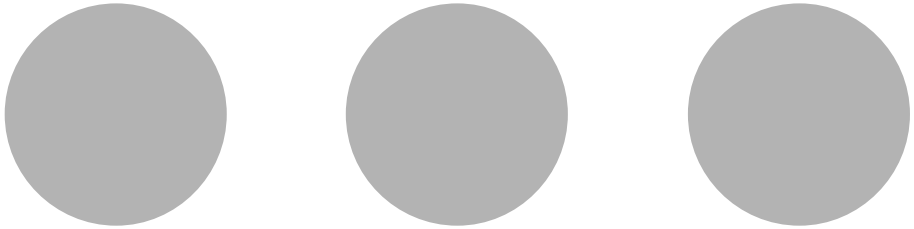
MARY

工作教室

2枚入り！組み合わせ自在！

超小型ARMマイコン基板

圓山宗智



AGENDA

Introduction 15:40 ~ 16:00

1.材料・ツールの確認
2.教材データのインストール
3.MARY概要

Color LED 16:00 ~ 16:30

4.[MB]x1 Lチカをやってみよう
5.[MB]x2 MB間連携・点滅方法変更

OLED Display 16:30~16:45

6.[OB/MB]x1 OLED表示・ランダム矩形表示
7.[OB/MB]x1 加速度センサの読み取り

MCU Array 16:45~17:10

8.[OB/MB]x2 OLED表示連携・UARTデバッグ
9.[OB/MB]x2 マルチボール・基板接続位置変更

Breakout Game 17:10-17:50

10.[OB/MB]x2 塩ビ板へ基板を取付け
11.[OB/MB]x2 ブロック崩しゲーム・ゲーム内容変更

Introduction

1.材料・ツールの確認

【材料】

●MB×2枚（コネクタ取付済み）
●OB×2枚
●アレイ接続ケーブル×2本（うち1本のみ使用）
●MBのCN6, CN7用2x2ピン・ソケット×2個（使用せず）
●塩ビ板×1枚
●MB取り付け用ネジセット×2式

●USBケーブル
●USBメモリ

【工具】
●ねじ回し（M2.6プラス用）

【ツール・インストール済みを前提】

- CP210x Virtual COM Port Driver
- MagicFlash
- LPCXpresso IDE
- TeraTerm

2.材料データのインストール

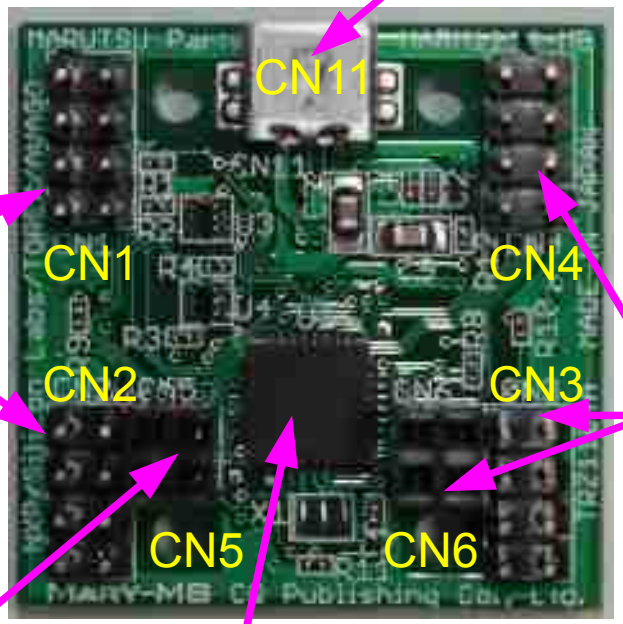
●USBメモリ
　　フォルダ「LPC1114_ejforum」
　　↓
　　C:\CQ\の下にコピー

●LPCXpressoにインポート
　　講師指示に従う

3.MARYの概要

付録基板MB (2枚入り)

表（部品面）



mini Bタイプ
USBコネクタ

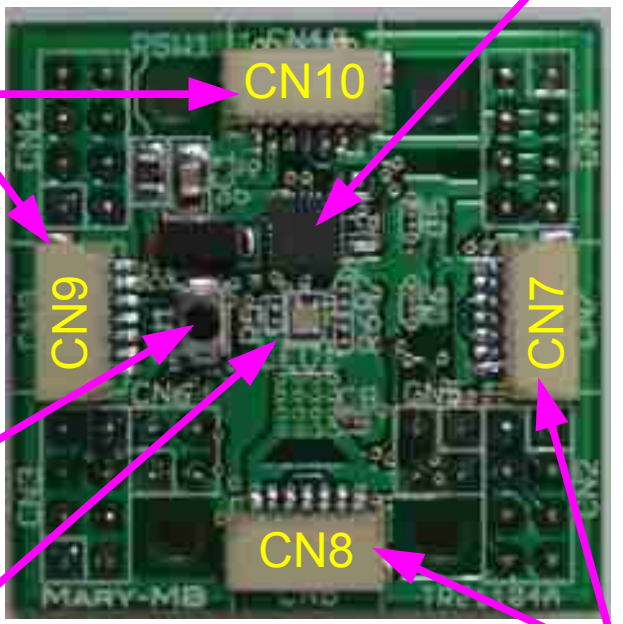
外部拡張用
コネクタ

外部拡張用
コネクタ

LPCXpresso
デバッガ接続
コネクタ(SWD)

32ビット Cortex-M0 MCU
LPC1114FHN33/301(NXP)

裏（半田面）



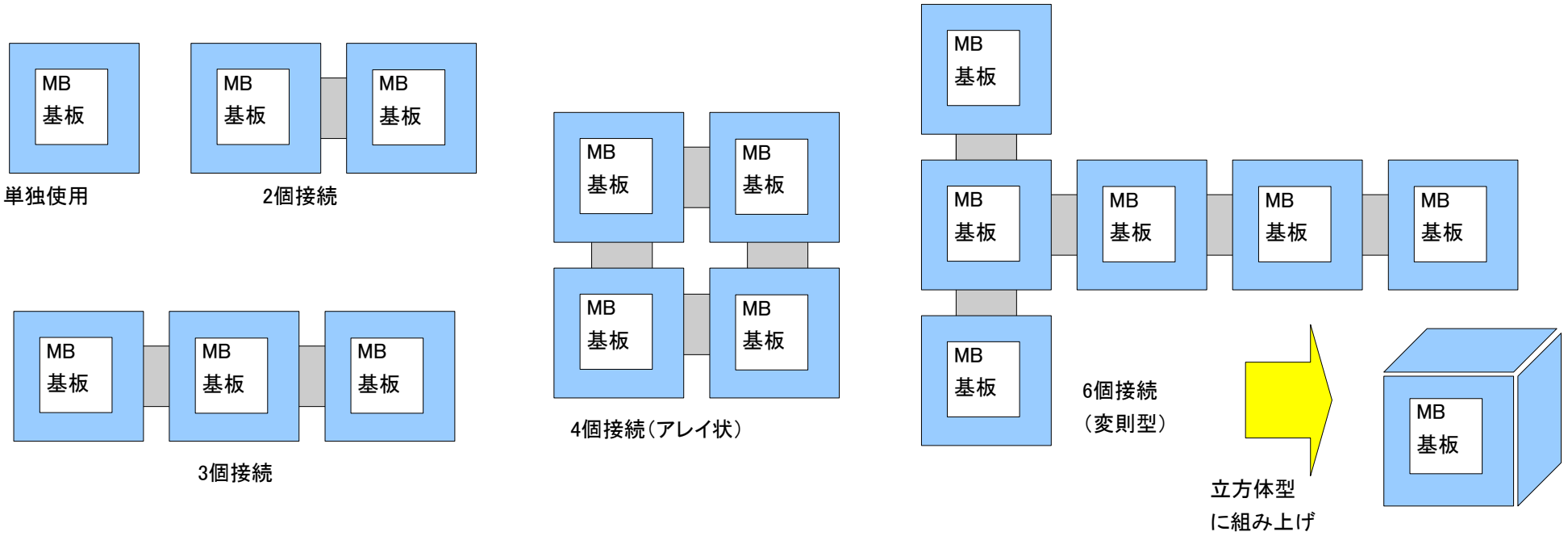
USB-シリアル変換IC
CP2104-E02-GM(Silicon Labs)

アレイ接続用
コネクタ

リセット
スイッチ
フル・カラー
チップLED

アレイ接続用
コネクタ

アレイ接続
→マルチ・プロセサ



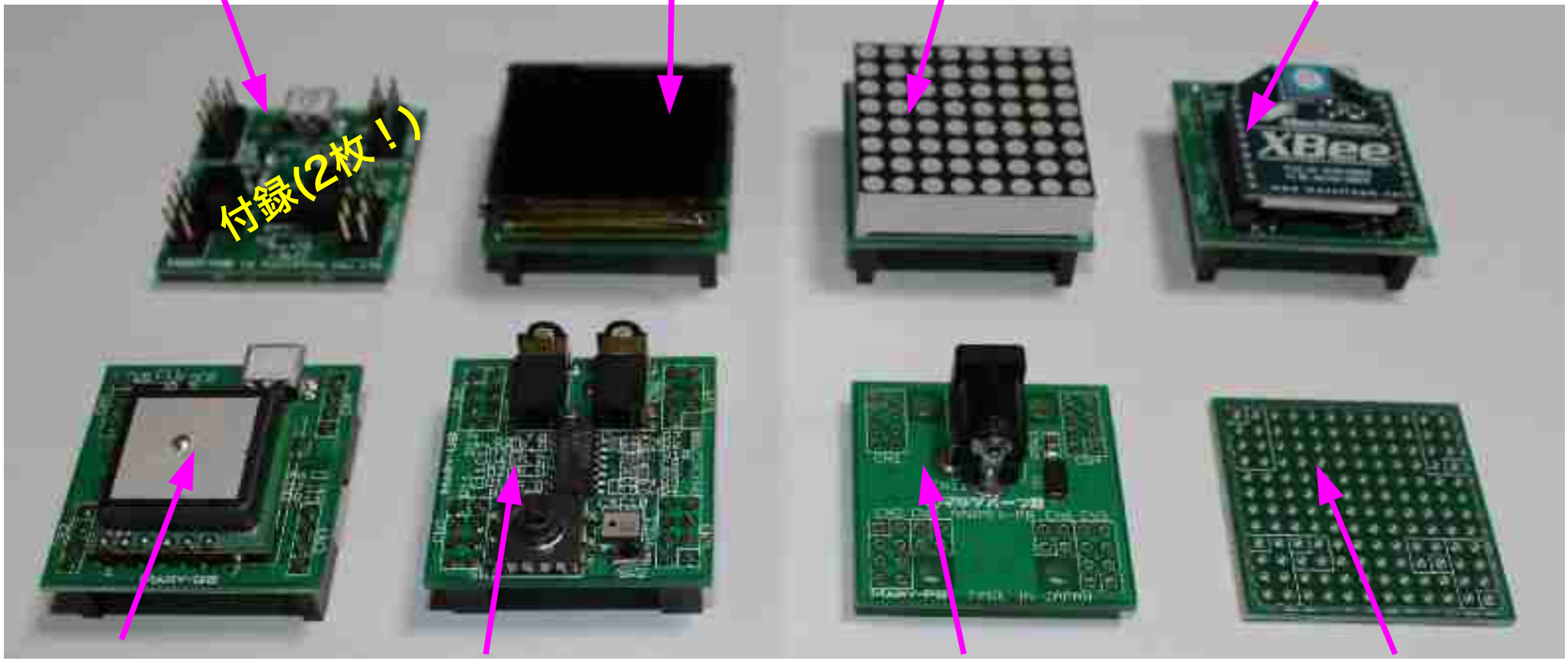
拡張基板

MB基板(MCU Board)
—Cortex-M0/LPC1114
—フル・カラーLED
—USB-シリアル変換IC

OB基板(OLED Board)
—カラーOLED(128x128)
—3軸加速度センサ

LB基板(LED Board)
—2色LEDアレイ(赤・緑)
—3色表示（赤・緑・橙）

XB基板(XBee Board)
—無線モジュールXBee
—micro SDカード・ソケット
—USB-シリアル変換IC



GB基板(GPS Board)
—GPSモジュール
—リアルタイム・クロック(RTC)
—バッテリ・バックアップ用
　CR1220ホルダ
—USB-シリアル変換IC

UB基板(UI Board)
—4方向スイッチ
—プッシュ・スイッチ
—アナログ信号入力
—アナログ信号出力
—オペ・アンプ
—MEMSシリコン・マイク
—圧電サウンダ

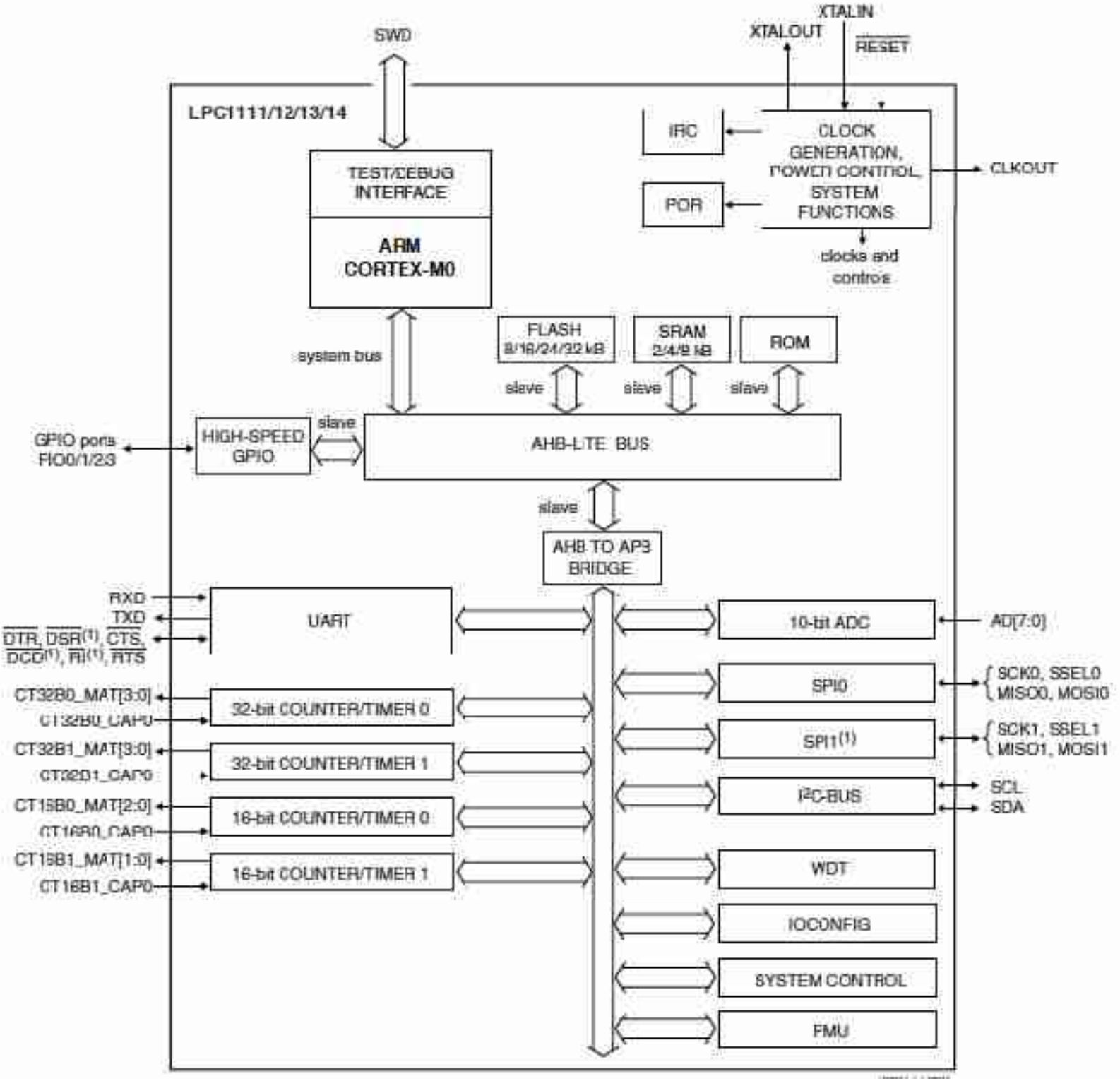
PB基板(Power Board)
—補助電源供給用ACアダプタ接続

CB基板(Craft Board)
—工作用専用ユニバーサル基板

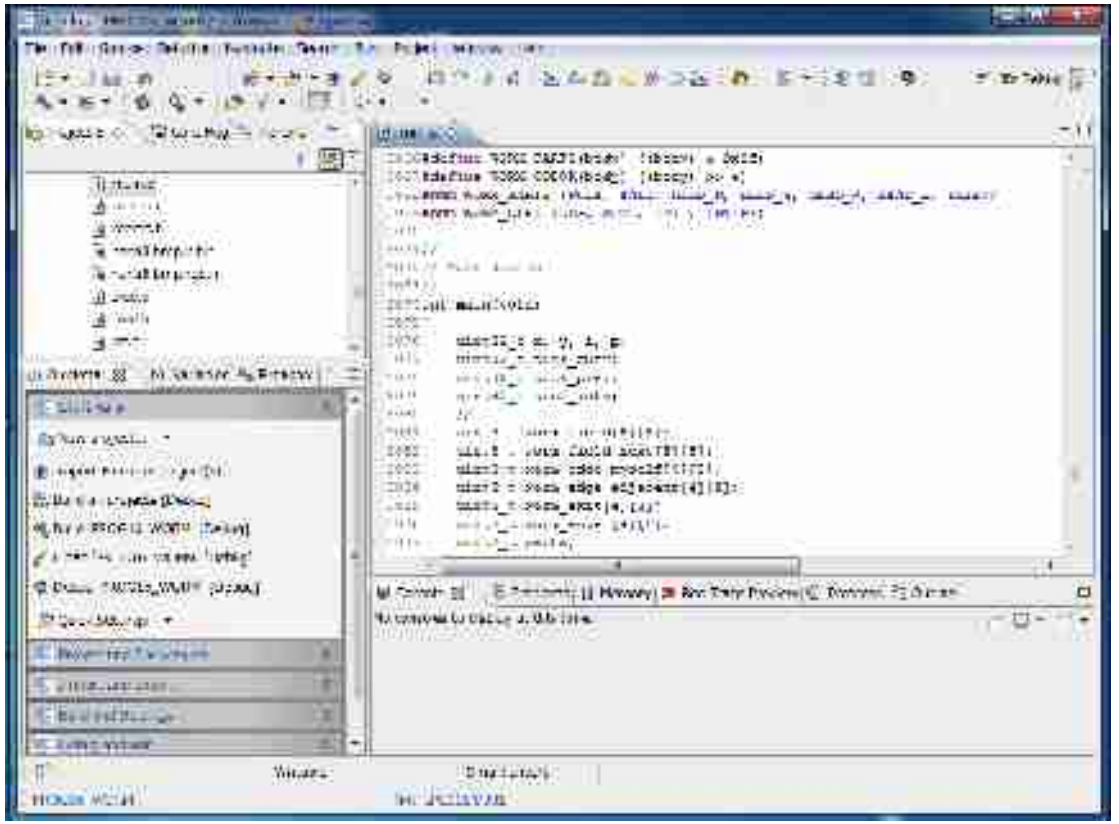
Cortex-M0

No.	項目	ARM7TDMI	Cortex-M0	Cortex-M3	Cortex-M4
1	特長	最も普及したコア	徹底的に小さいコア	標準向けコア	デジタル信号処理
2	搭載マイコン製品(NXP)	LH754xx、LH7952x	LPC111x	LPC13xx/17xx/18xx	LPC43xx
3	アーキテクチャ名	v4T	v6-M	v7-M	v7-ME
4	命令セット	ARM, Thumb	Thumb、Thumb-2	Thumb + Thumb2	Thumb + Thumb2、DSP、SIMD、FP
5	性能DMIPS/MHz	0.72 (Thumb) 0.95 (ARM)	0.9	1.25	1.25
6	バス・アーキテクチャ	フォンノイマン	フォンノイマン	ハーバード	ハーバード
7	割込みコントローラ(NVIC)	×	○	○	○
8	割込み本数	2(IRQ, FIQ)、NMIなし	1～32 + NMI	1～240 + NMI	1～240 + NMI
9	割込み優先度	×	4	8～256	8～256
10	ブレーク・ポイント、 ウォッチ・ポイント	2、2	4/2/0、2/1/0	8/4/0、2/1/0	8/4/0、2/1/0
11	デバッグ・インタフェース	JTAG	JTAG/SW(シリアル)	JTAG/SW(シリアル)	JTAG/SW(シリアル)
12	メモリ保護ユニット(MPU)	×	×	○(オプション)	○(オプション)
13	命令トレース(ETM)	○(オプション)	×	○(オプション)	○(オプション)
14	1サイクル乗算	×	○(オプション)	○	○
15	除算命令	×	×	○	○
16	ウェークアップ割込み	×	○	○	○
17	ビット操作対応	×	×	○	○
18	1サイクル積和演算	×	×	×	○
19	浮動小数点演算	×	×	×	○

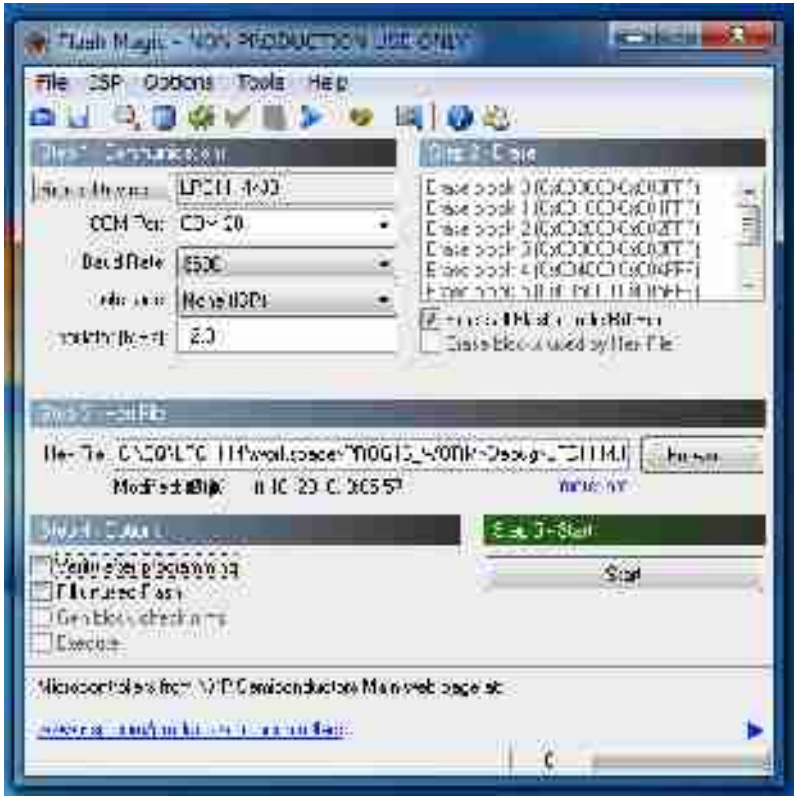
LPC1114


LPC1111/12/13/14
SWD
XTALIN
XTALOUT
RESET
TEST/DEBUG INTERFACE
ARM CORTEX-M0
IRC
POR
CLOCK GENERATION, POWER CONTROL, SYSTEM FUNCTIONS
CLKOUT
clocks and controls
system bus
FLASH 8/16/24/32 kB
SRAM 2/4/8 kB
ROM
slave
slave
slave
slave
slave
GPIO ports PIO0/1/2/3
HIGH-SPEED GPIO
AHB-LITE BUS
AHB TO APB BRIDGE
RXD
TXD
DTR, DSR(1), CTS, DCD(1), RI(1), RTS
UART
10-bit ADC
AD[7:0]
CT32B0_MAT[3:0]
CT32B0_CAP0
32-bit COUNTER/TIMER 0
SPI0
SCK0, SSEL0 MISO0, MOSI0
CT32B1_MAT[3:0]
CT32B1_CAP0
32-bit COUNTER/TIMER 1
SPI1(1)
SCK1, SSEL1 MISO1, MOSI1
CT16B0_MAT[2:0]
CT16B0_CAP0
16-bit COUNTER/TIMER 0
I2C-BUS
SCL
SDA
CT16B1_MAT[1:0]
CT16B1_CAP0
16-bit COUNTER/TIMER 1
WDT
IOCONFIG
SYSTEM CONTROL
PMU

開発環境



(1) LPCXpresso IDE



(2) Flash Magic

Color LED

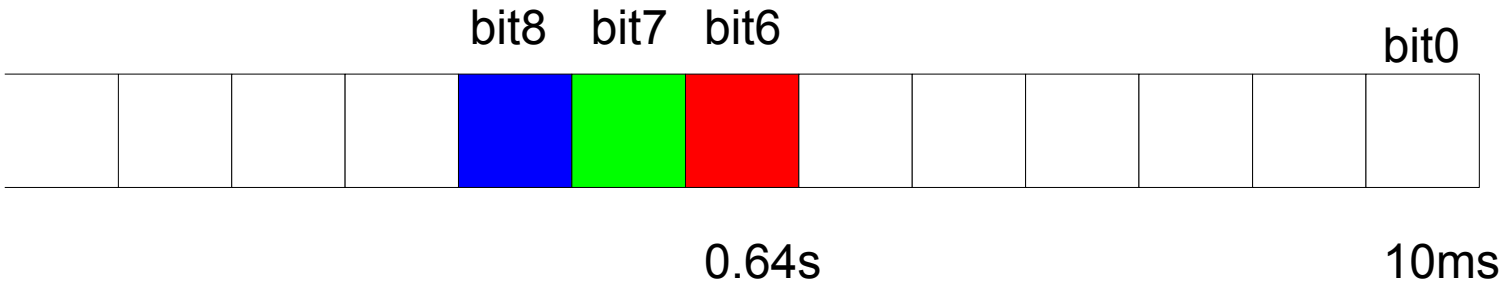
4.[MB]x1 Lチカをやってみよう

●CMSISをビルド

●PROG01_COLOR_LEDをビルド

ーInit_Systick() (Systick.h)
　　SysTick割込み(10ms)
　　割込みのたびにグローバル変数gTicksをインクリメント

—Draw_Color_LED() (color_led.h)
　　gTicksのbit8,6,7をBLU, GRN, REDに対応させて点灯



5.[MB]x2 MB間連携

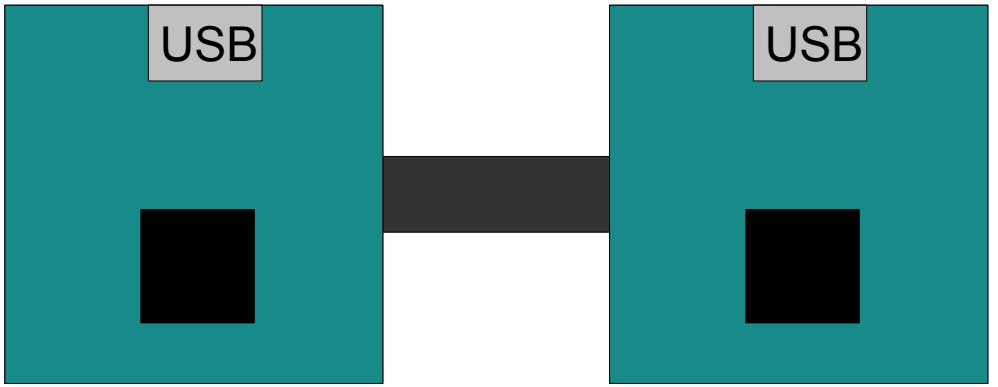
●PROG02_COLOR_LED_MULTIをビルド

－Init_Array_COM() (array_com.h)
　　MBアレイ接続の初期化

－Array_COM_ID_Assignment() (array_com.h)
　　MBアレイのチェックとIDアサイン

－Sync_Ticks() (array_com.h)
　　Ticks値を送信し、受信側は自動的にTicksを同期

●点滅方法変更
　講師指示に従う



OLED Display

6.[OB/MB]x1 OLED表示

●MBにOBを搭載（使うのは1組のみ）

●PROG03_OLED_BMPをビルド

ーInit_OLED() (oled.h)
　　OLEDの初期化

ーOLED_Draw_Dot() (oled.h)
　　OLED上にドットを打つ

※ビットマップ画像

```
#include "../bmp/board8.bmp.plt.h"（パレット）
#include "../bmp/board8.bmp.rgb.h"（画像本体）
```

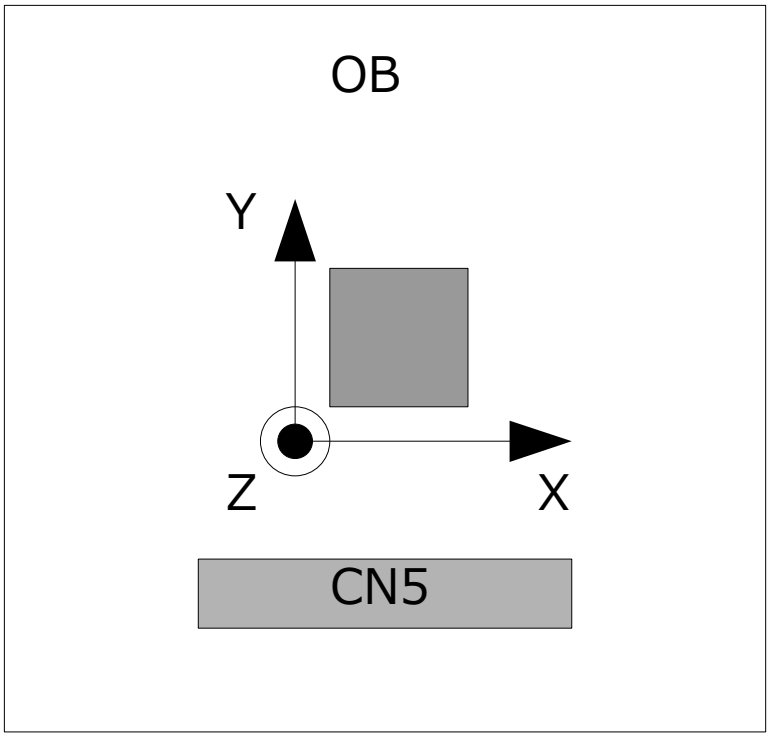
●ランダム矩形表示
　講師指示に従う

7.[OB/MB]x1 加速度センサ

●PROG04_OLED_MEMSをビルド

ーInit_MEMS() (mems.h)
　　MEMSの初期化

ーMEMS_Get_X() (mems.h)
　　MEMSからX軸側の値を得る



●矢印の向きに加速度を感じている場合

（矢印と反対方向に重力で引かれている場合）

その軸の読み取り値は正数。逆なら負数。

●8ビット符号付きデータとしての読み取り値の

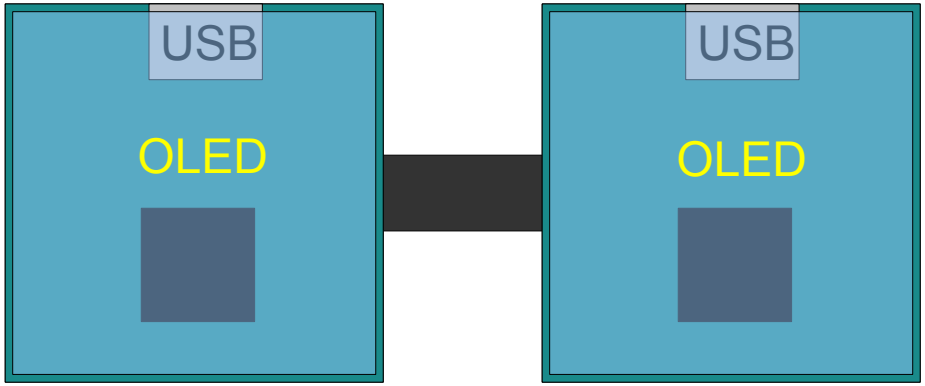
フル・スケールは±2Gに対応。+1Gの場合、

約+64前後の値になる。

MCU Array

8.[OB/MB]x2 OLED連携

●PROG04_OLED_MEMSのオリジナル




```
//=========================
// Demo Selection
//=========================
#define OLED_SCENE_1ST 1
#define OLED_SCENE_01  1 // Status Display
#define OLED_SCENE_02  2 // Lissajous
#define OLED_SCENE_03  3 // Single Ball
// if == 0, the scene is disable.
// if  > 0, the value shows next scene.
```

●PROG04_OLED_MEMS／リサージュ図形

```
//=========================
// Demo Selection
//=========================
#define OLED_SCENE_1ST  2
#define OLED_SCENE_01   1 // Status Display
#define OLED_SCENE_02   2 // Lissajous
#define OLED_SCENE_03   3 // Single Ball
// if == 0, the scene is disable.
// if  > 0, the value shows next scene.
```

●PROG04_OLED_MEMS／基板間ボール移動

```
//=========================
// Demo Selection
//=========================
#define OLED_SCENE_1ST  3
#define OLED_SCENE_01   1 // Status Display
#define OLED_SCENE_02   2 // Lissajous
#define OLED_SCENE_03   3 // Single Ball
// if == 0, the scene is disable.
// if  > 0, the value shows next scene.
```

●PROG04_OLED_MEMS／基板間ボール移動

※UART経由のデバッグ方法

1.講師指示に従ってmain.cを変更して再ビルド
2.プログラムをMBにダウンロード
3.Teraterm起動
4.Teraterm New connection
5.Teraterm マクロ実行
6.基板をリセット
7.Teraterm Disconnect

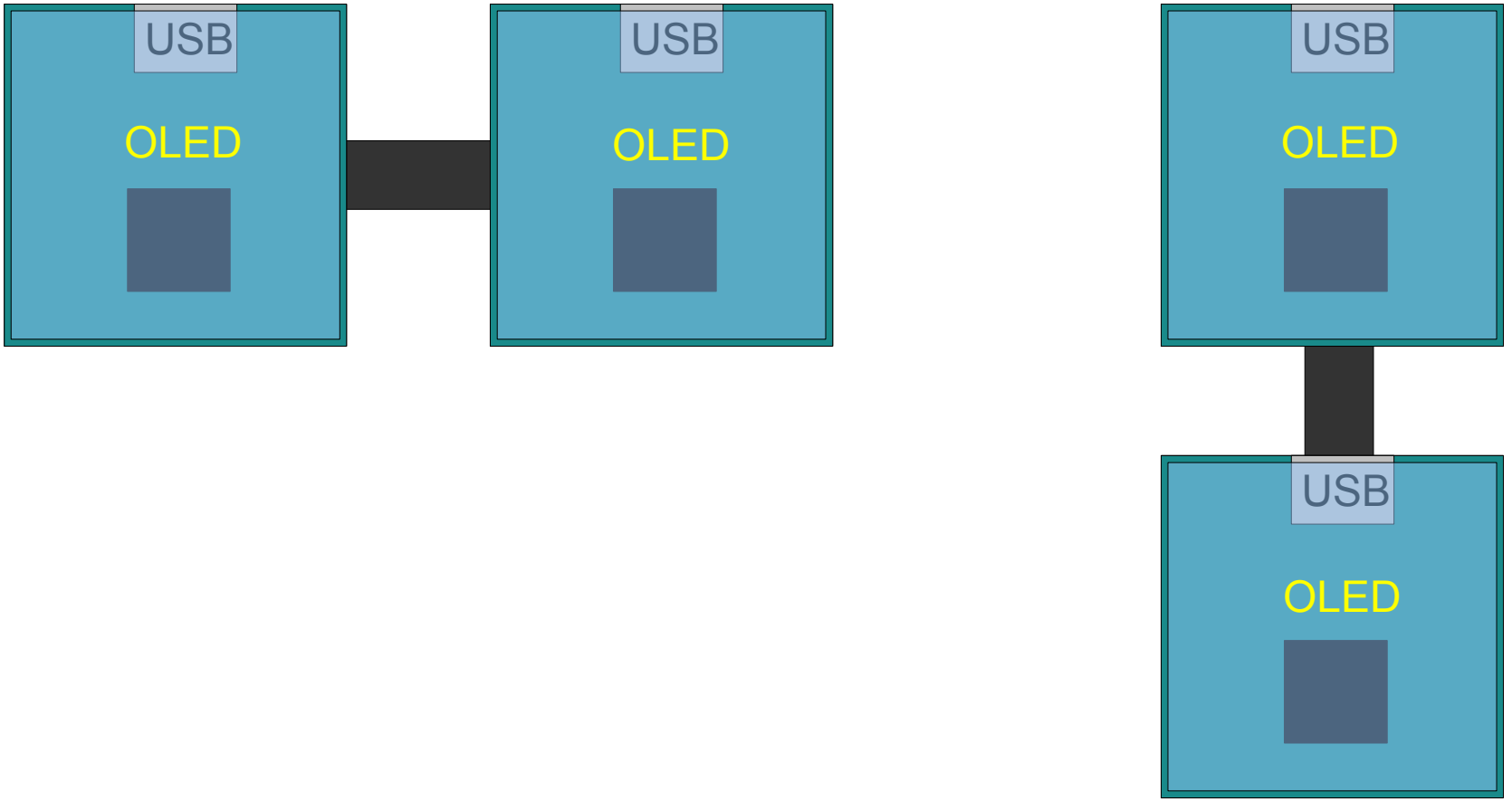
8.プログラムを修正（デバッグ）
2.プログラムをMBにダウンロード
3.Teraterm起動
…

●PROG04_OLED_MEMS／連続デモ実行

```
//=========================
// Demo Selection
//=========================
#define OLED_SCENE_1ST  1
#define OLED_SCENE_01   2 // Status Display
#define OLED_SCENE_02   3 // Lissajous
#define OLED_SCENE_03   4 // Single Ball
// if == 0, the scene is disable.
// if  > 0, the value shows next scene.
```

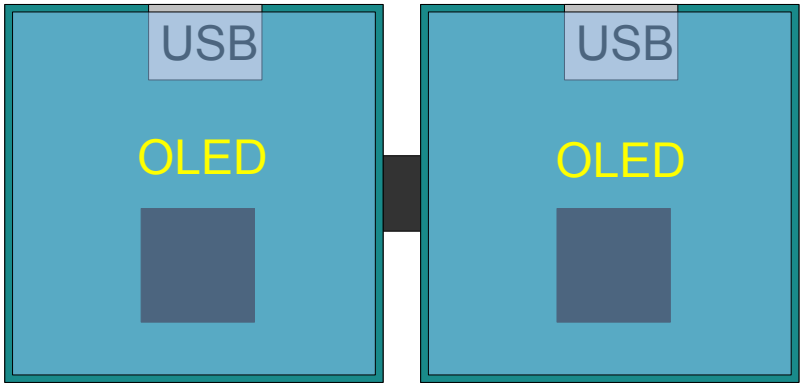
9.[OB/MB]x2 マルチボール

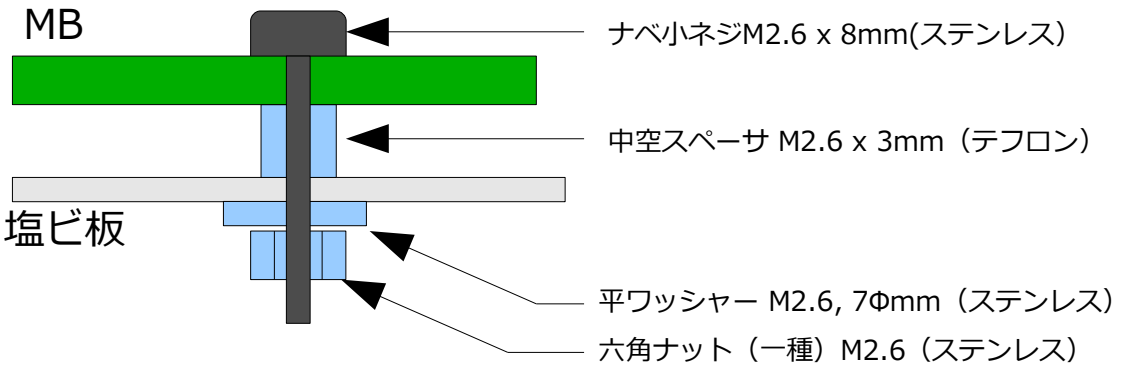
●PROG05_OLED_MULTIBALL

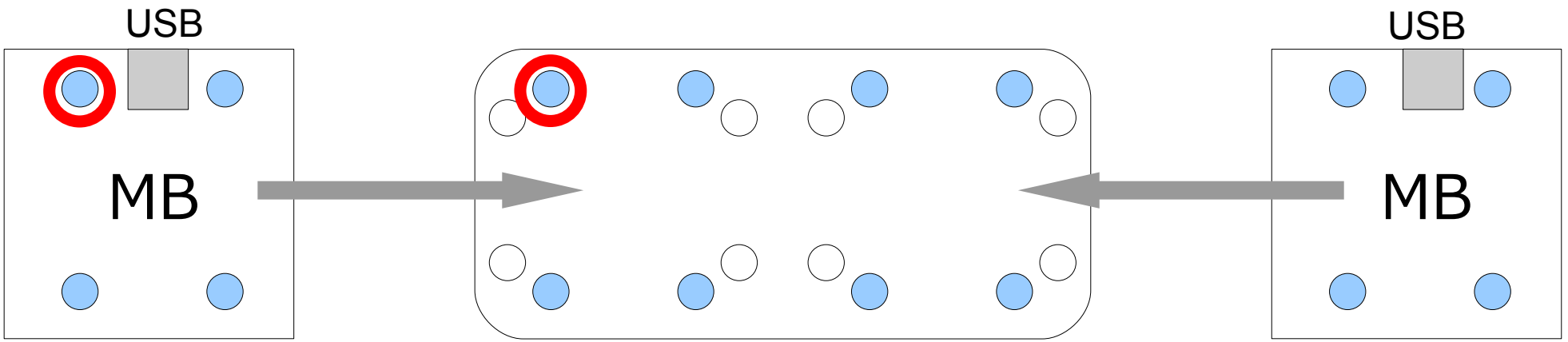


Breakout Game

10.[OB/MB]x2 取り付け

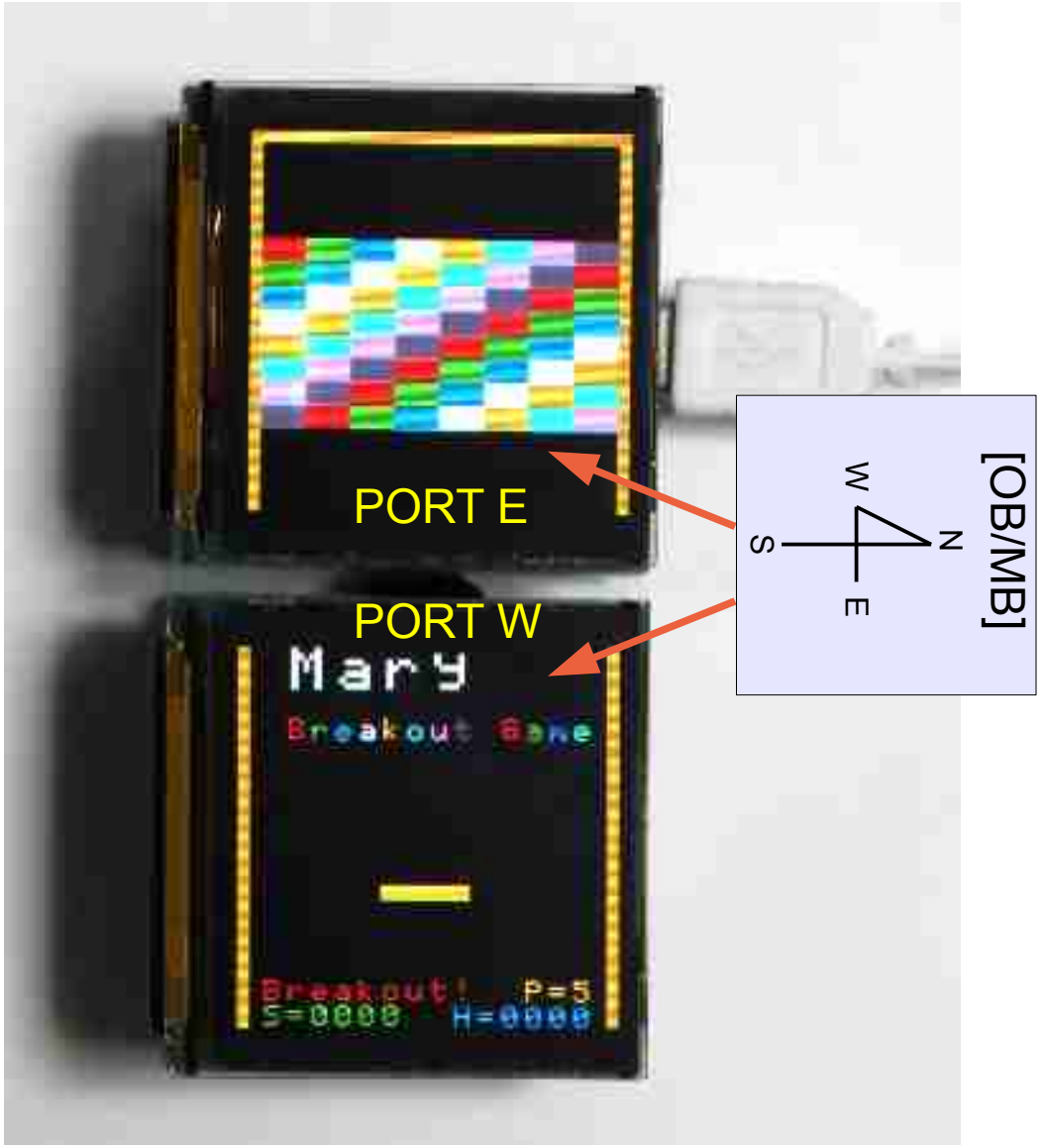






11.[OB/MB]x2 ブロック崩し

●PROG14_BREAKOUT



(1)リセット直後

(2)手前に起こして
ゲーム開始

(3)向こうに倒して
ボール発射

ミス5回で
ゲームオーバー

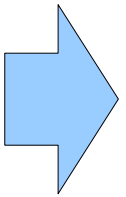
ミス～！

発射！

(4)左右に揺らしてパドルを操作してボールを打ち返し

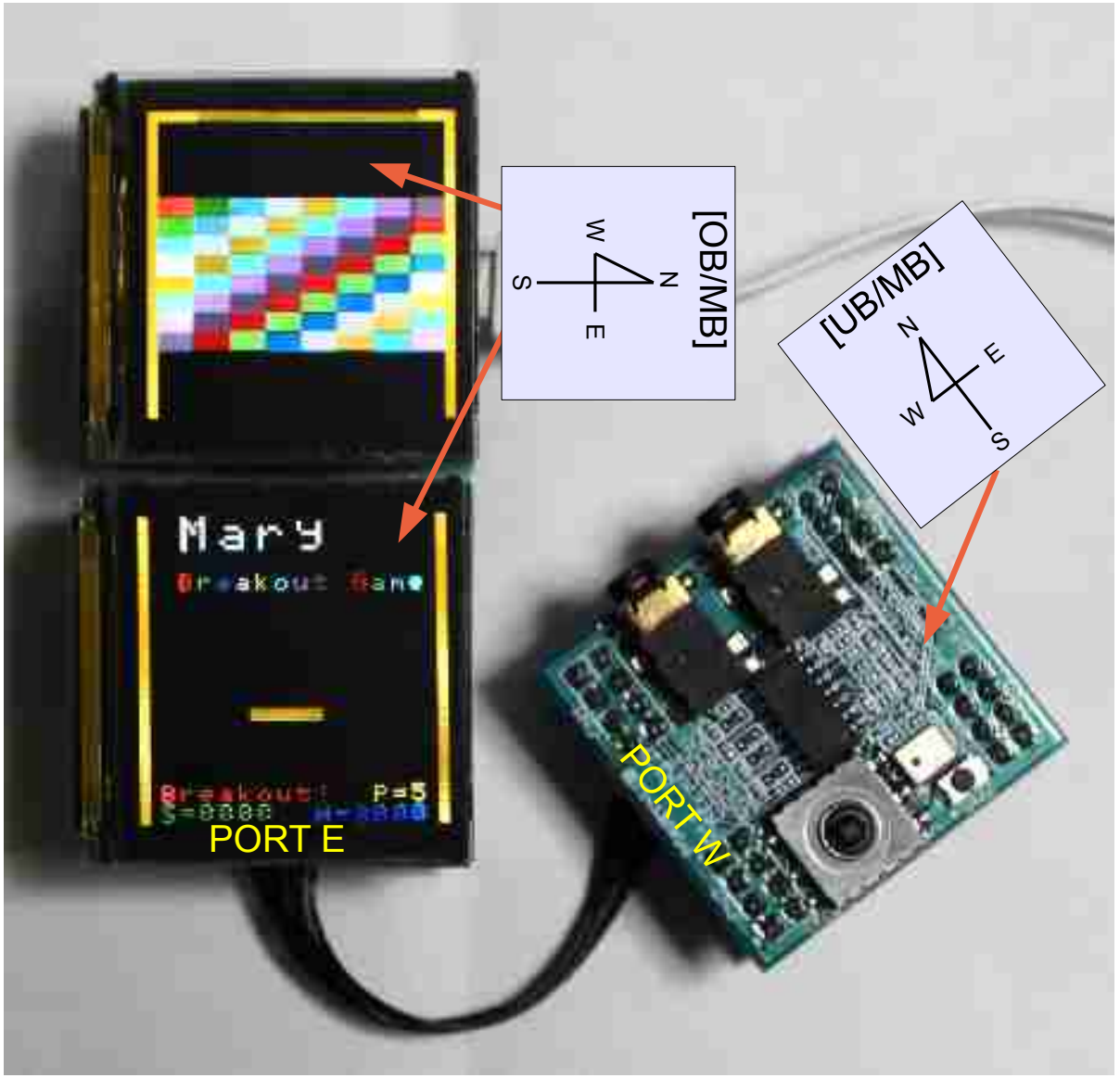
●PROG14_BREAKOUT：ゲームの変更

```
//
// Block Patterns
//
static const uint32_t BLK_PATTERN[..][..][..]
= {
        {
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1}
        },
```

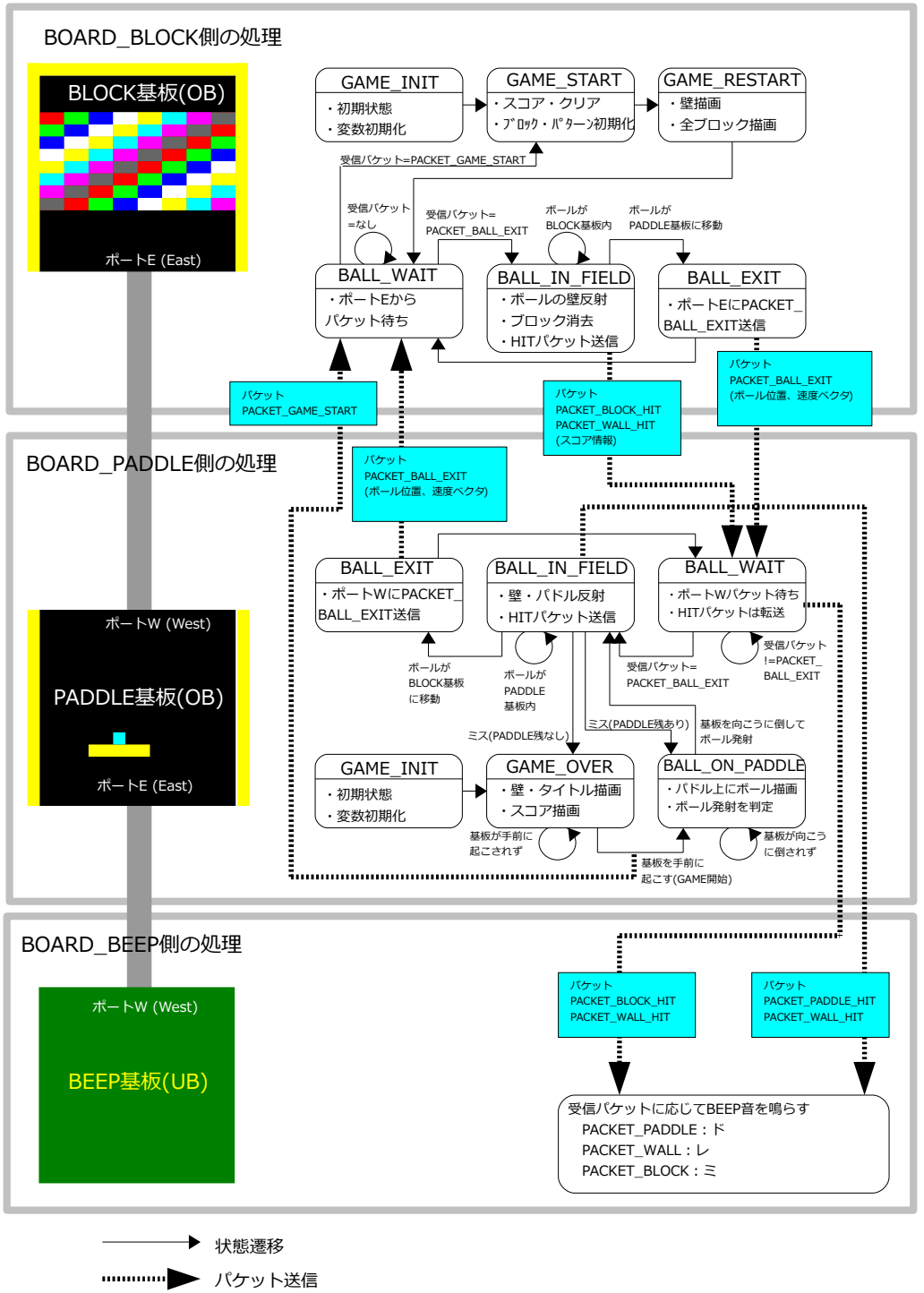


```
//
// Block Patterns
//
static const uint32_t BLK_PATTERN[..][..][..]
= {
        {
            {1, 1, 1, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 0, 0, 0, 0},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {1, 1, 1, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 0, 0, 0, 0},
            {1, 1, 1, 1, 1, 1, 1, 1}
        },
```

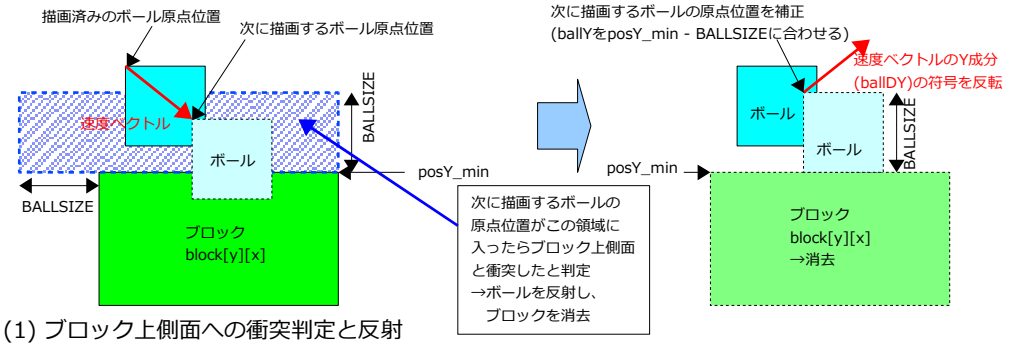
●PROG14_BREAKOUT：音を鳴らす



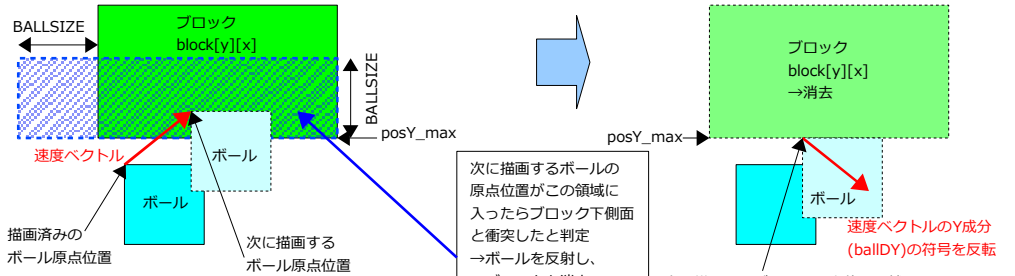
プログラム状態遷移図



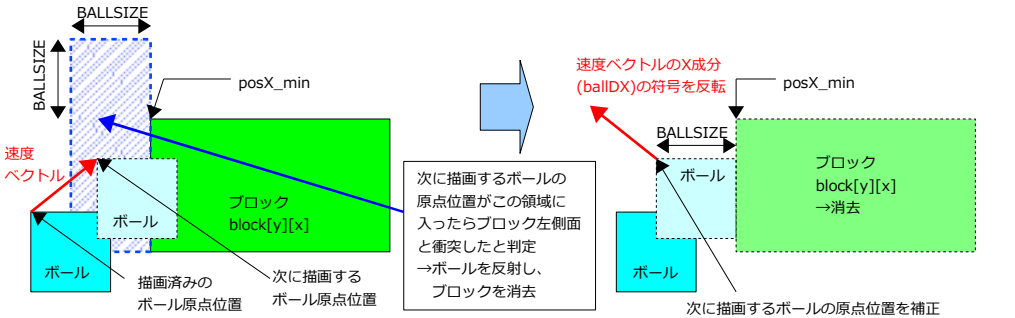
ボールの衝突判定



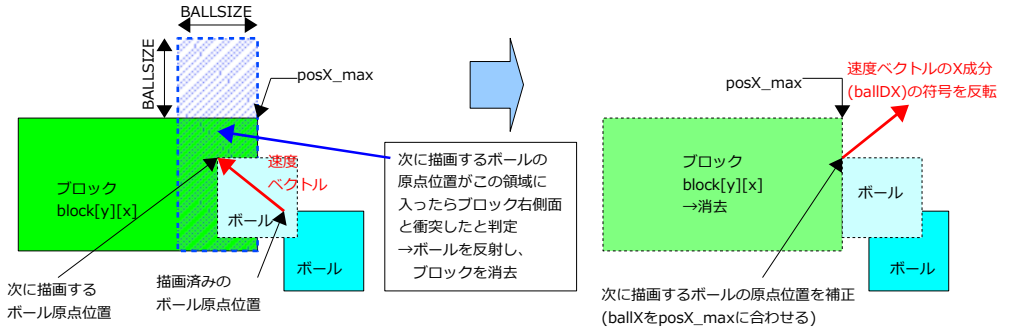
(1) ブロック上側面への衝突判定と反射



(2) ブロック下側面への衝突判定と反射



(3) ブロック左側面への衝突判定と反射



(4) ブロック右側面への衝突判定と反射

Information

●今後のMaryボードに関するCQ出版社の活動

・CQ出版社主催の「読者サポート・セミナ」
を5月～8月に巣鴨のCQ出版社
セミナ・ルームにて実施予定。

・設計コンテストを実施中。
応募締め切りは9/1。10月に受賞者決定。

・「Maryユーザ・フォーラム」を企画予定。

Thank you.