



実践ロボットプログラミング

LEGO Mindstorms NXT で目指せロボコン!

WEB : <http://robot-programming.jp/>

著者 : 藤吉弘亘, 藤井隆司, 鈴木裕利, 石井成郎

E-mail : support@robot-programming.jp

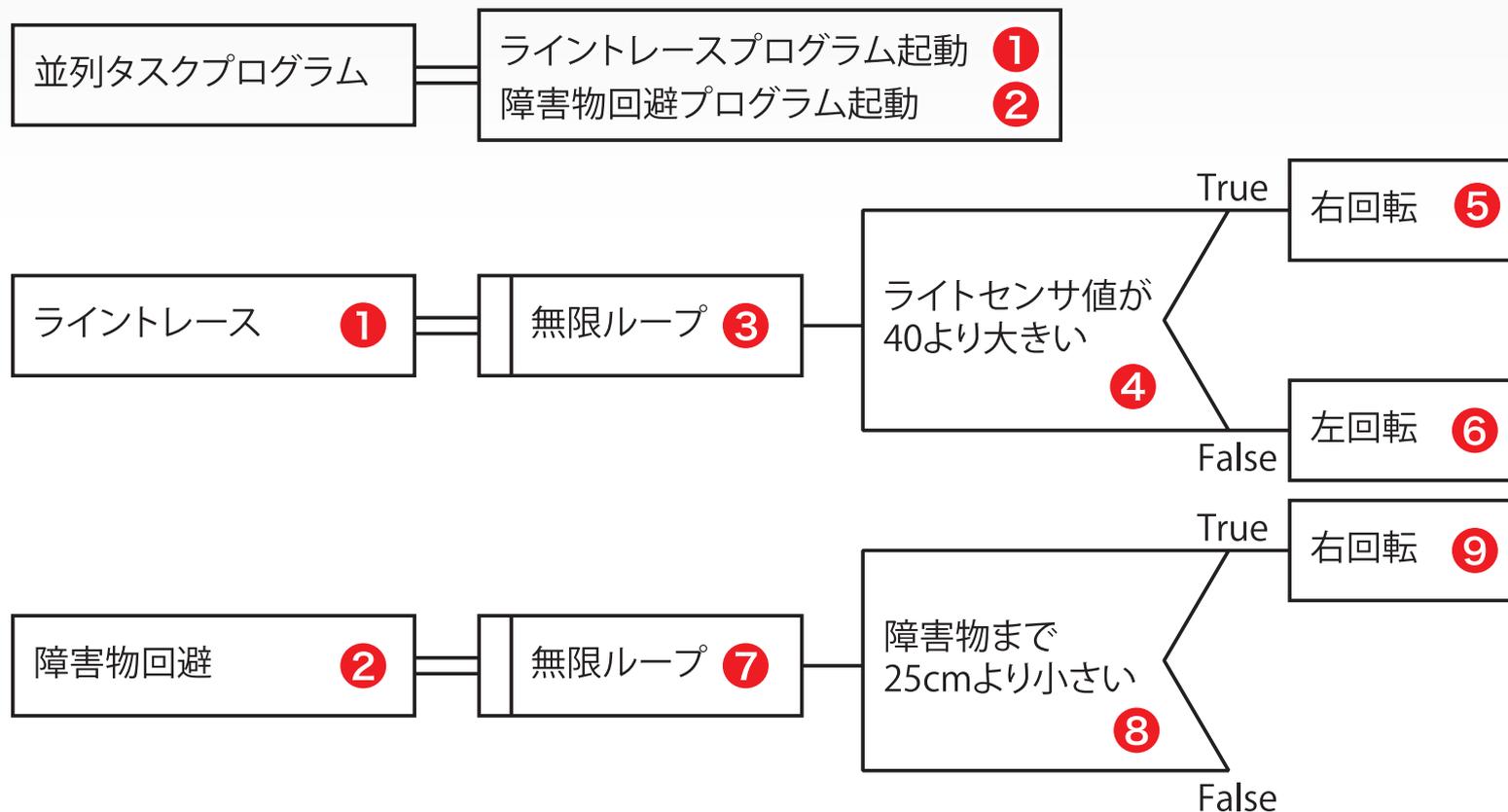


■ 並列タスク



PADによる並列タスクの図示

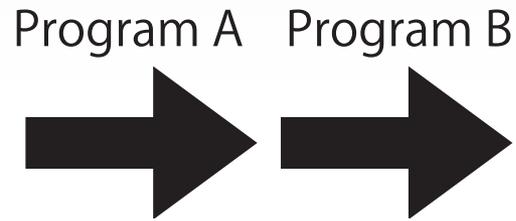
・ 並列タスク



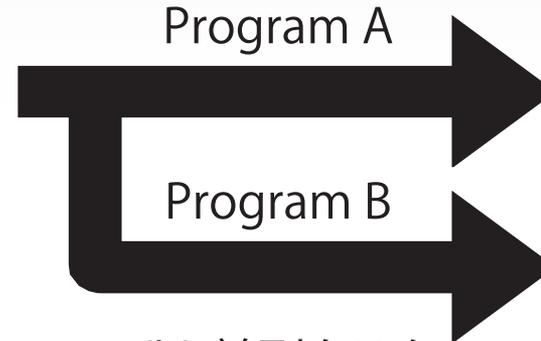


シングルタスク

- ・ シングルタスクと並列タスク



(a) シングルタスク



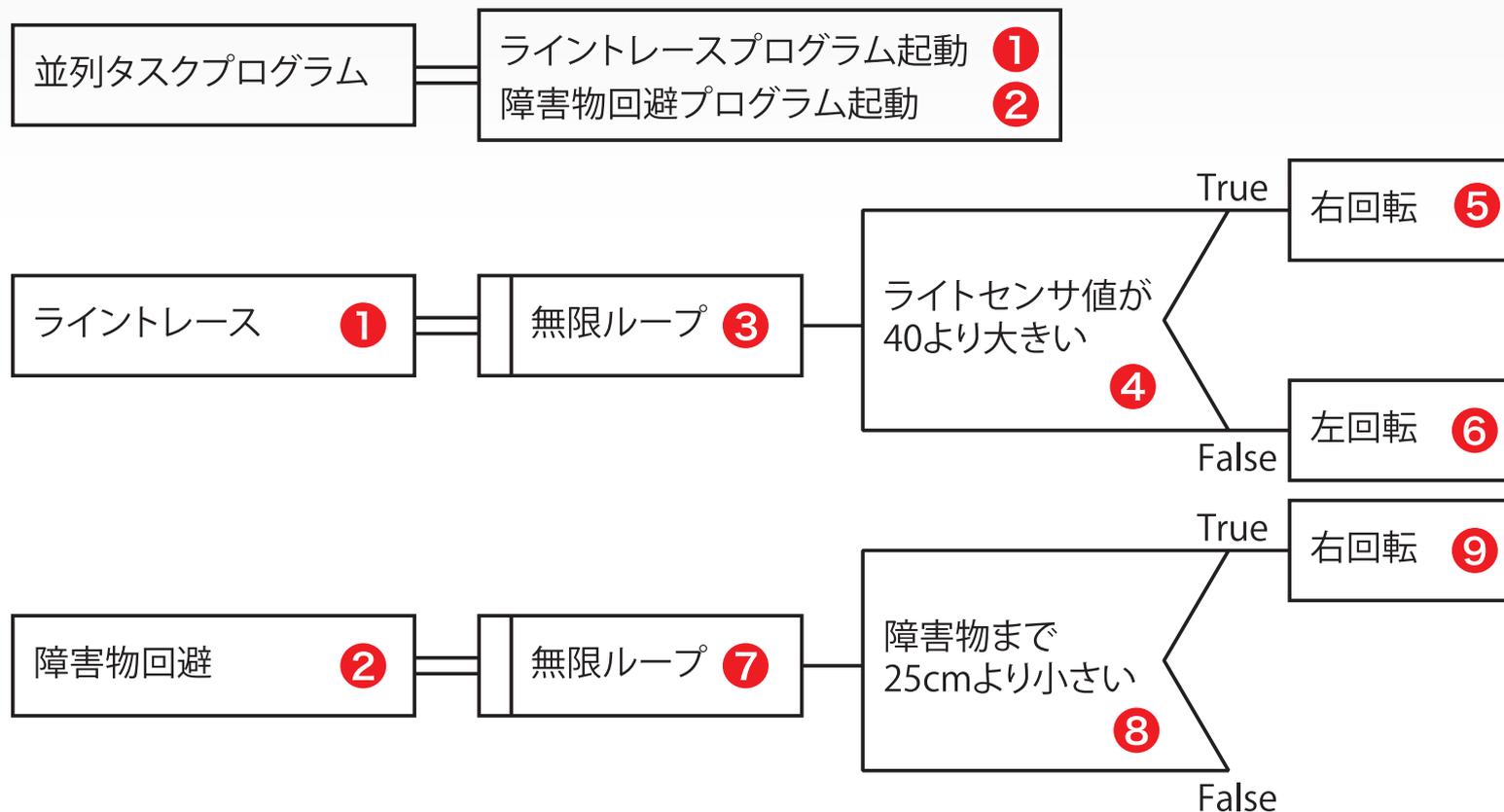
(b) 並列タスク

→ ライトレースと障害物回避を同時に行うには並列タスク化する



PADによる並列タスクの図示

・ 並列タスク





並列タスク (p.83: parallel_task1.nxc)

parallel_task1.nxc

```
#define TURN_RIGHT 100          // 回転する時間
#define POW        75

void turn_right(int time){省略} // ロボットを右回転
void turn_left(int time){省略}  // ロボットを左回転
```

```
1 task line_trace()           // ライントレース
{
  3 while(true){
    4 if(Sensor(IN_2) > 40){
      5 turn_right(1);          // 右回転
    }else{
      6 turn_left(1);          // 左回転
    }
  }
}
```

続く⇒



並列タスク (p.83: parallel_task1.nxc)

続き

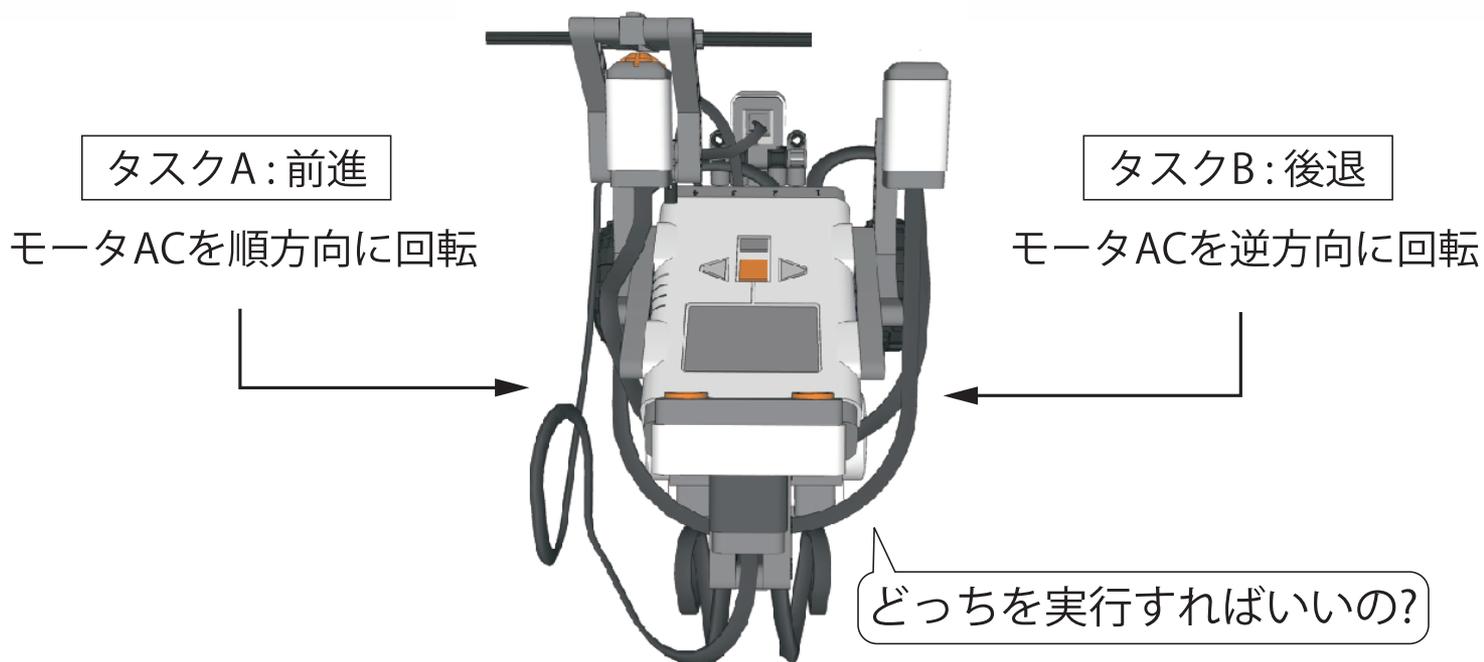
parallel_task1.nxc

```
2 task collision_avoidance() // 障害物回避
{
  7 while(true){
    8 if(SensorUS(IN_4) < 25){
      9 turn_right(TURN_RIGHT);
    }
  }
}

task main()
{
  SetSensorLight(IN_2); // 入力ポート 2 をライトセンサに設定
  SetSensorLowspeed(IN_4); // 入力ポート 4 を超音波センサに設定
  Precedes(line_trace, collision_avoidance); // 並列タスクの起動
}
```

並列タスクにおける問題

- 命令の衝突（コンフリクト）
 - タスクAがタスクBの実行を妨害
 - 同時にモータを制御





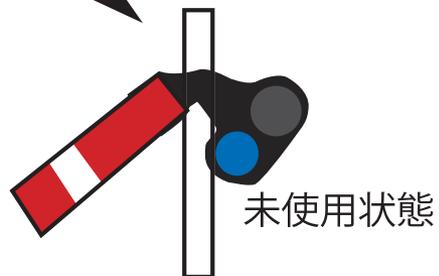
セマフォ (信号灯)

① モーターの未使用状態

タスク A

タスク B

リクエスト



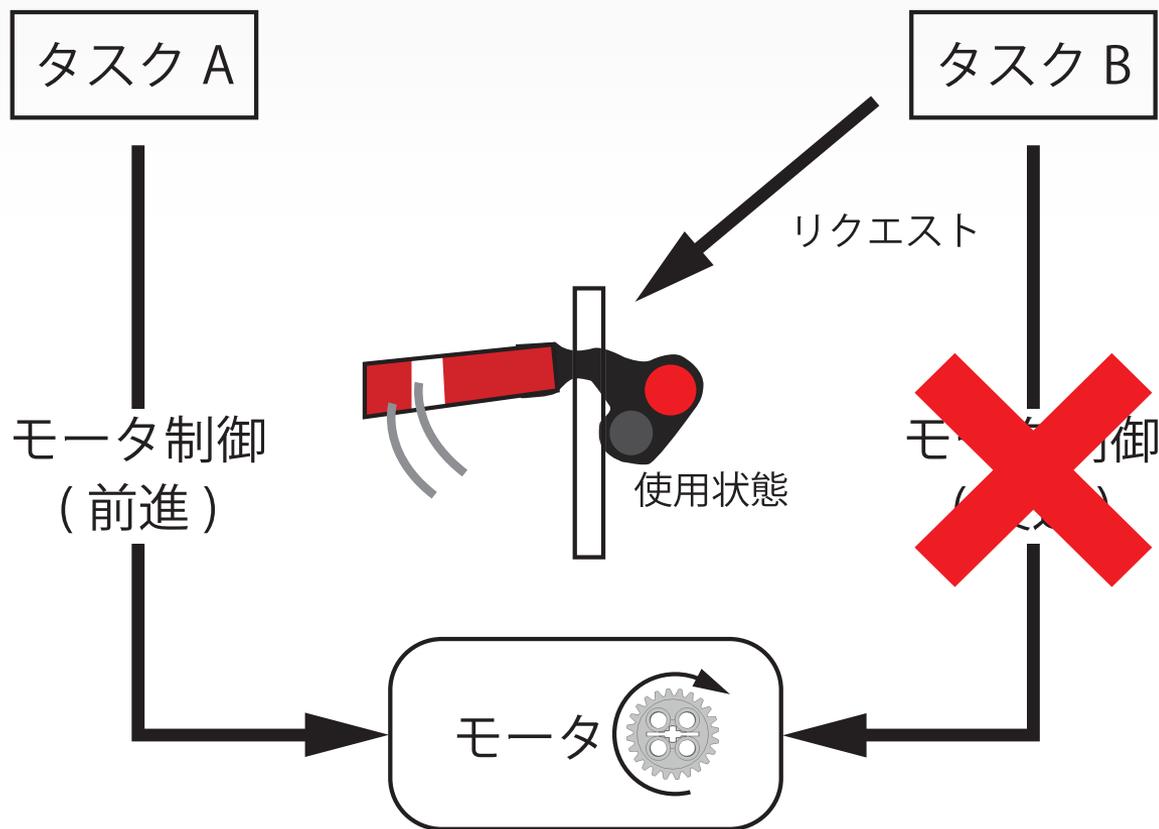
モーター





セマフォ (信号灯)

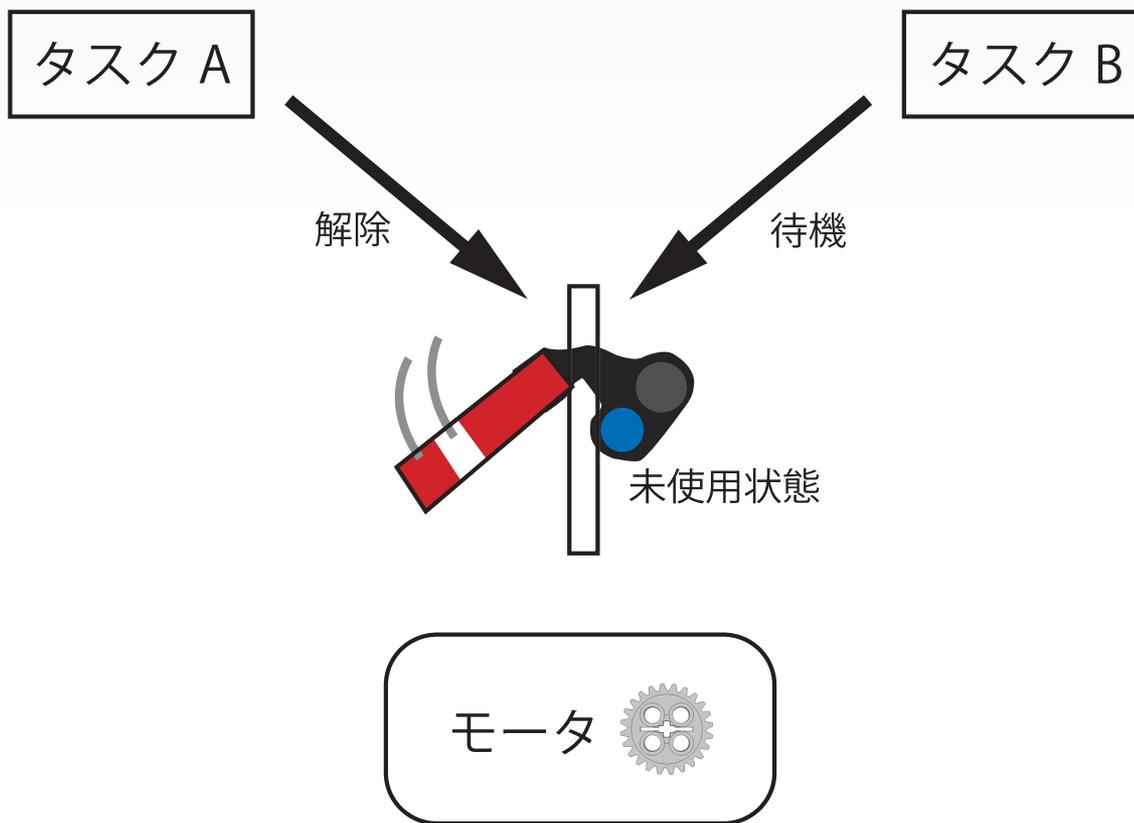
② タスク A によるモータ制御





セマフォ (信号灯)

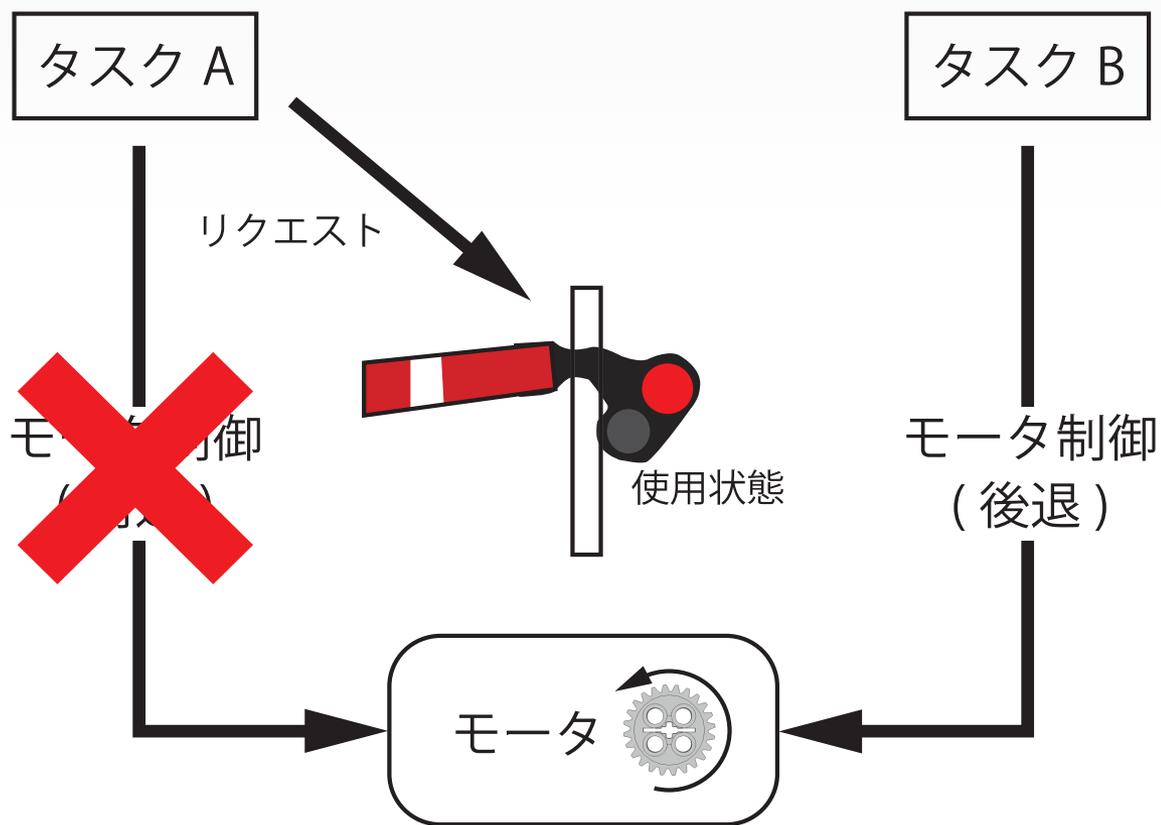
③ タスク A によるモータ制御終了





セマフォ (信号灯)

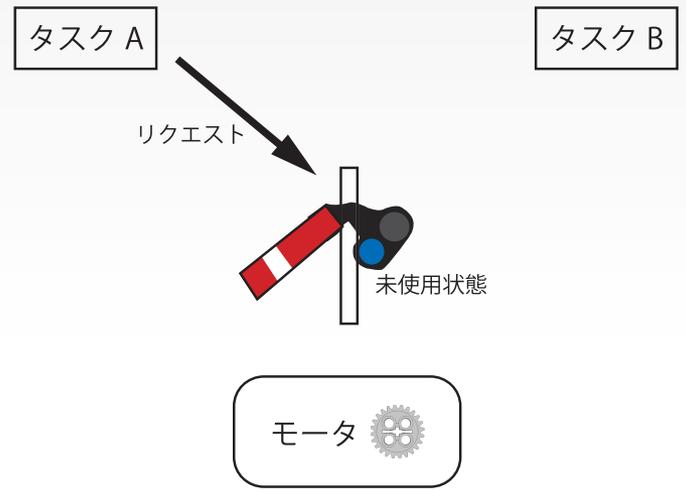
④ タスク B によるモータ制御



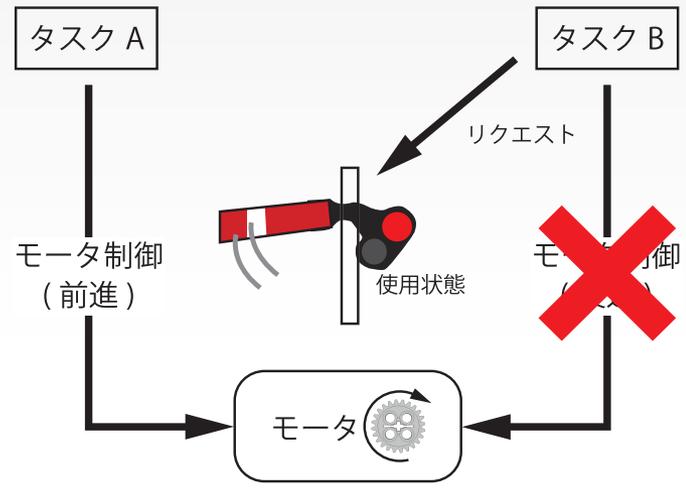
セマフォによる並列タスクのコンフリクトを回避



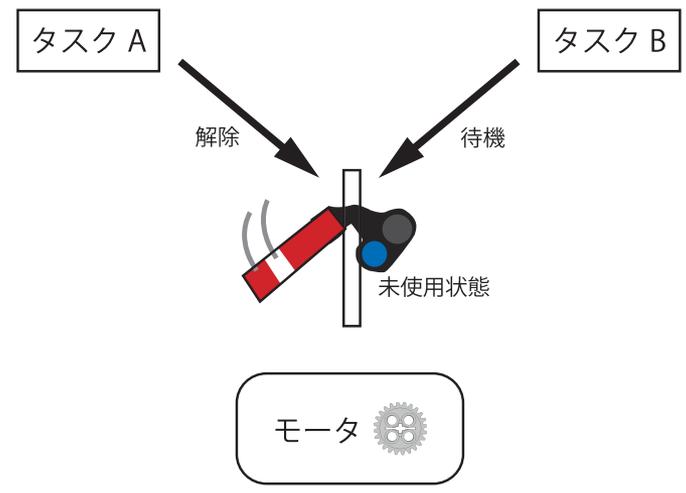
① モータの未使用状態



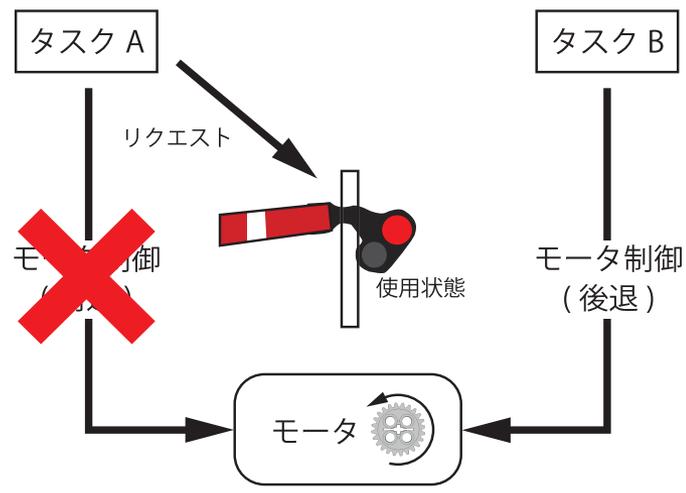
② タスク A によるモータ制御

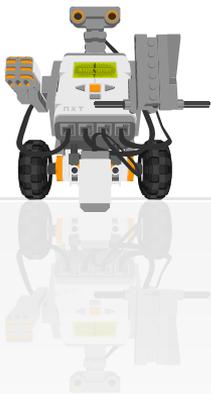


③ タスク A によるモータ制御終了



④ タスク B によるモータ制御



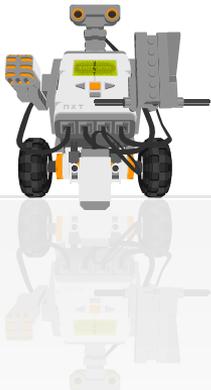


mutex変数の導入 (NXC独自)

- ・ モーターが使用されているかどうかを判断
 - Acquire(mutex)
 - ・ モータが使用されていない状態になるまで待つ
- ・ モータの使用後、解放
 - Release(mutex)
 - ・ モータを解放

```
mutex MoveMutex;

Acquire(moveMutex);
OnFwd(OUT_AC, 75);
Wait(1000);
Release(moveMutex);
```



並列タスク (p.87: parallel_task2.nxc)

```
#define TURN90 100
```

```
① mutex moveMutex;
```

```
void turn_right(int time){省略} // 右回転
```

```
void turn_left(int time){省略} // 左回転
```

```
task line_trace() // ライントレース
```

```
{
```

```
while(true){
```

```
② if(Sensor(IN_2) > 40){
```

```
③ Acquire(moveMutex);
```

```
⑤ turn_right(0); // 右回転
```

```
⑥ Release(moveMutex);
```

```
}else{
```

```
Acquire(moveMutex);
```

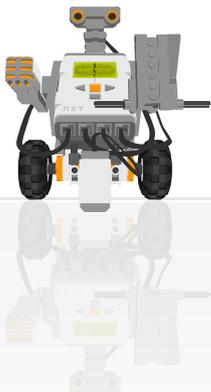
```
turn_left(0); // 左回転
```

```
Release(moveMutex);
```

```
}
```

```
}
```

```
}
```



並列タスク (p.87: parallel_task2.nxc)

```
task colision_avoidance()                // 障害物回避
{
    while(true){
        ⑦ if(SensorUS(IN_4) < 25){
            Acquire(moveMutex);
            ⑧ turn_right(TURN90);
            Release(moveMutex);
        }
    }
}

task main()
{
    SetSensorLight(IN_2);
    SetSensorLowspeed(IN_4);
    Precedes(line_trace, colision_avoidance); // 並列タスクの起動
}
```