

WRBB4ボード 搭載 Rubyファーム 説明資料 ver2.0

Wakayama. rb
山本三七男(たろサ)

ハード仕様

MCU

32ビットCPU RX63NBDDFP (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 (micro-B)

LED 1個

I/Oピン 20ピン

シリアル 6個 (+1個可能)

SPI 1個

A/D 4個

RTC

I2C 4個 (+1個可能)

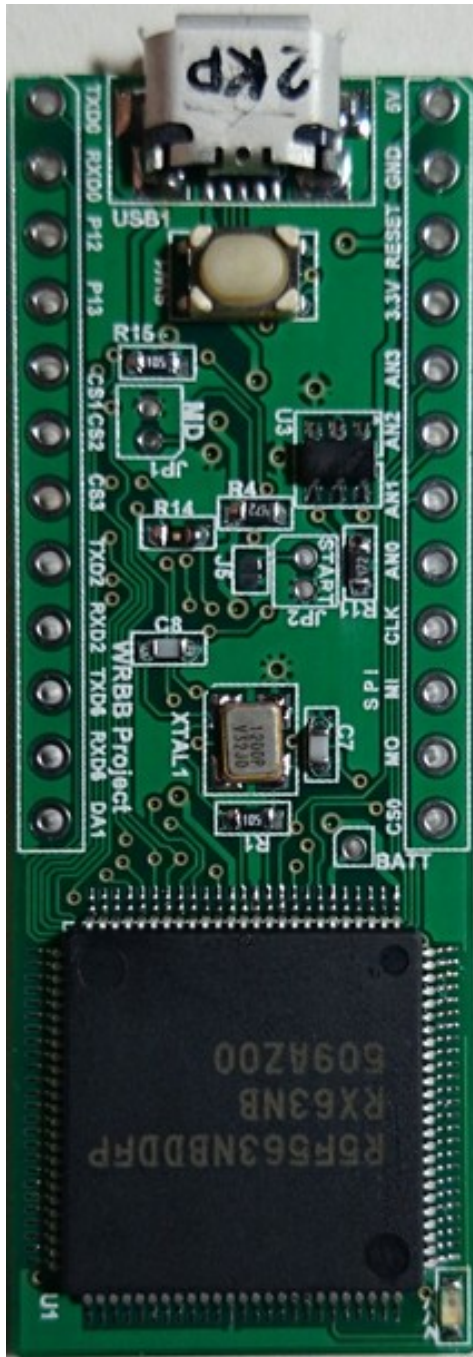
PWM、Servoは自由割当てです。

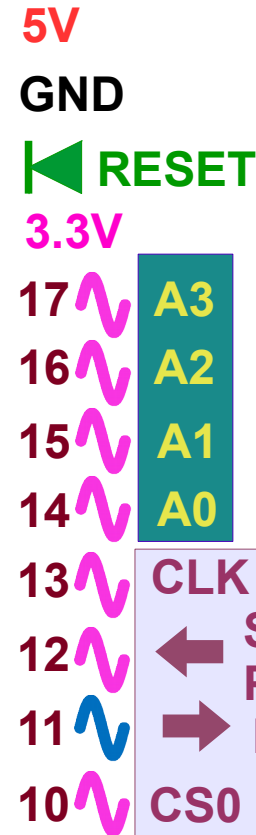
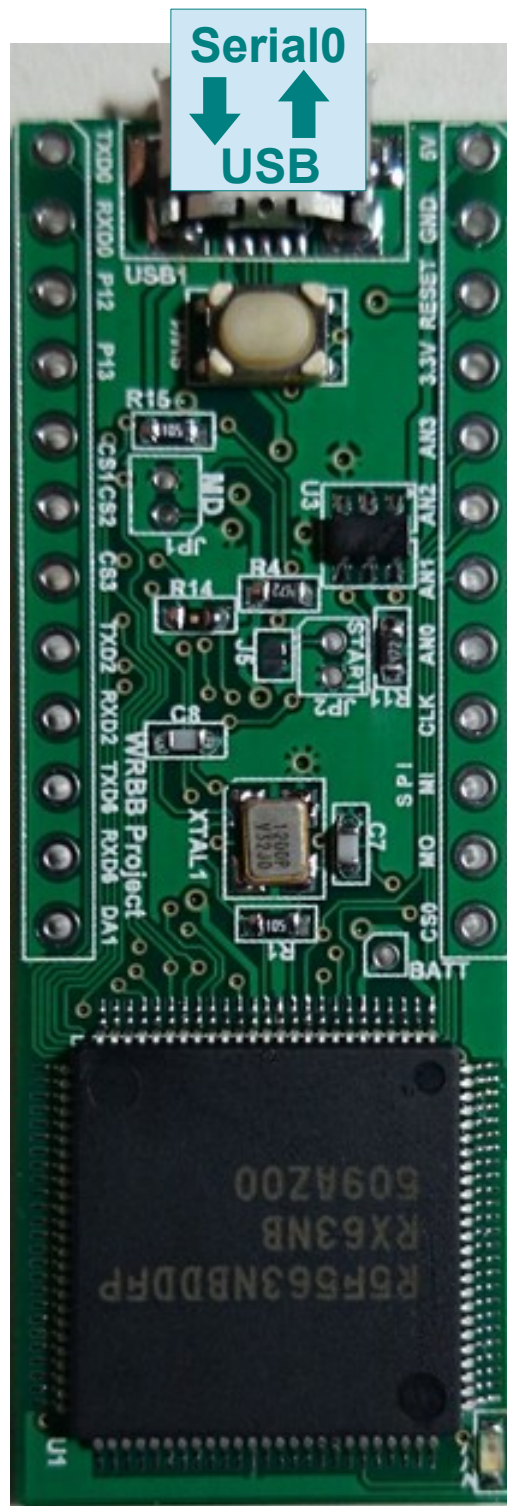
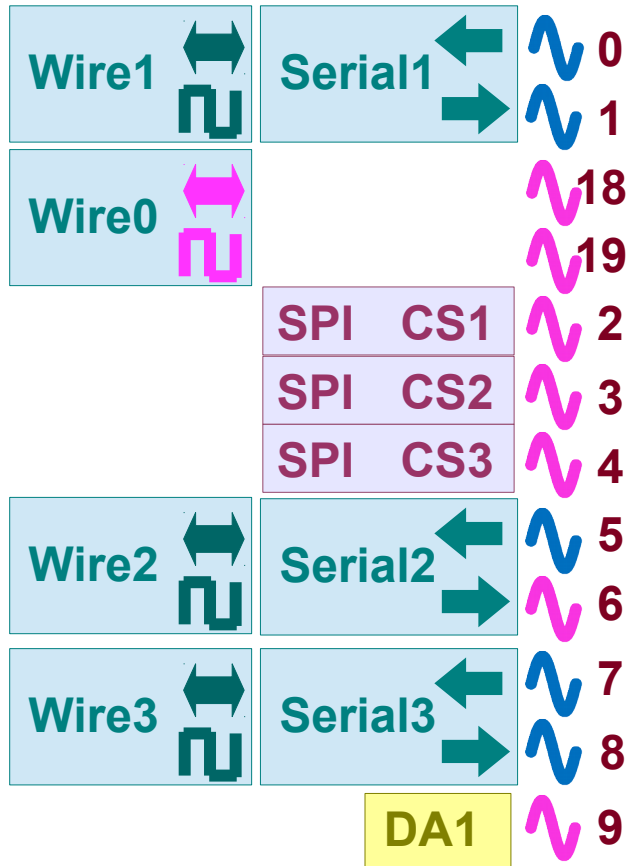
電源



5V (USBバスパワー)

サイズ

50 × 18mm



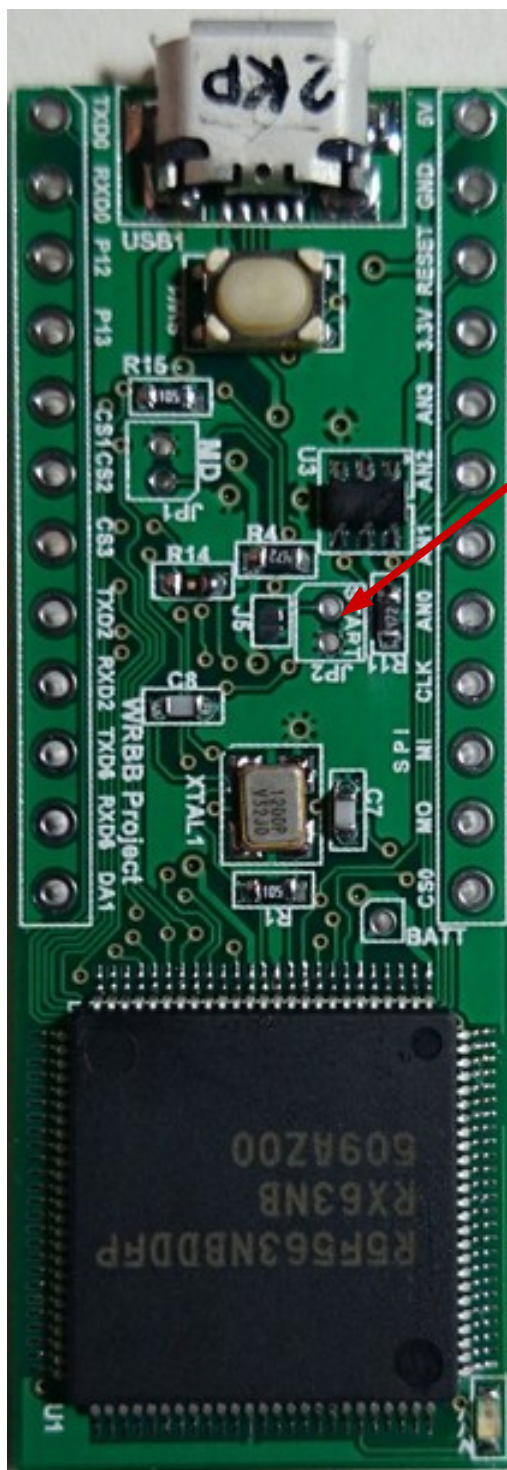


 analogWrite,tone(PWM)
 software PWM

RX63Nピン番号

	P20	0
	P21	1
	P12	18
P15	P13	19
P31	PC0	2
P30	PC1	3
	PC2	4
P34	P50	5
P55	P52	6
	P32	7
	P33	8
P26	P05	9

赤文字ピン番は
5Vトレラント



5V

GND

◀ RESET

3.3V

17

16

15

14

13

12

11

10

P43

P42

P41

P40

PC5

PC7

PC6

PC4

P35

NMI

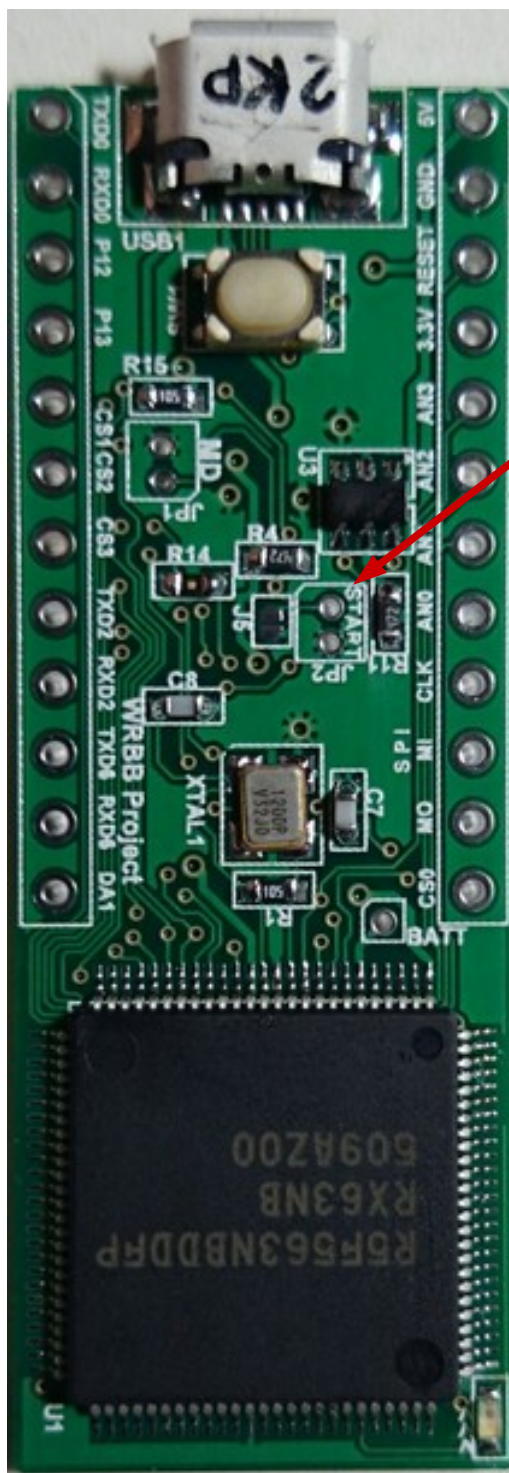
RX63Nピン番号

GR-SAKURAとのピン対比

GR-SAKURA割当番号

	1
	0
	30
33	31
TMS	22
TDI	23
	8
TRST	24
29	26
	6
	7
TDO	53

	P20	0
	P21	1
	P12	18
P15	P13	19
P31	PC0	2
P30	PC1	3
	PC2	4
P34	P50	5
P55	P52	6
	P32	7
	P33	8
P26	P05	9



5V
GND
◀ RESET
3.3V

17	P43	PE1
16	P42	PB5
15	P41	PB3
14	P40	P27
13	PC5	
12	PC7	
11	PC6	
10	PC4	

P35	54
-----	----

17	45
16	
15	
14	TCK
13	
12	
11	
10	

GR-SAKURA割当番号

JTAGの端子

TMS TDI MD

TRST

TDO

EMLE



GND RESET V_{3.3}

TCK

SPIMISO

ジャンパの説明

J4

P27(TCK)と14番を接続します。
P40ともショートになります。

J3

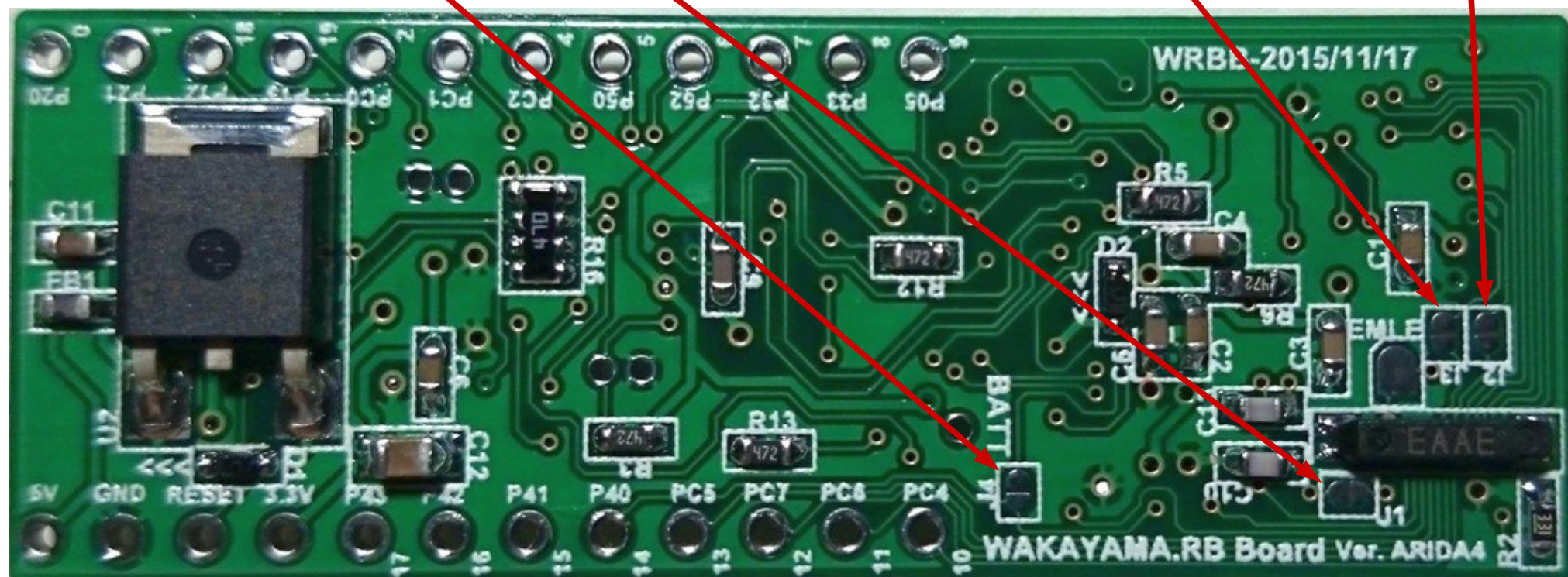
PB3と15番を接続します。
P41ともショートになります。

J1

PE1と17番を接続します。
P43ともショートになります。

J2

PB5と16番を接続します。
P42ともショートになります。



ジャンパの説明

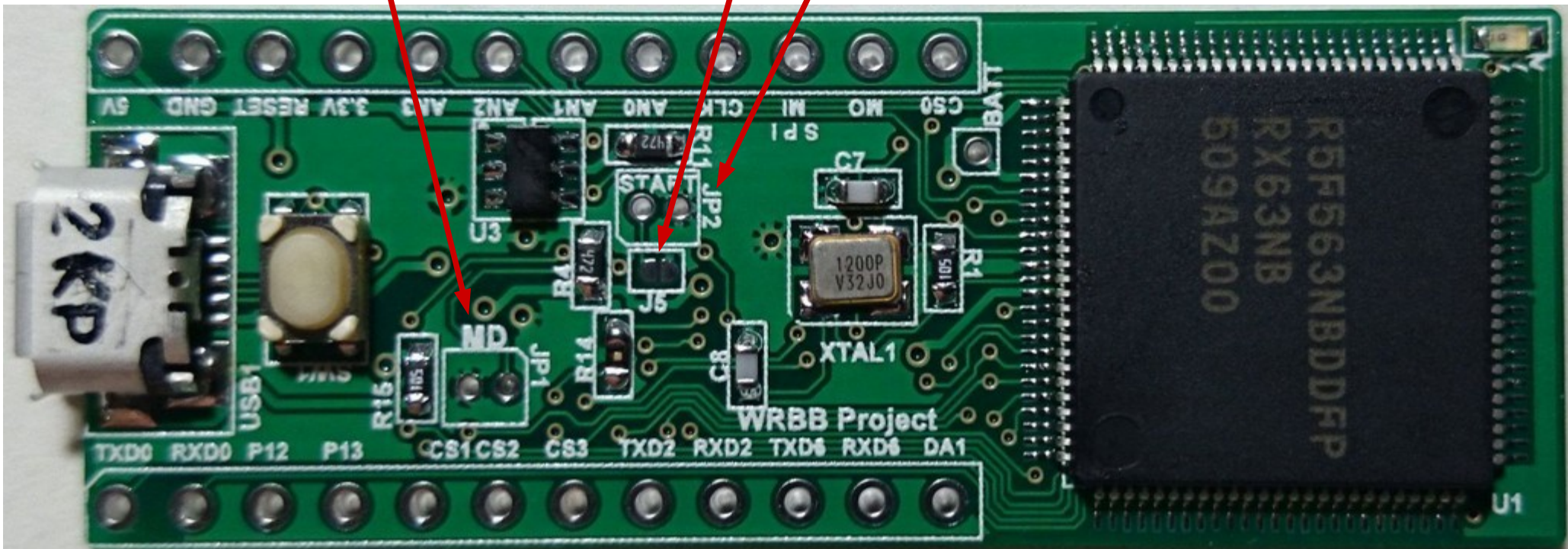
JP1

MDをGNDに落とします。
ファームを書き換えるときに、GND
とショートさせます。

J10

P35(NMI)をGNDに落とします。
電源ON時にデフォルトでGNDにしたいと
きに使用します。

JP2もJ10と同じです。



Rubyフレーム仕様(バインドしているmrbgem)

9

```
mruby-math          #->Math.sin  
mruby-numeric-ext   #->10.chr 0.zero?
```

カーネルクラス

```
pinMode(pin, mode)
digitalWrite(pin, value)
digitalRead(pin)
analogRead(number)
pwm(pin, value)
analogReference(mode)
initDac()
analogDac(value)
delay(value)
millis()
micros()
led(sw)
tone(pin, freq[, duration])
noTone(pin)
randomSeed(value)
random([min, ] max)
```

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)
MemFile.cp(src, dst[, mode])
MemFile.rm(filename)
```

シリアルクラス

```
Serial.new(number[, bps])
  bps(bps)
  print([str])
  println([str])
  available()
  read()
  write(buf, len)
  flush()
```

I2Cクラス

```
I2c.new(num)
  write(deviceID, address, data)
  read(deviceID, addL[, addH])
  begin(deviceID)
  lwrite(data)
  end()
  request(address, count)
  lread()
  available()
```

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.attached?(ch)
Servo.detach(ch)
```

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()  
System.reset()  
System.useSD()  
System.useWiFi()  
System.use(className[, options])  
System.use?(className[, options])  
System.getMrbPath()
```

グローバル変数

```
ON      = 1  
OFF     = 0  
HIGH    = 1  
LOW     = 0  
OUTPUT  = 1  
INPUT   = 0
```

リアルタイムクロッククラス

```
Rtc.getTime()  
Rtc.setTime(array)  
Rtc.deinit()  
Rtc.init()
```


SDカードクラス

- SD.exists(filename)
- SD.mkdir(dirname)
- SD.remove(filename)
- SD.copy(srcfilename, distfilename)
- SD.rmdir(dirname)
- SD.open(number, filename[, mode])
- SD.close(number)
- SD.read(number)
- SD.seek(number, byte)
- SD.write(number, buf, len)
- SD.flush(number)
- SD.size(number)
- SD.position(number)
- SD.cpmem(sdfilename, memfilename[, mode])

WiFiクラス

```
WiFi.at(command[, mode])  
WiFi.bypass()  
WiFi.cClose(number)  
WiFi.connect(SSID, Passwd)  
WiFi.disconnect()  
WiFi.httpGet(URL[, Headers])  
WiFi.httpGetSD(Filename, URL[, Headers])  
WiFi.httpPost(URL, Headers, Body)  
WiFi.httpPostSD(URL, Headers, Filename)  
WiFi.httpServer([Port])  
WiFi.ipconfig()  
WiFi.multiConnect(mode)  
WiFi.recv(number)  
WiFi.send(number, Data[, length])  
WiFi.serialOut(mode[, serialNumber])  
WiFi.setMode(mode)  
WiFi.udpOpen(number, IP_Address, SendPort, ReceivePort)  
WiFi.version()
```

***ruby*プログラムの実行**

Rubyファームは、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。

Rubyファームは、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

Rubyファームは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、RubyファームからデータをPCに出力したりします。



シリアル通信



CoolTerm



TeraTerm

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やCoolTermがあります。

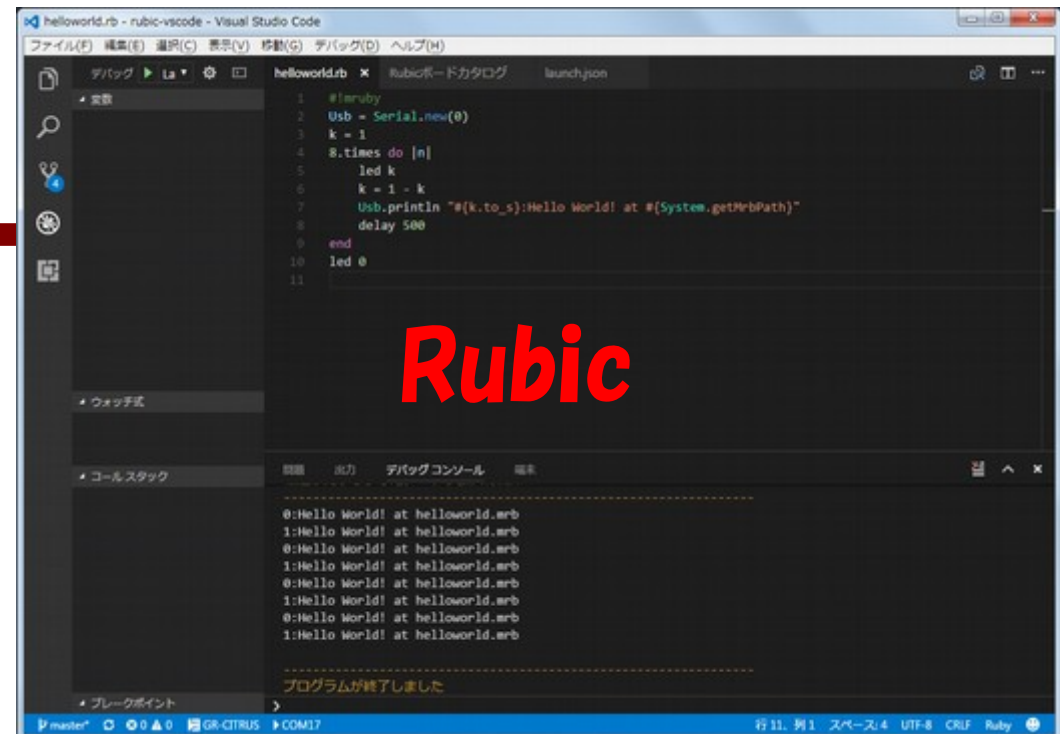
プログラムの書き込み

Rubic - vscode

きむしゅさんが開発している「Rubic-vscode」を使用すると、Rubyファームに接続したまま楽にプログラム開発ができます。mrbファイルへのコンパイルもRubic-vscodeが行います。



<https://github.com/kimushu/rubic-vscode>

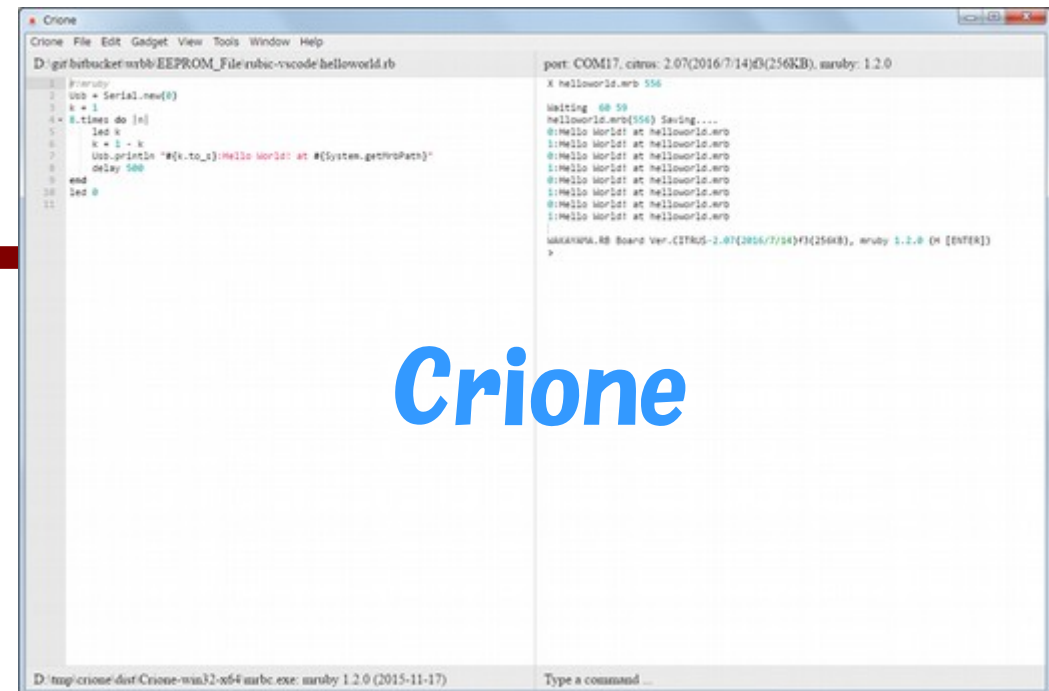


プログラムの書き込み Crione

ogomさんが開発している「Crione」を使用すると、Rubyファームに接続したまま楽にプログラム開発ができます。mrbファイルへのコンパイルもCrione内部でmrbcを呼び出して自動的に行います。



<https://github.com/ogom/crione>



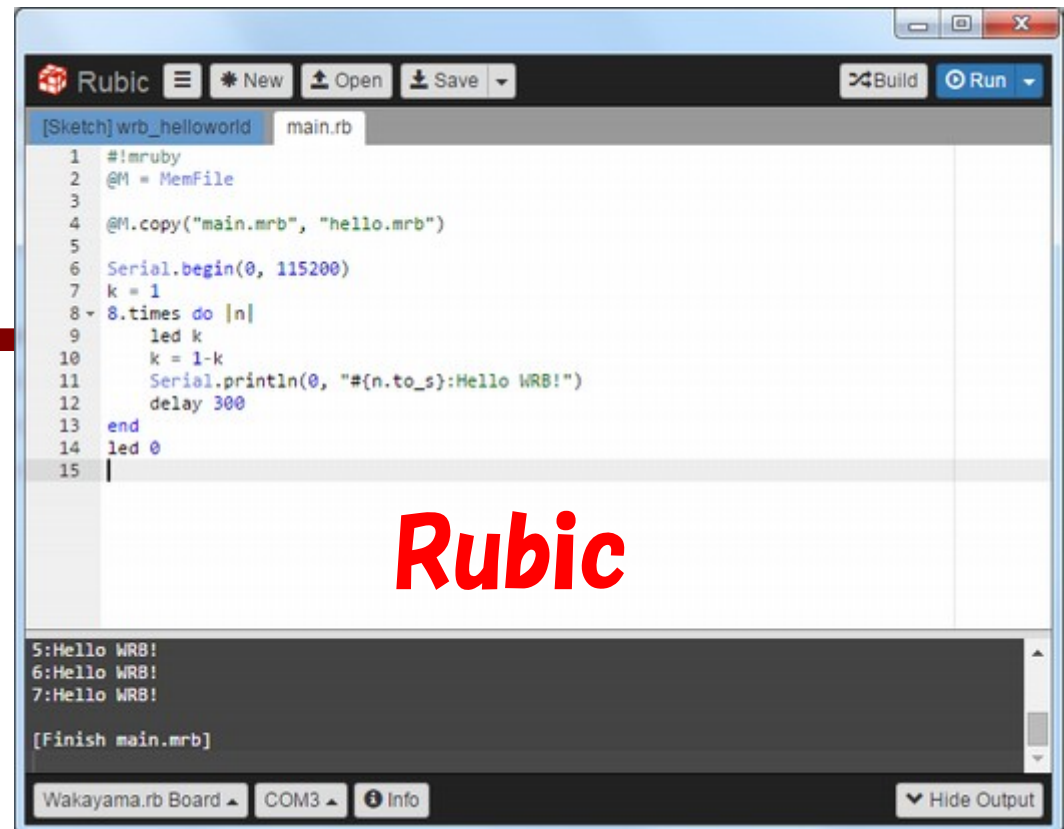
Crione

プログラムの書き込み

Chrome版 Rubic

きむしゅさんが開発している「Rubic」を使用すると、Rubyファームに接続したまま楽にプログラム開発ができます。mrbファイルへのコンパイルもRubicが行います。

※Chromeアプリ版のRubicは、Chromeの仕様変更によりサポートされない可能性があります。

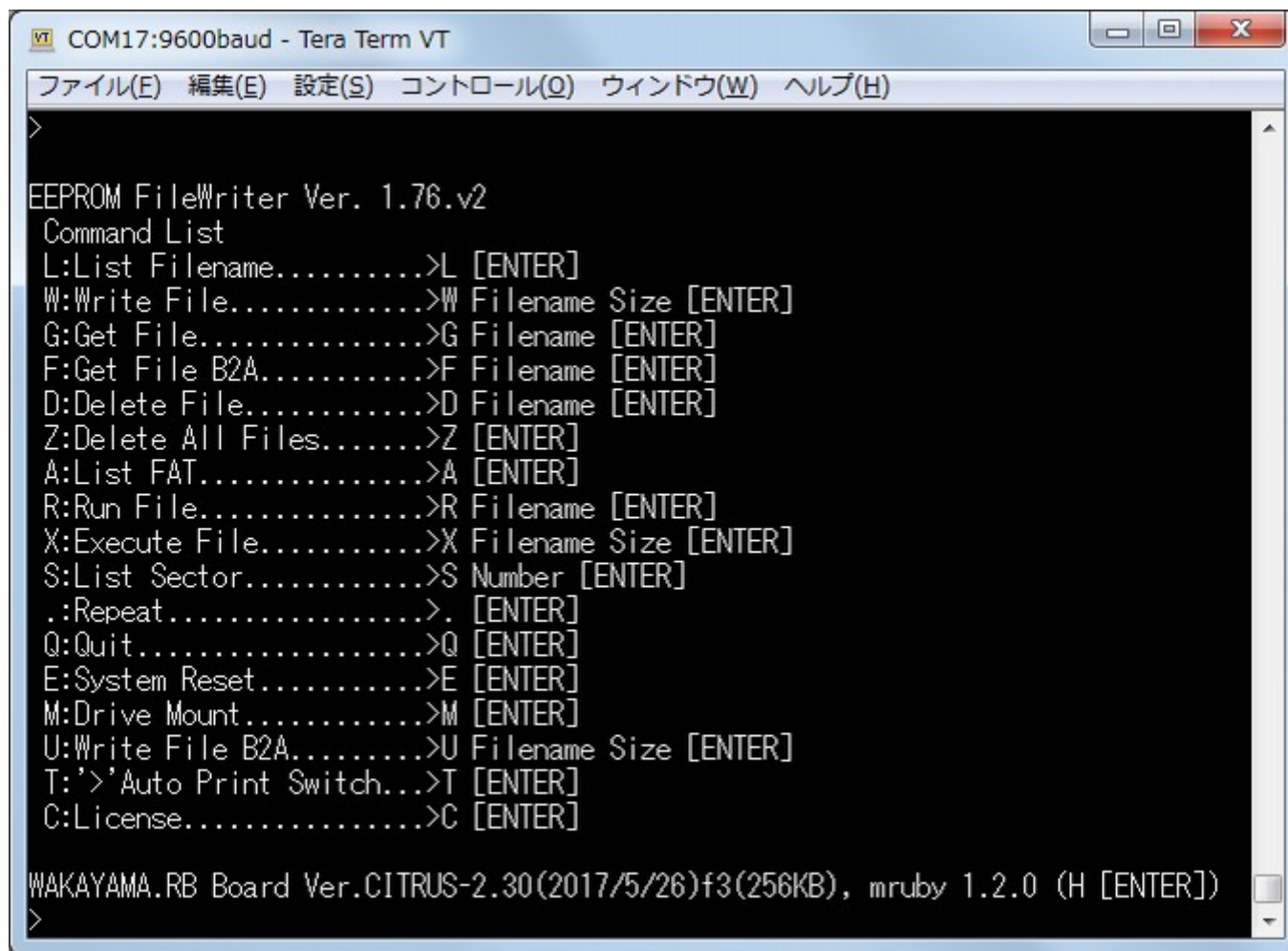


<https://github.com/kimushu/rubic>

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。



```
COM17:9600baud - 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)
>
EEPROM FileWriter Ver. 1.76.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
M:Drive Mount.....>M [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
T:'''Auto Print Switch...>T [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.CITRUS-2.30(2017/5/26)f3(256KB), mruby 1.2.0 (H [ENTER])
>
```

Rubyファームの起動画面

コマンドの種類

```

COM17:9600baud - 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. 1.76.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
M:Drive Mount.....>M [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
T:'>'Auto Print Switch...>T [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.CITRUS-2.30(2017/5/26)f3(256K)
>
  
```

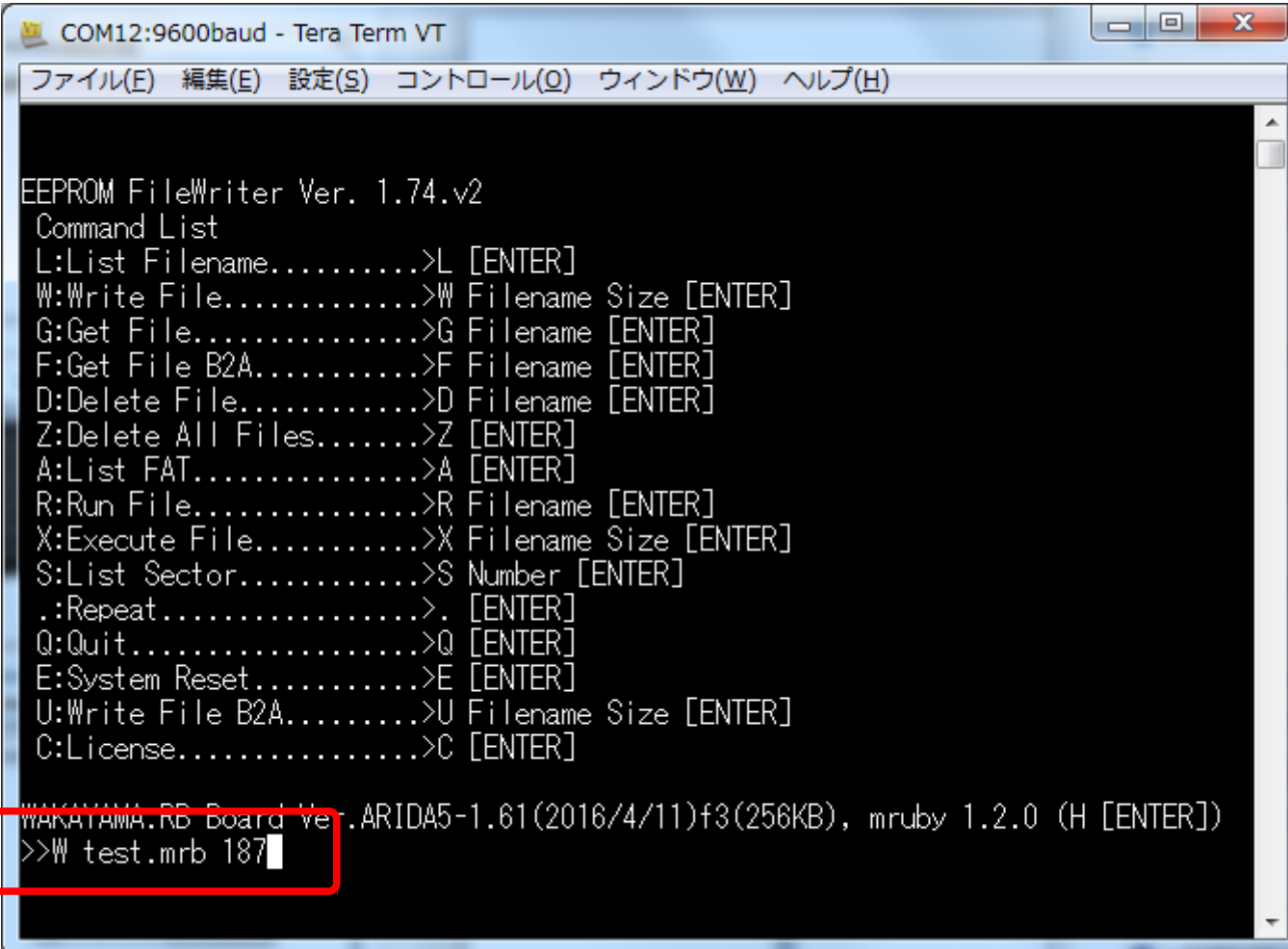
- L: 保存しているファイルを一覧します。
- W: ファイルを書き込みます。
- G: ファイルを送信します。
- F: ファイル内容を16進数テキストで送信します。
- D: ファイルを削除します。
- Z: 全てのファイルを削除します。
- A: セクタを一覧表示します。
- R: 実行ファイルをセットします。
- X: ファイルを書き込んで直ぐ実行します。
- S: セクタ情報を表示します。
- .: 直前のコマンドを再実行します。
- Q: コマンドを終了します。
- E: システムをリセットします。
- M: ドライブとしてマウントします。
- U: 16進数テキストでデータを受信します。
- T: '>'の自動送信を切り替えます。
- C: ライセンスを表示します。

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



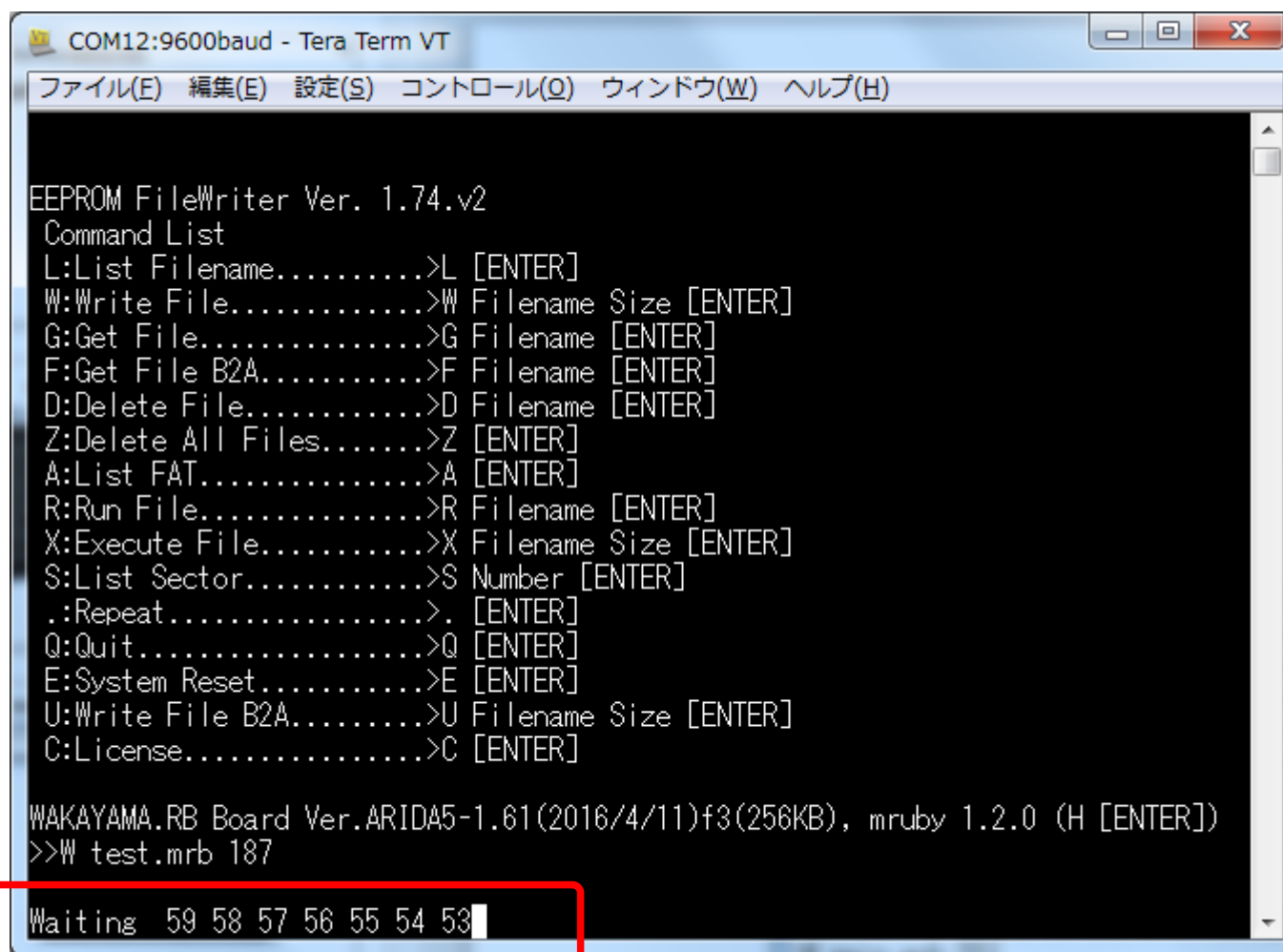
```
COM12:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)

EEPROM FileWriter Ver. 1.74.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKATAMA.RB Board Ver. ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W test.mrb 187
```

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。



```
COM12:9600baud - 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)

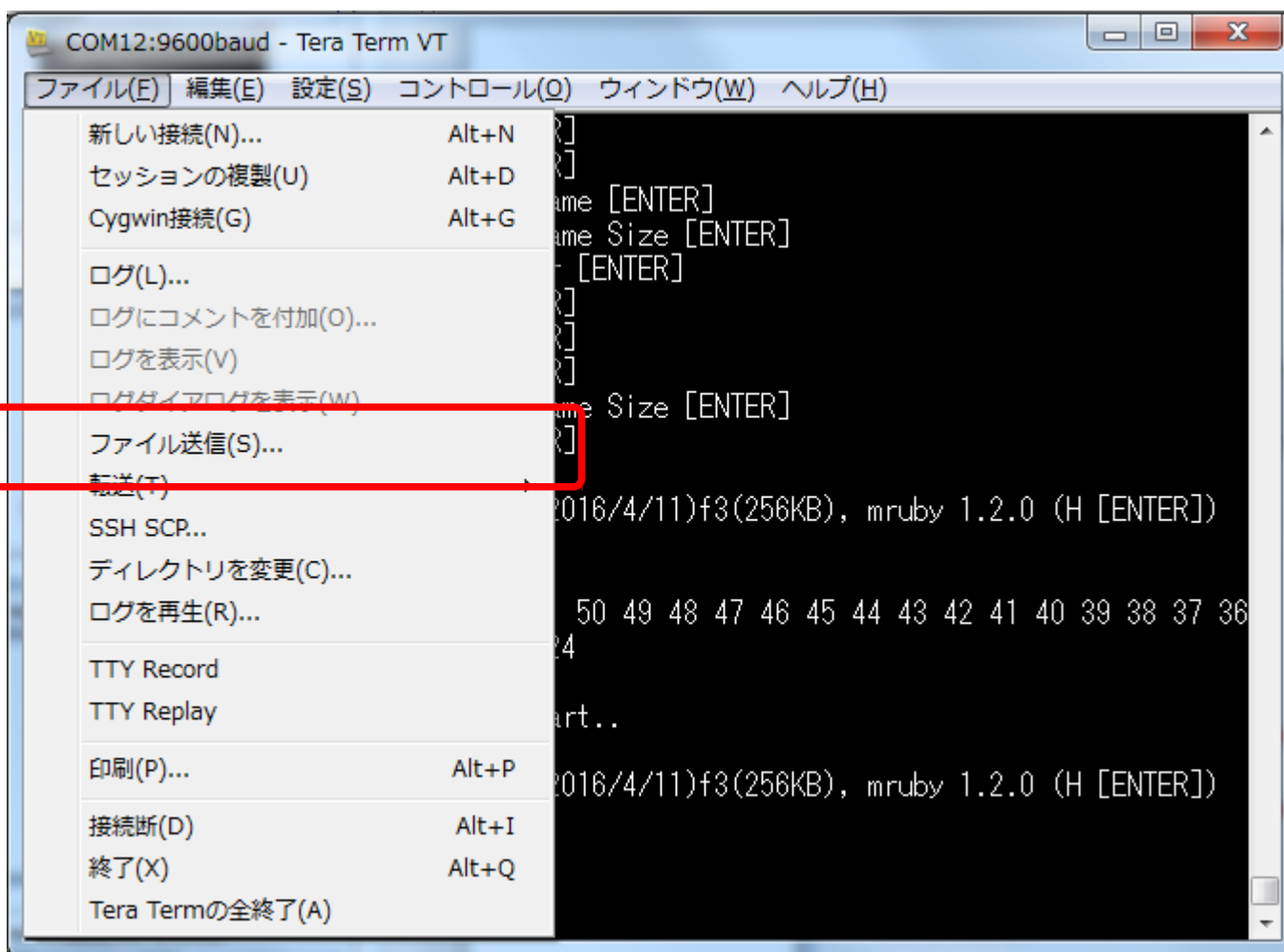
EEPROM FileWriter Ver. 1.74.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W test.mrb 187

Waiting 59 58 57 56 55 54 53
```

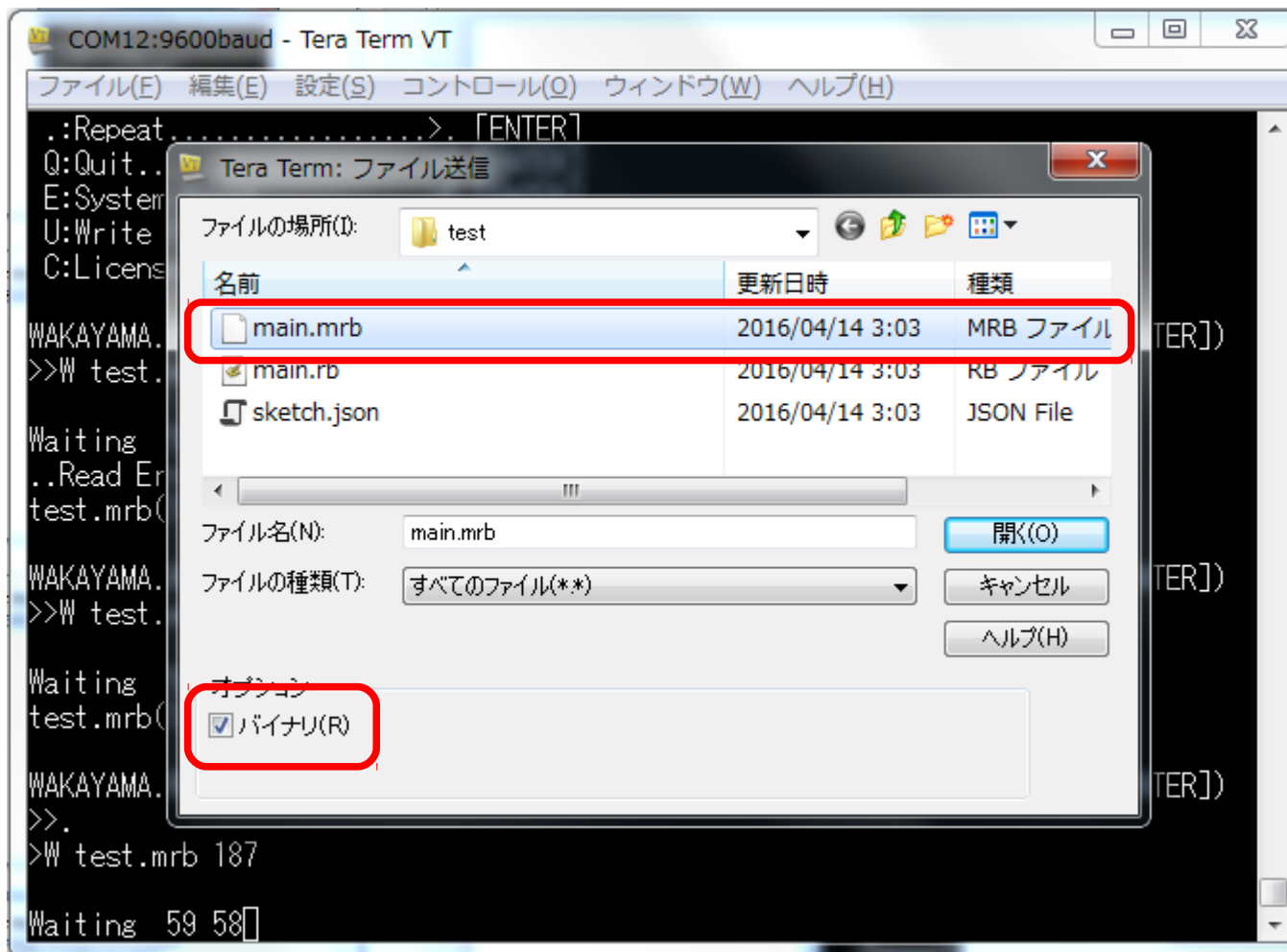
プログラムを書き込みます。(W コマンド)

Tera Termの場合、ファイル→ファイル送信 を選択します。



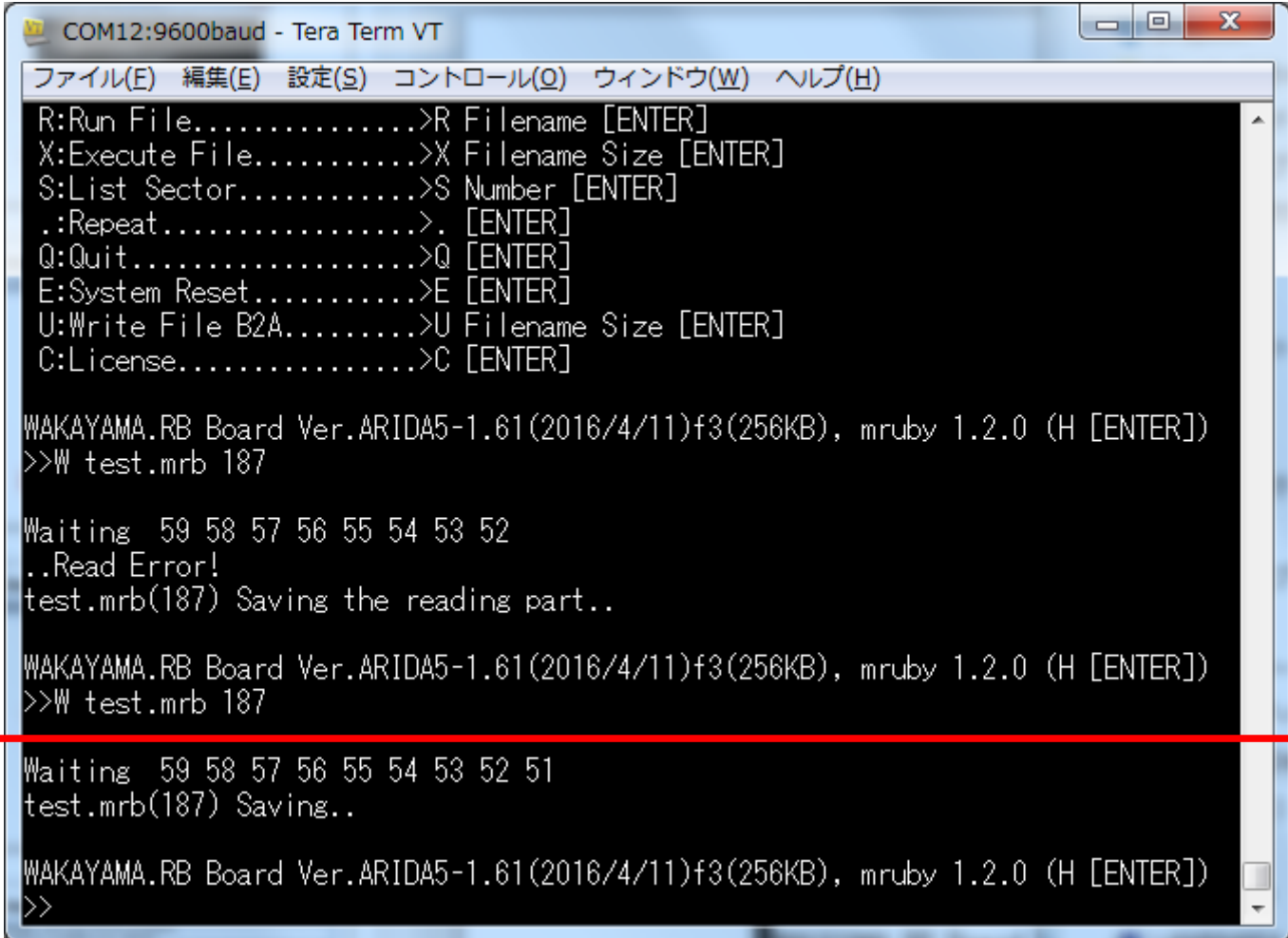
プログラムを書き込みます。(W コマンド)

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。



```
COM12:9600baud - 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)
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W test.mrb 187

Waiting 59 58 57 56 55 54 53 52
..Read Error!
test.mrb(187) Saving the reading part..

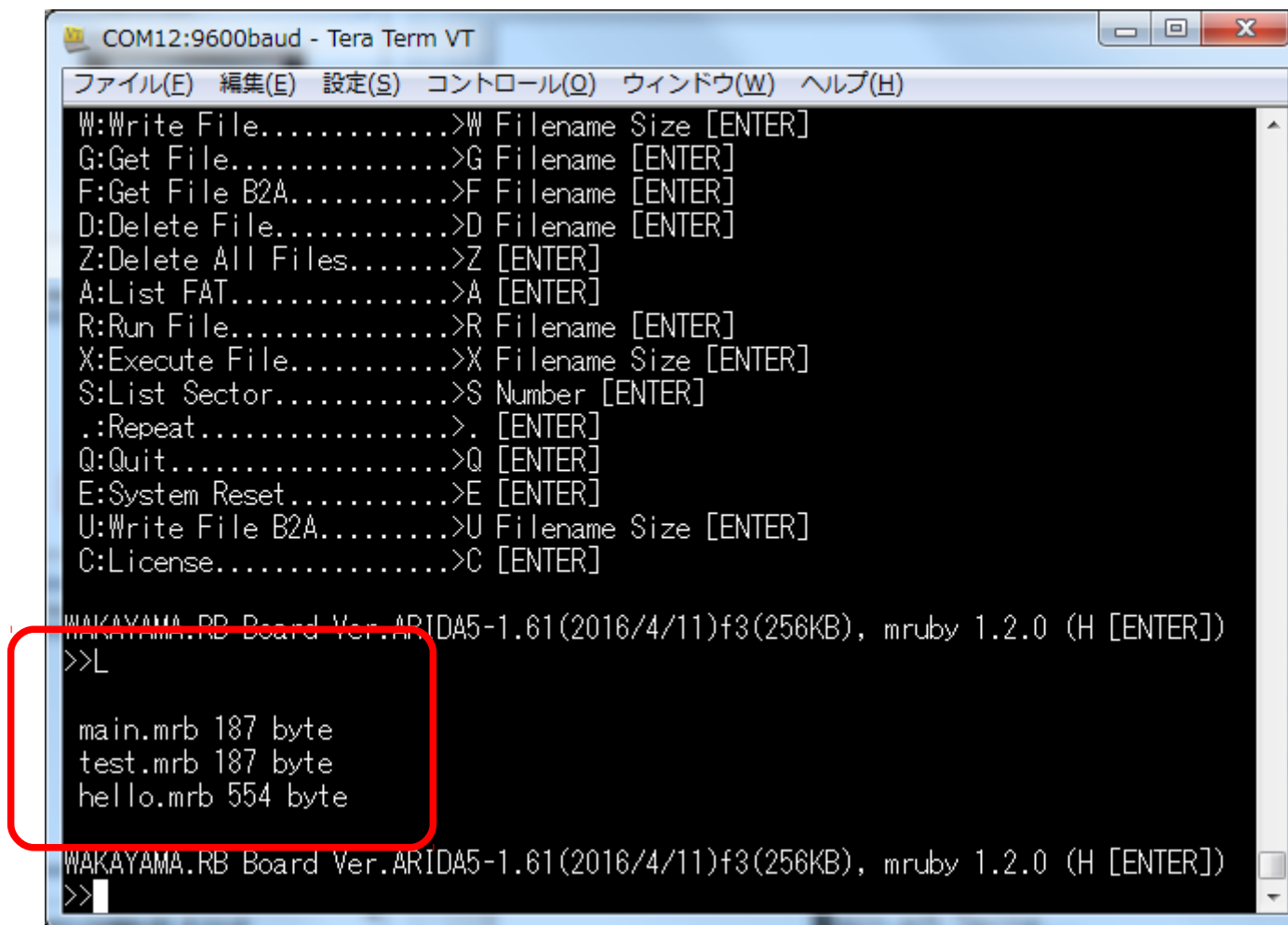
WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W test.mrb 187

Waiting 59 58 57 56 55 54 53 52 51
test.mrb(187) Saving..

WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (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 "COM12:9600baud - 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 command list includes: W:Write File, G:Get File, F:Get File B2A, D>Delete File, Z>Delete All Files, A:List FAT, R:Run File, X:Execute File, S:List Sector, .:Repeat, Q:Quit, E:System Reset, U:Write File B2A, and C:License. The prompt "WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])" is shown. The user has entered ">>L", and the output, highlighted by a red box, is: "main.mrb 187 byte", "test.mrb 187 byte", and "hello.mrb 554 byte". The prompt ">>" is shown at the bottom.

```
COM12:9600baud - 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]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]
WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L
main.mrb 187 byte
test.mrb 187 byte
hello.mrb 554 byte
WAKAYAMA.RB Board Ver.ARIDA5-1.61(2016/4/11)f3(256KB), mruby 1.2.0 (H [ENTER])
>>
```

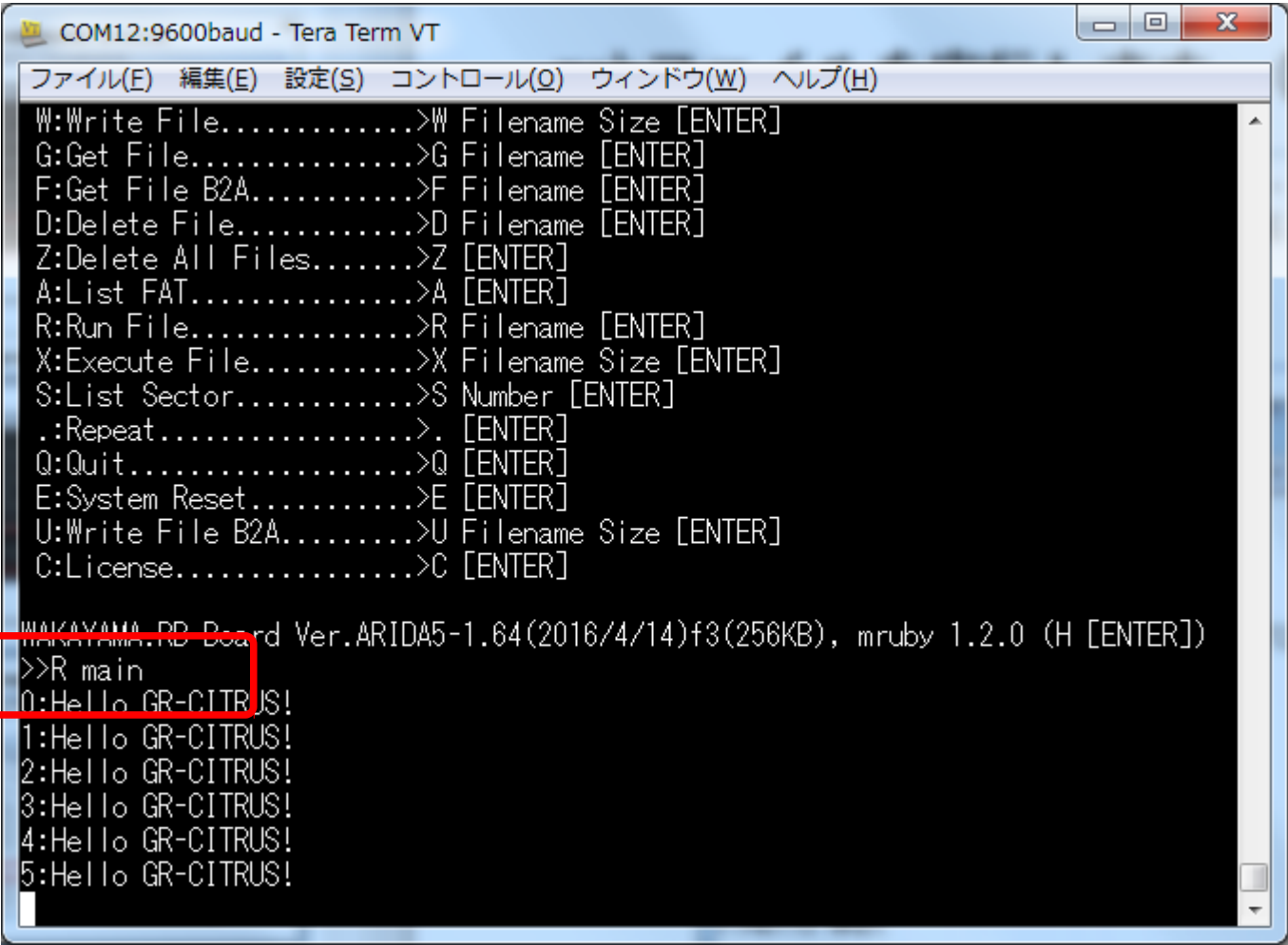
mrubyファイルを実行します。(R コマンド)

Rコマンドは、mruby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。

.mrubyは省略可能です。

>R ファイル名



The screenshot shows a Tera Term VT window titled "COM12:9600baud - 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)", and "ヘルプ(H)". The command list on the left includes: W:Write File, G:Get File, F:Get File B2A, D>Delete File, Z>Delete All Files, A:List FAT, R:Run File, X:Execute File, S:List Sector, .:Repeat, Q:Quit, E:System Reset, U:Write File B2A, and C:License. The main window displays the following text:

```
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

MANKAYAMA-RD Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
```

A red rectangle highlights the command ">>R main" and the subsequent output lines.

mrubyを実行します。(R コマンド)

実行が終了するとコマンドモードに戻ります。

```
COM12:9600baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
Z:Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.ARIDAS-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
6:Hello GR-CITRUS!
7:Hello GR-CITRUS!

WAKAYAMA.RB Board Ver.ARIDAS-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>
```

プログラムを書き込み実行します。(X コマンド)

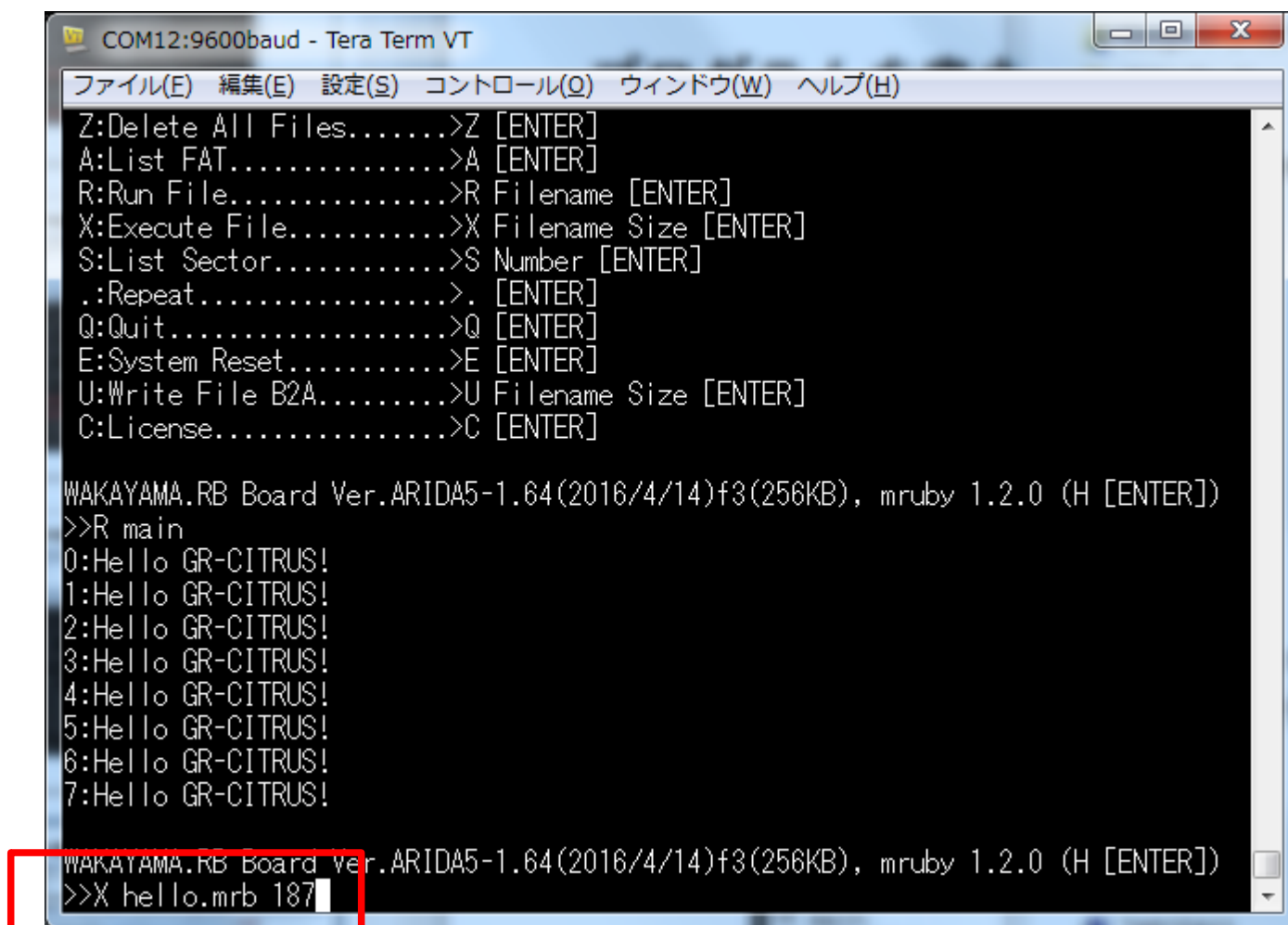
Xコマンドを用いて、mrbファイルを書き込み後直ぐ実行します。

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。

.mrbは省略可能です。

>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。



```
COM12:9600baud - 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)
Z:Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

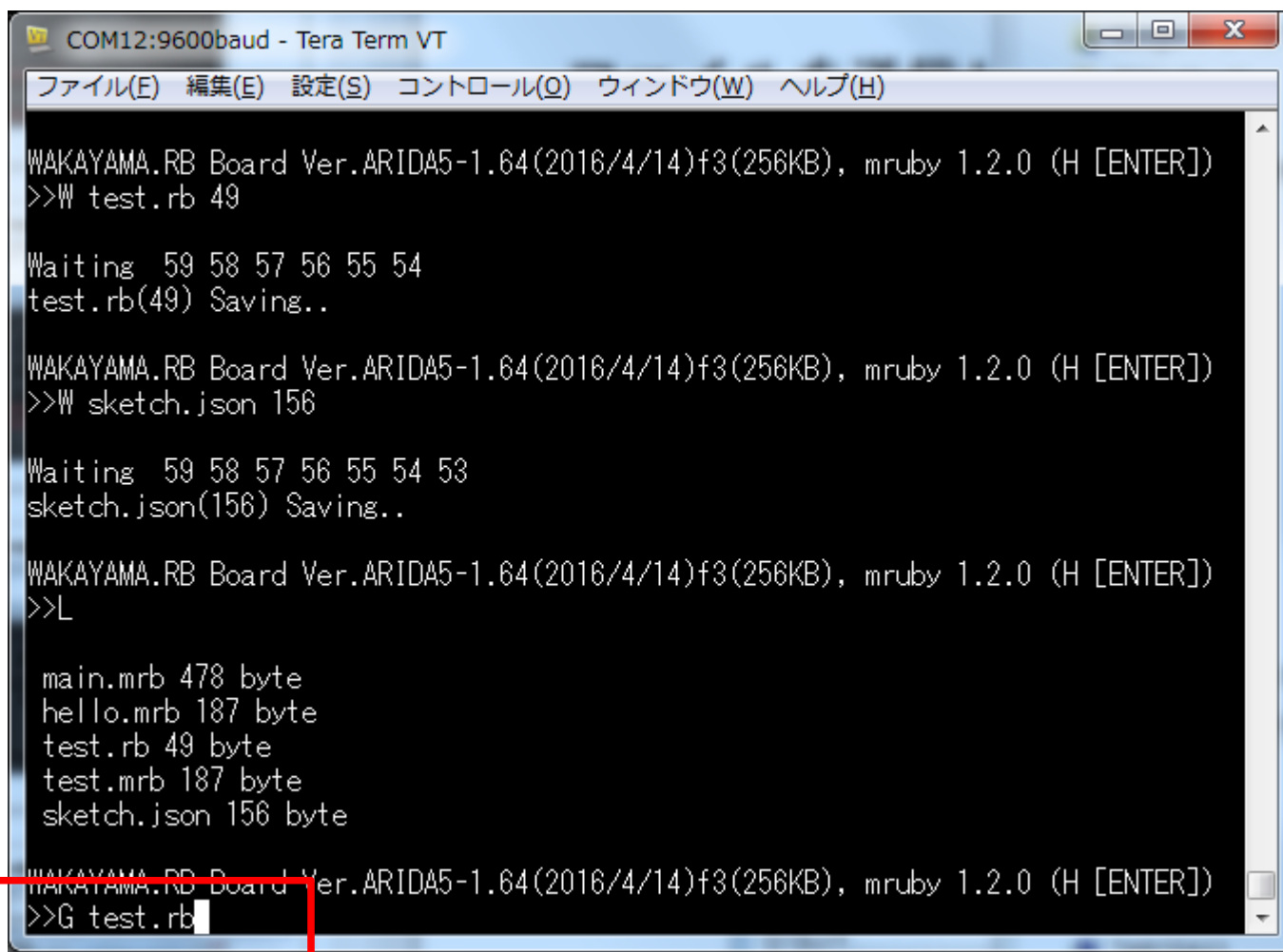
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
6:Hello GR-CITRUS!
7:Hello GR-CITRUS!

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>X hello.mrb 187
```


ファイルを送信します。(G コマンド)

Gコマンドを用いるとGR-CITRUSに保存されているファイルをPCに読み出すことができます。
Gの後にスペースで区切って、ファイル名を書き、ENTERキーを押します。

>G ファイル名



```
COM12:9600baud - 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)

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W test.rb 49

Waiting 59 58 57 56 55 54
test.rb(49) Saving..

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W sketch.json 156

Waiting 59 58 57 56 55 54 53
sketch.json(156) Saving..

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L

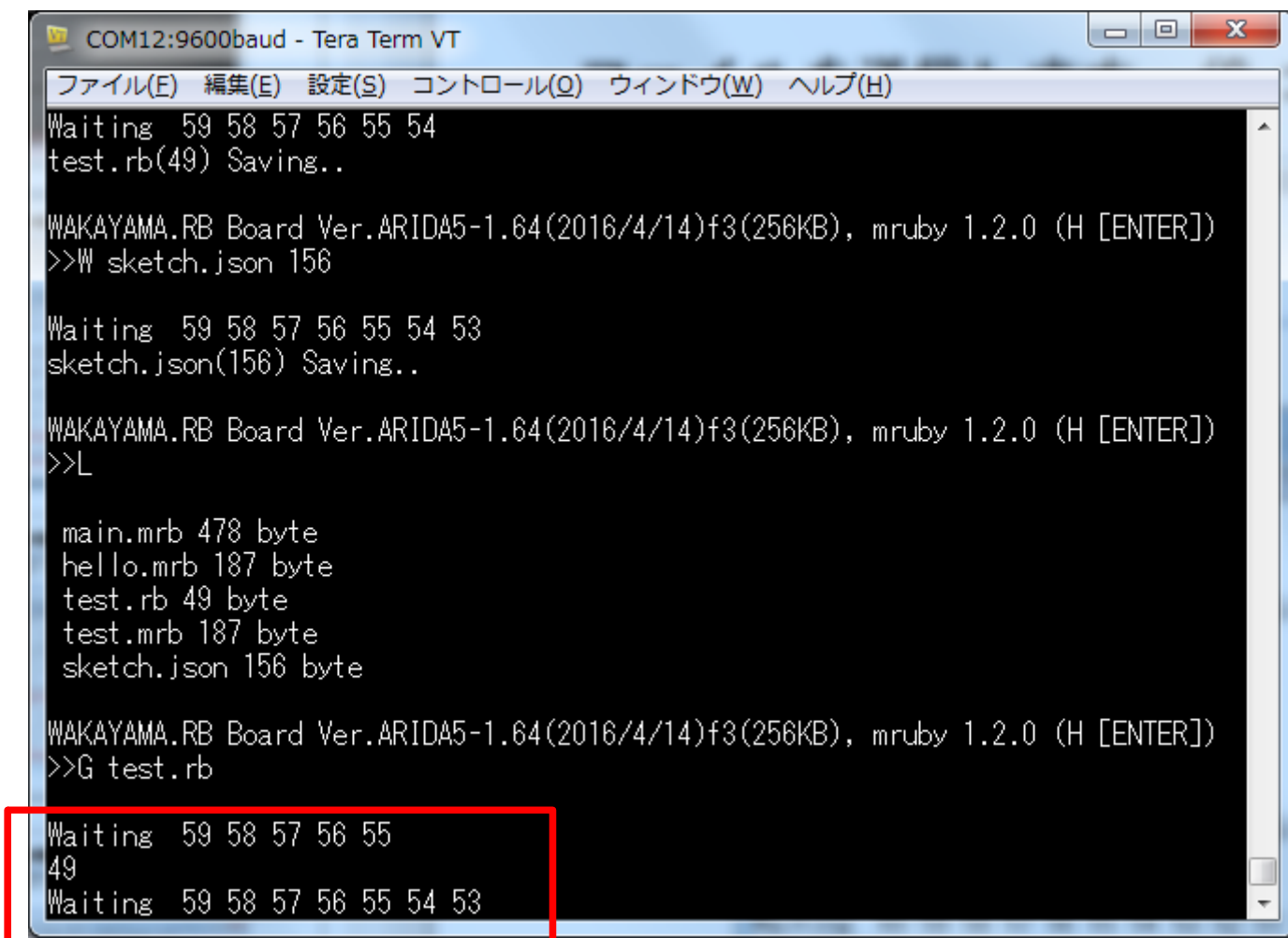
main.mrb 478 byte
hello.mrb 187 byte
test.rb 49 byte
test.mrb 187 byte
sketch.json 156 byte

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>G test.rb
```

ファイルを送信します。(G 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PCから1バイト送信すると、送信するファイルのファイルサイズが送信されます。

そして、再びカウントダウンが始まります。



```
COM12:9600baud - 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)
Waiting 59 58 57 56 55 54
test.rb(49) Saving..

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>W sketch.json 156

Waiting 59 58 57 56 55 54 53
sketch.json(156) Saving..

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L

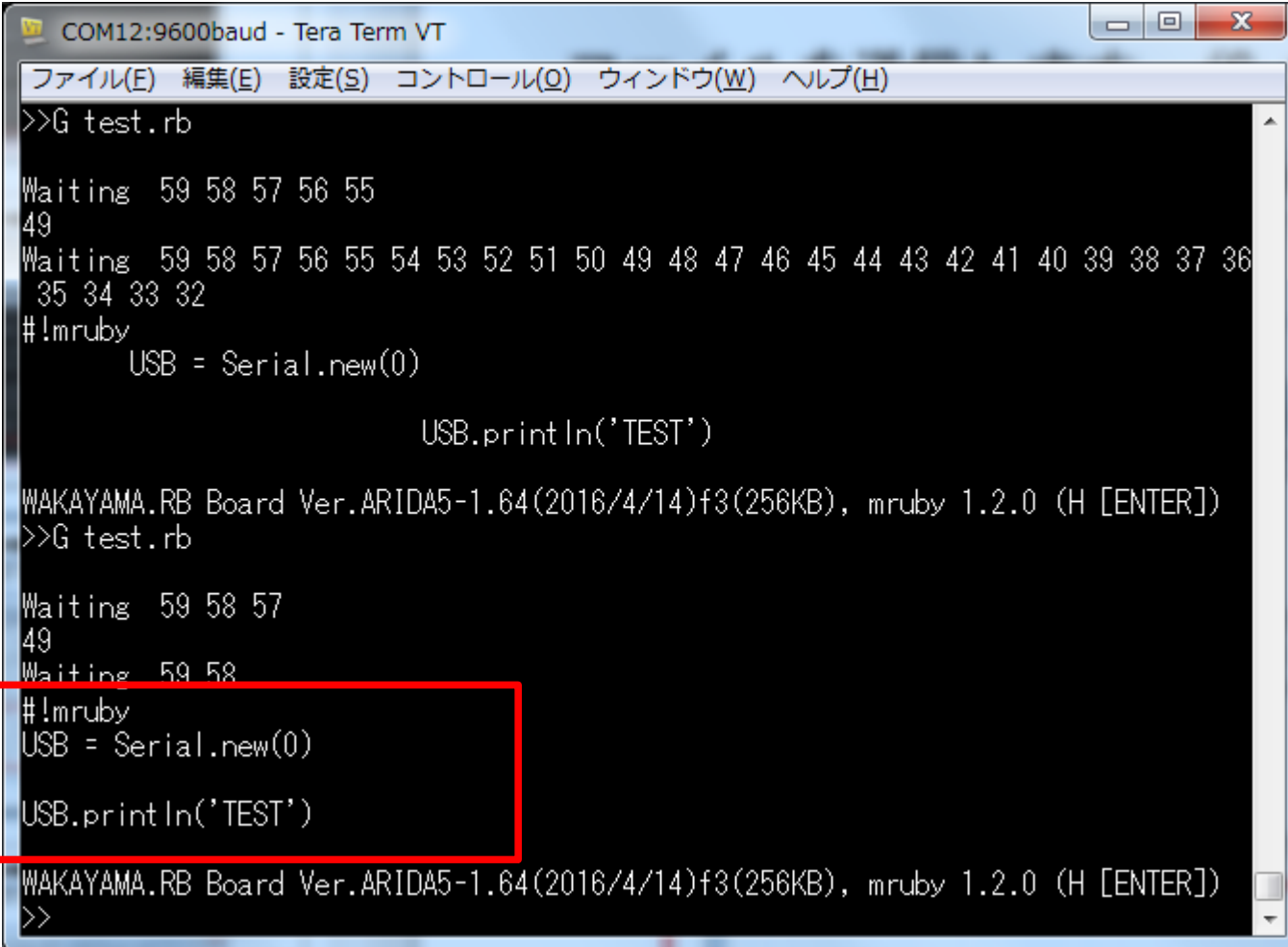
main.mrb 478 byte
hello.mrb 187 byte
test.rb 49 byte
test.mrb 187 byte
sketch.json 156 byte

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>G test.rb

Waiting 59 58 57 56 55
49
Waiting 59 58 57 56 55 54 53
```

ファイルを送信します。(G コマンド)

2回目のカウントダウンの60sec以内にPCから1バイト送信すると、指定したファイルが送信されます。
バイナリファイルの場合もバイナリのまま送信されます。



The screenshot shows a Tera Term VT terminal window titled "COM12:9600baud - 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)", and "ヘルプ(H)". The terminal displays the following sequence of events:

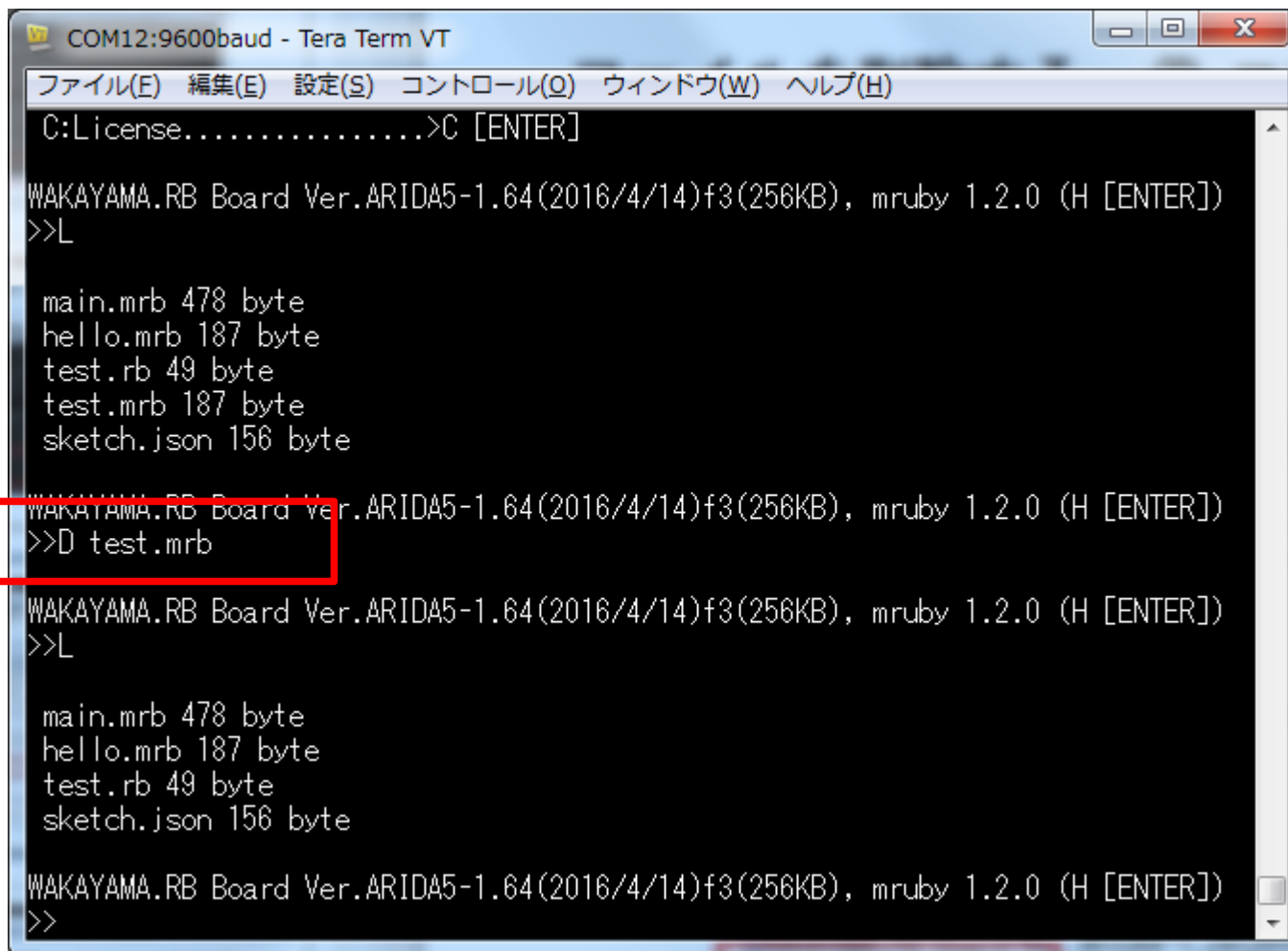
```
>>G test.rb  
  
Waiting 59 58 57 56 55  
49  
Waiting 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36  
35 34 33 32  
#!mruby  
    USB = Serial.new(0)  
  
        USB.println('TEST')  
  
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])  
>>G test.rb  
  
Waiting 59 58 57  
49  
Waiting 59 58  
#!mruby  
USB = Serial.new(0)  
  
USB.println('TEST')  
  
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])  
>>
```

A red rectangular box highlights the second execution of the mruby code block, specifically the lines: `#!mruby`, `USB = Serial.new(0)`, and `USB.println('TEST')`.

ファイルを削除する。(D コマンド)

DコマンドはWRBボード保存しているファイルを削除します。
Dの後にスペースで区切って、ファイル名を書き、ENTERキーを押します。

>D ファイル名



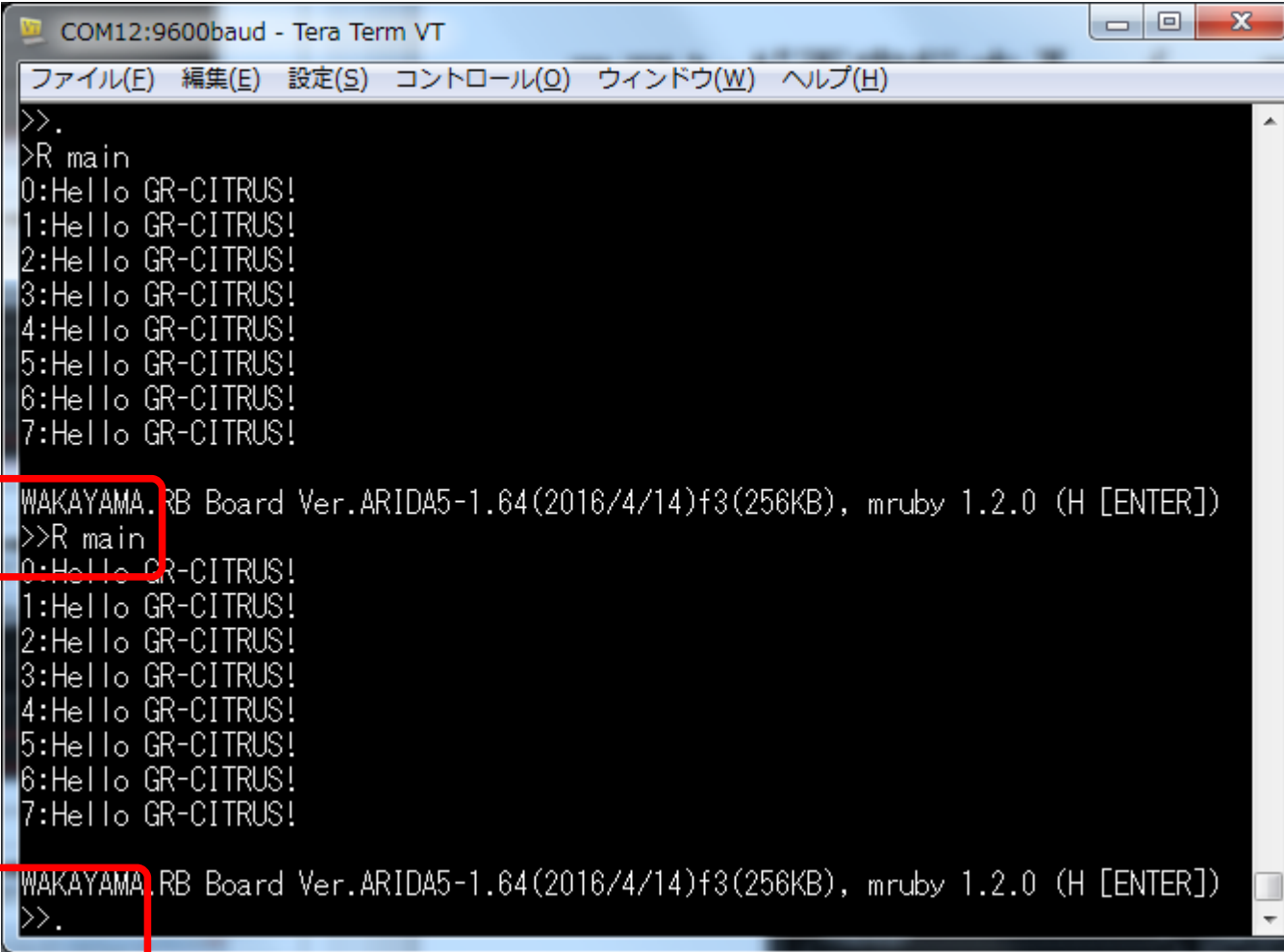
The screenshot shows a Tera Term VT window titled "COM12:9600baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(O)", "ウィンドウ(W)", and "ヘルプ(H)". The command prompt shows the following sequence of commands and outputs:

```
C:\License.....>C [ENTER]
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L
main.mrb 478 byte
hello.mrb 187 byte
test.rb 49 byte
test.mrb 187 byte
sketch.json 156 byte
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>D test.mrb
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L
main.mrb 478 byte
hello.mrb 187 byte
test.rb 49 byte
sketch.json 156 byte
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>
```

A red rectangle highlights the command ">>D test.mrb", indicating the successful execution of the file deletion command.

コマンド再実行する。(. コマンド)

. コマンドを入力すると、直前に実行したコマンドを再実行します。



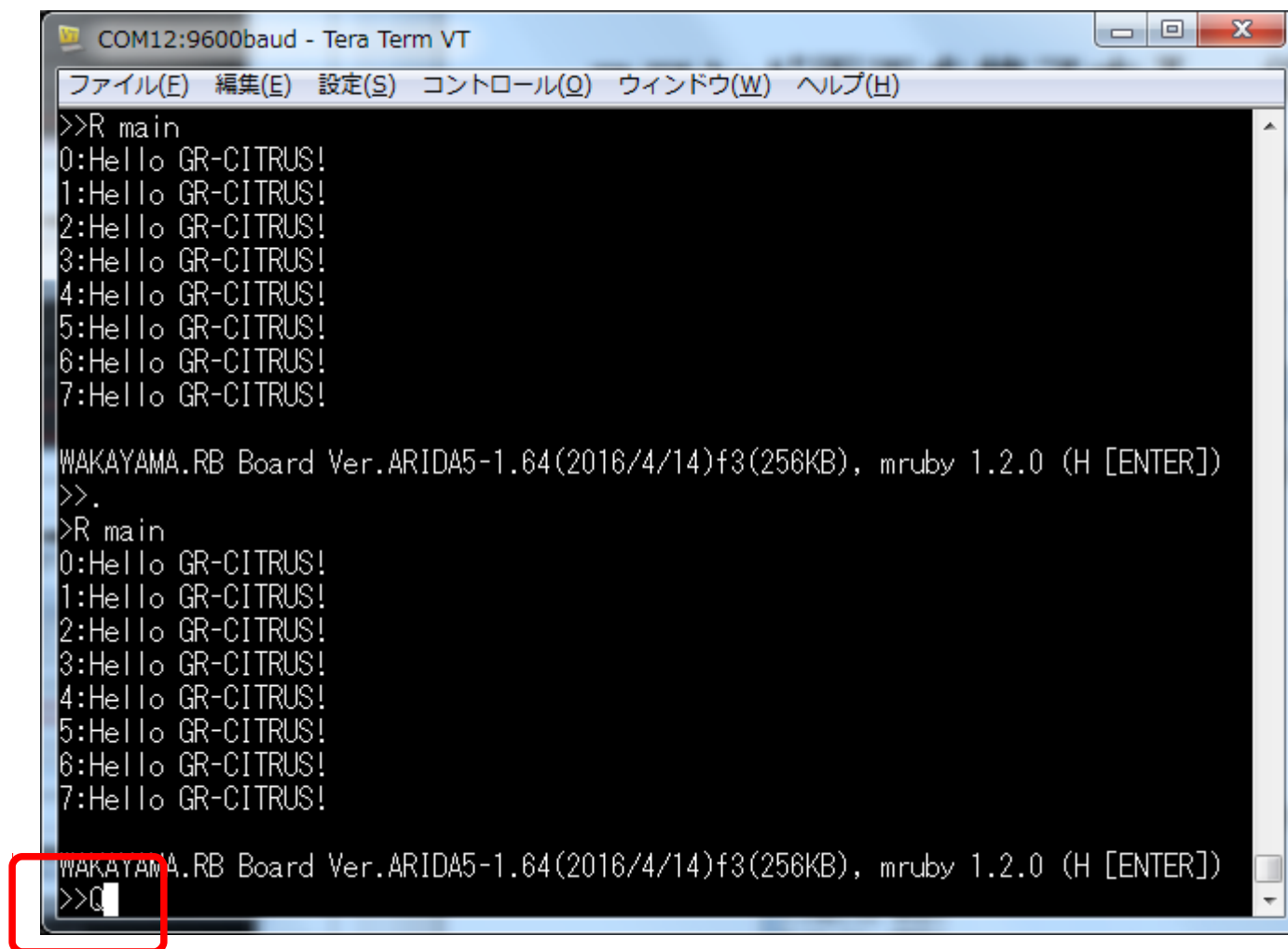
The screenshot shows a Tera Term VT window titled "COM12:9600baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(Q)", "ウィンドウ(W)", and "ヘルプ(H)". The terminal output shows a sequence of commands and their results:

```
>>.
>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
6:Hello GR-CITRUS!
7:Hello GR-CITRUS!
WAKAYAMA:RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
6:Hello GR-CITRUS!
7:Hello GR-CITRUS!
WAKAYAMA:RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>.
```

Two red boxes highlight the input lines: the first box is around "WAKAYAMA:RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])" followed by ">>R main", and the second box is around the same prompt followed by ">>.". This demonstrates how the "." command repeats the last command entered.

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。



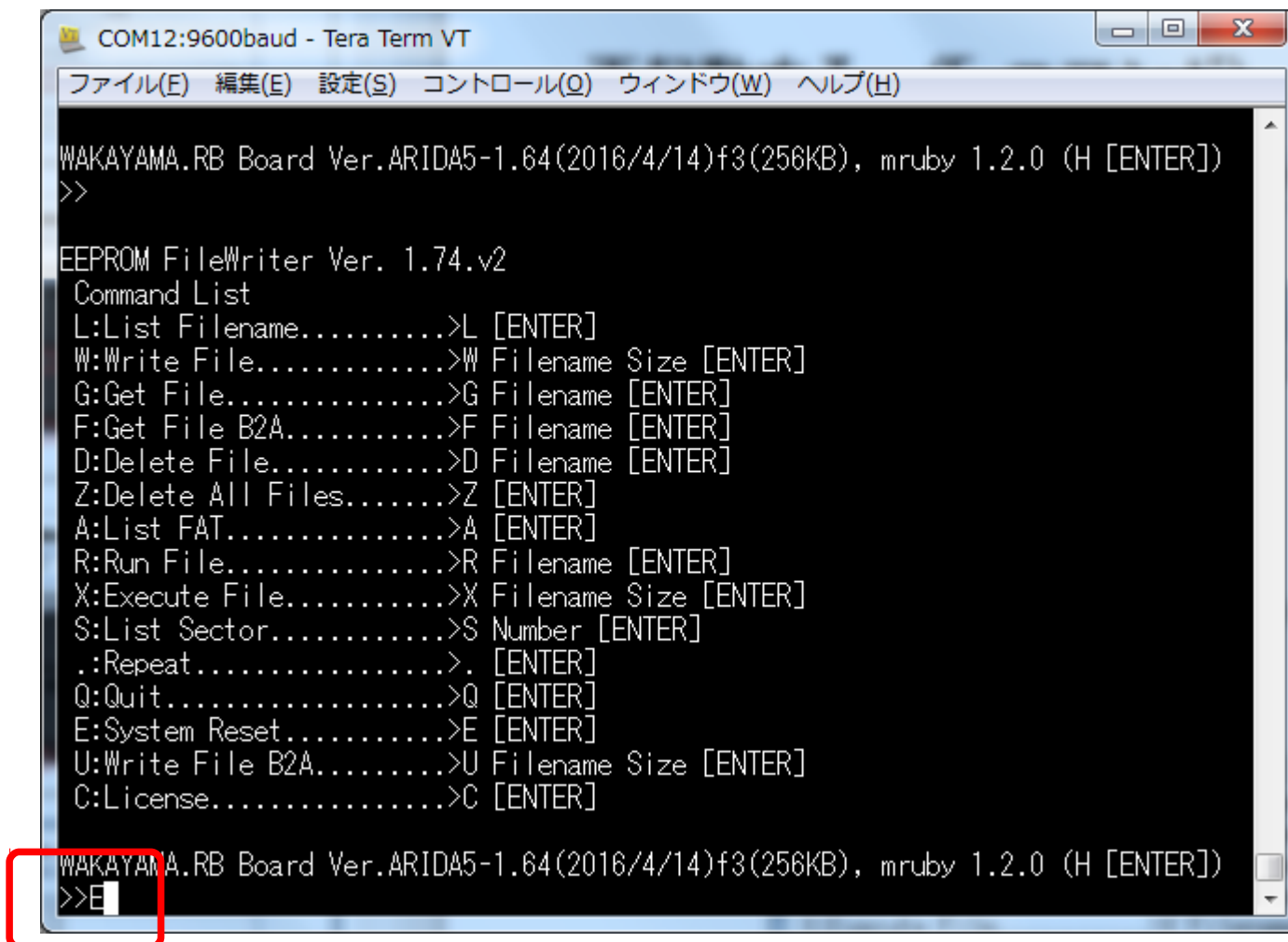
```
COM12:9600baud - Tera Term VT
ファイル(E) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
>>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
6:Hello GR-CITRUS!
7:Hello GR-CITRUS!

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>.
>R main
0:Hello GR-CITRUS!
1:Hello GR-CITRUS!
2:Hello GR-CITRUS!
3:Hello GR-CITRUS!
4:Hello GR-CITRUS!
5:Hello GR-CITRUS!
6:Hello GR-CITRUS!
7:Hello GR-CITRUS!

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (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。



The screenshot shows a Tera Term VT window titled "COM12:9600baud - 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)", and "ヘルプ(H)". The main text area displays the following content:

```
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>

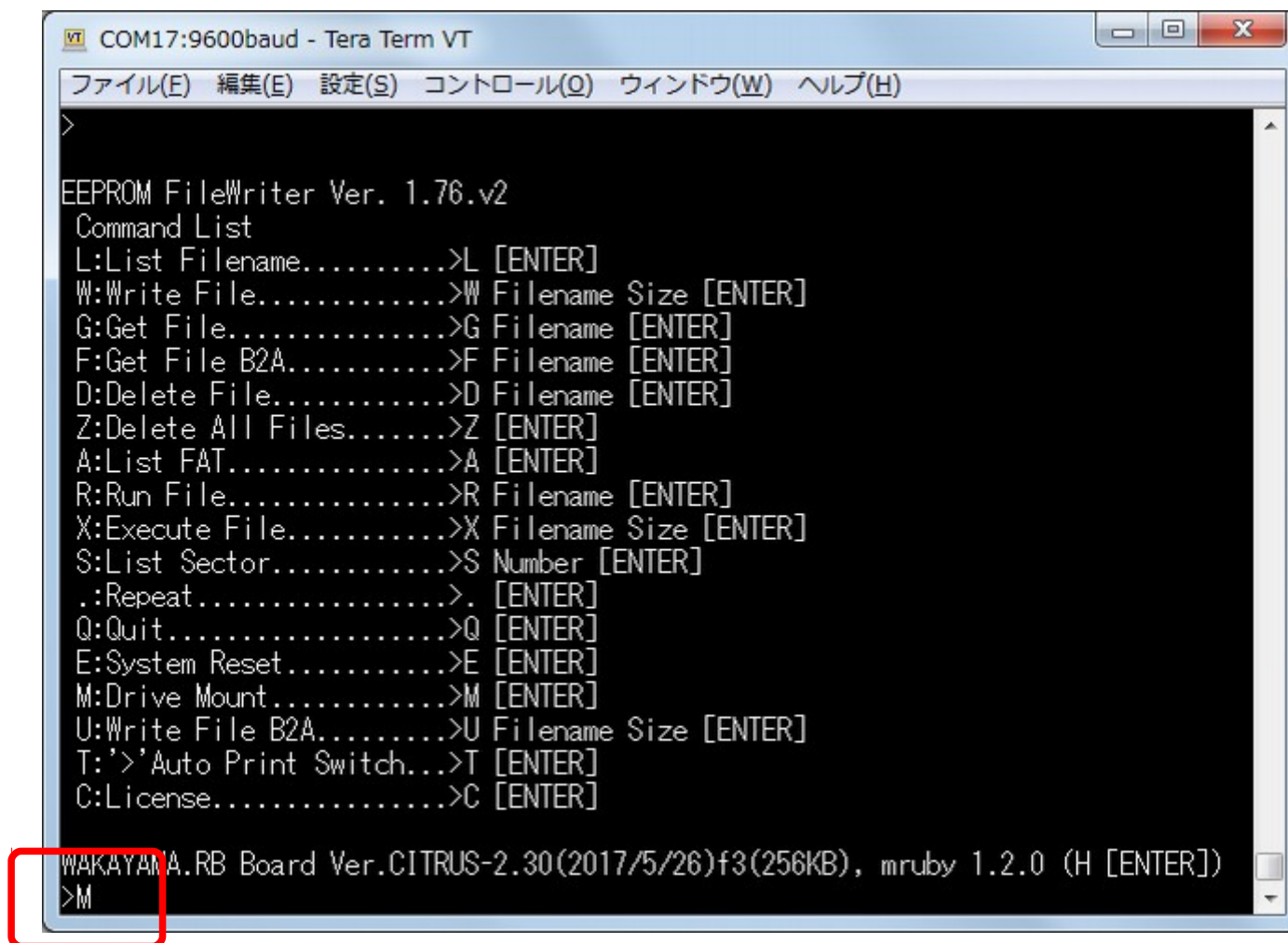
EEPROM FileWriter Ver. 1.74.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>E
```

The "E" command is highlighted with a red box, indicating it is the command to be entered to restart the microcontroller.

ドライブとしてマウントします。(M コマンド)

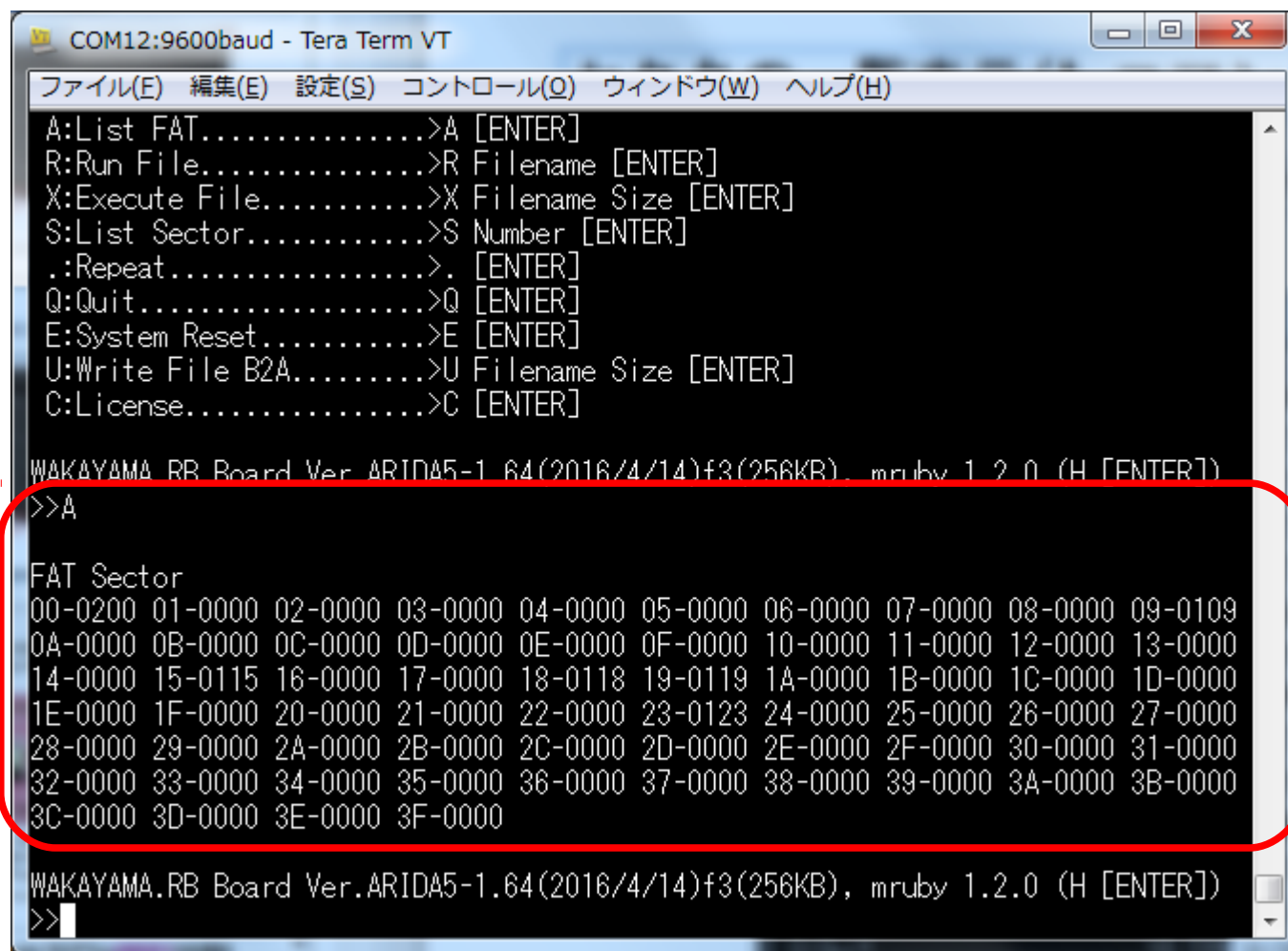
Mコマンドを入力すると、USBドライブとしてマウントされ、binファイルが書き込めるモードになります。
リセットボタンを押した動作と同じです。



```
COM17:9600baud - 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)
>
EEPROM FileWriter Ver. 1.76.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
M:Drive Mount.....>M [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
T:'>'Auto Print Switch...>T [ENTER]
C:License.....>C [ENTER]
WAKAYAMA A.RB Board Ver.CITRUS-2.30(2017/5/26)f3(256KB), mruby 1.2.0 (H [ENTER])
>M
```

セクタの一覧表示 (A コマンド)

Aコマンドを入力すると、セクタを一覧表示します。



The screenshot shows a Tera Term VT window titled "COM12:9600baud - 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 command list shows "A:List FAT.....>A [ENTER]". The output of the A command is displayed, showing a list of FAT sectors. A red rounded rectangle highlights the output text from "FAT Sector" to the final "WAKAYAMA.RB Board" line.

```
COM12:9600baud - 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)
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]
WAKAYAMA.RB Board Ver. ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>A
FAT Sector
00-0200 01-0000 02-0000 03-0000 04-0000 05-0000 06-0000 07-0000 08-0000 09-0109
0A-0000 0B-0000 0C-0000 0D-0000 0E-0000 0F-0000 10-0000 11-0000 12-0000 13-0000
14-0000 15-0115 16-0000 17-0000 18-0118 19-0119 1A-0000 1B-0000 1C-0000 1D-0000
1E-0000 1F-0000 20-0000 21-0000 22-0000 23-0123 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. ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>
```

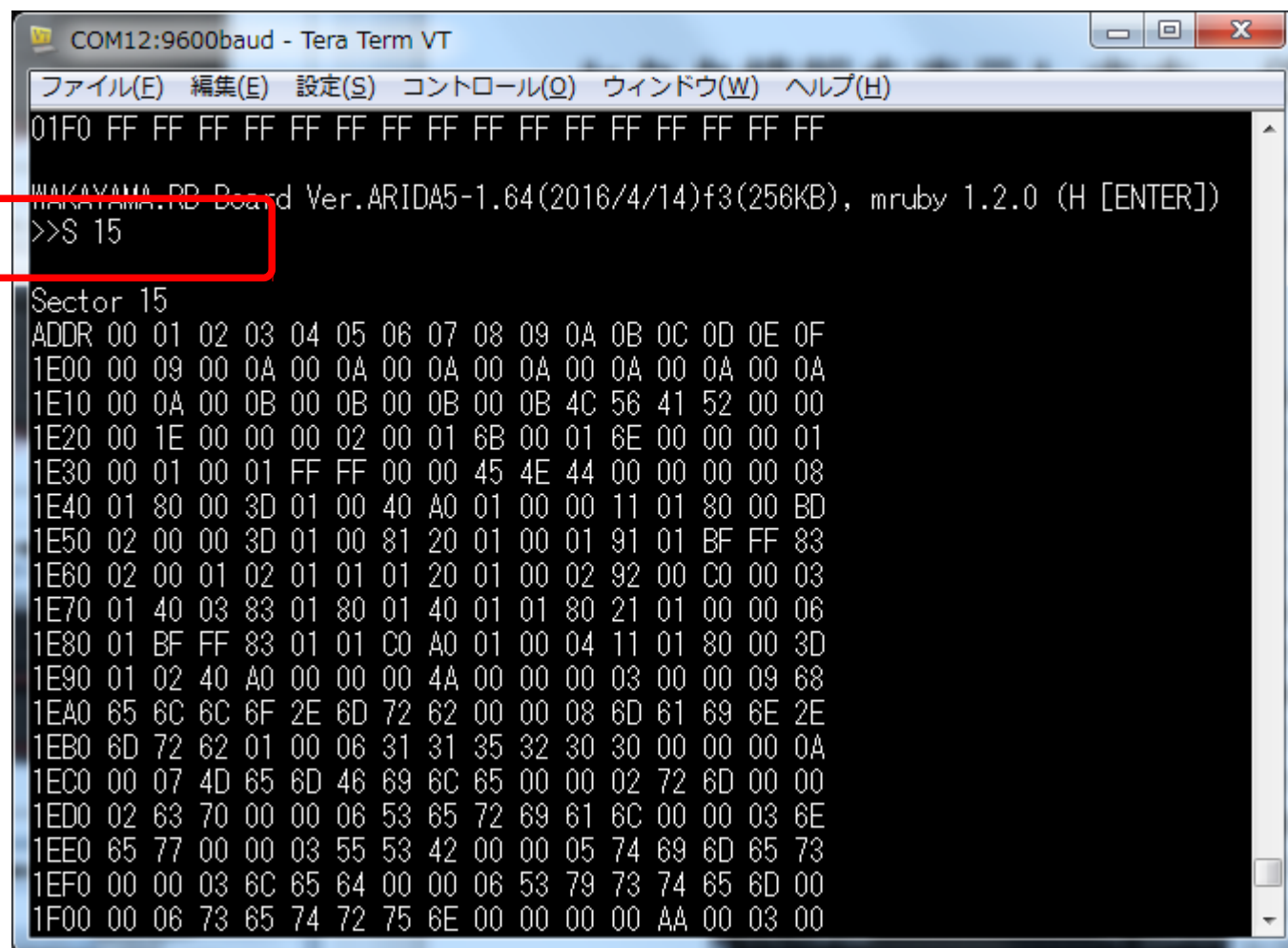

セクタ情報を表示します。(S コマンド)

Sコマンドは、セクタの情報を表示することができます。

Sの後にスペースで区切って、表示したいセクタ番号を書き、ENTERを押します。

番号は、Aコマンドで表示される番号です。

>S 番号



The screenshot shows a Tera Term VT window titled "COM12:9600baud - 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)", and "ヘルプ(H)". The main text area displays the following content:

```
01F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  
WAKAYAMA.RD-Board Ver.ARIDAS-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])  
>>S 15  
Sector 15  
ADDR 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  
1E00 00 09 00 0A 00 0A 00 0A 00 0A 00 0A 00 0A 00  
1E10 00 0A 00 0B 00 0B 00 0B 00 0B 4C 56 41 52 00 00  
1E20 00 1E 00 00 00 02 00 01 6B 00 01 6E 00 00 00 01  
1E30 00 01 00 01 FF FF 00 00 45 4E 44 00 00 00 00 08  
1E40 01 80 00 3D 01 00 40 A0 01 00 00 11 01 80 00 BD  
1E50 02 00 00 3D 01 00 81 20 01 00 01 91 01 BF FF 83  
1E60 02 00 01 02 01 01 01 20 01 00 02 92 00 C0 00 03  
1E70 01 40 03 83 01 80 01 40 01 01 80 21 01 00 00 06  
1E80 01 BF FF 83 01 01 C0 A0 01 00 04 11 01 80 00 3D  
1E90 01 02 40 A0 00 00 00 4A 00 00 00 03 00 00 09 68  
1EA0 65 6C 6C 6F 2E 6D 72 62 00 00 08 6D 61 69 6E 2E  
1EB0 6D 72 62 01 00 06 31 31 35 32 30 30 00 00 00 0A  
1EC0 00 07 4D 65 6D 46 69 6C 65 00 00 02 72 6D 00 00  
1ED0 02 63 70 00 00 06 53 65 72 69 61 6C 00 00 03 6E  
1EE0 65 77 00 00 03 55 53 42 00 00 05 74 69 6D 65 73  
1EF0 00 00 03 6C 65 64 00 00 06 53 79 73 74 65 6D 00  
1F00 00 06 73 65 74 72 75 6E 00 00 00 00 AA 00 03 00
```

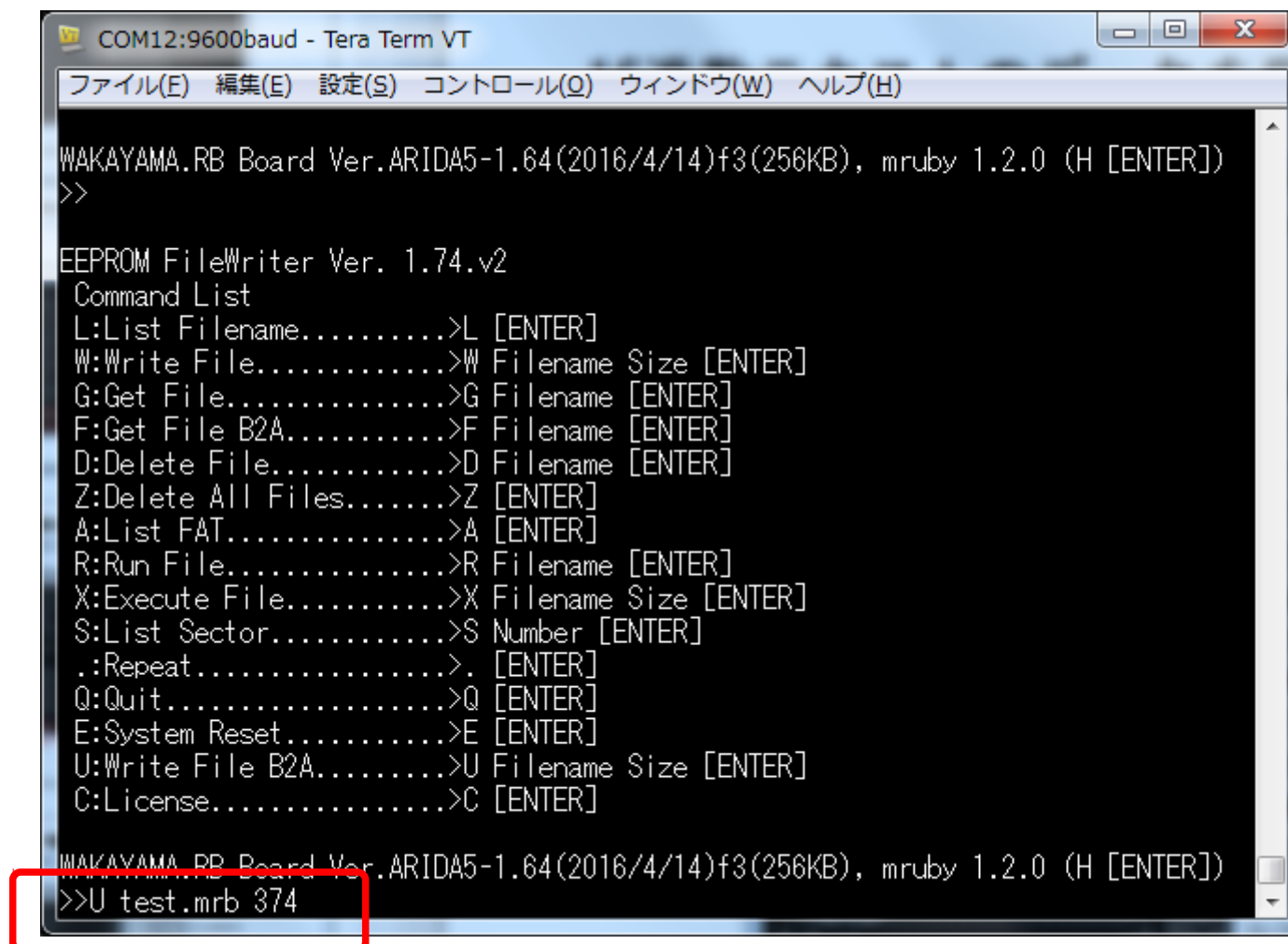
16進数テキストのデータを受信します。(U コマンド)

Uコマンドを用いて、16進数テキストデータを書き込みます。

U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。

ファイル受信方法はWコマンドと同じです。

>U ファイル名 ファイルサイズ



```
COM12:9600baud - 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)

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>

EEPROM FileWriter Ver. 1.74.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D>Delete File.....>D Filename [ENTER]
Z>Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>U test.mrb 374
```

Fの後にスペースで区切って、ファイル名を書き、ENTERキーを押します。

送信方法はGコマンドと同じですが、ファイル内容は16進数テキストで送信されます。

>F ファイル名

COM12:9600baud - 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)

test.rb 49 byte
test.mrb 187 byte
sketch.json 156 byte

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>F main.mrb

Waiting 50 50
478
Waiting 50 50 50

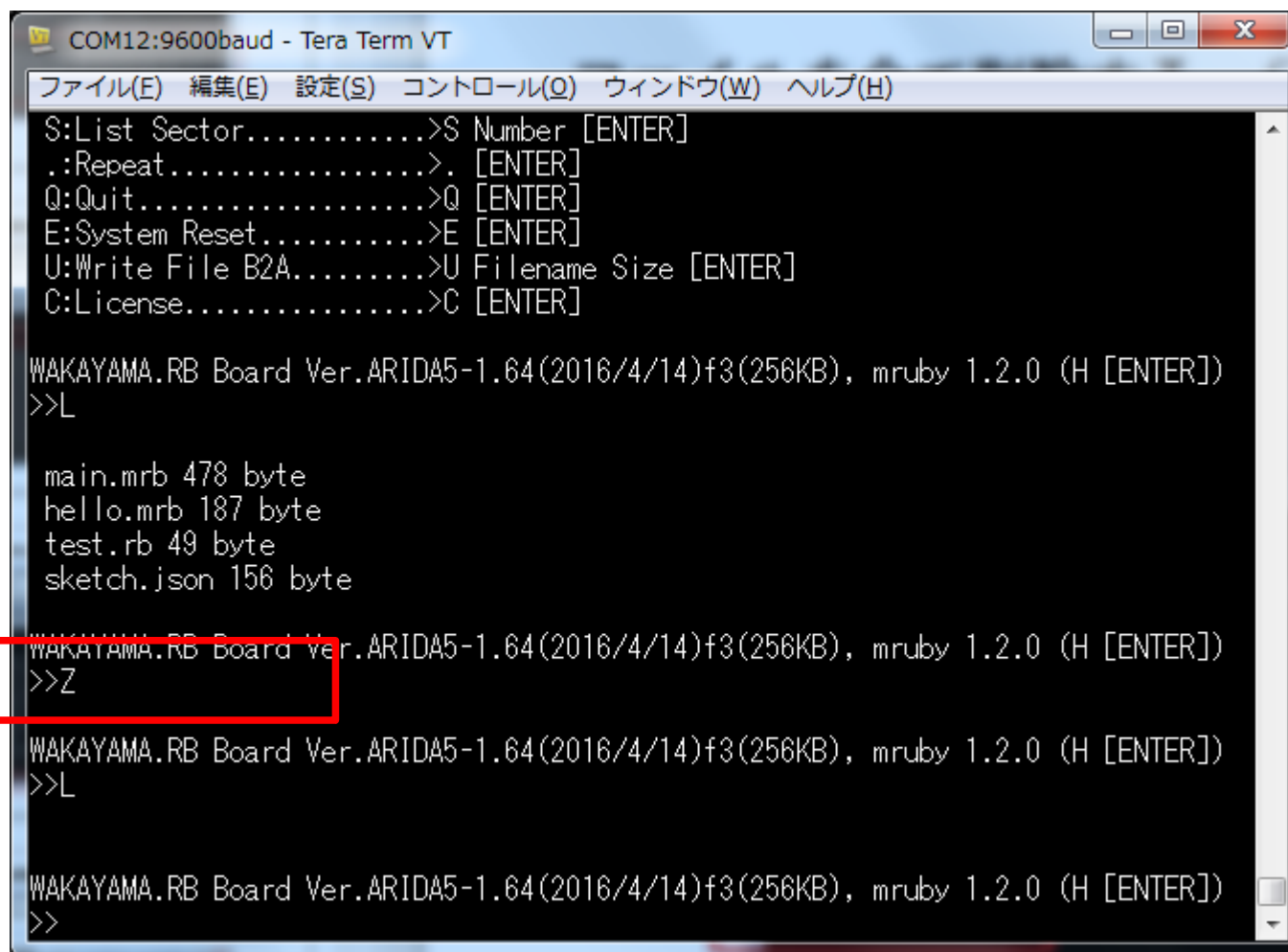
5249544530303033D6D0000001DE4D41545A30303030495245500000012A3030303000000007A0002
000600010000000D0100001101BFFF8302000002010041200100011200C000030140038301800140
0100C0210100000601BFFF83010100A000000004A0000000101000631313532303000000005000653
657269616C00000036E657700000355534200000574696D657300000036C656400000000AA00030007
000000000014000002000260180000602004015018000A001C0000302004015018040AE01804016
018001110200003D02804001028100200201403E028000BD0201403E0180C0A00180000602409583
018140A0018000290000000020000000000113A48656C6C6F2047522D434954525553210000000600
036C65640000012D0000035553420000077072696E746C6E000004746F5F7300000564656C617900
4442470000000078000100082F6D61696E2E72620000002B00010000000000000000D00000500
050005000500050006000700070007000D000D000D000D000000390001000000000000000001400
0007000800080008000900090009000900A000A000A000A000A000A000A000A000B000B000B000B
4C5641520000001E0000000200016B00016E0000000100010001FFFF0000454E440000000008

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>

ファイルを全て削除する。(Z コマンド)

ZコマンドはGR-CITRUSに保存しているファイルを削除します。
実際にはファイルシステムを初期化しています。

>Z [ENTER]



The screenshot shows a Tera Term VT window titled "COM12:9600baud - Tera Term VT". The menu bar includes "ファイル(F)", "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 a series of menu options and their corresponding actions, followed by a list of files and their sizes. The command ">>Z" is entered, which is highlighted by a red rectangle. The output shows the files being deleted.

```
COM12:9600baud - Tera Term VT
ファイル(F) 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]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L

main.mrb 478 byte
hello.mrb 187 byte
test.rb 49 byte
sketch.json 156 byte

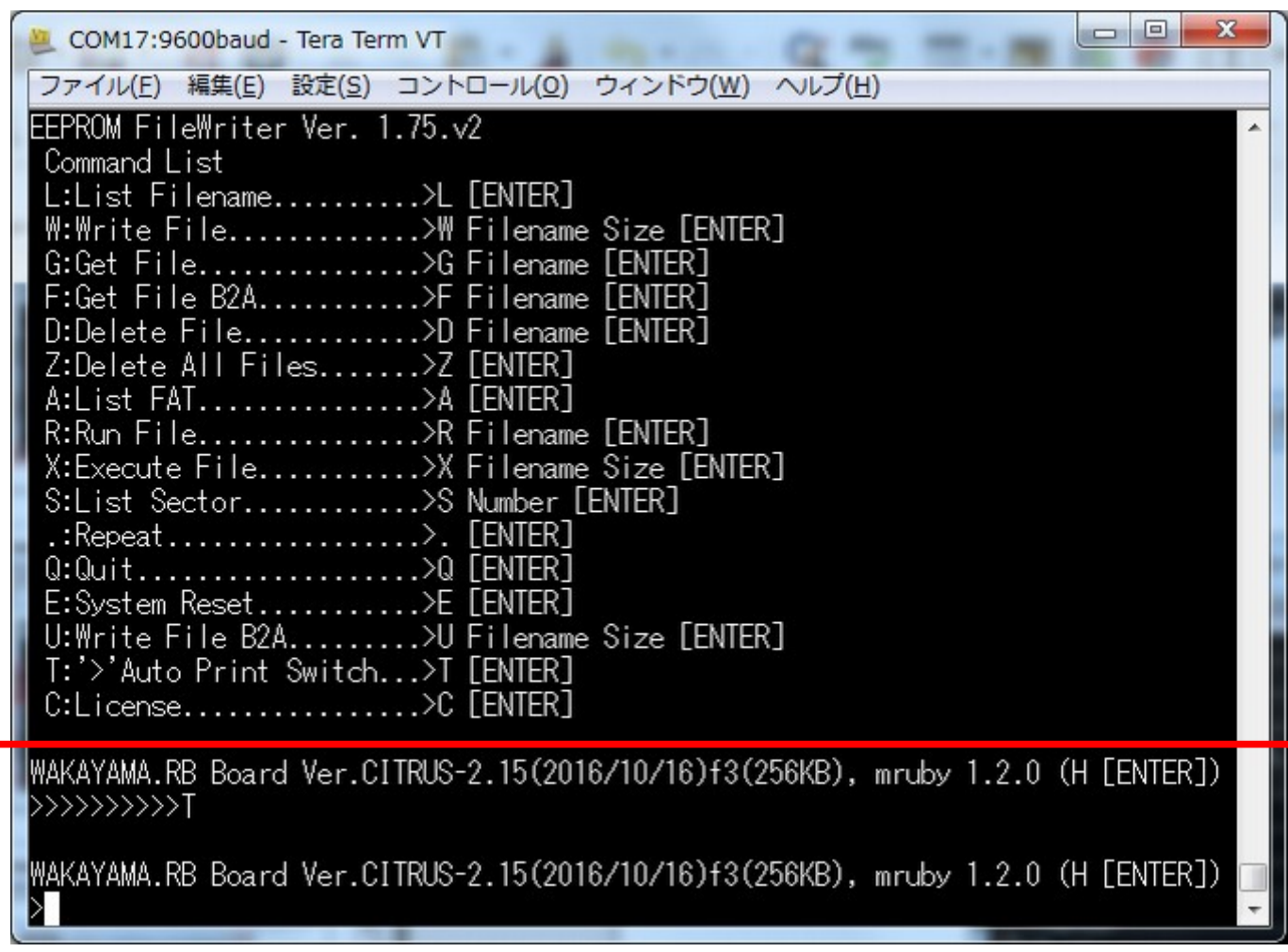
WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>Z

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>L

WAKAYAMA.RB Board Ver.ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>
```

ライセンス情報の表示 (T コマンド)

Tコマンドを入力すると、'>'の自動送信を停止/再開できます。



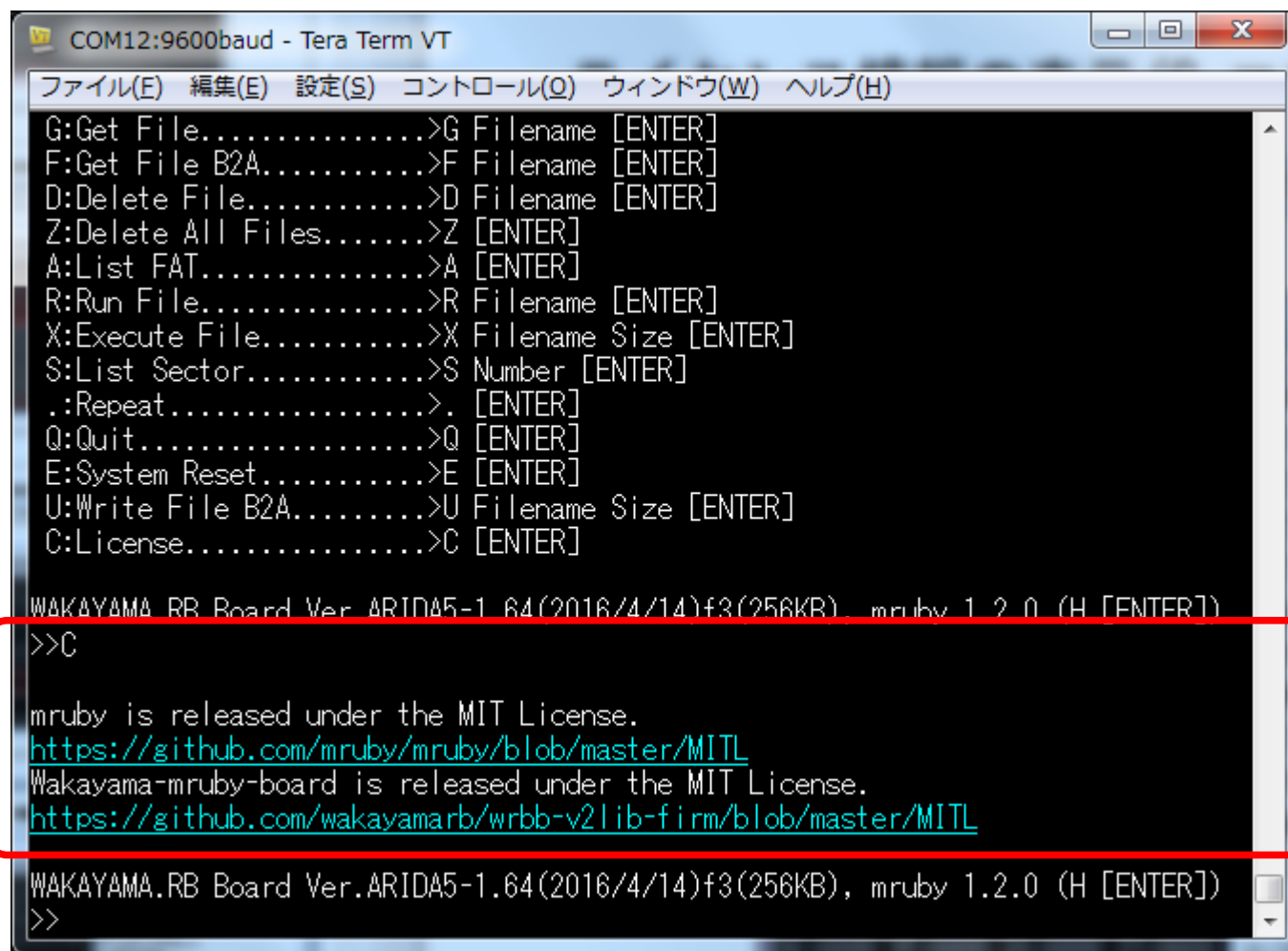
The screenshot shows a Tera Term VT window titled "COM17:9600baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(Q)", "ウィンドウ(W)", and "ヘルプ(H)". The main text area displays the "EEPROM FileWriter Ver. 1.75.v2" command list, which includes commands like L (List Filename), W (Write File), G (Get File), F (Get File B2A), D (Delete File), Z (Delete All Files), A (List FAT), R (Run File), X (Execute File), S (List Sector), . (Repeat), Q (Quit), E (System Reset), U (Write File B2A), T ('>' Auto Print Switch), and C (License). Below the command list, the text "WAKAYAMA.RB Board Ver.CITRUS-2.15(2016/10/16)f3(256KB), mruby 1.2.0 (H [ENTER])" is displayed. The character ">>>>>>>>T" is entered, and the text "WAKAYAMA.RB Board Ver.CITRUS-2.15(2016/10/16)f3(256KB), mruby 1.2.0 (H [ENTER])" is displayed again. A red rectangle highlights the bottom portion of the window, including the command list and the text area.

```
COM17:9600baud - 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. 1.75.v2
Command List
L:List Filename.....>L [ENTER]
W:Write File.....>W Filename Size [ENTER]
G:Get File.....>G Filename [ENTER]
F:Get File B2A.....>F Filename [ENTER]
D:Delete File.....>D Filename [ENTER]
Z:Delete All Files.....>Z [ENTER]
A:List FAT.....>A [ENTER]
R:Run File.....>R Filename [ENTER]
X:Execute File.....>X Filename Size [ENTER]
S:List Sector.....>S Number [ENTER]
.:Repeat.....>. [ENTER]
Q:Quit.....>Q [ENTER]
E:System Reset.....>E [ENTER]
U:Write File B2A.....>U Filename Size [ENTER]
T:'>' Auto Print Switch...>T [ENTER]
C:License.....>C [ENTER]

WAKAYAMA.RB Board Ver.CITRUS-2.15(2016/10/16)f3(256KB), mruby 1.2.0 (H [ENTER])
>>>>>>>>T
WAKAYAMA.RB Board Ver.CITRUS-2.15(2016/10/16)f3(256KB), mruby 1.2.0 (H [ENTER])
>
```


ライセンス情報の表示 (C コマンド)

Cコマンドを入力すると、ライセンス情報を示します。

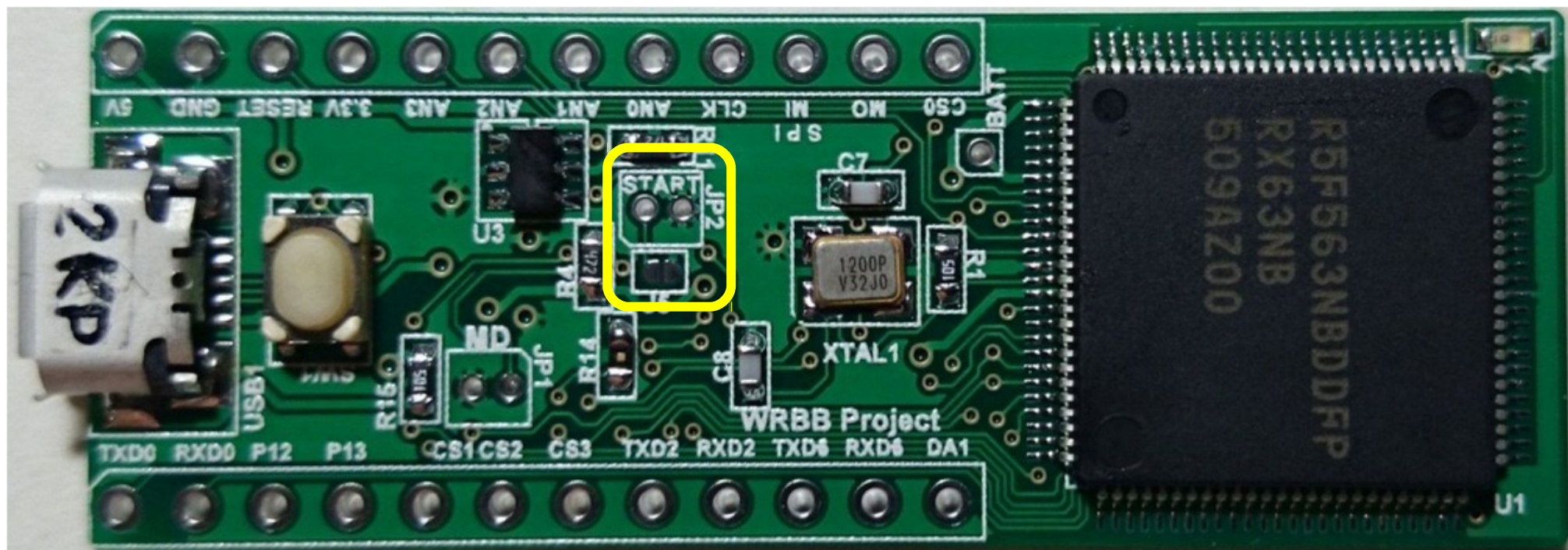


The screenshot shows a Tera Term VT window titled "COM12:9600baud - Tera Term VT". The menu bar includes "ファイル(E)", "編集(E)", "設定(S)", "コントロール(Q)", "ウィンドウ(W)", and "ヘルプ(H)". The command menu is displayed, listing various commands and their prompts. The "C:License.....>C [ENTER]" command is selected. Below the menu, the output of the "C" command is shown, enclosed in a red rounded rectangle. The output text is as follows:

```
WAKAYAMA RB Board Ver. ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>C
mruby is released under the MIT License.
https://github.com/mruby/mruby/blob/master/MITL
Wakayama-mruby-board is released under the MIT License.
https://github.com/wakayamarb/wrbb-v2lib-firm/blob/master/MITL
WAKAYAMA.RB Board Ver. ARIDA5-1.64(2016/4/14)f3(256KB), mruby 1.2.0 (H [ENTER])
>>
```

電源ONで即実行する方法

電源をONしてプログラムを即実行したい場合は、J10をショートさせるか、JP2にハーフピッチジャンパを取り付けて、ジャンパをショートさせてください。



即実行時に、Rubyファームにwrbb.xmlファイルがあり、Startタグで開始プログラム名が書かれているときには、該当プログラムを実行します。無い場合はmain.mrbが実行されます。main.mrbが無い場合は、コマンドモードになります。ただし、SDカードが読み込める場合はSDカードを検索し、SDカードにmrbファイルがある場合はそれを実行します。

rubyプログラム自動実行の仕組み

自動実行する場合のrubyプログラム実行条件 条件(1)

Rubyファームは、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。

条件(2)

mrbファイルが見つからない場合は、SDカードが読み込める状態であれば、SDカード内のmrbファイルを検索し、見つければそのmrbファイルをRubyボード内にコピーして実行します。

rubyプログラム自動実行の仕組み

条件(3)

wrbbs.xml ファイルが見つからない場合は、main.mrbファイルを検索します。
main.mrb ファイルが見つければ、main.mrbファイルを実行します。

条件(4)

main.mrbファイルが見つからない場合は、SDカードが読み込める状態であれば、SDカード内のmain.mrbファイルを検索し、見つければmain.mrbファイルをRubyボード内にコピーして実行します。

条件(5)

wrbbs.xml、main.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 GR-CITRUS!と10回出力されます。

```
usbout = Serial.new(0)
10.times do
  usbout.println("Hello GR-CITRUS!")
  delay(500)
end
```

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、実行が終了後、指定されたrubyプログラムが呼び出されます。

main.mrb 実行

```
sw = 1
10.times do
  led(sw)
  sw = 1 - sw
  delay(500)
end
System.setrun("hello.mrb")
```

hello.mrb 実行

```
usbout = Serial.new(0)
usbout.println(0, "Hello GR-CITRUS!")
led(1)
```

hello.mrb 終了

mrbcファイルの作成方法

コマンドラインからmrbcを実行してください。

```
$ ./mrbc main.rb
```

```
$ ls -l main.rb
```

```
-----rwx-----+ 1 minao None 865 6月 26 23:29 main.rb
```

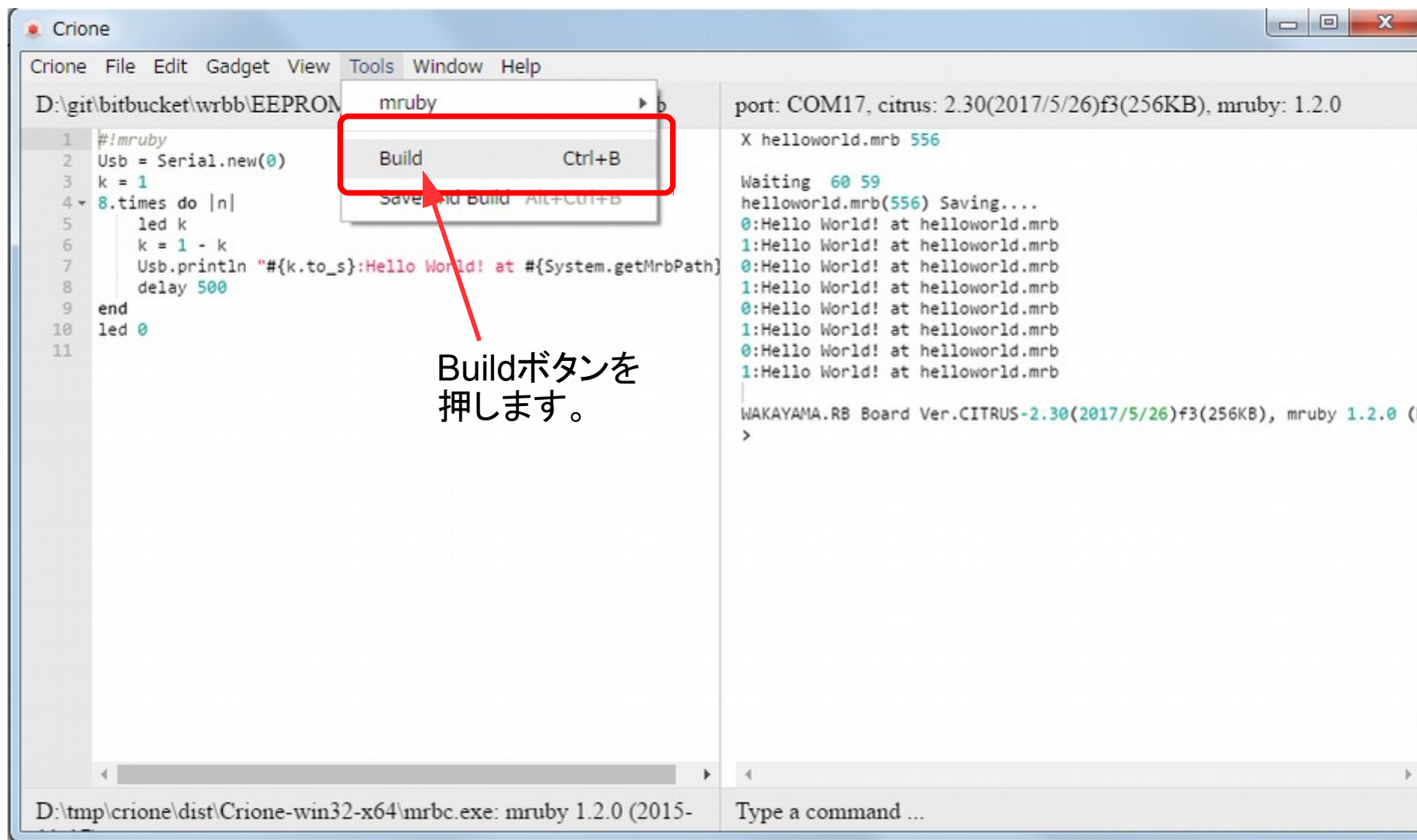
プログラムをデバッグしたい場合は、コンパイルオプションに `-g` を付けてコンパイルすることをお勧めします。エラーの行番号など詳しいエラーメッセージが出力されます。

```
$ ./mrbc -g main.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
```

mrubyファイルの作成方法

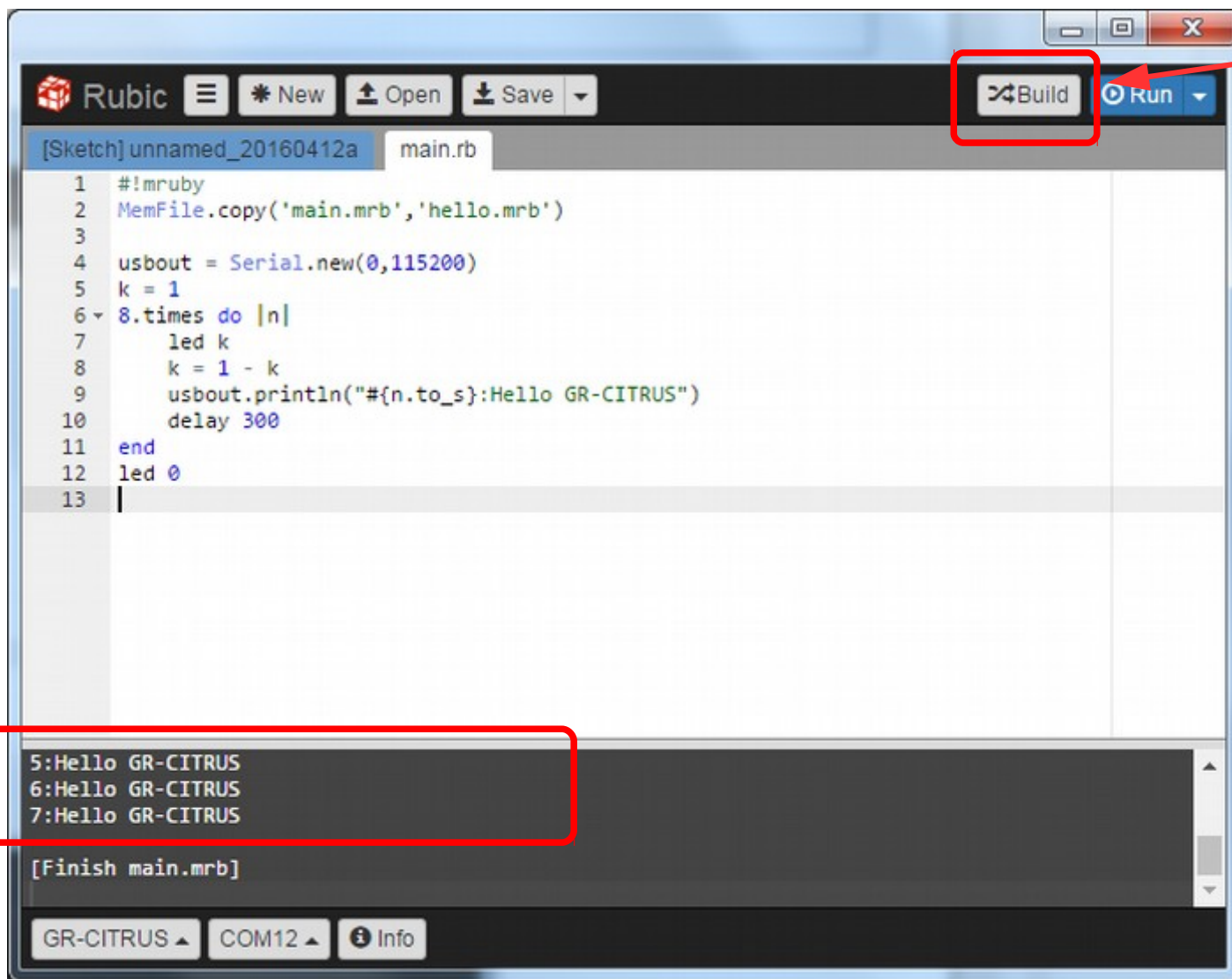
ogomさんが開発している「Crione」を使用すると、コマンドラインからmrbcを使うことなくmrubyファイルを作成することができます。



mrbcファイルの作成方法

きむしゅさんが開発している「Rubic」を使用すると、コマンドラインからmrbcを使うことなくmrbcファイルを作成することができます。

※Chromeアプリ版のRubicは、Chromeの仕様変更によりサポートされない可能性があります。



Buildボタンを
押します。

メソッドの説明 (V2ライブラリ)

グローバル定数

グローバル定数として以下の変数を使用可能です。

ON	1
OFF	0
HIGH	1
LOW	0
INPUT	0
OUTPUT	1

メソッドの説明 (V2ライブラリ)

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 (V2ライブラリ)

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ピン初期化 `initDac()`

アナログ出力ピンを初期化します。
初期化しないとアナログ出力しません。

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 (V2ライブラリ)

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設定後に、他のピンのpinMode設定をしてください。一度PWMに設定したピンは、リセットするまで変更できません。ショートしているPIOはINPUTに設定しておいてください。

アナログリファレンス `analogReference(mode)`

アナログ入力で使われる基準電圧を設定します。

mode: 0:DEFAULT : 5.0V Arduino互換, 1:INTERNAL : 1.1V 内蔵電圧, 2:EXTERNAL : AVREFピン供給電圧,
3:RAW12BIT : 3.3V

ディレイ `delay(value)`

指定の時間 (ms) 動作を止めます。

value: 時間 (msec)

※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

起動からのミリ秒数

メソッドの説明 (V2ライブラリ)

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

トーンを出力 `tone(pin, frequency[, duration])`

トーンを出力します。

pin: ピン番号

frequency: 周波数 Hz

duration: 出力を維持する時間[ms]。省略時、0指定時は出力し続ける。

トーンを停止 `noTone(pin)`

トーンを出力を停止します。

pin: ピン番号

メソッドの説明 (V2ライブラリ)

カーネルクラス

乱数の設定 `randomSeed (value)`

乱数を得るための種を設定します。
value: 種となる値

乱数の `random([min,] max)`

乱数を取得します。
min: 乱数の取りうる最小値。省略可
max: 乱数の取りうる(最大値 + 1)
maxは乱数の取りうる最大値に+1したものです。

メソッドの説明 (V2ライブラリ)

カーネルクラス

使用例

```
pinMode(4, INPUT)
pinMode(5, OUTPUT)

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

10.times do
  led(ON)
  delay(1000)
  led(OFF)
  delay(1000)
end
```

メソッドの説明 (V2ライブラリ)

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

コマンドモードの呼び出し `System.fileload()`

コマンドモードを呼び出します。

メソッドの説明 (V2ライブラリ)

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

システムのリセット System.reset()

システムをリセットします。電源ONスタート状態となります。

メソッドの説明 (V2ライブラリ)

システムクラス

SDカードを使えるようにします `System.useSD()`

SDカードを使えるように設定します。

戻り値

0: 使用不可, 1: 使用可能

WA-MIKANボード (WiFi) を使えるようにします `System.useWiFi()`

WA-MIKANボード (WiFi) を使えるように設定します。

戻り値

0: 使用不可, 1: 使用可能

MP3再生を使えるようにします: `System.useMP3(pausePin, stopPin)`

MP3再生を行えるように設定します。

0番ピンとGNDの間にスピーカーを接続してください。

pausePin: 再生中の一時停止に使用するピン番号です。LOWになると一時停止/再開を繰り返します。

stopPin: 再生を止めるときに使用するピン番号です。LOWになると停止します。

戻り値

0: 使用不可, 1: 使用可能

設定できるピンは、1, 3, 4, 6, 9, 10, 14, 15, 16, 17, 18番ピンの11個です。

メソッドの説明 (V2ライブラリ)

システムクラス

実行しているmrbファイルパスを取得します: `System.getMrbPath()`

実行しているmrbファイルパスを取得します。

戻り値

実行しているmrbファイルパス(ファイル名です)。

追加クラスを使用できるようにします: `System.use(className[, options])`

追加クラスを使用できるようにする。

className: クラス名です。'SD'、'WiFi'、'MP3' のいずれかです。

options: オプションの配列です。

SDはオプション無し。

WiFiはオプション無し。

MP3は再生中の一時停止に使用するピン番号と、再生を止めるときに使用するピン番号の配列を指定します。例) [3, 4]

戻り値

0: 使用不可

1: 使用可能

メソッドの説明 (V2ライブラリ)

システムクラス

追加クラスを使用できるようにします: `System.use?(ClassName[, Options])`

追加クラスを使用できるようにする。

ClassName: クラス名です。'SD'、'WiFi'、'MP3' のいずれかです。

Options: オプションの配列です。

SDはオプション無し。

WiFiはオプション無し。

MP3は再生中の一時停止に使用するピン番号と、再生を止めるときに使用するピン番号の配列を指定します。例) [3, 4]

戻り値

true: 使用可能

false: 使用不可

メソッドの説明 (V2ライブラリ)

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。
```

```
Usb = Serial.new(0, 115200)
```

```
if(System.use?("WiFi") == false) then
  Usb.println "WiFi Card can't use."
  System.exit()
end
```

```
Usb.println "WiFi Ready"
```

```
if(System.use?("MP3", [3, 4]) == false) then
  Usb.println "MP3 can't use."
  System.exit()
end
```

```
Usb.println "MP3 Ready"
```

メソッドの説明 (V2ライブラリ)

シリアルクラス

このクラスはポート毎にインスタンスを生成して使います。

シリアル通信の初期化 `Serial.new(num, bps)`

シリアル通信を初期化します。シリアル通信を使用する場合は、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
4: 12ピン送信/11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。

戻り値
シリアルのインスタンス

ボーレートの設定 `bps (baudrate)`

シリアル通信のボーレートを設定します。

baudrate: ボーレート

シリアルポートへの出力 `print([str])`

シリアルポートに出力します。

str: 文字列。省略時は何も出力しません設定できます。

メソッドの説明 (V2ライブラリ)

シリアルクラス

このクラスはポート毎にインスタンスを生成して使います。

シリアルポートへの出力 (¥r¥n付き) `println([str])`

シリアルポートに¥r¥n付きで出力します。
str: 文字列。省略時は改行のみ

シリアル受信チェック `available()`

シリアルポートに受信データがあるかどうか調べます。

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データ取得 `read()`

シリアルポートの受信データを取得します。

戻り値

読み込んだデータ配列

データは0x00~0xFFの値

メソッドの説明 (V2ライブラリ)

シリアルクラス

このクラスはポート毎にインスタンスを生成して使います。

シリアルポートへデータ出力 `write(buf, len)`

シリアルポートにデータを出力します。

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

シリアルデータをフラッシュします `flash()`

シリアルデータをフラッシュします。

メソッドの説明 (V2ライブラリ)

シリアルクラス

このクラスはポート毎にインスタンスを生成して使います。

使用例

```
USB_Out = Serial.new(0, 115200)
sw = 0

while(USB_Out.available() > 0) do    #何か受信があった
  USB_Out.read()
end

50.times do
  while(USB_Out.available() > 0) do #何か受信があった
    c = USB_Out.read()              #文字取得
    USB_Out.print c                  #読み込んだ文字をprintします
  end

  #LEDを点滅させます
  led sw
  sw = 1 - sw

  delay 500
end
```

```
USB_Out = Serial.new(0, 115200)
data = 0x30.chr + 0x31.chr + 0.chr + 0x32.chr + 0x33.chr + 0x0d.chr + 0x0a.chr
USB_Out.write(data, 7)

System.exit()
```

メソッドの説明 (V2ライブラリ)

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 (V2ライブラリ)

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

ファイルをコピーします `MemFile.cp(srcFilename, dstFilename[, mode])`

ファイルをコピーします。
srcFilename: コピー元ファイル名
dstFilename: コピー先ファイル名
mode: 0:上書きしない, 1:上書きする 省略時は上書きしない。

戻り値
成功: 1, 失敗: 0

メソッドの説明 (V2ライブラリ)

MemFileクラス (Flashメモリをメディアのように扱うクラス)

ファイルを削除します `MemFile.rm(Filename)`

ファイルを削除します。

Filename: 削除するファイル名

戻り値

成功: 1, 失敗: 000~0xFFが返る。ファイルの最後だったら-1が返る。

メソッドの説明 (V2ライブラリ)

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
  MemFile.write(0, data, 5 )
MemFile.close(0)

MemFile.cp('sample.txt', 'memfile.txt', 1)

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

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

メソッドの説明 (V2ライブラリ)

I2cクラス

このクラスはポート毎にインスタンスを生成して使います。

I2C通信を行うピンの初期化 `I2c.new(number)`

I2C通信を行うピンの初期化を行います。

num: 通信番号

1: SDA-0ピン, SCL-1ピン

2: SDA-5ピン, SCL-6ピン

3: SDA-7ピン, SCL-8ピン

4: SDA-12ピン, SCL-11ピン

戻り値

I2cのインスタンス

アドレスからデータを読み込み: `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

アドレスにデータを書き込みます `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

常に 0

メソッドの説明 (V2ライブラリ)

I2cクラス

このクラスはポート毎にインスタンスを生成して使います。

I2Cデバイスに対して送信を開始するための準備をする: `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 `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
常に 0

デバイスに受信シーケンスを発行しデータを読み出す `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 (V2ライブラリ)

I2cクラス

このクラスはポート毎にインスタンスを生成して使います。

送信バッファの末尾に数値を追加する `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 `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

受信バッファ内にあるデータ数を調べる `available()`

デバイスに対して受信バッファ内にあるデータ数を調べます。

戻り値

データ数

メソッドの説明 (V2ライブラリ)

I2cクラス

このクラスはポート毎にインスタンスを生成して使います。

使用例

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

#センサ接続ピンの初期化 (12番SDA, 11番SCL)
sensor = I2c.new(3)
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
sensor.write(@APTemp, APTemp_CTRL_REG1, APTemp_SAMPLING)
delay(100)

#気圧を取得します -----
#Address 0x28, 0x29, 0x2A, 0x2B, 0x2C
v0 = sensor.read( @APTemp, 0x28, 0x29)
v1 = sensor.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 = sensor.read( @APTemp, 0x2B, 0x2C)
if v2 > 32767
  v2 = v2 - 65536
end
t = v2 / 480.0 + 42.5
USB.println(a.to_s + ", " + t.to_s)
```

メソッドの説明 (V2ライブラリ)

I2cクラス

このクラスはポート毎にインスタンスを生成して使います。

使用例

```
USB = Serial.new(0, 115200)          #USBシリアル通信の初期化
#センサ接続ピンの初期化 (12番SDA, 11番SCL)
sensor = I2c.new(3)
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
sensor.write(@APTemp, APTemp_CTRL_REG1, APTemp_SAMPLING)    # 7Hz
delay(100)

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

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

メソッドの説明 (V2ライブラリ)

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 (V2ライブラリ)

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: サーボのチャンネル 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

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: サーボのチャンネル 0~9まで指定できます

戻り値

true: 割り当てられている

false: 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: サーボのチャンネル 0~9まで指定できます

メソッドの説明 (V2ライブラリ)

サーボクラス

使用例

```
g_pos = 0
g_inc = 10

USB = Serial.new(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
  end
end

Servo.detach(0)
```


メソッドの説明 (V2ライブラリ)

リアルタイムクロッククラス

RTCを起動します: `Rtc.init()`

戻り値

0: 起動失敗

1: 起動成功

`init()` を実行すると日時がリセットされます。

RTCを停止します: `Rtc.deinit()`

RTCを停止します。

戻り値

0: 失敗

1: 成功

RTCの日時をセットします: `Rtc.setTime(array)`

RTCの日時をセットします。

array: 年 (0000-9999), 月 (1-12), 日 (1-31), 時 (0-23), 分 (0-59), 秒 (0-59) の配列

戻り値

0: 失敗

1: 成功

メソッドの説明 (V2ライブラリ)

リアルタイムクロッククラス

RTCの日時を取得します: `Rtc.getTime()`

RTCの日時を取得します。

戻り値は以下の値が配列で返ります

year: 年 (2000-2099)

month: 月 (1-12)

day: 日 (1-31)

hour: 時 (0-23)

minute: 分 (0-59)

second: 秒 (0-59)

weekday: 曜日 (0-6) 0: 日, 1: 月, 2: 火, 3: 水, 4: 木, 5: 金, 6: 土

メソッドの説明 (V2ライブラリ)

リアルタイムクロッククラス

使用例

```
USB = Serial.new(0, 115200)          #USBシリアル通信の初期化
Rtc.init
Rtc.setDateTime([2016, 4, 16, 17, 0, 0])

15.times do |i|
  led(i % 2)
  year, mon, da, ho, min, sec = Rtc.getTime()
  USB.println(year.to_s + "/" + mon.to_s + "/" + da.to_s + " " + ho.to_s + ":" + min.to_s + ":" +
sec.to_s)
  delay(500)
end
```

メソッドの説明 (V2ライブラリ)

SDカードクラス

System.useSD() を呼んでおく必要があります。

ファイルのオープン SD.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 SD.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 SD.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 (V2ライブラリ)

SDカードクラス `System.useSD()` を呼んでおく、または `System.use('SD')` しておく必要があります。

Openしたファイルからの読み込み `SD.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 `SD.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

Openしたファイルの書き込みをフラッシュします: `SD.flush(number)`

Openしたファイルの書き込みをフラッシュします。

number: ファイル番号 0 または 1

メソッドの説明 (V2ライブラリ)

SDカードクラス `System.useSD()` を呼んでおく、または `System.use('SD')` しておく必要があります。

Openしたファイルのサイズを取得します: `SD.size(number)`

Openしたファイルのサイズを取得します。
number: ファイル番号 0 または 1

戻り値
ファイルサイズ

Openしたファイルのseek位置を取得します: `SD.position(number)`

Openしたファイルのseek位置を取得します。
number: ファイル番号 0 または 1

戻り値
シーク位置

メソッドの説明 (V2ライブラリ)

SDカードクラス `System.useSD()` を呼んでおく、または `System.use('SD')` しておく必要があります。

ディレクトリを作成する: `SD.mkdir(dirname)`

ディレクトリを作成する。

`dirname`: 作成するディレクトリ名

戻り値

成功: 1, 失敗: 0

ファイルを削除します: `SD.remove(filename)`

ファイルを削除します。

`filename`: 削除するファイル名

戻り値

成功: 1, 失敗: 0

ファイルをコピーする: `SD.copy(srcfilename, distfilename)`

ファイルをコピーする。

`srcfilename`: コピー元ファイル名

`distfilename`: コピー先ファイル名

戻り値

成功: 1, 失敗: 0

メソッドの説明 (V2ライブラリ)

SDカードクラス `System.useSD()` を呼んでおく、または `System.use('SD')` しておく必要があります。

ディレクトリを削除します: `SD.rmdir(dirname)`

ディレクトリを削除します。
dirname: 削除するディレクトリ名

戻り値
成功: 1, 失敗: 0

ファイルが存在するかどうか調べる: `SD.exists(filename)`

ファイルが存在するかどうか調べます。
filename: 調べるファイル名

戻り値
存在する: 1, 存在しない: 0

ファイルをフラッシュメモリにコピーします: `SD.cpmem(SDFile, MemFile[, mode])`

SDカードのファイルをフラッシュメモリにコピーします。
SDFile: SDカードのファイル名
MemFile: フラッシュメモリのコピー先ファイル名
mode: 0上書きしない, 1:上書きする

戻り値
成功: 1, 失敗: 0

メソッドの説明 (V2ライブラリ)

SDカードクラス `System.useSD()` を呼んでおく、または `System.use('SD')` しておく必要があります。

使用例

```
if(System.use?('SD') == false)then
  System.exit
end

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

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

SD.open(0, 'sample.txt', 0)
while(true)do
  c = SD.read(0)
  if(c < 0)then
    break
  end
  USB.write(c.chr, 1)
end

SD.close(0)
System.exit()
```

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

ステーションモードを設定する: WiFi.setMode(mode)

ステーションモードを設定します。

mode: 1:Station, 2:SoftAP, 3:Station + SoftAP1

戻り値

ESP8266の戻り値

応答のシリアル出力設定: WiFi.serialOut(mode[, serialNumber])

ESP8266に送信したコマンドの応答をシリアル出力するときに設定します。

mode: 0:出力しない, 1:出力する

serialNumber: 出力先のシリアル番号

戻り値

無し

ATコマンドを送信する: WiFi.at(command[, mode])

ATコマンドを送信します。

commnad: ATコマンド文字列

mode: 0:'AT+' を自動追加する、1:'AT+' を自動追加しない

戻り値

ESP8266の戻り値

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

WiFi接続する: WiFi.connect(SSID, Passwd)

WiFiアクセスポイントの接続します。

SSID: WiFiのSSID

Passwd: パスワード

戻り値

ESP8266の戻り値

IPアドレスとMACアドレスの表示: WiFi.ipconfig()

IPアドレスとMACアドレスを表示します

戻り値

ESP8266の戻り値

USBポートとESP8266をシリアルで直結します: WiFi.bypass()

USBポートとESP8266をシリアルで直結します。

リセットするまで、処理は戻りません。

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

ESP8266のソフトのバージョンを取得する: `WiFi.version()`

ESP8266のソフトのバージョンを取得します。

戻り値
ESP8266の戻り値

WiFiを切断します: `WiFi.disconnect()`

WiFiを切断します。

戻り値
ESP8266の戻り値

複数接続可能モードの設定: `WiFi.multiConnect(mode)`

複数接続可能モードの設定をします。
mode: 0:1接続のみ, 1:4接続まで可能

戻り値
ESP8266の戻り値

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

http GET結果をSDカードに保存する: WiFi.httpGetSD(Filename, URL[, Headers])

http GET結果をSDカードに保存します。

Filename: 保存するファイル名

URL: URL

Headers: ヘッダに追記する文字列の配列

戻り値

0: 失敗

1: 成功

2: SDカードが使えない

3: 送信データファイルをオープンできなかった

4: 送信データサイズを読み込めなかった

5: 送信データファイルを読み込めなかった

6: 受信用ファイルをオープンできなかった

7: 受信したファイルの生成に失敗した

http GETプロトコルを送信する: WiFi.httpGet(URL[, Headers])

http GETプロトコルを送信します。送信のみで、結果の受信しません。

URL: URL

Headers: ヘッダに追記する文字列の配列

戻り値

0: 失敗

1: 成功

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

TCP/UDPの接続を閉じる: WiFi.cClose(number)

TCP/UDPの接続を閉じます。
number: 接続番号 (1~4)

戻り値
ESP8266の戻り値

UDP接続を開始する: WiFi.udpOpen(number, IP_Address, SendPort, ReceivePort)

UDP接続を開始します。
number: 接続番号 (1~4)
IP_Address: 通信相手アドレス
SendPort: 送信ポート番号
ReceivePort: 受信ポート番号

戻り値
ESP8266の戻り値

指定接続番号にデータを送信する: WiFi.send(number, Data[, length])

指定接続番号にデータを送信します。
number: 接続番号 (1~4)
Data: 送信するデータ
length: 送信データサイズ

戻り値
送信データサイズ

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

指定接続番号からデータを受信する: WiFi.recv(number)

指定接続番号からデータを受信します。

number: 接続番号 (1~4)

戻り値

受信したデータの配列 ただし、256以下

SDカードのファイルをhttpPOSTする: WiFi.httpPostSD(URL, Headers, Filename)

SDカードのファイルをhttp POSTします。

URL: URL

Headers: ヘッダに追記する文字列の配列

Filename: POSTするファイル名

戻り値

0: 失敗

1: 成功

2: SDカードが使えない

2: SDカードが使えない

3: 送信データファイルサイズを読み込めなかった

4: 送信ヘッダファイルを書き込みオープンできなかった

5: 送信ヘッダファイルサイズを読み込めなかった

6: 送信ヘッダファイルを読み込みオープンできなかった

7: 送信データファイルを読み込みオープンできなかった

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

http POSTする: WiFi.httpPost(URL, Headers, data)

http POSTします。送信のみで結果は受信しません。

URL: URL

Headers: ヘッダに追記する文字列の配列

Data: POSTデータ

戻り値

0: 失敗

1: 成功

httpサーバを開始します: WiFi.httpServer([Port])

httpサーバを開始します。アクセスの有無で返り値が変わります。

SDカードが必須となります。

ポート番号を省略したときはアクセス確認します。

Port: 待ち受けポート番号

-1: サーバ停止

戻り値

0: アクセスはありません

1: アクセスあり

2: SDカードが使いません

3: ファイルのアクセスに失敗しました

クライアントからアクセスがあるとき

GET: パスが返ります

GET以外、ヘッダの1行目が返ります

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

使用例

```
#ESP8266を一度停止させる(リセットと同じ)
pinMode(5, 1)
digitalWrite(5, 0)    # LOW:Disable
delay 500
digitalWrite(5, 1)    # LOW:Disable

Usb = Serial.new(0, 115200)

if( System.use?('WiFi') == false)then
  Usb.println "WiFi Card can't use."
  System.exit()
end

Usb.print WiFi.version
```

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

使用例

```
#ESP8266を一度停止させる(リセットと同じ)
pinMode(5, 1)
digitalWrite(5, 0)    # LOW:Disable
delay 500
digitalWrite(5, 1)    # LOW:Disable
```

```
Usb = Serial.new(0, 115200)
```

```
if( System.use?('WiFi') == false)then
  Usb.println "WiFi Card can't use."
  System.exit()
end
```

```
Usb.println "GR-CITRUS[bypass]"
```

```
WiFi.bypass()
```

#GR-CITRUSはESP8266とUSBシリアル通信が接続したままとなります。
#ターミナルを使って、ESP8266との通信テストをすることができます。

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

使用例

```
#ESP8266を一度停止させる(リセットと同じ)
pinMode(5, 1)
digitalWrite(5, 0)    # LOW:Disable
delay 500
digitalWrite(5, 1)    # LOW:Disable

Usb = Serial.new(0, 115200)

if( System.use?('WiFi') == false)then
  Usb.println "WiFi Card can't use."
  System.exit()
end

Usb.println WiFi.disconnect
Usb.println WiFi.setMode 3 #Station-Mode & SoftAPI-Mode
Usb.println WiFi.connect("TAROSAY", "****")
Usb.println WiFi.ipconfig
Usb.println WiFi.multiConnect 1

for value in 1..10
  Usb.println WiFi.httpGet("192.168.1.58:3000/?value1=" + value.to_s + "&value2=" +
    (value*value).to_s).to_s
end

heds=["User-Agent: curl"]
Usb.println WiFi.httpGetSD("wether1.htm", "wttr.in/wakayama").to_s
Usb.println WiFi.httpGetSD("wether2.htm", "wttr.in/wakayama", heds).to_s
Usb.println WiFi.httpGetSD("yahoo.htm", "www.yahoo.co.jp").to_s
Usb.println WiFi.httpGetSD("google.htm", "www.google.co.jp").to_s
```

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

使用例

```
#UDP通信, WA-MIKAN 受信:5555, 送信: 5556
Usb.println WiFi.udpOpen(4, "192.168.1.44", 5555, 5556)

Usb.println "UDP受信した分がarray配列で返ります"
1000.times do
  array = WiFi.recv 4 #受信データがない場合は array[0]に -1 が返ります
  if(array[0] >= 0) then
    for var in array do
      Usb.println var.to_s
    end
  end
  delay 10
end
```

```
#UDP通信, WA-MIKAN 受信:5555, 送信: 5556
Usb.println WiFi.udpOpen(4, "192.168.1.44", 5555, 5556)

100.times do
  WiFi.send 4, "hoho01111122222¥r¥n"
  delay 25
end
Usb.println WiFi.send(4, 0x02.chr + "bcdefghijklmn" + 0x03.chr + "ddd¥r¥n").to_s

Usb.println WiFi.cClose 4
Usb.println WiFi.disconnect
```

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

使用例

```
#http POST
```

```
header=["User-Agent: gr-citrus", "Accept: application/json", "Content-type: application/json"]  
body = ' { "name" : "tarosay" } '
```

```
WiFi.httpPost("192.168.1.52:3000", header, body)
```

```
# body.json ファイルをPOSTします
```

```
WiFi.httpPostSD("192.168.1.52:3000", header, "body.json")
```

メソッドの説明 (V2ライブラリ)

WiFiクラス

System.WiFi() を呼んでおく、またはSystem.use('WiFi') しておく必要があります。

使用例

```
header0 = "HTTP/1.1 200 OK\r\nServer: GR-CITRUS\r\nContent-Type: text/html\r\n"
header0 += "Date: Sun, 13 Nov 2016 12:00:00 GMT\r\nConnection: close\r\n"
body0 = '<html><head><meta http-equiv="Content-Type" content="text/html; charset=utf-8">'
body0 += '<title>RAMUNE SHOOTER</title></head>'

Usb.println WiFi.httpServer(80).to_s #-> 80ポートでhttp受信します
while(true) do
  res = WiFi.httpServer #-> アクセス確認しています。
  Usb.println res.to_s #-> 0のときはアクセスなし、GETのときはパスをそれ以外はヘッダ先頭行が返る

  if(res == "/exit")
    body1 = '<body><h1 align="center">終了します。</h1></body>' + "\r\n\r\n"
    header1 = "Content-Length: " + (body0 + body1).length.to_s + "\r\n\r\n"
    WiFi.send(0, header0)
    WiFi.send(0, header1)
    WiFi.send(0, body0)
    WiFi.send(0, body1)
    break
  elsif(res != 0)
    Usb.println "Else:" + res.to_s
    body1 = '<body><h1 align="center">エラーです。</h1></body>' + "\r\n\r\n"
    header1 = "Content-Length: " + (body0 + body1).length.to_s + "\r\n\r\n"
    WiFi.send(0, header0)
    WiFi.send(0, header1)
    WiFi.send(0, body0)
    WiFi.send(0, body1)
  end
  delay 100
end
WiFi.httpServer(-1) #-> サーバを停止します
```