

1. ゴール

PIO_R の16ビット設定値を導き出す。

2. 出力形式

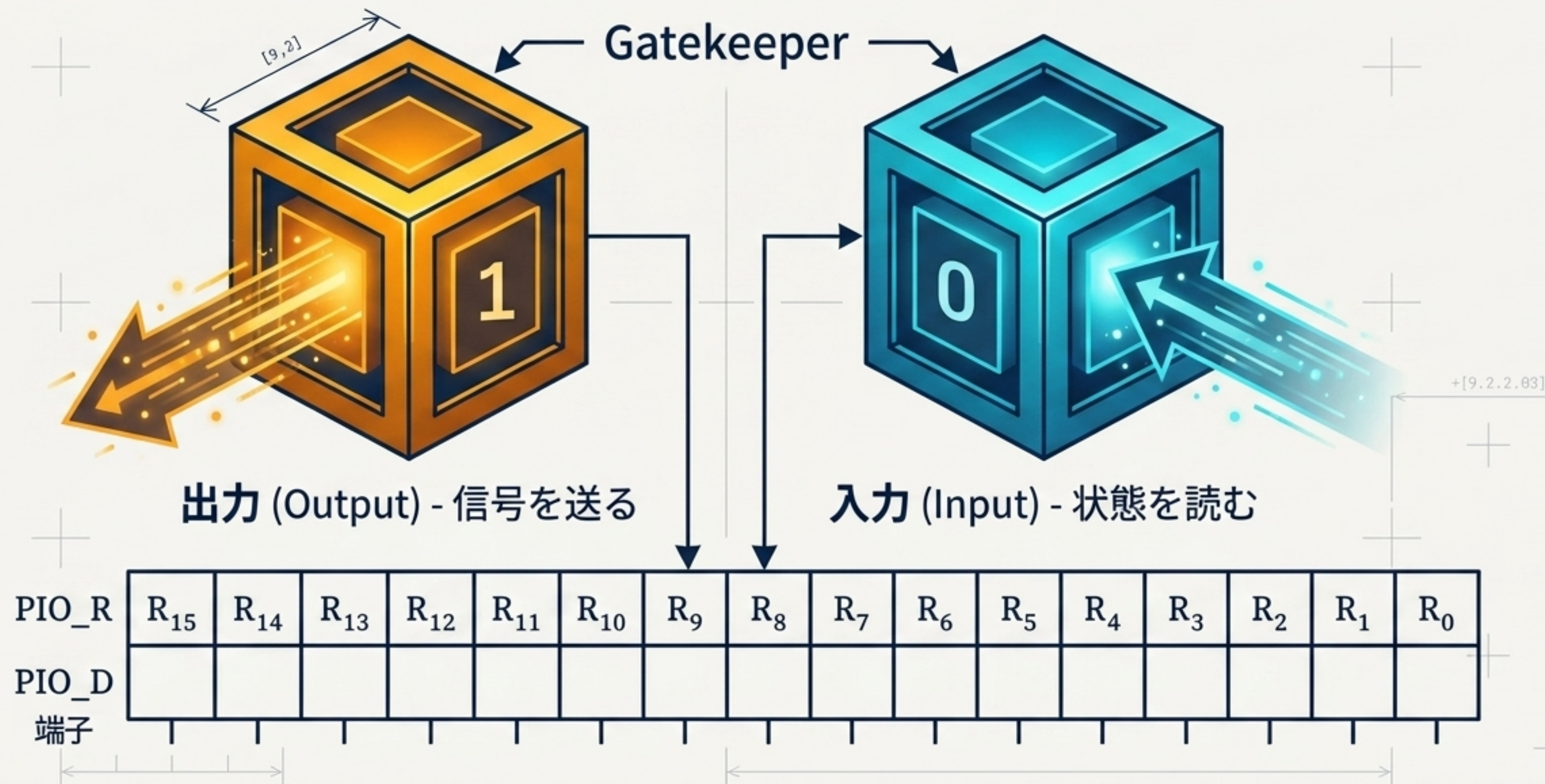
16進数4けた

初期化プログラムによって、PIO_Rを初期化する設定値を16進数4けたで答えよ。
ただし、PIO_Dの未使用のビットは入力として設定すること。

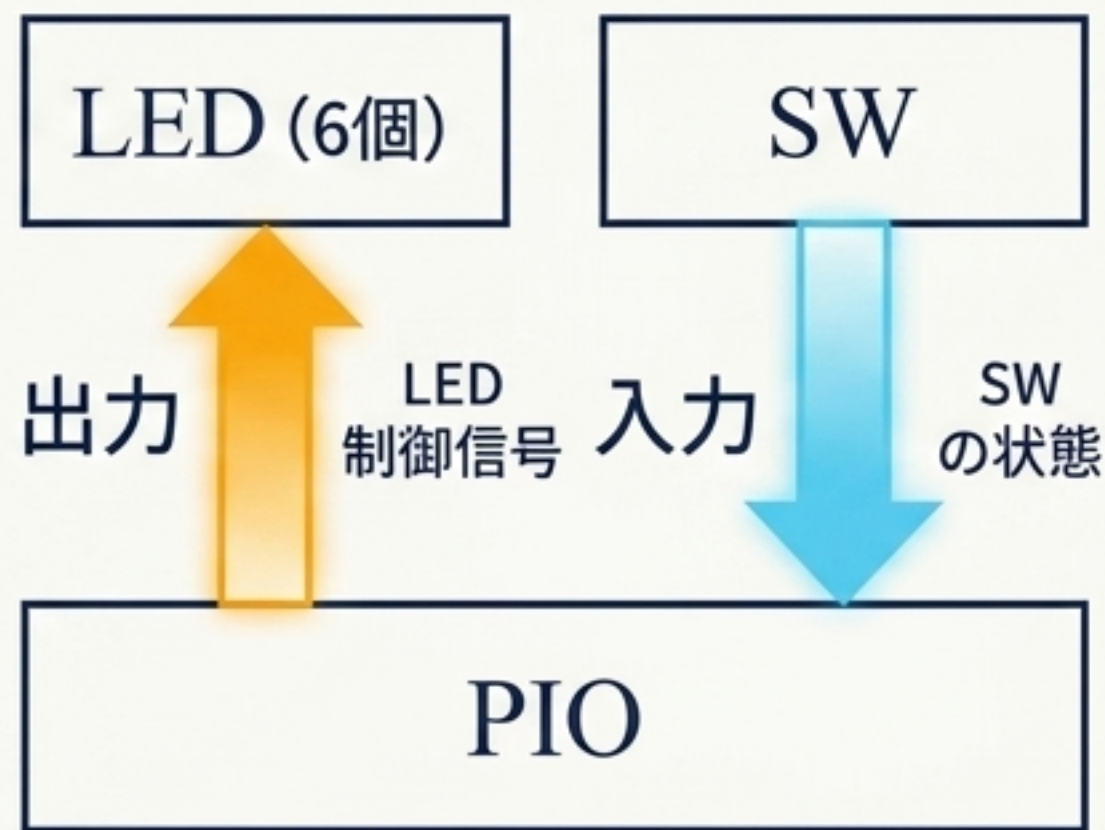
⚠ 重要制約

未使用ビット = 「入力」として設定。

PIO_R（入出力制御レジスタ）の絶対ルール： PIO_Dの各端子（ビット）のデータ通信方向を決定する。



ソフトウェアでPIOを制御するには、
「物理的なハードウェアが
どのピンに接続されているか」を
把握する必要がある。



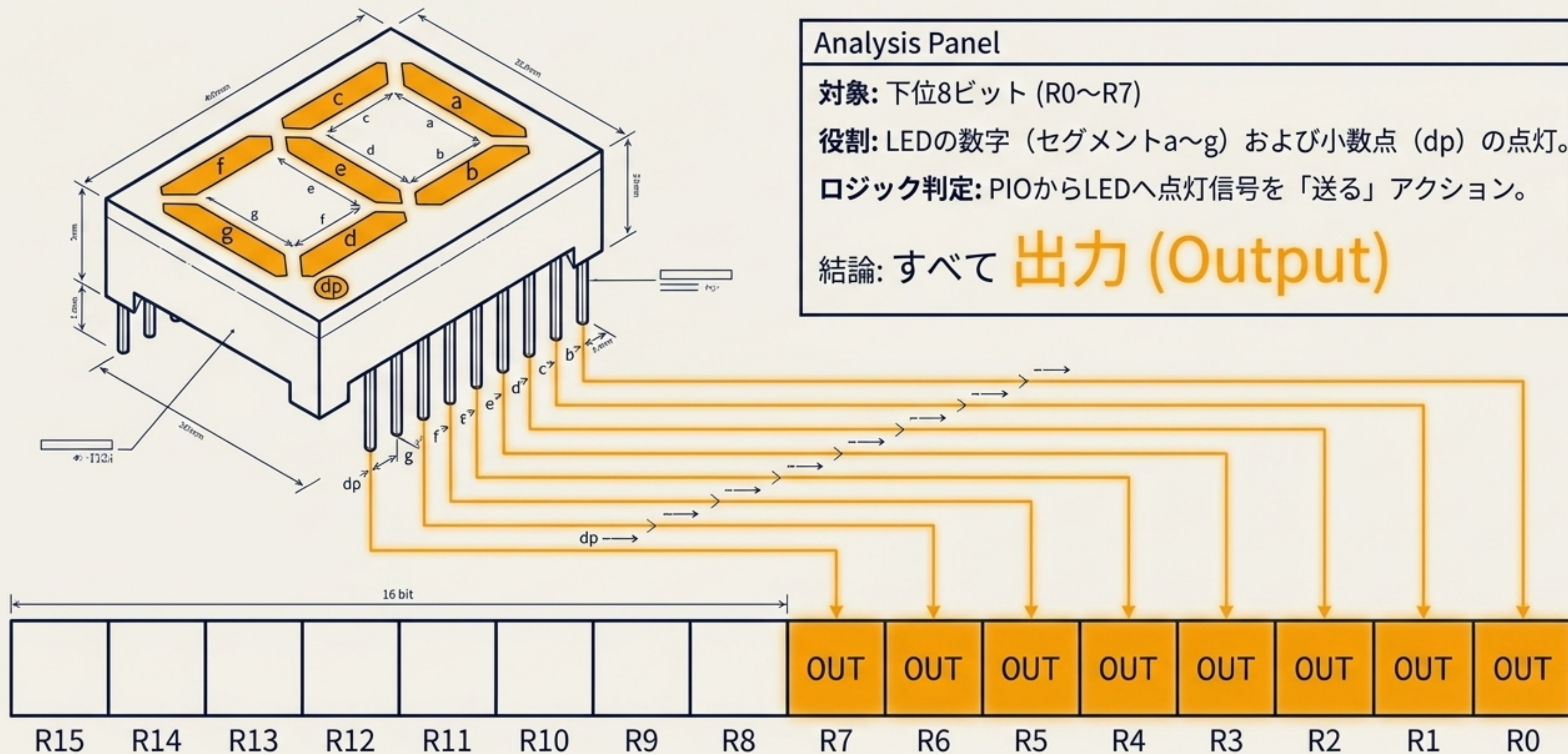
+ [9.2.2.03]

[上位8ビット (R8-R15)] :
LEDの選択・スイッチ入力・未使用枠

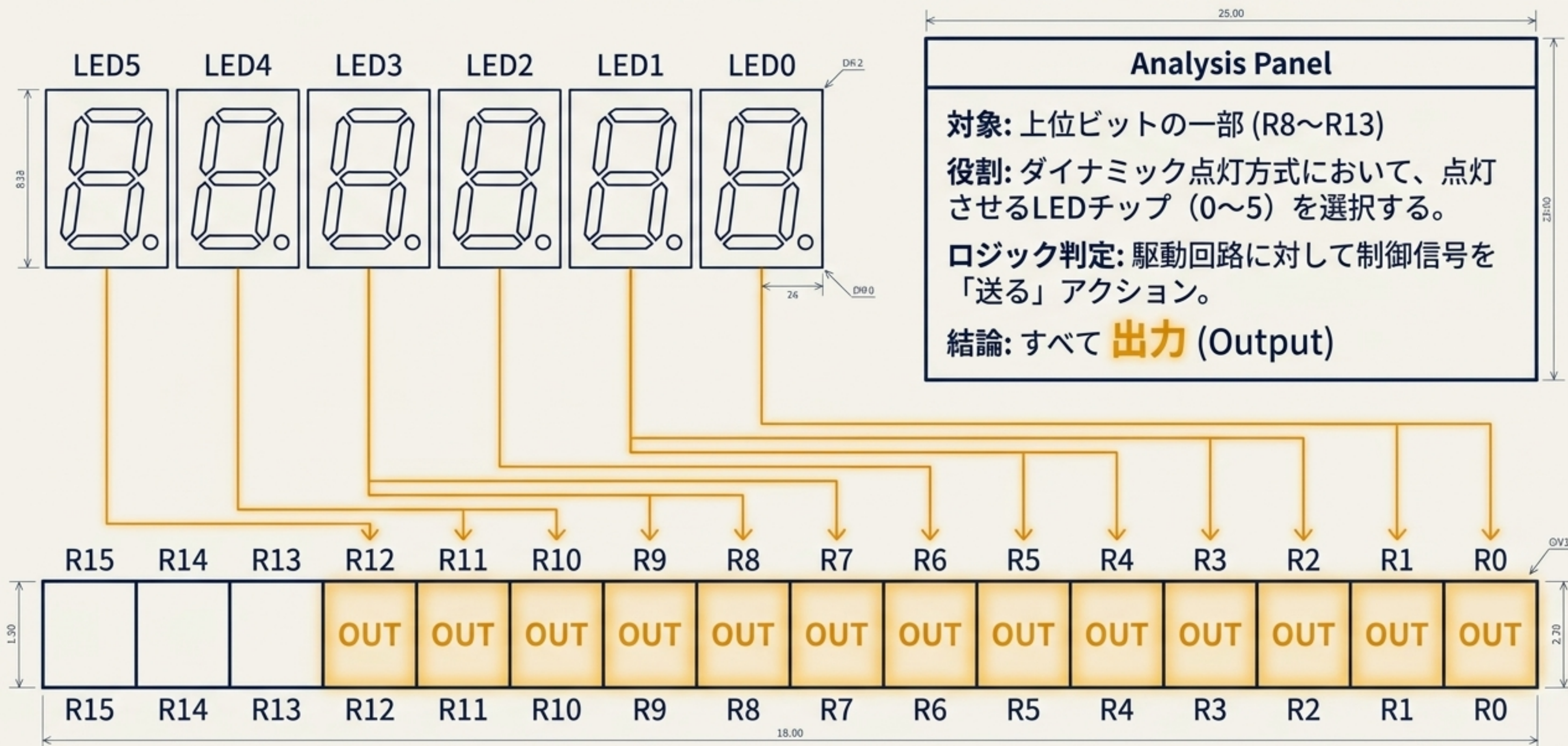
[下位8ビット (R0-R7)] :
LEDセグメントの表示制御

+ [9.2.2.03]

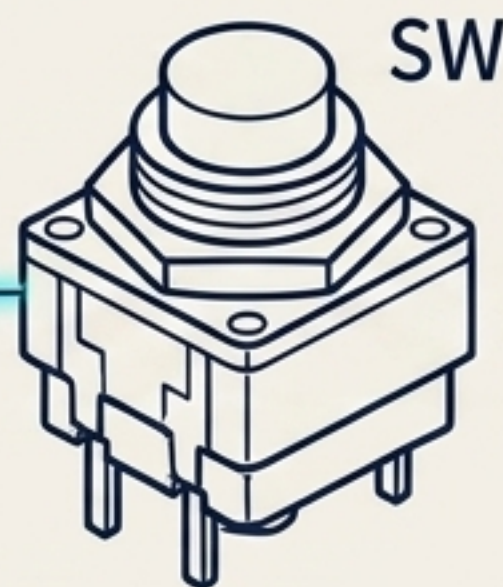
PIO_D 下位8ビット (R0-R7): LEDセグメント制御 (出力)



PIO_D 上位ビット (R8-R13): LED選択制御 (出力)



PIO_D 第14ビット (R14): スイッチ入力制御



Analysis Panel

対象: 第14ビット (R14)

役割: 押しボタン式スイッチ (SW) の接続。

ロジック判定: 物理的なスイッチのON/OFF状態をPIOが「読み取る」アクション。

結論: **入力 (Input)**

R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
	IN	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT
R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0

System Constraint

Analysis Panel

対象: 最上位ビット (R15)

役割: 未使用 (Unused)

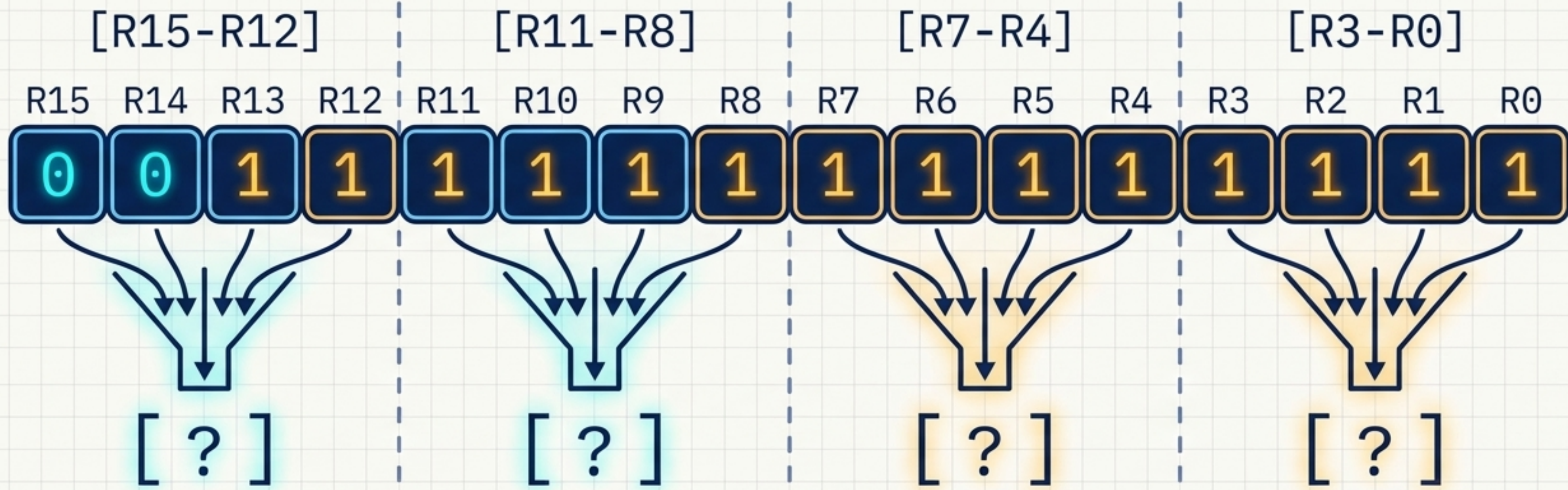
ロジック判定: 設問の絶対条件
「未使用のビットは入力として
設定すること」を適用。

結論: 強制入力 (Input)

R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
IN	IN	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT

R15	R14	R13	R12	R1	R0	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0
IN	IN	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT

[ビット範囲]	[接続ハードウェア]	[信号の方向]	[論理状態 (PIO_R)]
R0 - R7	7セグメント & DP表示	PIO → LED (送る)	出力 (OUT)
R8 - R13	桁選択 (LED0 - LED5)	PIO → 駆動回路 (送る)	出力 (OUT)
R14	スイッチ (SW)	SW → PIO (読む)	入力 (IN)
R15	未使用	制約による強制指定	入力 (IN)



最終工程：2進数から16進数への変換

Step 1: 判定したIN(0) / OUT(1)の配列を4ビットごとに分割する。

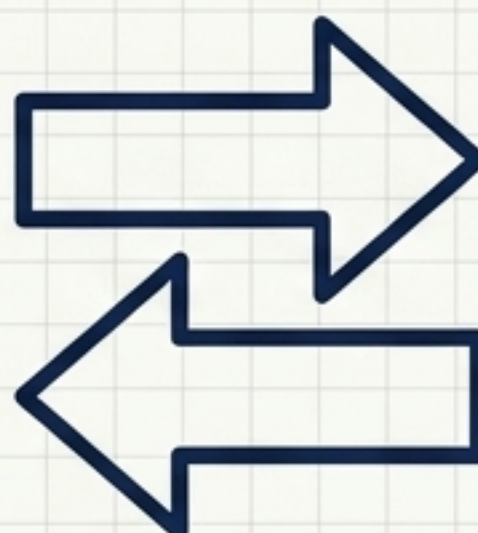
Step 2: 各4ビットのブロックを、1つの16進数文字 (0~9, A~F) に変換する。

構築されたロジック回路をもとに、最終的な4けたの解答を導き出してください。



仕様と物理の 結びつけ

ハードウェアコンポーネント（LED, SW）が、論理レジスタのどのビット空間にマッピングされているかを正確に図解から読み取る。



「方向」による 入出力定義

信号を「駆動する・送る」のか、「状態を読む」のか。電氣的なデータの流れをイメージすることで、出力(1)と入力(0)の迷いをなくす。



見落としやすい 制約条件

「未使用ビットの扱い」など、設問の末尾に隠された例外ルールはシステム全体の挙動（および解答）を左右する。必ずマークする。