

# ハードウェア I/O 1 day1 15:00 ~

KEK, High Energy Accelerator Research Organization

帯名 崇

(takashi.obina@kek.jp)

2018/11/01 ~ 02 EPICS入門セミナー@KEK つくばキャンパス (3号館1階会議室)

## このセッションでのおはなし

Raspberry Pi + EPICS という環境で、ハードウェアを制御する。今回は最も簡単な例として、GPIO (General Purpose Input/output) を使ってLED を点灯させる。

1. 実習で使うモノの確認
  - Raspberry Pi, ブレッドボード、LED、抵抗他
  - 壊さないための注意
2. EPICSを使わずに動作確認
3. EPICSでLED制御
4. GUIを作成して遠隔制御
5. 最後に

お手元の手引書に実習の詳細は記載してあります

# Raspberry Pi とは何か

Homepage: <https://www.raspberrypi.org/>

Wikipediaの記載↓

## Raspberry Pi

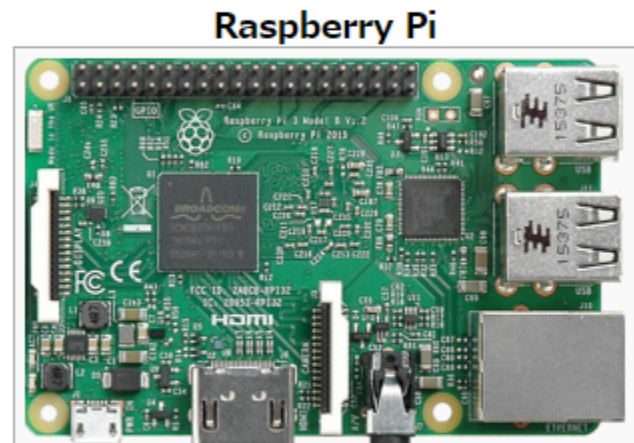
出典: フリー百科事典『ウィキペディア (Wikipedia) 』

**Raspberry Pi** (ラズベリー パイ) は、ARMプロセッサを搭載したシングルボードコンピュータ。イギリスのラズベリーパイ財団によって開発されている。日本では略称として**ラズパイ**とも呼ばれる。

主に教育で利用することを想定しているが、IoTが隆盛した2010年代後半以降は、安価に入手できるIoT機器として趣味や業務に広く用いられている。IoT教育においては、ソフトウェア開発に強いRaspberry Piと、ハードウェア開発に強いArduinoの組み合わせが一般的である。

### 目次 [非表示]

- 1 概要
- 2 主な仕様
- 3 ギャラリー
- 4 ソフトウェア
  - 4.1 オペレーティングシステム
  - 4.2 Mathematica
- 5 参照



Raspberry Pi 3 Model B

製造元	Raspberry Pi Foundation
種別	シングルボードコンピュータ
発売日	2012年2月29日
売上台数	累計1,100万台（2016年11月25日時点） <sup>[1]</sup>

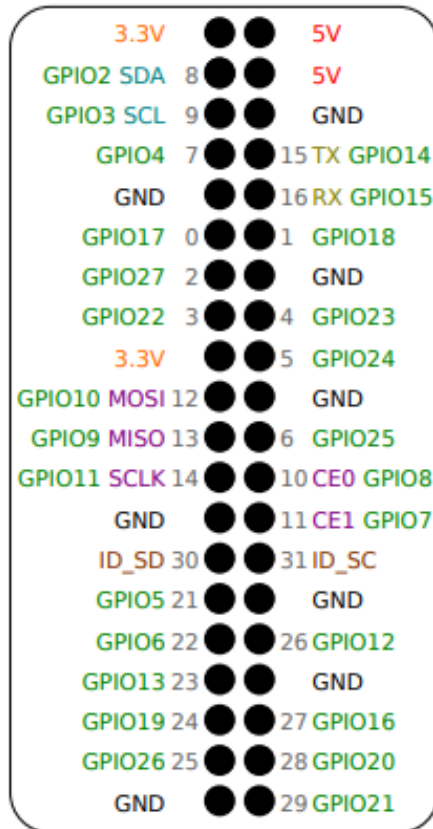
# お手元にあるのは

- Raspberry Pi 3 model B+
  - 2018年3月に発売開始。Ethernetが1Gbpsになった。
  - RSコンポーネンツから購入可能。
  - 今回は使用しませんが、無線LANも搭載
  - Pi2 に比べると消費電力も大きいのでACアダプタ推奨(2.5A)
- 主な外部インターフェース
  - USB × 4
  - Ethernet
  - GPIO
  - HDMI
  - microUSB : (ほぼ電源用)

※今回の実習では使いませんが、キーボード、マウス、USBをつなげば端末として使用可能です。GUI（デスクトップ画面）が必要な人はVNC 経由で接続してください

# GPIO

- <https://github.com/splitbrain/rpiplusleaf>
- <https://pinout.xyz>
- [https://elinux.org/RPi\\_Low-level\\_peripherals](https://elinux.org/RPi_Low-level_peripherals)



## Raspberry Pi B+ Leaf

Power (5 Volts)

Power (3 Volts)

Ground

WiringPi GPIO

BCM GPIO

I2C Interface

UART Interface

SPI Interface

ID EEPROM Interface

splitbrain.org

このLeafは各自に配布しています。  
穴をあけるとピンに嵌めることができ便利です。



# pinoutコマンド

各自、実行して確認してください

```
kektaro@raspberrypi: ~ $ pinout
-----
 00000000000000000000 J8
 10000000000000000000
-----
Pi Model ???V1.3
+-----+
| D |   | SoC |
| S |   +-----+
| I |   |
+-----+
+-----+
| USB |
+-----+
+-----+
| USB |
+-----+
+-----+
| C |   | Net |
| S |   +-----+
| I |   |
| A |   |
| V |   +-----+
+-----+
pwr  | HDMI |
+-----+
Revision      : a020d3
SoC           : BCM2837
RAM          : 1024Mb
Storage      : MicroSD
USB ports    : 4 (excluding power)
Ethernet ports : 1
Wi-fi       : False
Bluetooth    : False
Camera ports (CSI) : 1
Display ports (DSI): 1
```


```
J8:
 3V3 (1) (2) 5V
GPI02 (3) (4) 5V
GPI03 (5) (6) GND
GPI04 (7) (8) GPI014
GND (9) (10) GPI015
GPI017 (11) (12) GPI018
GPI027 (13) (14) GND
GPI022 (15) (16) GPI023
 3V3 (17) (18) GPI024
GPI010 (19) (20) GND
GPI09 (21) (22) GPI025
GPI011 (23) (24) GPI08
GND (25) (26) GPI07
GPI00 (27) (28) GPI01
GPI05 (29) (30) GND
GPI06 (31) (32) GPI012
GPI013 (33) (34) GND
GPI019 (35) (36) GPI016
GPI026 (37) (38) GPI020
GND (39) (40) GPI021
```

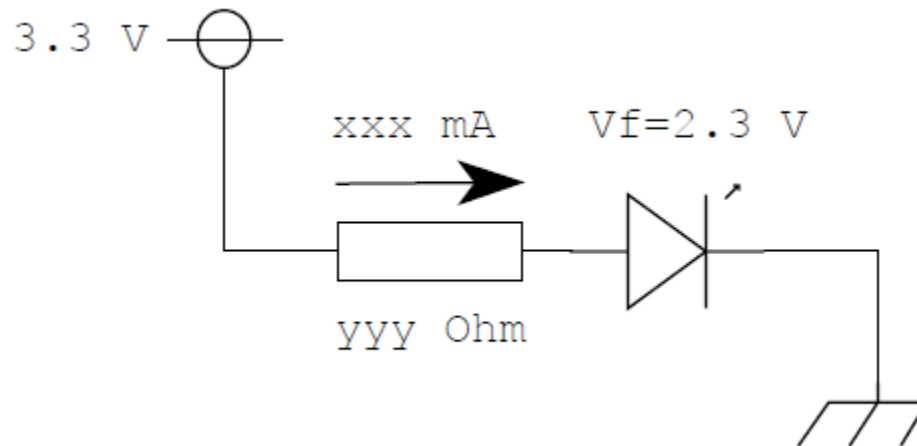
# LEDを光らせるには

ダイオードを光らせるには「順方向電圧」が必要です

- スペックシートを読みましょう (今回のLEDでは 1.8 - 2.6 V)
- $V_f$  と記載してあることが多い

最大電流も (本当は) スペックシートで確認してください

- 今回のLEDでは Max 50 mA, Typical 20 mAですが、試してみると適当な明るさで光らせるためには 10 mA も流せば十分なようです。
- 電流を制限するための抵抗を取り付けます  何Ω必要?
- 抵抗無しだと、LEDかRasPiのどちらかが壊れます



## GPIOピンはどこまで流せる？

電源ピンと異なり、GPIOピンは電流をたくさん流せません

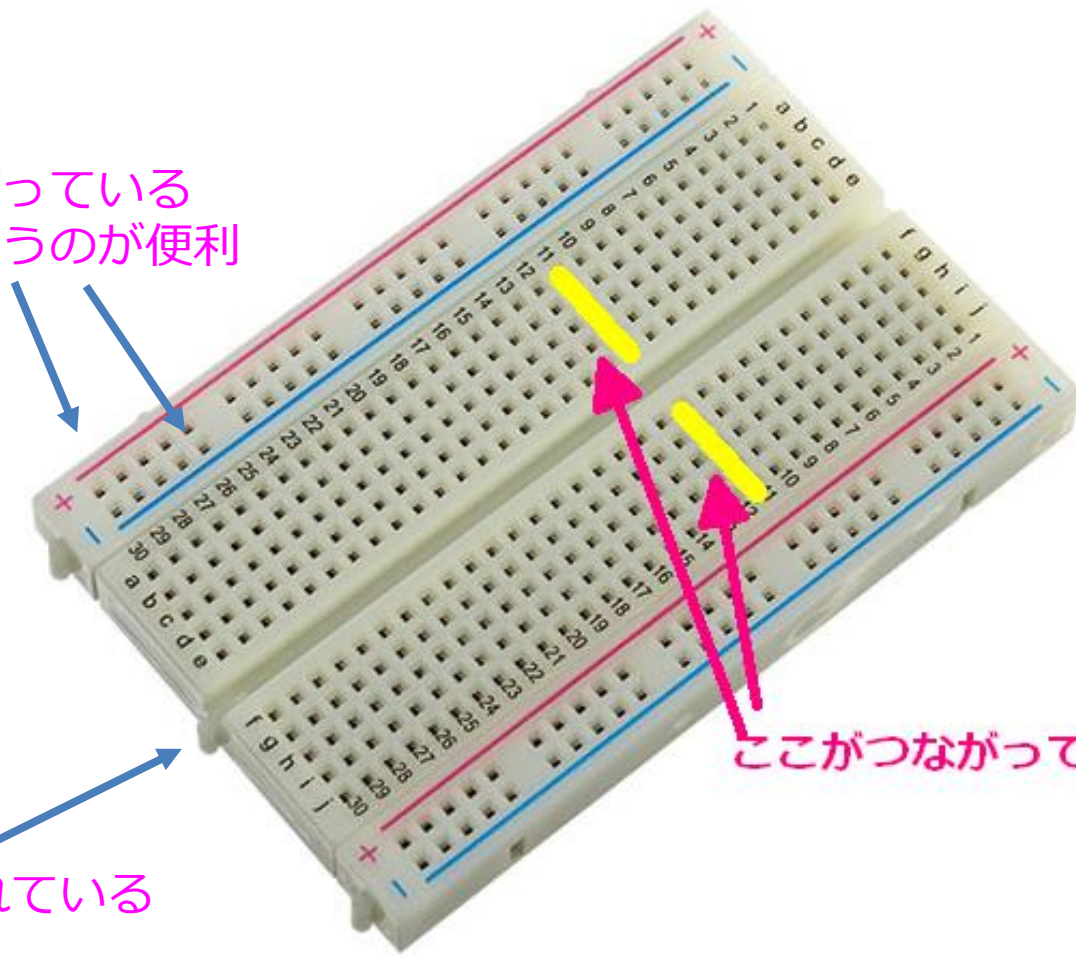
- GPIO : 各ピン 16 mA まで、合計 50 mA まで
- 3.3 V 電源系 : 最大 50 mA
- 5.0 V 電源系 : 300 mA 程度までは流せる



# ブレッドボード

ブレッドボードとは、はんだを使わずに回路を試作できる、とても便利なモノです。両端は電源線として使い、横方向5個がつながっています。中央は切れているので、例えばICを置くのに便利（次ページ）。

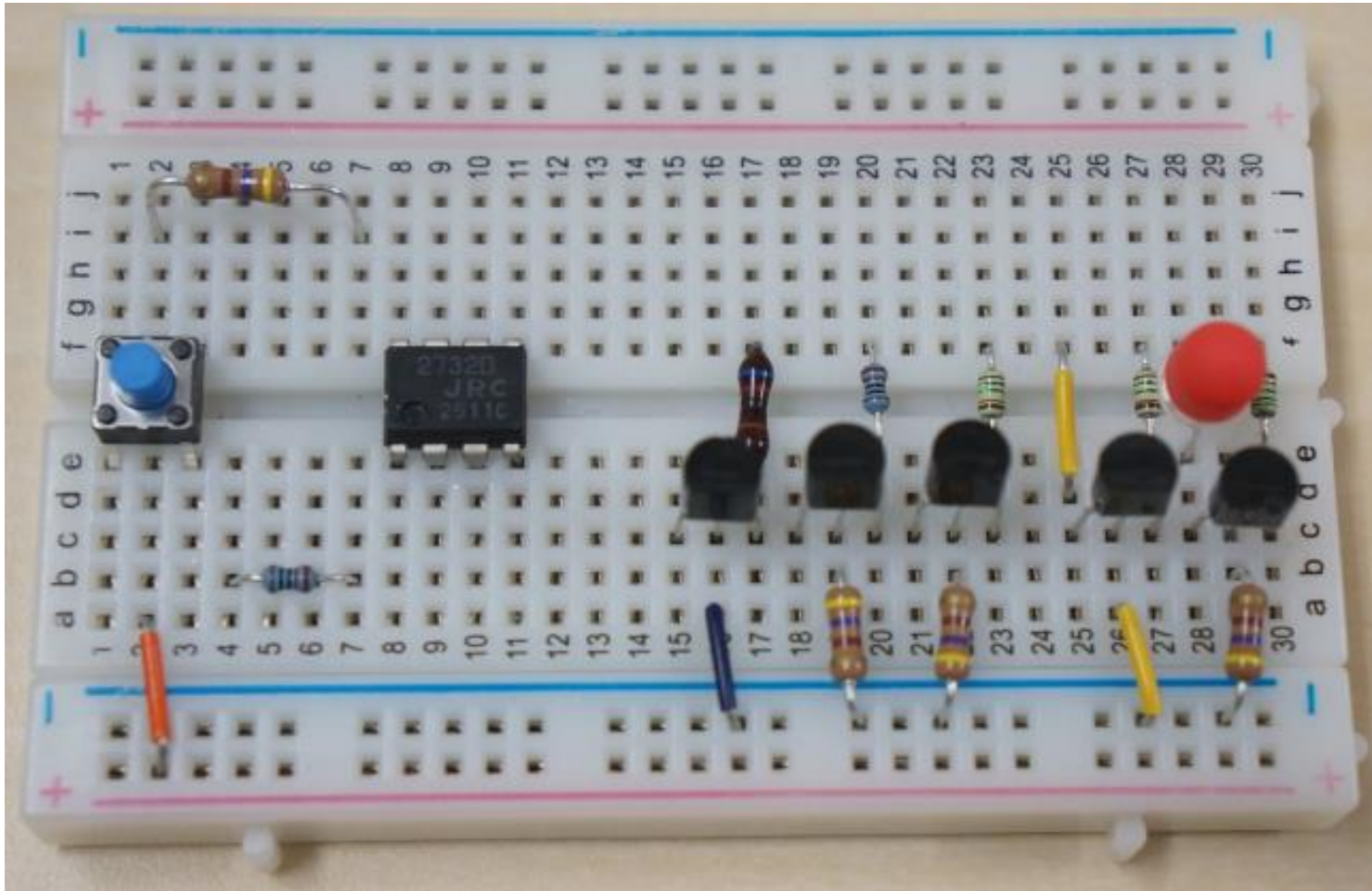
両端は縦方向につながっている  
→電源ラインとして使うのが便利



ここがつながっている

中央は切れている

例えばICを置くときは中央に



# 配線してみましょう

今回は、GPIO17番のピンを使います

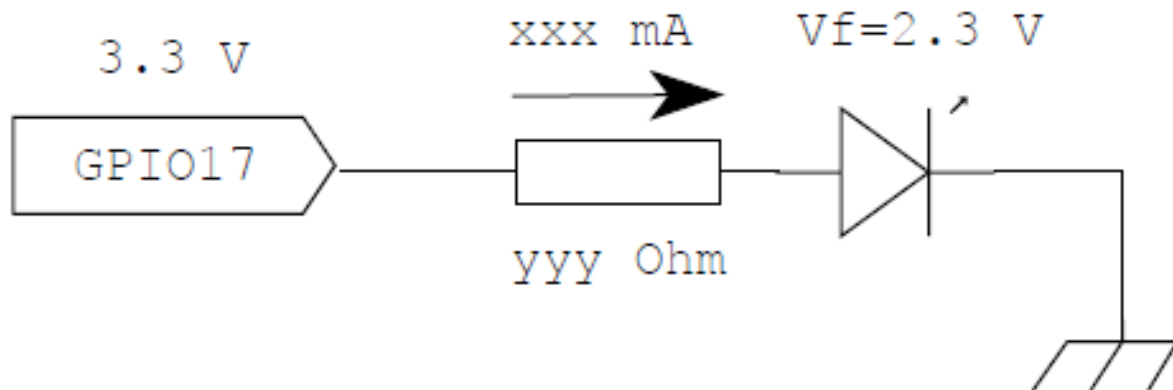
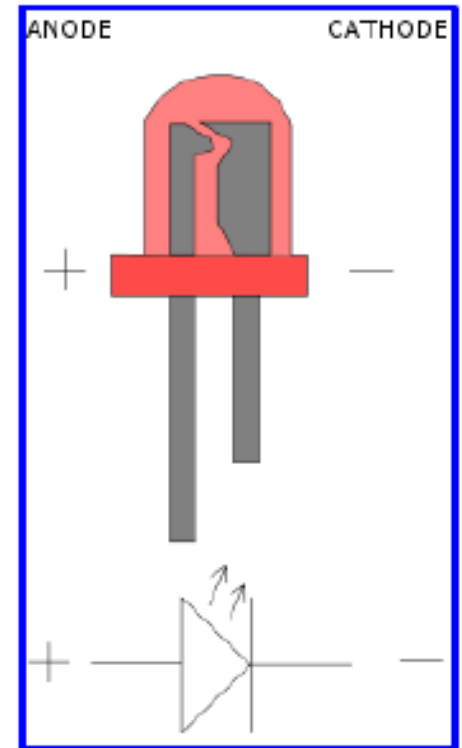


図 5 GPIO17 を LED に接続



## EPICSを使わずに確認

動作確認のため、まずはEPICSを使わずに本当に点灯するかどうかを確認します。以下のコマンドを入力してください。

※この段階で点灯しないならば、何かが間違っています。

```
$ gpio -g mode 17 out    <-- GPIO #17 を出力モードに設定
$ gpio -g write 17 1     <-- 電圧HIGH 出力
$ gpio -g write 17 0     <-- 電圧LOW 出力
```

## EPICSを使わずに確認（その2） sysfs : 紹介のみ

### 0. 作業前の状態を確認

```
$ ls /sys/class/gpio/  
export gpiochip0 gpiochip128 unexport
```

### 1. gpio ピンを使う宣言をおこなう

```
$ echo 17 > /sys/class/gpio/export  
$ ls /sys/class/gpio/  
export gpio17 gpiochip0 gpiochip128 unexport  
$ ls /sys/class/gpio/gpio17/  
active_low device direction edge power subsystem uevent  
value
```

## EPICSを使わずに確認（その2） sysfs : 紹介のみ

### 2. 方向の確認

```
$ cat /sys/class/gpio/gpio17/direction  
in
```

となっているので、out に書き直す

```
$ echo out > /sys/class/gpio/gpio17/direction  
$ cat /sys/class/gpio/gpio17/direction  
out
```

### 3. 値を設定して ON/OFF

```
$ cat /sys/class/gpio/gpio17/value  
0  
$ echo 1 > /sys/class/gpio/gpio17/value  
$ cat /sys/class/gpio/gpio17/value  
1
```

## EPICSを使わずに確認（その2） sysfs : 紹介のみ

### 4. 最後に、消灯してから

```
$ echo 0 > /sys/class/gpio/gpio17/value
```

### 使用終了宣言をおこなう

```
$ echo 17 > /sys/class/gpio/unexport
```

```
$ ls /sys/class/gpio/
```

```
export gpiochip0 gpiochip128 unexport
```

## EPICSを使わずに確認（その3） python：紹介のみ

gpiozero モジュールを使用する例。  
(ほかにもpythonモジュールは存在する)

```
$ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>> from gpiozero import LED
>>> from time import sleep
>>> led = LED(17)
>>> led.on()
>>> led.off()
```



# EPICS Application 作成

最終的に出来上がるディレクトリ構成は以下の通り

```
├── epics
│   └── app
│       ├── gpiol
│           ├── configure
│           │   ├── Makefile
│           │   └── RELEASE
│           ├── gpiolApp
│           │   ├── Db
│           │   │   ├── Makefile
│           │   │   └── test.db
│           │   ├── Makefile
│           │   └── src
│           │       ├── gpiolMain.cpp
│           │       └── Makefile
│           ├── iocBoot
│           │   ├── iocgpiol
│           │   │   ├── Makefile
│           │   │   └── st.cmd
│           └── example
│               └── stream
```

## makeBaseApp.pl

makeBaseApp.pl テンプレートとして "ioc" を使う

```
$ mkdir -p epics/app/gpiol
$ cd epics/app/gpiol/
$ makeBaseApp.pl -t ioc gpiol
$ makeBaseApp.pl -i -t ioc gpiol
Using target architecture linux-arm (only one available)
The following applications are available:
    gpiol
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed
above.
Application name?    <----- Enter を入れるのみ
```

## configure/RELEASE編集

ファイルを編集して、以下のコメントを外す

```
#SNCSEQ=/opt/epics/R315.6/modules/soft/seq/2.2.4  
#ASYN=/opt/epics/R315.6/modules/soft/asyn/4-31  
#STREAM=/opt/epics/R315.6/modules/soft/stream/2-7-7  
#STREAM=/opt/epics/R315.6/modules/soft/stream/2-7-7_I2C  
RPIGPIO=/opt/epics/R315.6/modules/soft/gpio/20160308  
#RPII2C=/opt/epics/R315.6/modules/soft/i2c/20170603
```

## databaseファイル作成

<TOP>/gpio1App/Dbディレクトリに移動し、データベースを作成

```
record(bo, "$ (head):GPIO17:OUT") {  
  field(DTYP, "devgpio")  
  field(OUT, "@17 H")  
  field(ZNAM, "OFF")  
  field(ONAM, "ON")  
}
```

## srcディレクトリで Makefile 編集

<TOP>/gpio1App/srcディレクトリに移動し、Makefile編集

```
...  
gpio1_DBD += devgpio.dbd  
gpio_registerRecordDeviceDriver pdbname  
...  
  
gpio1_LIBS += devgpio  
...
```

その後、<TOP>ディレクトリに戻って make かける。  
エラーが出ていないことを確認

## startup script 編集

<TOP>/iocBoot/iocgpio1ディレクトリに移動し、st.cmd 編集

```
#!../../bin/linux-arm/gpio1
....
dbLoadDatabase "dbd/gpio1.dbd"
dbLoadRecords ("db/test.db", "head="ET_kektaro") <--追加
GpioConstConfigure ("RASPI B+") <---追加
cd "${TOP}/iocBoot/${IOC}"
iocInit
```

実行フラグをつけてから、iocを起動する

```
$ chmod +x st.cmd
$ ./st.cmd
```

エラーが出ないことを確認。

# コマンドラインから制御

## dbfコマンドでデータベース確認

```
epics> dbf  
ET_kektaro:GPIO17:OUT
```

## dbpfで値を設定→LEDを確認

```
epics> dbpf ET_kektaro:GPIO17:OUT 1  
DBR_STRING:          "ON"  
epics> dbpf ET_kektaro:GPIO17:OUT 0  
DBR_STRING:          "OFF"
```

## コマンドラインから制御

別の端末を開く

caget, caput で値設定、確認

```
$ caget ET_kektaro:GPIO17:OUT  
ET_kektaro:GPIO17:OUT OFF
```

```
$ caput ET_kektaro:GPIO17:OUT 1  
Old : ET_kektaro:GPIO17:OUT OFF  
New : ET_kektaro:GPIO17:OUT ON
```

```
$ caget ET_kektaro:GPIO17:OUT  
ET_kektaro:GPIO17:OUT ON
```

```
$ caput ET_kektaro:GPIO17:OUT OFF  
Old : ET_kektaro:GPIO17:OUT ON  
New : ET_kektaro:GPIO17:OUT OFF
```



# GUIから制御

- CSSでGUIを作成

## 時間がある人へ

- 他の人のLED をモニター(or 制御!)
- bo レコードに 0/1 以外の数値を入れてみる。例えば 0.3 とか 7 等
- bo レコードの HIGH フィールドを設定してパルス出力（モーメンタリー出力）に変更してみる→変更したあとは make 必要
- ioc を終了する際に CTRL-C で停止すると次に起動したときに警告表示が出る(動作に問題はない)。この原因は? exit コマンドを使って ioc を終了すれば警告表示は出ない。
- devgpio のソースコード  
/opt/epics/R315.6/modules/soft/gpio/20160308 を眺めてみる
- 複数の LED を同時に設定したい：ビット列のデータを同時に設定する(mbbo または longout)：デバイスサポートは bi/bo のみなので、DB Link で実現するのがとりあえずは簡単。