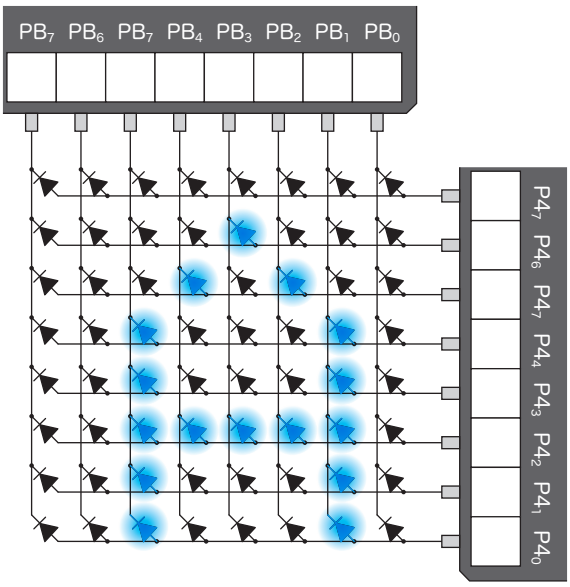
07 STEP

ドットマトリクス LED に「A」と点灯させる

今まではただ LED を点灯させるだけでしたが、
この STEP では、ドットマトリクス LED に「A」と表示させましょう。

問題　7-1

スタティック点灯でドットマトリクス LED に「A」と点灯してみましょう。
STEP06 の復習も兼ねて、下図を使って「A」と表示できるか考えてください。



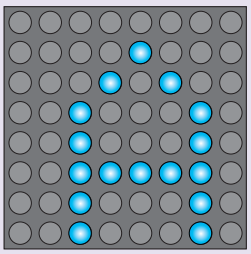
STEP 07 ドットマトリクス LED に「A」と点灯させる

頭をひねられた方、ごめんなさい。スタティック点灯では「A」と表示させることはできません。

実は、ドットマトリクス LED を使用する場合、スタティック点灯では点灯パターンに制限があります。STEP01 でも説明したとおり、ドットマトリクス LED は、LED のアノードとカソードをそれぞれ 8 つずつまとめて配線されています。ですから、スタティック点灯では構造上「A」という表示は不可能なのです。

課題　7-1

ダイナミック点灯で図のように、ドットマトリクス LED に「A」と点灯させる。



7.1　ダイナミック点灯で A を表示させる

ダイナミック点灯で「A」と表示させるには、以下のような点灯パターンを 1 行ずつ高速に切り替える必要があります。なお、図 7-2 は横行で走査する例ですが、縦列で走査する方法もあります。

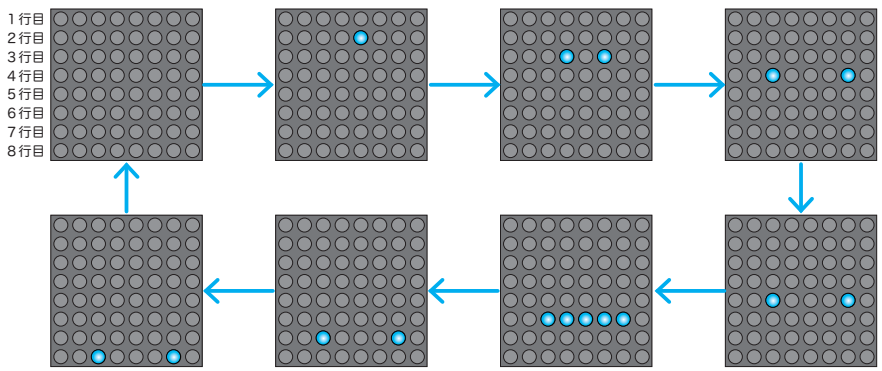


図 7-2　ドットマトリクス LED の点灯の変化

問題　7-2

「A」と表示されるように、各行ごとに PBDR に入れる値を考えましょう。

	P4DR（bin）	PBDR（bin）	PBDR（hex）
1 行目	1000 0000	1111 1111	0xFF
2 行目	0100 0000		
3 行目	0010 0000		
4 行目	0001 0000		
5 行目	0000 1000		
6 行目	0000 0100		
7 行目	0000 0010		
8 行目	0000 0001		

答えは p.94 参照

では、次にフローチャートを考えてみましょう。図 7-3 はダイナミック点灯で「A」と表示させるフローチャートです。初期化関数と待ち時間関数は省略してあります。また、待ち時間は 1m 秒に設定することにします。

※ 本 STEP 以降は、初期化関数と待ち時間関数のフローチャートは省略します。

待ち時間はなぜ必要？

待ち時間を入れずにプログラムを実行すると、速度が速すぎて LED の点灯が間に合わず残像が残るため「A」と表示しているように見えません。実際どうなるかプログラムの待ち時間をコメントアウトして確かめてみてもいいでしょう。

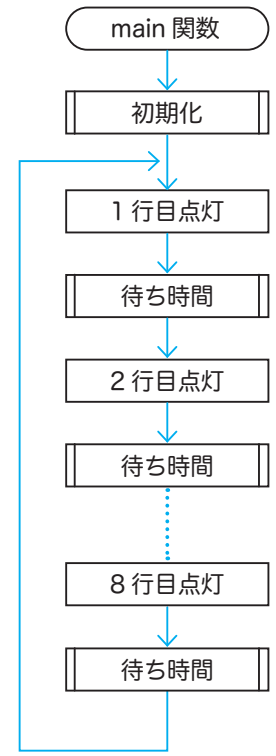


図 7-3　フローチャート

STEP 07 ドットマトリクス LED に「A」と点灯させる

プログラム例　7-1

```
/*
 *  main 関数
 */
int main(void)
{
    initIO(); // 初期化関数の呼び出し

    // 永久ループ
    while (1)
    {
        // 1 行目点灯
        P4.DR.BYTE = a_p4[0];
        PB.DR.BYTE = 0xFF;   // ○○○○○○○○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 2 行目点灯
        P4.DR.BYTE = a_p4[1];
        PB.DR.BYTE = 0xF7;   // ○○○○●○○○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 3 行目点灯
        P4.DR.BYTE = a_p4[2];
        PB.DR.BYTE = 0xEB;   // ○○○●○●○○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 4 行目点灯
        P4.DR.BYTE = a_p4[3];
        PB.DR.BYTE = 0xDD;   // ○○●○○○●○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 5 行目点灯
        P4.DR.BYTE = a_p4[4];
        PB.DR.BYTE = 0xDD;   // ○○●○○○●○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 6 行目点灯
        P4.DR.BYTE = a_p4[5];
        PB.DR.BYTE = 0xC1;   // ○○●●●●●○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 7 行目点灯
        P4.DR.BYTE = a_p4[6];
        PB.DR.BYTE = 0xDD;   // ○○●○○○●○
        waitMs(1);           // 待ち時間関数の呼び出し

        // 8 行目点灯
        P4.DR.BYTE = a_p4[7];
        PB.DR.BYTE = 0xDD;   // ○○●○○○●○
        waitMs(1);           // 待ち時間関数の呼び出し
    }

    return 0;
}
```

7.2　プログラムを整理する

PBDR に入れる点灯パターンも配列にして、プログラムを整理しましょう。
フローチャートは以下のようになるでしょう。

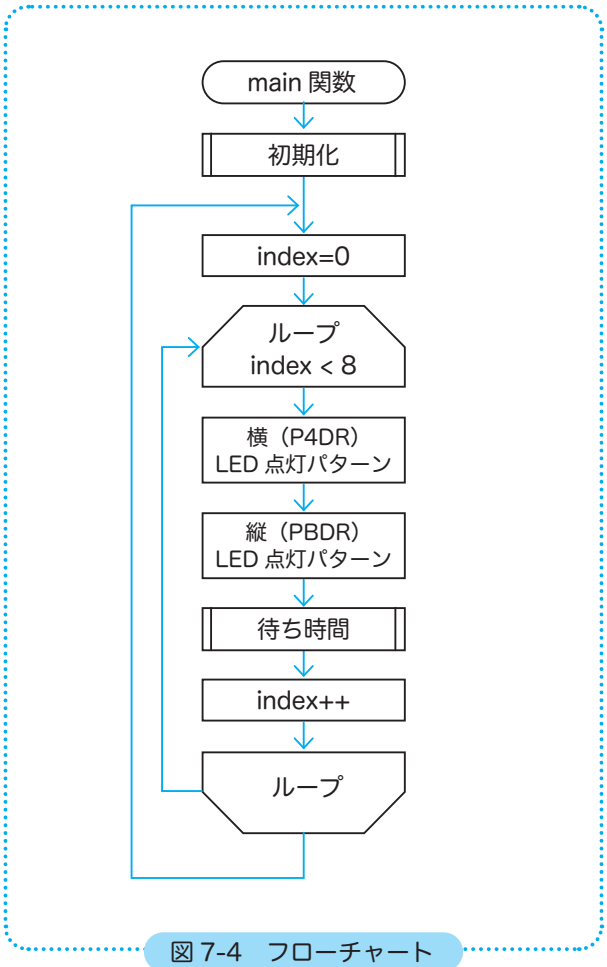


図 7-4　フローチャート

ドットマトリクス LED に「A」と点灯させる

プログラム例　7-2

```
07  // 横点灯行の配列
08  int a_p4[8] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
09
10  // 点灯パターン A の配列
11  int a_pb[8] = {0xFF, 0xF7, 0xEB, 0xDD, 0xDD, 0xC1, 0xDD, 0xDD};

    中略

40  /*
41   *  main 関数
42   */
43  int main(void)
44  {
45      initIO(); // 初期化関数の呼び出し
46
47      // 永久ループ
48      while (1)
49      {
50          int index = 0; // 両配列に使う添え字の変数
51
52          for (index = 0; index < 8; index++)
53          {
54              P4.DR.BYTE = a_p4[index];
55              PB.DR.BYTE = a_pb[index];
56
57              waitMs(1); // 待ち時間関数の呼び出し
58          }
59      }
60
61      return 0;
62  }
```

a_pb 配列を変更すれば、好きな図柄を表示させることができますね。