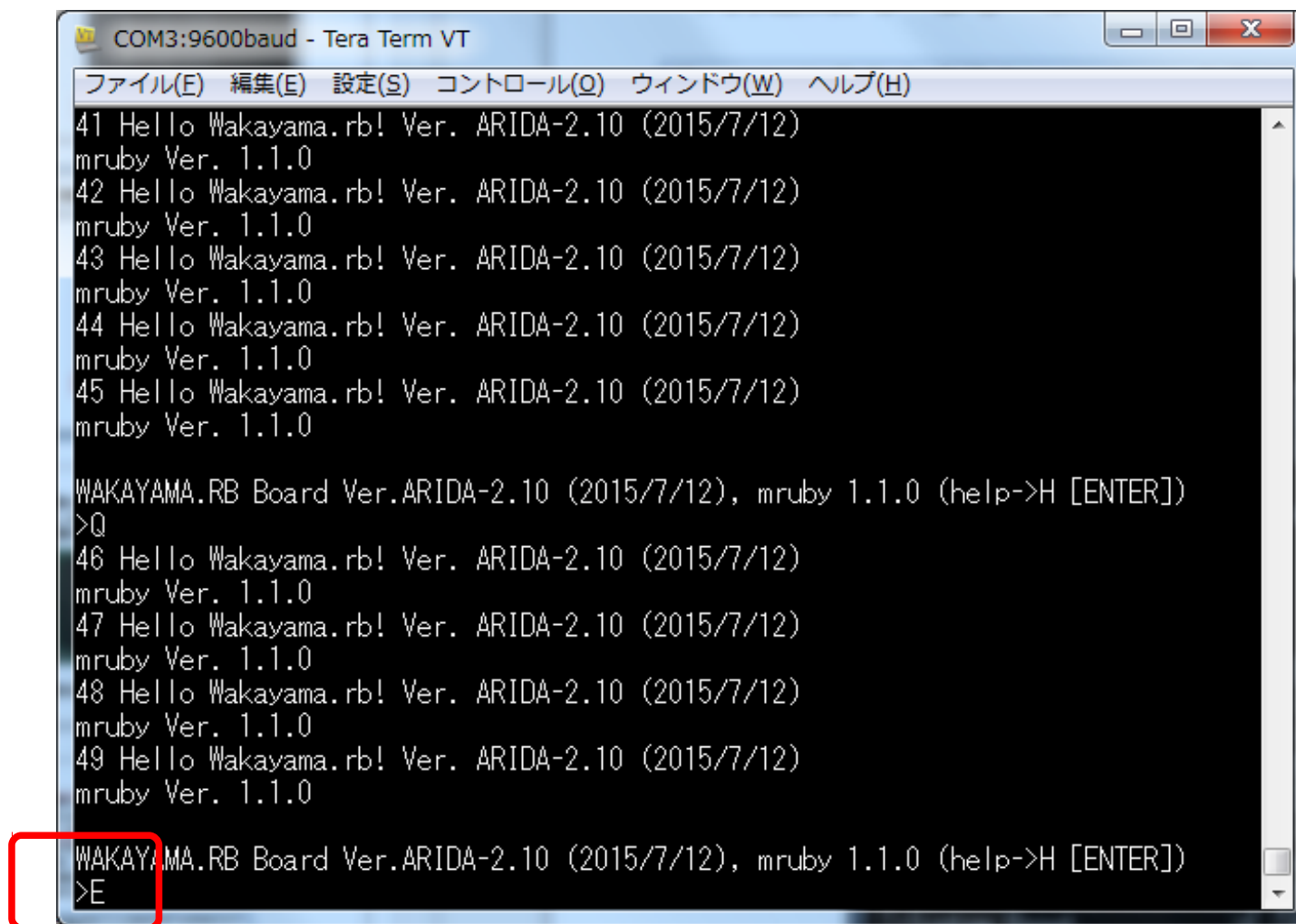


```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]

WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>Q
38 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
39 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
40 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
41 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
42 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
43 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
44 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
45 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>Q
```

再起動する。(E コマンド)

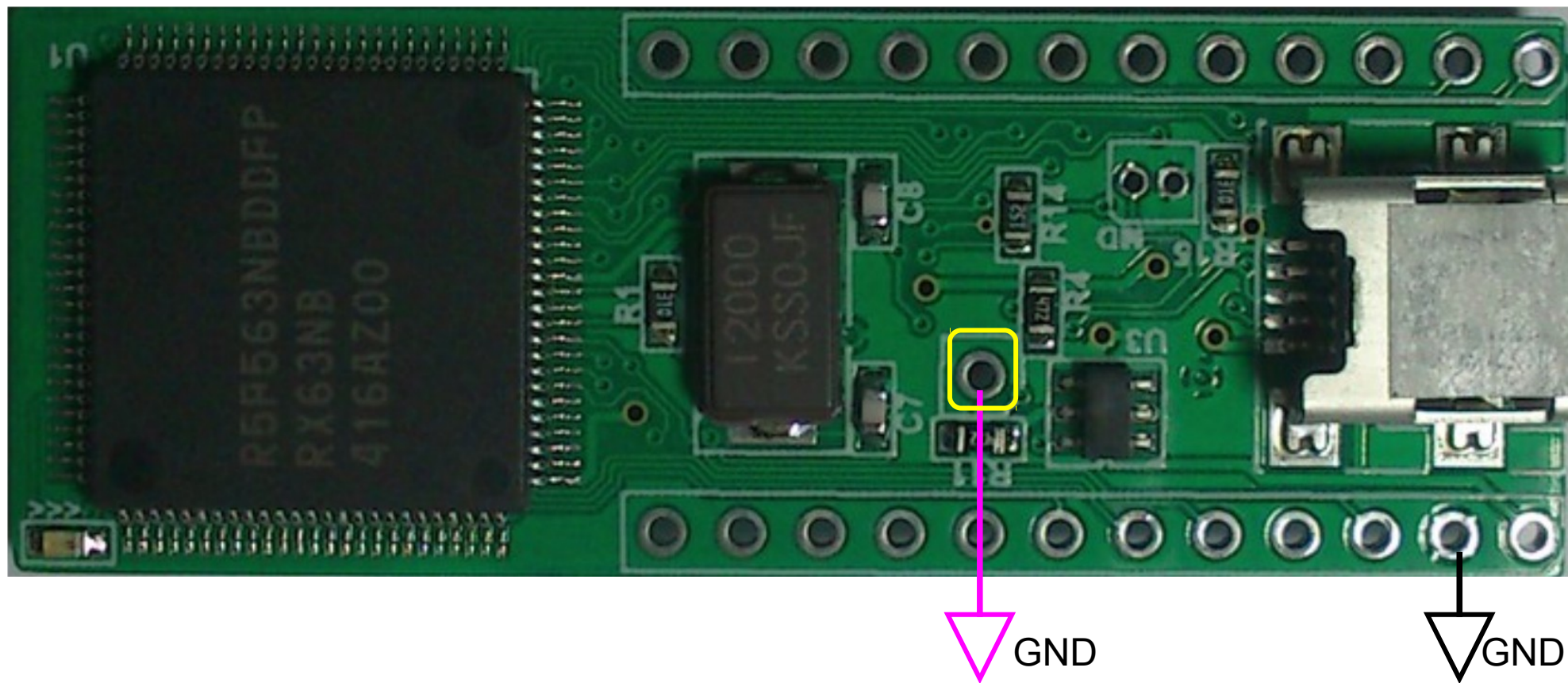
Eコマンドを入力すると、マイコンを再起動します。



```
COM3:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(Q) ウィンドウ(W) ヘルプ(H)
41 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
42 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
43 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
44 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
45 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>Q
46 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
47 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
48 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
49 Hello Wakayama.rb! Ver. ARIDA-2.10 (2015/7/12)
mruby Ver. 1.1.0
WAKAYAMA.RB Board Ver.ARIDA-2.10 (2015/7/12), mruby 1.1.0 (help->H [ENTER])
>E
```


電源ONで即実行する方法

電源をONすると、直ぐにコマンドモードが起動します。改行を入力するとメニューが表示されます。電源ONで即プログラムを実行したい場合は下記の場所をGNDに落としてください。



印の部分をつなぐと、電源ONでプログラムが走ります。

rubyプログラム実行の仕組み

自動実行する場合のrubyプログラム実行条件
条件(1)

WRBBは、先ず wrbb.xml ファイルを検索します。wrbb.xmlとはXML形式で書かれたファイルです。

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Config>
  <Start file="wrbb.mrb" />
</Config>
```

Startタグのfile要素に実行するmrbファイル名を書いておくと、そのプログラムを実行します。

rubyプログラム実行の仕組み

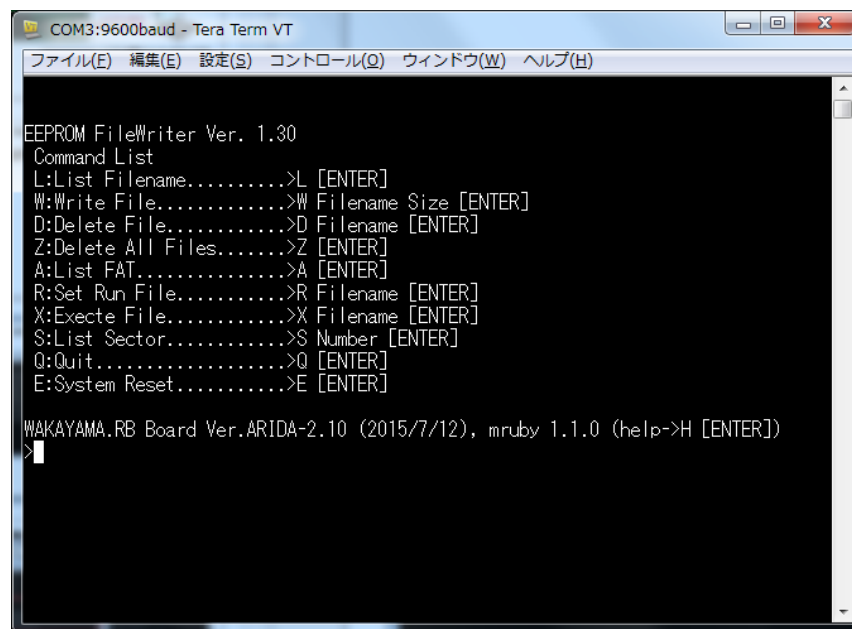
条件(2)

wrbb.xml ファイルが見つからない場合は、wrbb.mrbファイルを検索します。

wrbb.mrb ファイルが見つければ、wrbb.mrbファイルを実行します。

条件(3)

wrbb.xml、wrbb.mrb 両方のファイルが見つからない場合は、USB接続先にコマンド画面を表示します。



*ruby*プログラム実行の仕組み

rubyプログラム例

LEDを5回 ON/OFFさせます。

```
sw = 1
10.times do
  led(sw)
  sw = 1 - sw
  delay(500)
end
```

以下のように書いても同じです。

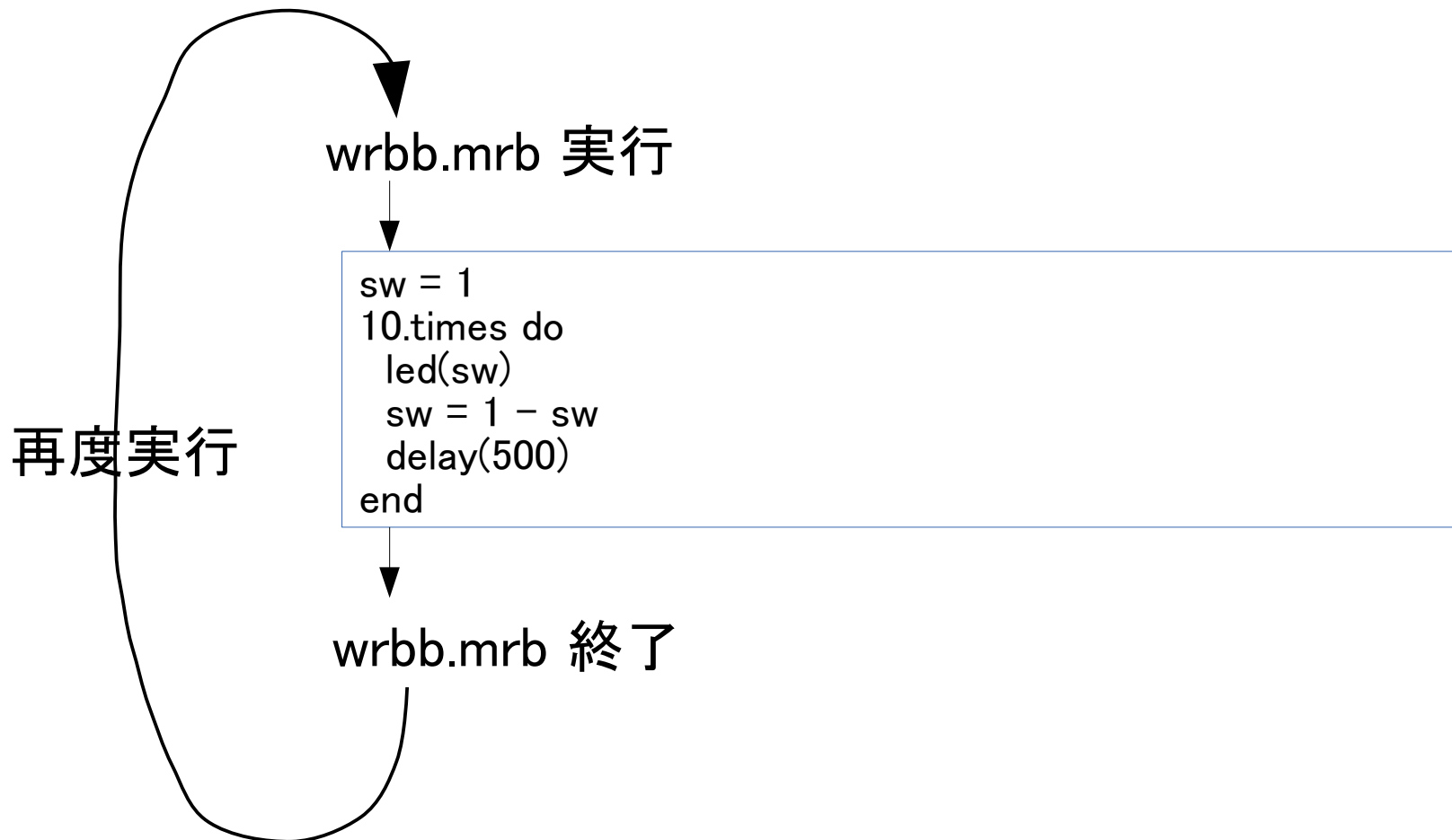
```
sw = 1
for i in 1..10 do
  led(sw)
  sw = 1 - sw
  delay(500)
end
```

Hello WAKAYAMA.RB Board!と10回出力されます。

```
10.times do
  Serial.println(0, "Hello WAKAYAMA.RB Board!")
  delay(500)
end
```

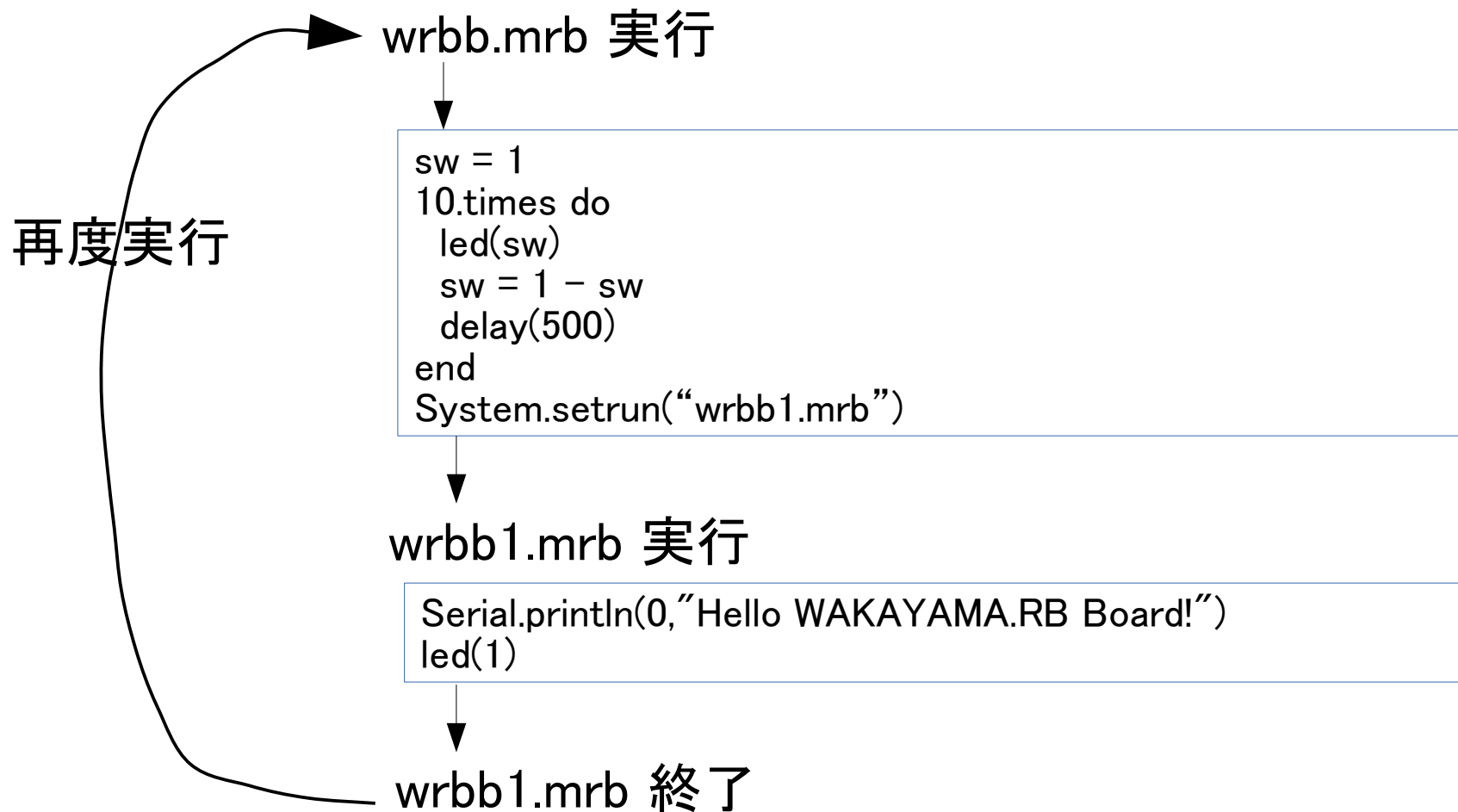
rubyプログラム実行の仕組み

rubyプログラム例を実行すると、永遠にLED点滅を繰り返す場合があります。それは、プログラムが終了した後、再び、rubyプログラムが呼び出されているからです。



rubyプログラム実行の仕組み

rubyプログラム中に `System.setrun` 命令を用いて、次に呼び出すrubyプログラムを指定しておくと、実行が終了後、`System.setrun` されたrubyプログラムが呼び出されます。



mrbcファイルの作成方法

Web-mrbcコンパイラを用いて、Rubyプログラムからmrbcファイルを生成できる予定ですが、まだできません。

コマンドラインからmrbcを実行してください。

```
$ ./mrbc wrbb.rb
```

```
$ ls -l wrbb.mrb
```

```
----rwx---+ 1 minao None 865 6月 26 23:29 wrbb.mrb
```

プログラムをデバッグしたい場合は、コンパイルオプションに `-g` を付けてコンパイルすることをお勧めします。エラーの行番号など詳しいエラーメッセージが出力されます。

```
$ ./mrbc -g wrbb.rb
```

```
$ ./mrbc -h
Usage: ./mrbc [switches] programfile
switches:
-c          check syntax only
-o<outfile> place the output into <outfile>
-v          print version number, then turn on verbose mode
-g          produce debugging information
-B<symbol>  binary <symbol> output in C language format
-e          generate little endian iseq data
-E          generate big endian iseq data
--verbose   run at verbose mode
--version   print the version
--copyright print the copyright
```


メソッドの説明

カーネルクラス

PINのモード設定 `pinMode(pin, mode)`

ピンのデジタル入力と出力を設定します。

pin: ピンの番号

mode: 0: INPUTモード

1: OUTPUTモード

デフォルトは入力(INPUT)モードです。

デジタルライト `digitalWrite(pin, value)`

ピンのデジタル出力のHIGH/LOWを設定します。

pin: ピンの番号

value: 0: LOW

1: HIGH

デジタルリード `digitalRead(pin)`

ピンのデジタル入力値を取得します。

pin: ピンの番号

戻り値

0: LOW

1: HIGH

メソッドの説明

カーネルクラス

アナログリード `analogRead(pin)`

ピンのアナログ入力値を取得します。
pin: アナログピンの番号 (14, 15, 16, 17)

戻り値
10ビットの値 (0~1023)

アナログDAC出力 `analogDac(value)`

ピンからアナログ電圧を出力します。
value: 10bit精度 (0~4095) で0~3.3V

LEDオンオフ `led(sw)`

基板のLEDを点灯します。
sw: 0: 消灯
1: 点灯

メソッドの説明

カーネルクラス

PWM出力 `pwm(pin, value)`

ピンのPWM出力値をセットします。

pin: ピンの番号

value: 出力PWM比率(0~255)

PWM周波数設定 `pwmHz(value)`

PWM出力するときの周波数を設定します。

value: 周波数(12~184999)Hz

ディレイ `delay(value)`

指定の時間(ms)動作を止めます。

value: 時間(ms)

※delay中に強制的にGCを行っています。

ミリ秒を取得します `millis()`

システムが稼動してから経過した時間を取得します。

戻り値

起動してからのミリ秒数

メソッドの説明

カーネルクラス

マイクロ秒を取得します `micros()`

システムが稼動してから経過した時間を取得します。

戻り値

起動してからのマイクロ秒数

メソッドの説明

カーネルクラス

使用例

```
pinMode(4, 0)
pinMode(5, 1)

x = digitalRead(4)
digitalWrite(5, 0)

10.times do
  led(1)
  delay(1000)
  led(0)
  delay(1000)
end
```

メソッドの説明

システムクラス

システムのバージョン取得 `System.version([R])`

システムのバージョンを取得します。
R: 引数があればmrubyのバージョンを返します。

プログラムの終了 `System.exit()`

プログラムを終了させます。
`System.setRun`により次に実行するプログラムがセットされていれば、そのプログラムが実行されます。

実行するプログラムの設定 `System.setRun(filename)`

次に実行するプログラムを設定します。
filename: mrbファイル名

管理カーネルの呼び出し `System.fileload()`

システムの管理カーネル「EEPROM File Writer」を呼び出します。

メソッドの説明

システムクラス

フラッシュメモリに書き込み System.push(address, buf, length)

フラッシュメモリに値を書き込みます。

address: 書き込み開始アドレス (0x0000~0x00ff)

buf: 書き込むデータ

length: 書き込むサイズ (MAX 32バイト)

戻り値

1: 成功

0: 失敗

※ここに書き込んだ値は、電源を切っても消えません。

フラッシュメモリから読み出し System.pop(address, length)

フラッシュメモリから値を読み出します。

address: 読み込みアドレス (0x0000~0x00ff)

length: 読み込みサイズ (MAX 32バイト)

戻り値

読み込んだデータ分

メソッドの説明

システムクラス

使用例

#アドレス0x0000から0x0005に{0x3a, 0x39, 0x38, 0x00, 0x36}の5バイトのデータを書き込みます
buf = 0x3a.chr+0x39.chr+0x38.chr+0x0.chr+0x36.chr

System.push(0x0000, buf, 5)

#アドレス0x0000から5バイトのデータを読み込みます
ans = System.pop(0x0000, 5)

System.setRun(' sample.mrb') #次に実行するプログラム名をセットします

System.exit() #このプログラムを終了します。

メソッドの説明

シリアルクラス

シリアル通信の初期化 `Serial.begin(num, bps)`

シリアル通信を初期化します。シリアル通信を私用する場合は、初めに初期化を行ってください。

num: 初期化する通信番号
0: USB
1: 0ピン送信/1ピン受信
2: 5ピン送信/6ピン受信
3: 7ピン送信/8ピン受信

bps: ボーレート (bps) 基本的に任意の値が設定できます。

シリアル通信のデフォルト通信番号の設定 `Serial.setDefault(num)`

シリアル通信のデフォルト通信番号を設定します。

num: 通信番号 (番号はSerial.begin参照)

※システム内部でエラーなどが発生した時に、出力されるメッセージの出力先となります。

シリアルポートへの出力 `Serial.print(num[, str])`

シリアルポートに出力します。

num: 通信番号 (番号はSerial.begin参照)
str: 文字列。省略時は何も出力しません設定できます。

シリアルポートへの出力 (¥r¥n付き) `Serial.println(num[, str])`

シリアルポートに¥r¥n付きで出力します。

num: 通信番号 (番号はSerial.begin参照)
str: 文字列。省略時は改行のみ

メソッドの説明

シリアルクラス

シリアル受信チェック `Serial.available(num)`

シリアルポートに受信データがあるかどうか調べます。
num: 通信番号(番号はSerial.begin参照)

戻り値

シリアルバッファにあるデータのバイト数。0の場合はデータなし。

シリアルポートから1バイト取得 `Serial.read(num)`

シリアルポートの受信データを1バイト取得します。
num: 通信番号(番号はSerial.begin参照)

戻り値

0x00~0xFFの値、データが無いときは-1が返ります。

シリアルポートへデータ出力 `Serial.write(num, buf, len)`

シリアルポートにデータを出力します。
num: 通信番号(番号はSerial.begin参照)
buf: 出力データ
len: 出力データサイズ

戻り値

出力したバイト数

メソッドの説明

シリアルクラス

シリアルポートを閉じます `Serial.end(num)`

シリアルポートを閉じます。
num: 通信番号(番号はSerial.begin参照)

メソッドの説明

シリアルクラス

使用例

```
Serial.begin(0, 115200)      #USBシリアル通信の初期化
```

```
Sw = 0
```

```
while(true) do  
  while(Serial.available(0) > 0) do #何か受信があった  
    c = Serial.read(0).chr          #1文字取得  
    Serial.print(0, c)              #エコーバック  
  end  
end
```

```
#LEDを点滅させます
```

```
led(Sw)
```

```
Sw = 1 - Sw
```

```
if(Serial.available(0) > 0) then  
  System.fileload()  
end
```

```
delay(500)  
end
```

```
Serial.begin(1, 115200)      #0ピンと1ピンのシリアル通信初期化
```

```
data = 0x30.chr + 0x31.chr + 0.chr + 0x32.chr + 0x33.chr + 0x0d.chr + 0x0a.chr  
Serial.write(1, data, 7 )    #1番ポートに7バイトのデータを出力
```

```
System.exit()
```

メソッドの説明

MemFileクラス (Flashメモリをメディアのように扱うクラス)

ファイルのオープン `MemFile.open(number, filename[, mode])`

ファイルをオープンします。

number: ファイル番号 0 または 1

filename: ファイル名 (8.3形式)

mode: 0:Read, 1:Append, 2:New Create

戻り値

成功: 番号, 失敗: -1

※同時に開けるファイルは2つまでに限定しています。

ファイルのクローズ `MemFile.close(number)`

ファイルをクローズします。

number: クローズするファイル番号 0 または 1

ファイルの読み出し位置に移動 `MemFile.seek(number, byte)`

Openしたファイルの読み出し位置に移動します。

number: ファイル番号 0 または 1

byte: seekするバイト数 (-1) でファイルの最後に移動する

戻り値

成功: 1, 失敗: 0

メソッドの説明

MemFileクラス (Flashメモリをメディアのように扱うクラス)

Openしたファイルからの読み込み `MemFile.read(number)`

Openしたファイルから1バイト読み込みます。

number: ファイル番号 0 または 1

戻り値

0x00~0xFFが返る。ファイルの最後だったら-1が返る。

Openしたファイルにバイナリデータを書き込む `MemFile.write(number, buf, len)`

Openしたファイルにバイナリデータを書き込みます。

number: ファイル番号 0 または 1

buf: 書き込むデータ

len: 書き込むデータサイズ

戻り値

実際に書いたバイト数

メソッドの説明

MemFileクラス

使用例

```
MemFile.open(0, 'sample.txt', 2)
  MemFile.write(0, 'Hello mruby World', 17)
  data = 0x30.chr + 0x31.chr + 0.chr + 0x32.chr + 0x33.chr
  Serial.write(0, data, 5)
MemFile.close(0)
```

```
Serial.begin(0, 115200)      #USBシリアル通信の初期化
```

```
MemFile.open(0, 'sample.txt', 0)
while(true) do
  c = MemFile.read(0)
  if(c < 0) then
    break
  end
  Serial.write(0, c.chr, 1)
end
MemFile.close(0)
System.exit()
```


メソッドの説明

I2cクラス

I2C通信を行うピンの初期化 `I2c.sdascI(sda, scl)`

I2C通信を行うピンを設定します。

sda: データピン
scl: クロックピン

アドレスにデータを書き込みます `I2c.write(deviceID, address, data)`

アドレスにデータを書き込みます。

deviceID: デバイスID
address: 書き込みアドレス
data: データ

戻り値

- 0: 成功
- 1: 送信バッファ溢れ
- 2: スレーブアドレス送信時にNACKを受信
- 3: データ送信時にNACKを受信
- 4: その他のエラー

アドレスからデータを読み込み `I2c.read(deviceID, addressL[, addressH])`

アドレスからデータを読み込みます。

deviceID: デバイスID
addressL: 読み込み下位アドレス
addressH: 読み込み上位アドレス

戻り値

読み込んだ値

メソッドの説明

I2cクラス

I2Cデバイスに対して送信を開始するための準備をする: `I2c.begin(deviceID)`

I2Cデバイスに対して送信を開始するための準備をします。この関数は送信バッファを初期化するだけで、実際の動作は行わない。繰り返し呼ぶと、送信バッファが先頭に戻る。

deviceID: デバイスID 0~0x7Fまでの純粋なアドレス

デバイスに対してI2Cの送信シーケンスの発行 `I2c.end()`

デバイスに対してI2Cの送信シーケンスを発行します。I2Cの送信はこの関数を実行して初めて実際に行われる。

戻り値

- 0: 成功
- 1: 送信バッファ溢れ
- 2: スレーブアドレス送信時にNACKを受信
- 3: データ送信時にNACKを受信
- 4: その他のエラー

デバイスに受信シーケンスを発行しデータを読み出す `I2c.request(address, count)`

デバイスに対して受信シーケンスを発行しデータを読み出します。

address: 読み込み開始アドレス

count: 読み出す数

戻り値

実際に受信したバイト数

メソッドの説明

I2cクラス

送信バッファの末尾に数値を追加する I2c.lwrite(data)

送信バッファの末尾に数値を追加します。

data: セットする値

戻り値

送信したバイト数(バッファに溜めたバイト数)を返す。

送信バッファ(260バイト)に空き容量が無ければ失敗して0を返す。

デバイスに受信シーケンスを発行しデータを読み出す I2c.lread()

デバイスに対して受信シーケンスを発行しデータを読み出します。

戻り値

読み込んだ値

周波数を変更する I2c.freq(Hz)

周波数を変更します。

Hz: クロックの周波数をHz単位で指定する。

有効な値は1~200000程度。基本的にソフトでやっているなので400kHzは出ない。

メソッドの説明

I2cクラス

使用例

```

@APTemp = 0x5D          # 0b01011101 圧力・温度センサのアドレス
Serial.begin(0, 115200)  # USBシリアル通信の初期化

#センサ接続ピンの初期化(17番SDA, 16番SCL)
I2c.sdascl( 17, 16 )
delay(300)

#気圧と温度センサの初期化
@APTemp = 0x5D          # 0b01011101
APTemp_CTRL_REG1 = 0x20 # Control register
APTemp_SAMPLING = 0xA0  # A0:7Hz, 90:1Hz
# 7Hz
I2c.write(@APTemp, APTemp_CTRL_REG1, APTemp_SAMPLING)
delay(100)

#気圧を取得します -----
#Address 0x28, 0x29, 0x2A, 0x2B, 0x2C
v0 = I2c.read( @APTemp, 0x28, 0x29)
v1 = I2c.read( @APTemp, 0x2A)
a = v0 + v1 * 65536
a = a / 4096.0          # hPa単位に直す
#温度を取得します -----
v2 = I2c.read( @APTemp, 0x2B, 0x2C)
if v2 > 32767
  v2 = v2 - 65536
end
t = v2 / 480.0 + 42.5
Serial.println(0, a.to_s + ", " + t.to_s)

```

メソッドの説明

I2cクラス

使用例

```
Serial.begin(0, 115200)      #USBシリアル通信の初期化
#センサ接続ピンの初期化(17番SDA, 16番SCL)
I2c.sdascl( 17, 16 )
delay(300)
#気圧と温度センサの初期化
@APTemp = 0x5D               # 0b01011101
APTemp_CTRL_REG1 = 0x20      # Control register
APTemp_SAMPLING = 0xA0       # A0:7Hz, 90:1Hz
I2c.write(@APTemp, APTemp_CTRL_REG1, APTemp_SAMPLING)  # 7Hz
delay(100)

#Address 0x2B, 0x2C
I2c.begin(@APTemp)
I2c.lwrite(0x2B)
I2c.end()
I2c.request(@APTemp, 1)
datL = I2c.lread()

I2c.begin(@APTemp)
I2c.lwrite(0x2C)
I2c.end()
I2c.request(@APTemp, 1)
datH = I2c.read()
v = datL + datH * 256
if v > 32767
  v = v - 65536
end
t = v / 480.0 + 42.5
Serial.println(0, t.to_s)
```

メソッドの説明

サーボクラス

サーボ出力を任意のピンに割り当てます `Servo.attach(ch, pin[, min, max])`

ch: サーボのチャンネル 0~9まで指定できます
pin: 割り当てるピン番号
min: サーボの角度が0度のときのパルス幅(マイクロ秒)。デフォルトは544
max: サーボの角度が180度のときのパルス幅(マイクロ秒)。デフォルトは2400

サーボの角度をセットします: `Servo.write(ch, angle)`

ch: サーボのチャンネル 0~9まで指定できます
angle: 角度 0~180バイスに対して受信シーケンスを発行しデータを読み出します。

サーボモータにus単位で角度を指定します: `Servo.us(ch, us)`

ch: サーボのチャンネル 0~9まで指定できます
us: 出力したいパルスの幅 1~19999, 0で出力 OFF
サーボモータに与えられるパルスは20ms周期で、1周期中のHighの時間を直接指定する。
実質的にPWM出力。連続回転タイプのサーボでは、回転のスピードが設定することができる。

最後に設定された角度を読み出します: `Servo.read(ch)`

ch: サーボのチャンネル 0~9まで指定できます

戻り値
マイクロ秒単位。ただし `us(ch)` で与えた値は読みとれません。

メソッドの説明

サーボクラス

ピンにサーボが割り当てられているかを確認します: `Servo.attached(ch)`

ch: サーボのチャネル 0~9まで指定できます

戻り値

- 1: 割り当てられている
- 0: 割り当てはない

サーボの動作を止め、割り込みを禁止します: `Servo.detach(ch)`

ch: サーボのチャネル 0~9まで指定できます

メソッドの説明

サーボクラス

使用例

```
g_pos = 0
g_inc = 10

Serial.begin(0, 115200)      #USBシリアル通信の初期化
#8番ピンをサーボ用ピンに割り当てる。
Servo.attach(0, 8)
Servo.write(0, g_pos) #サーボの角度設定

#サーボを10度ずつ50回動かす
50.times do
  delay(100)
  g_pos = g_pos + g_inc
  Servo.write(0, g_pos)
  if(g_pos >= 180 || g_pos <= 0) then
    g_inc = g_inc * -1
  end
end
end
```


メソッドの説明

リアルタイムクロッククラス

RTCを起動します: `Rtc.begin()`

戻り値

- 0: 起動失敗
- 1: 起動成功
- 2: RTCは既に起動していた

RTCの時計をセットします: `Rtc.setDateTime(Year, Month, Day, Hour, Minute, Second)`

Year: 年 0-99
Month: 月 1-12
Day: 日 0-31
Hour: 時 0-23
Minute: 分 0-59
Second: 秒 0-59

戻り値

- 0: 失敗
- 1: 成功

RTCの時計を取得します: `Rtc.getDateTime()`

戻り値

Year: 年
Month: 月
Day: 日
Hour: 時
Minute: 分
Second: 秒

メソッドの説明

リアルタイムクロッククラス

使用例

```
Serial.begin(0, 115200)      #USBシリアル通信の初期化  
Rtc. setDateTime(2015, 6, 27, 0, 0, 1)  
  
10.times do|i|  
  led(i % 2)  
  year, mon, da, ho, min, sec = Rtc. getDateTime()  
  Serial.println(0, year.to_s + "/" + mon.to_s + "/" + da.to_s + " " + ho.to_s + ":" + min.to_s +  
  ":" + sec.to_s)  
  delay(500)  
end
```

JTAGも頑張ればつながります。

