**Trường Đại học Bách Khoa**

**Khoa Điện – Điện Tử**

**Bộ môn Viễn Thông**  
 **---------------o0o---------------**

**BÁO CÁO**

**LẬP TRÌNH HỆ THỐNG NHÚNG**

**GVHD:** Th.s Nguyễn Phan Hải Phú

**Học kì: 251 Lớp: L01**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **HỌ VÀ TÊN** | **MSSV** | **Đóng góp** |
| 1 | Võ Sơn Khoa | 2211663 | 100% |
| 2 | Tạ Duy Khiêm | 2211575 | 100% |
| 3 | Nguyễn Trường Huy | 2211252 | 100% |
| 4 | Nguyễn Tầm Chân Lý | 2211975 | 100% |
| 5 | Lê Quốc Hoàng | 2434002 | 100% |

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2025**

**Mục lục**

[Phần A: I/O PORT 3](#_Toc216209811)

[1. Giới thiệu tổng quan về ngoại vi I/O PORT 3](#_Toc216209812)

[1.1. Tổng quan 3](#_Toc216209813)

[1.2. Đặc điểm 3](#_Toc216209814)

[2. Mô tả các thanh ghi 5](#_Toc216209815)

[**2.1.** **Thanh ghi điều khiển port 1 (Port Control Register)** 5](#_Toc216209816)

[**2.2.** **Thanh ghi điều khiển port 2 (Port Control Register)** 5](#_Toc216209817)

[2.3. Thanh ghi điều khiển port 3 (Port Control Register) 5](#_Toc216209818)

[**2.4.** **Thanh ghi điều khiển port 4 (Port Control Register)** 5](#_Toc216209819)

[**2.5.** **PmnPFS/PmnPFS\_HA/PmnPFS\_BY : Port mn Pin Function Select Register (m = 0 to 9, A, B, n = 00 to 15)** 5](#_Toc216209820)

[**2.6.** **PWPR : Write-Protect Register** 5](#_Toc216209821)

[**2.7.** **PFENET : Ethernet Control Register** 5](#_Toc216209822)

[**2.8.** **PWPRS : Write-Protect Register for Secure** 5](#_Toc216209823)

[3. CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA I/O PORT 5](#_Toc216209824)

[**3.1.** **Chế độ GPIO (General Purpose Input/Output)** 5](#_Toc216209825)

[**3.2.** **Chế độ Peripheral Function (chức năng ngoại vi)** 5](#_Toc216209826)

[**3.3.** **Chế độ Analog (ADC input hoặc comparator)** 5](#_Toc216209827)

[**3.4.** **Chế độ Event Link Controller (ELC)** 5](#_Toc216209828)

[**3.5.** **Chế độ Open-Drain Output** 5](#_Toc216209829)

[**3.6.** **Chế độ Drive Strength (sức kéo dòng)** 5](#_Toc216209830)

[**3.7.** **Chế độ Secure/Non-Secure (TrustZone)** 5](#_Toc216209831)

[**3.8.** **Chế độ Ethernet (RMII/MII)** 5](#_Toc216209832)

[**3.9.** **Chế độ Double-edge Detection (EVENT ON RISING/FALLING)** 5](#_Toc216209833)

[4. Chương trình ví dụ về ngoại vi 5](#_Toc216209834)

[Phần B: GENERAL PWM TIMER (GPT) 5](#_Toc216209835)

[1. Giới thiệu tổng quan về ngoại vi 5](#_Toc216209836)

[**1.1** **Cấu trúc cơ bản và Nguồn Clock:** 5](#_Toc216209837)

[**1.2** **Hệ thống Thanh ghi và Buffer:** 5](#_Toc216209838)

[**1.3** **Khả năng Kích hoạt và Đồng bộ:** 5](#_Toc216209839)

[**1.4** **Tính năng Bảo vệ và Điều khiển Đầu ra (POEG):** 5](#_Toc216209840)

[2. Mô tả các thanh ghi liên quan đến GPT 5](#_Toc216209841)

[3. Mô tả các chế độ hoạt động của GPT 5](#_Toc216209842)

[**3.1.** **Các chế độ hoạt động chính của GPT được chọn thông qua các bit MD[2:0] trong thanh ghi GTCR:** 5](#_Toc216209843)

[**3.2.** **Các chức năng hoạt động khác:** 5](#_Toc216209844)

[4. Các chương trình ví dụ về ngoại vi 5](#_Toc216209845)

[PHẦN C: GIẢI THUẬT LẬP TRÌNH CK RA6M5 ĐỌC CẢM BIẾN VÀ GỬI CHUỖI DỮ LIỆU QUA UART 6](#_Toc216209846)

[Lưu đồ giải thuật 6](#_Toc216209847)

[PHẦN D : THIẾT KẾ WEB SERVER HIỂN THỊ DỮ LIỆU ĐỌC TỪ PORT UART 6](#_Toc216209848)

[1. Lựa chọn framework: 6](#_Toc216209849)

[+ Tốc độ Phát triển Nhanh chóng (Rapid Development) 6](#_Toc216209850)

[2. Khởi tạo server: 6](#_Toc216209851)

# Phần A: I/O PORT

## Giới thiệu tổng quan về ngoại vi I/O PORT

### 1.1. Tổng quan

I/O Port (Input/Output Port) là ngoại vi cơ bản của vi điều khiển, cho phép giao tiếp giữa MCU và thế giới bên ngoài thông qua các chân vật lý (pin). Trên RA6M5, các cổng I/O được chia thành các port (P000 – P915): Port 0, Port 1, … Port 9, mỗi port gồm tối đa 16 chân. Mỗi chân (pin) của MCU có thể được cấu hình làm:

Ngõ vào (Input): để đọc tín hiệu logic (0 hoặc 1).

Ngõ ra (Output): để xuất tín hiệu điều khiển.

Điều khiển trực tiếp bằng phần mềm, hoặc giao cho các chức năng ngoại vi khác (UART, SPI, I2C, ADC,...).

### 1.2. Đặc điểm

Mức điện áp logic: 0 V (Low) và 3.3 V (High).

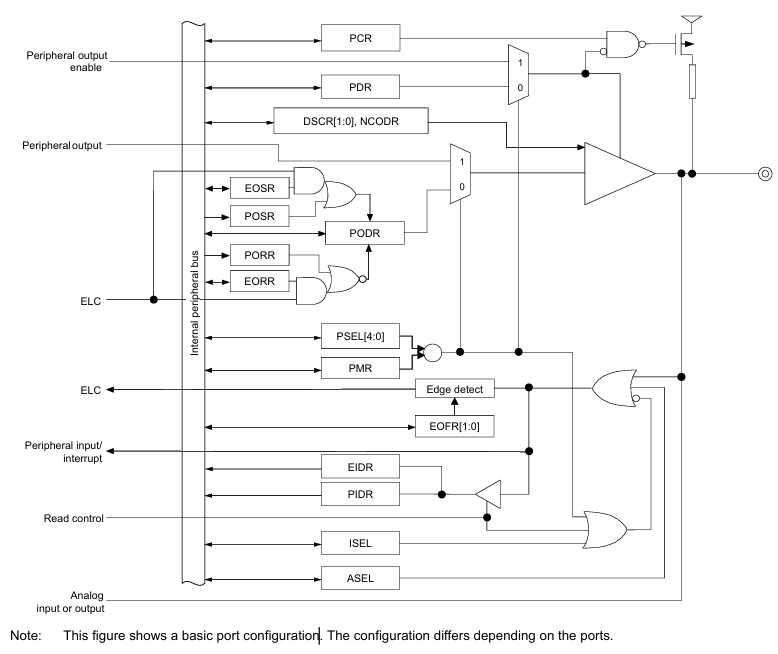
Có thể kích hoạt điện trở kéo lên / kéo xuống (pull-up / pull-down).

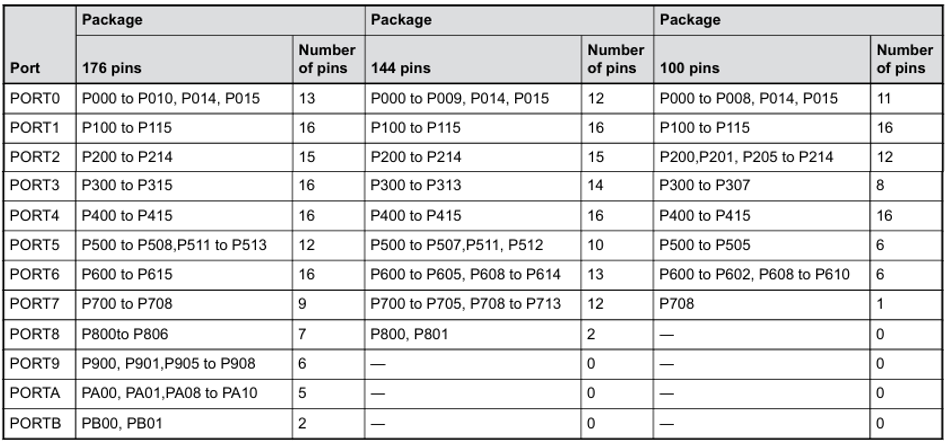
Hỗ trợ tùy chọn chế độ ngắt (interrupt) khi thay đổi mức logic.

Có bảo vệ ghi cho thanh ghi cấu hình.

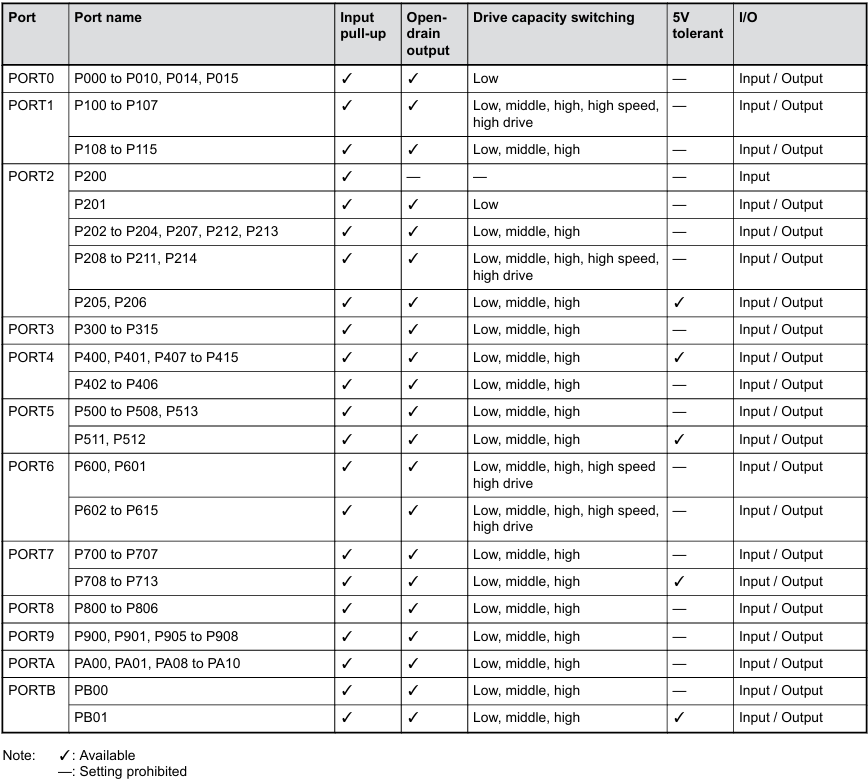
Mỗi chân có chức năng đa năng (multi-function pin), được định nghĩa qua module Function Select Pin (FSP).

Tất cả các chân ngoại trừ P109 (là TDO của cổng JTAG) hoạt động như các chân đầu vào ngay sau khi reset, và chức năng của các chân được chuyển đổi bằng các thiết lập thanh ghi. Các cổng I/O và mô-đun ngoại vi cho mỗi chân được chỉ định trong các thanh ghi liên quan.



**Hình 1: Sơ đồ kết nối cho các thanh ghi cổng I/O**

**Bảng 1: Các thông số kỹ thuật của cổng I/O theo gói**



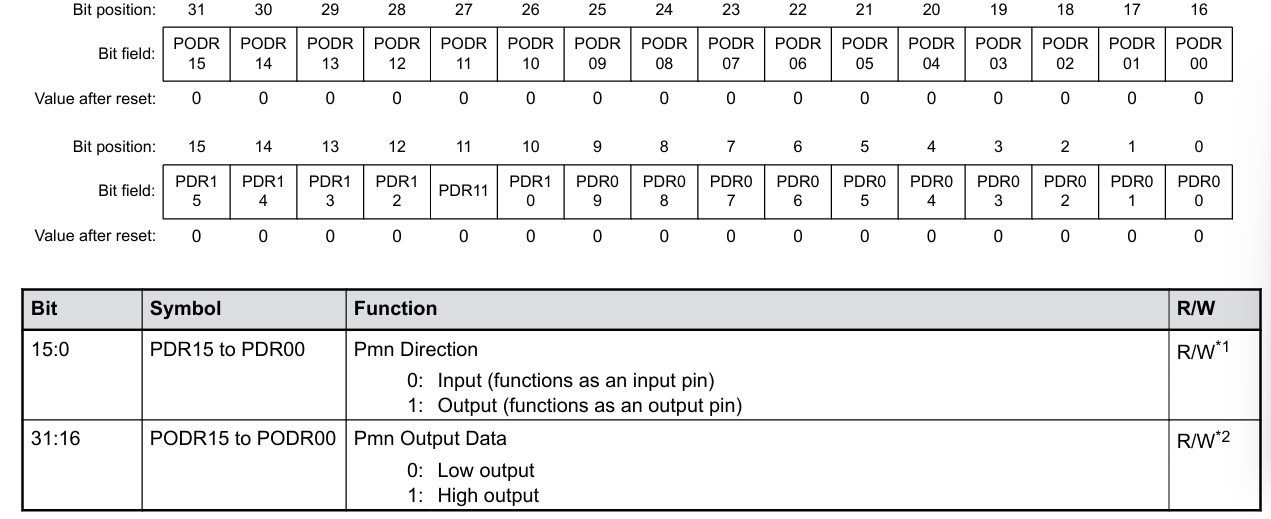
**Bảng 2: Chức năng các cổng I/O**

## Mô tả các thanh ghi

1. **Thanh ghi điều khiển port 1 (Port Control Register)**

Base address: PORTm = 0x4008\_0000 + 0x0020 × m (m = 0 to 9, A, B)

Offset address: 0x000 (PCNTR1/PODR) 0x002 (PDR)

****

1. **PDRn bits (Pmn Direction)**

Quy định mỗi chân là Input (0) hay Output (1).

R\_PORTm->PDR\_b.PDRn, (m,n: Thay thành Port, Pin tương ứng)

Ví dụ: R\_PORT1->PDR\_b.PDR5 = 1; // P105 = Output

1. **PODRn bits (Pmn Output Data)**

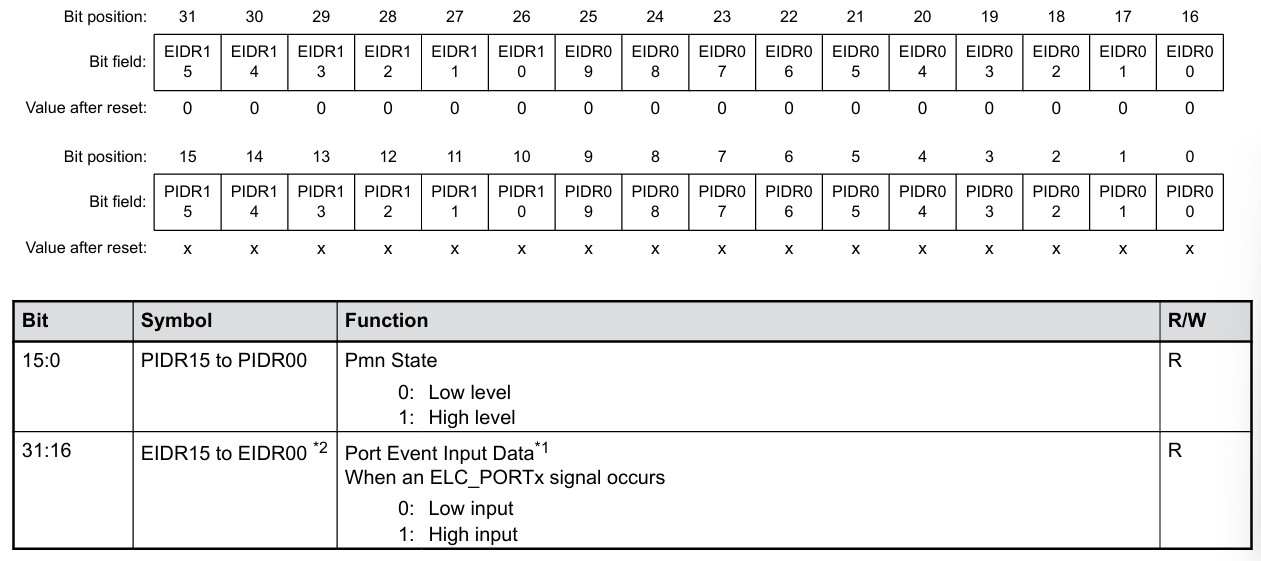
Ghi giá trị xuất ra chân OUT khi chân đang ở chế độ Output.

R\_PORTm->PODR\_b.PODRn (m,n: Thay thành Port, Pin tương ứng)

Ví dụ: R\_PORT1->PODR\_b.PODR5 = 1; // P105 = mức High

1. **Thanh ghi điều khiển port 2 (Port Control Register)**

Base address: PORTm = 0x4008\_0000 + 0x0020 × m (m = 0 to 9, A, B)

Offset address: 0x004 (PCNTR2/EIDR) 0x006 (PIDR)

1. **PIDRn bits (Pmn State)**

Đọc trạng thái hiện tại của chân input.

R\_PORTm->PIDR\_b.PODRn

PIDRn = 0 → chân đang ở mức 0

PIDRn = 1 → chân đang ở mức 1

Ví dụ: uint32\_t val = R\_PORT1->PIDR\_b.PIDR3;

1. **EIDRn bits (Port Event Input Data)**

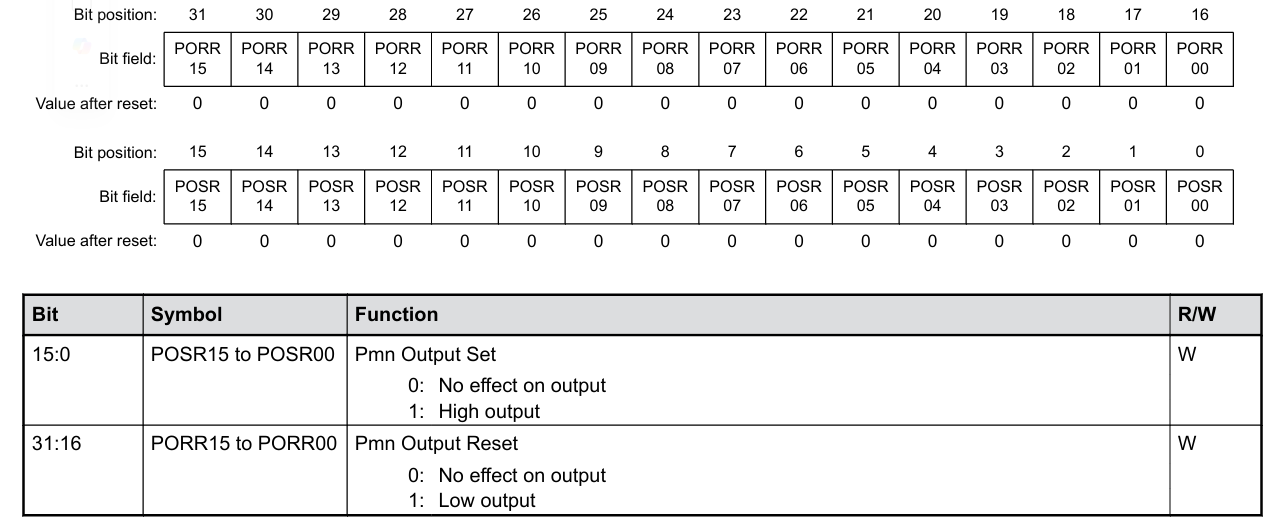
Đây là thanh ghi dùng để đọc trạng thái của tín hiệu sự kiện (Event Input) từ các chân I/O khi chân được cấu hình làm nguồn sự kiện EL (Event Link Controller).

EIDR **không phải là input thông thường** (PIDR mới là input bình thường để đọc trạng thái chân).

EIDR dùng cho **Event Link Controller** khi ta cấu hình chân I/O làm nguồn sự kiện cho: Timer (GPT), Capture unit, ICU (Interrupt), Peripheral trigger…

### Thanh ghi điều khiển port 3 (Port Control Register)

Base address: PORTm = 0x4008\_0000 + 0x0020 × m (m = 0 to 9, A, B)

Offset address: 0x008 (PCNTR3/PORR) 0x00A (POSR)

1. **POSRn bits (Pmn Output Set)**

POSRn là thanh ghi chỉ ghi (write-only) dùng để set (đặt lên 1) các bit OUT của các chân I/O mà không làm ảnh hưởng tới các bit khác trong cùng port.

R\_PORTm->POSR\_b.POSRn (m,n: Thay thành Port, Pin tương ứng)

Ví dụ: R\_PORT1->POSR\_b.POSR5 = 1; // Set output HIGH tại P105

1. **PORRn bits (Pmn Output Reset)**

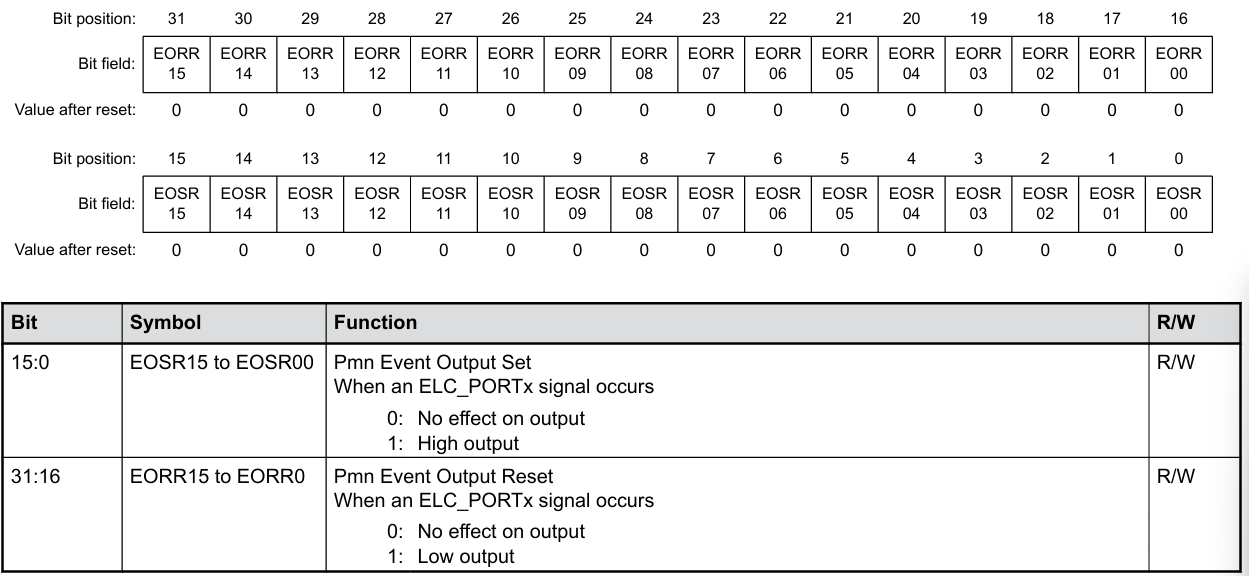
PORRn là thanh ghi chỉ ghi (write-only) dùng để Đặt (reset) mức Output của chân I/O về LOW (0) mà không làm thay đổi các chân khác trong cùng port.

R\_PORTm->PORR\_b.PORRn (m,n: Thay thành Port, Pin tương ứng)

Ví dụ: R\_PORT1->PORR\_b.PORR5 = 1; // đặt P105 về LOW.

1. **Thanh ghi điều khiển port 4 (Port Control Register)**

Base address: PORTm = 0x4008\_0000 + 0x0020 × m (m = 1 to 4)

Offset address: 0x00C (PCNTR4/EORR), 0x00E (EOSR)

1. **EOSRn bits (Pmn Event Output Set)**

EOSR là thanh ghi chỉ ghi (write-only) dùng để set (kích hoạt) tín hiệu Event Output ra chân I/O. Chân này phải được cấu hình trước trong PFS (Pin Function Select) để chạy chế độ Event Output (thuộc Event Link Controller – ELC).

EOSRn = 1 → chân Px n phát tín hiệu event output

EOSRn = 0 → không thay đổi

R\_PORTm->EOSR\_b.EOSRn

1. **EORRn bits (Pmn Event Output Reset)**

EORR là thanh ghichỉ ghi (write-only)dùng đểReset (tắt / xóa) tín hiệu Event Output tại chân Px n.Nó hoạt động ngược lại so với EOSR (Event Output Set).

EORRn = 1 → tắt / reset xung Event Output

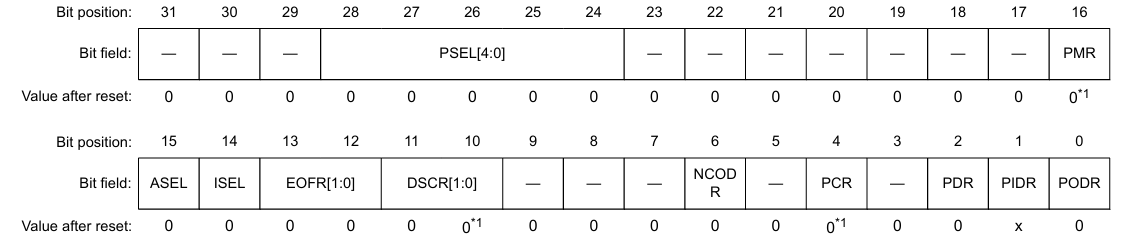
EORRn = 0 → không làm gì

R\_PORT1->EORR\_b.EORR5 = 1;

1. **PmnPFS/PmnPFS\_HA/PmnPFS\_BY : Port mn Pin Function Select Register (m = 0 to 9, A, B, n = 00 to 15)**

PmnPFS là thanh ghi cấu hình chức năng của từng chân I/O trên RA6M5. Mỗi chân (ví dụ P105) có một thanh ghi PFS riêng, quy định chân đó sẽ hoạt động như: GPIO input / output, Peripheral function (I2C, SPI, SCI, CAN, ADC trigger, GPT I/O…), Event link input/output và nhiều chế độ đặc biệt khác...

Base address: PFS = 0x4008\_0800

Offset address: 0x000 + 0x040 × m + 0x004 × n (PmnPFS, Chuẩn – dùng cho hầu hết chân); 0x002 + 0x040 × m + 0x004 × n (PmnPFS\_HA, High Accuracy – cho chân có cần độ chính xác cao); 0x003 + 0x040 × m + 0x004 × n (PmnPFS\_BY, Byte-access – cho chân hỗ trợ truy cập theo byte).

1. **PODR bit (Port Output Data), PIDR bit (Port State), PDR bit (Port Direction)**

Các bit PDR, PIDR và ​​PODR có chức năng tương tự như PCNTR. Khi các bit này được đọc, giá trị PCNTR được đọc.

PDR: Pmn Direction 🡪 Cấu hình input/output {R\_PORTm->PDR\_b.PDRn}

0: Input

1: Output

PIDR: Pmn Input Data 🡪 Đọc data {R\_PORTm->PIDR\_b.PODRn}

PODR: Pmn Output Data 🡪 Xuất data {R\_PORTm->PODR\_b.PODRn}

1. **PCR bit (Pull-up Control)**

Mỗi pin có 1 bit PCR dùng để bật/tắt điện trở kéo lên

{R\_PFS->PORT[m].PIN[n].PmnPFS\_b.PCR}

0: Tắt input pull-up

1: Bật input pull-up

1. **NCODR bit (N-Channel Open-Drain Control)**

Bit NCODR chỉ định loại đầu ra cho các chân cổng. Các bit liên quan đến các chân không tồn tại được dành riêng. Các bit được dành riêng được đọc là 0. Giá trị ghi phải là 0.

Chọn chế độ open-drain cho từng chân:

* Output chỉ có thể kéo LOW
* Khi cần HIGH → chân thả nổi (Hi-Z), phải dùng external pull-up
* Dùng trong:

+ I²C bus (SCL, SDA)

+ Giao tiếp multi-master

+ Giao tiếp mức điện áp khác

1. **DSCR[1:0] bits (Port Drive Capability)**

Điều chỉnh độ mạnh dòng lái (drive strength) của chân I/O.

Dòng lớn khi: điều khiển LED công suất ≥ 20 mA, điều khiển tín hiệu tốc độ cao

Dòng nhỏ khi: tiết kiệm năng lượng, giảm nhiễu EMI

|  |  |
| --- | --- |
| **DSCR1:0** | **drive strength** |
| 00 | Low drive (yếu nhất) |
| 01 | Middle drive |
| 10 | High drive |
| 11 | Maximum drive (mạnh nhất) |

1. **EOFR[1:0] bits (Event on Falling/Event on Rising)**

Điều khiển Event Link trigger theo sườn. Dùng khi chân cấu hình làm event input cho ELC. Chọn ngắt tại cạnh nào của chân {R\_PFS->PORT[m].PIN[n]. PmnPFS\_b.EOFR}

|  |  |
| --- | --- |
| **EOFR[1:0]** | **Function** |
| 00 | Không tạo event |
| 01 | Cạnh lên |
| 10 | Cạnh xuống |
| 11 | Event tại cả 2 cạnh |

1. **ISEL bit (IRQ Input Enable)**

Bit ISEL chỉ định các chân đầu vào IRQ. Thiết lập này có thể được sử dụng kết hợp với các chức năng ngoại vi, mặc dù một IRQn (ngắt chân ngoài) cùng số chỉ được bật cho một chân. Bit ISEL cho một IRQn không xác định được dành riêng. Để cho phép ngắt cần set bit này lên 1.

1. **ASEL bit (Analog Input Enable)**

Bit này dùng để chọn chức năng analog (ADC, ACMP). Khi ASEL = 1: Chân được kết nối tới bộ ADC/analog, PMR bị bỏ qua, các tính năng digital bị vô hiệu hóa (PDR/PODR…)

1. **PMR bit (Port Mode Control)**

Chọn chân chạy ở GPIO mode (PMR=0) hay Peripheral mode (PMR=1).

Điều kiện kích hoạt peripheral: PSEL chọn hàm trước → xong sau đó PMR = 1. Nếu PMR bật trước: không có tác dụng.

1. **PSEL[4:0] bits (Peripheral Select)**

Chọn chức năng ngoại vi cho chân. Có 32 giá trị (tối đa), mỗi giá trị map tới một peripheral. PSEL là trái tim của PFS, quyết định chức năng chân.

1. **PWPR : Write-Protect Register**

PWPR (Port Write Protect Register) là thanh ghi dùng để bật/tắt cơ chế bảo vệ ghi đối với toàn bộ các thanh ghi PmnPFS (Pin Function Select Register).

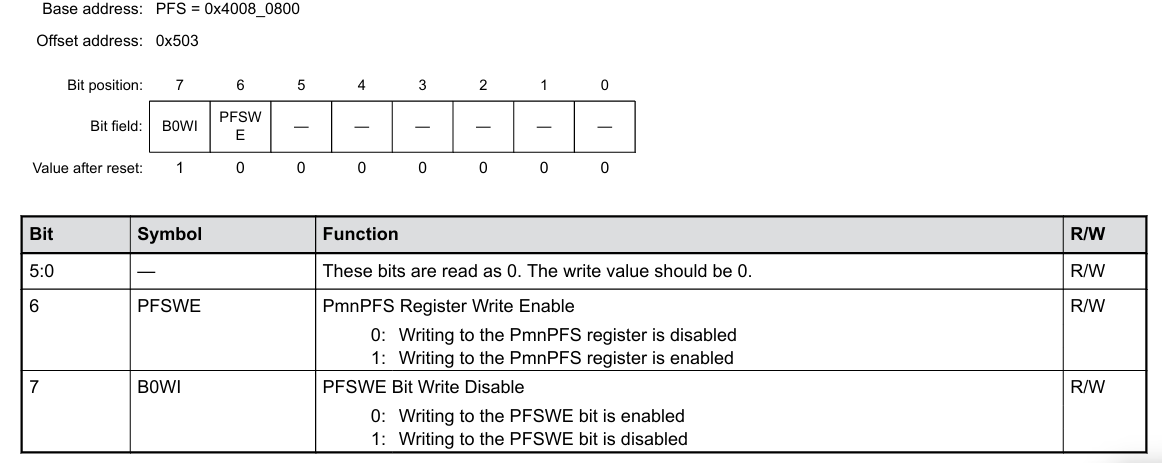
PFSWE (bit 7) – PFS Write Enable: Đây là bit bắt buộc phải bật lên 1 trước khi cấu hình pin.

1 → Cho phép ghi vào tất cả thanh ghi PFS

0 → Cấm ghi vào PFS (Protect mode)

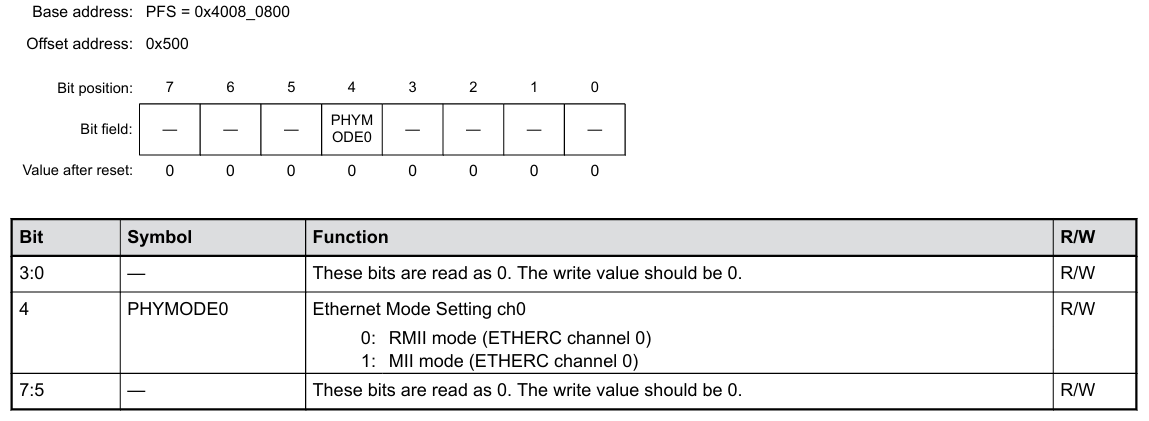
BOWI (bit 0) – Write Protection Disable: Nếu BOWI = 1 → bạn không thể kích hoạt chế độ ghi PFS.

0 → Cho phép thay đổi PFSWE (nghĩa là có thể mở khóa)

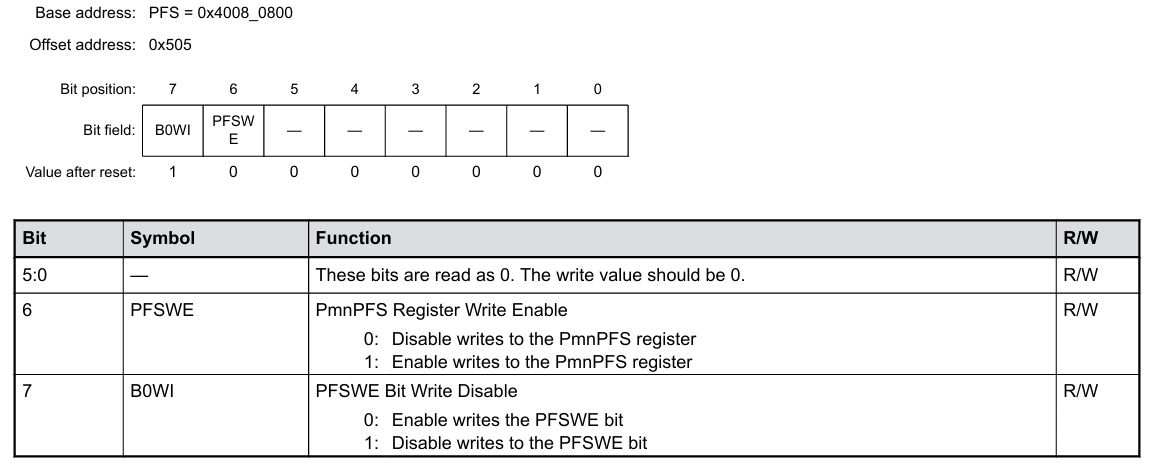
1 → Khóa cứng, không thể bật PFSWE → không thể ghi PFS

1. **PFENET : Ethernet Control Register**

PFENET dùng để cấu hình chức năng Ethernet lên các chân port. RA6M5 hỗ trợ Ethernet MAC + PHY RMII, và PFENET cho phép bật/tắt chức năng này. Khi PFENET.ENETEN = 1, các chân liên quan (PA0–PA7, PB0–PB3…) sẽ tự chuyển sang peripheral Ethernet.

Bit này quyết định RA6M5 sẽ dùng chuẩn RMII hay MII để giao tiếp với PHY Ethernet ngoài.

1. **PWPRS : Write-Protect Register for Secure**

PWPR là thanh ghi bảo vệ ghi (write-protection) của khối PFS (Pin Function Select)**.** PFS rất quan trọng (chọn chức năng chân: GPIO / Peripheral / IRQ / Analog…). Vì vậy RA6M5 không cho ghi PFS trực tiếp, phải mở khóa qua PWPR.

Để bật cho phép ghi cần làm:

\_B1; Clear bit B0WI

{R\_PMISC->PWPR\_b.B0WI = 0}

\_B2; Set bit PFSWE

{R\_PMISC->PWPR\_b.PFSWE = 1}

## CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA I/O PORT

1. **Chế độ GPIO (General Purpose Input/Output)**

Đây là chế độ mặc định của tất cả các chân gồm cấu hình port, thiết lập dữ liệu đầu ra và đọc dữ liệu đầu vào. Ứng dụng: nút nhấn, LED, tín hiệu logic đơn giản.

GPIO Output Mode: Pin được cấu hình để xuất tín hiệu số. Các thanh ghi điều khiển chính:

+ PDR = 1 → chọn output

+ PODR → ghi mức HIGH/LOW

+ POSR/PORR → set/reset atomic

GPIO Input Mode: Pin được cấu hình để đọc mức logic từ ngoài vào. Thanh ghi chính:

+ PDR = 0 → chọn input

+ PIDR → đọc trạng thái pin

+ PCR → enable pull-up

1. **Chế độ Peripheral Function (chức năng ngoại vi)**

Chân I/O được chuyển từ GPIO sang chức năng ngoại vi: UART (TX/RX), I2C (SDA/SCL), SPI (MOSI/MISO/SCK/CS), PWM (GPT), ADC input, CAN, Ethernet RMII/MII, IRQ input, USB, ...

Để kích hoạt chế độ này thì phải thiết lập các thanh ghi sau:

+ PMR = 1 → chuyển từ GPIO sang Peripheral

+ PSEL[4:0] → chọn chức năng cụ thể

+ ISEL / ASEL → tùy loại tín hiệu

Tất cả điều chỉnh trong PmnPFS.

1. **Chế độ Analog (ADC input hoặc comparator)**

Dùng cho chân kết nối ADC hoặc analog comparator. Khi vào chế độ analog thì: ASEL = 1, Vô hiệu hóa digital input/output, Pin dùng đường analog nội bộ → không bị ảnh hưởng bởi logic số.

Chỉ một số chân có thể dùng analog.

1. **Chế độ Event Link Controller (ELC)**

Chế độ này cho phép các ngoại vi truyền sự kiện cho nhau không cần CPU, giảm độ trễ. Ứng dụng: kích timer, kích ADC, kích DMA.

Chân I/O có thể làm:

* Event Input

+ Pin đọc mức event từ ngoài → đi vào ELC.

+ Thanh ghi liên quan: EIDR (Event Input Data), ISEL trong PmnPFS để bật IRQ/ELC input

* Event Output

+ Pin phát ra tín hiệu sự kiện (trigger) cho thiết bị khác.

+ Thanh ghi:

* + EOSR → set event output
  + EORR → reset event output
  + EOFR[1:0] → chọn cạnh Rising/Falling

1. **Chế độ Open-Drain Output**

Dùng cho bus I²C hoặc giao tiếp cần dòng kéo xuống.

+ NCODR = 1 → bật N-Channel open-drain

+ Tín hiệu chỉ có thể kéo LOW, không thể đẩy HIGH.

+ HIGH nhờ điện trở pull-up ngoài hoặc PCR (pull-up nội).

1. **Chế độ Drive Strength (sức kéo dòng)**

Điều khiển mức dòng output của chân:

DSCR[1:0]

00 = Low drive

01 = Medium drive

10 = High drive

11 = Ultra-high drive (tùy pin)

Dùng khi: Điều khiển LED công suất, giao tiếp tốc độ cao, Ethernet RMII/MII, SPI tốc độ cao.

1. **Chế độ Secure/Non-Secure (TrustZone)**

Một số chân port có thể được cấu hình: Secure hoặc Non-secure qua các thanh ghi:

+ PWPR → mở khóa / khóa ghi PFS

+ PWPRS → điều khiển quyền truy cập secure

Ứng dụng trong bảo mật phần mềm.

1. **Chế độ Ethernet (RMII/MII)**

Các chân port sang mode Ethernet khi:

+ PFENET = 1

+ PHYMODE0 cấu hình MII/RMII

+ PSEL chuyển sang chức năng Ethernet

Chế độ này override toàn bộ cấu hình GPIO.

1. **Chế độ Double-edge Detection (EVENT ON RISING/FALLING)**

Dùng cho chân event hoặc interrupt: EOFR[1:0]

00 = No event

01 = Event on Rising

10 = Event on Falling

11 = Event on Both edges

## Chương trình ví dụ về ngoại vi

**Ví dụ: Nhấn nút để chuyển trạng thái 3 LED xanh lá, đỏ, xanh dương**

**Cách 1: Lập trình thanh ghi**

void delay(void)

{

for (volatile int i = 0; i < 1000000; i++); // Tạo trễ

}

void hal\_entry(void) {

// Cấu hình LED làm Output

R\_PORT6->PDR\_b.PDR10 = 1; // LED Đỏ (P610)

R\_PORT6->PDR\_b.PDR9 = 1; // LED Xanh lá (P609)

R\_PORT6->PDR\_b.PDR1 = 1; // LED Xanh dương (P601)

// Cấu hình nút nhấn SW2 làm Input (P804)

R\_PORT8->PDR\_b.PDR4 = 0;

R\_PMISC->PWPR\_b.B0WI = 0;

R\_PMISC->PWPR\_b.PFSWE = 1;

R\_PFS->PORT[8].PIN[4].PmnPFS\_b.PCR = 1; // bật pull-up

R\_PMISC->PWPR\_b.PFSWE = 0;

R\_PMISC->PWPR\_b.B0WI = 1;

int state = 0; // biến lưu trạng thái LED hiện tại

R\_PORT6->PODR\_b.PODR10 = 0;

R\_PORT6->PODR\_b.PODR9 = 1;

R\_PORT6->PODR\_b.PODR1 = 0;

while (1)

{

if (R\_PORT8->PIDR\_b.PIDR4 == 0) // nếu nhấn nút

{

delay(); // chống dội

if (R\_PORT8->PIDR\_b.PIDR4 == 0)

{

state++;

if (state > 2) state = 0;

// --- Tắt hết LED ---

R\_PORT6->PODR\_b.PODR9 = 0;

R\_PORT6->PODR\_b.PODR10 = 0;

R\_PORT6->PODR\_b.PODR1 = 0;

// --- Bật LED theo state ---

switch (state)

{

case 0: R\_PORT6->PODR\_b.PODR9 = 1; break; // Xanh lá

case 1: R\_PORT6->PODR\_b.PODR10 = 1; break; // Đỏ

case 2: R\_PORT6->PODR\_b.PODR1 = 1; break; // Xanh dương

}

// --- Chờ thả nút ---

while (R\_PORT8->PIDR\_b.PIDR4 == 0);

delay();

}

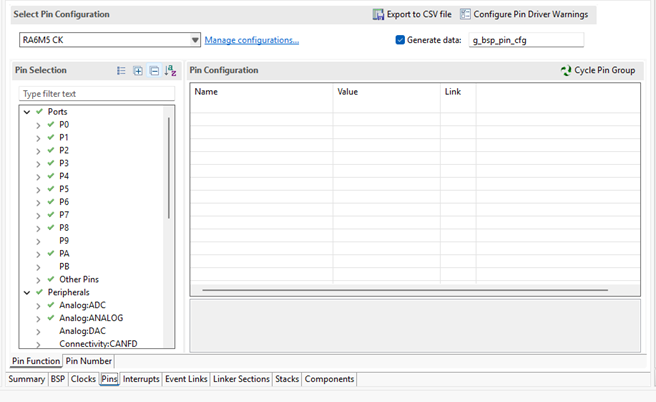
}

}

}

**Cách 2: sử dụng FSP**

Bước 1: Sau khi tạo xong project, Nhấn vào để cấu hình FSP

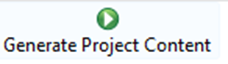
Bước 2: Để cấu hình I/O, chọn Pins, giao diện như sau:**2434002**

Bước 3: Do LED xanh lá được nối đến chân P609 nên ta cần cấu hình output cho chân này.

> Nhấn chọn P6 => P609

> Mode chọn Output mode (…)

Tương tự ta có LED xanh dương là P601 và LED đỏ là P610, mode chọn tương tự LED xanh lá



Sau đó CTRL + S để save và nhấn

Bước 4: Nhập code

#include "hal\_data.h"

void delay\_ms(int ms)

{

R\_BSP\_SoftwareDelay(ms, BSP\_DELAY\_UNITS\_MILLISECONDS);

}

int main(void){}

void hal\_entry(void)

{

int state = 0;

int last = 1; // nút thả (pull-up → đọc là 1)

while (1)

{

int now = R\_PORT1->PIDR\_b.PIDR10; // đọP205110 (SW1)

// phát hiện nhấn (1 → 0)

if (last == 1 && now == 0)

{

delay\_ms(20); // debounce

if (R\_PORT1->PIDR\_b.PIDR10 == 0) // kiểm tra lại

{

state++;

if (state > 2) state = 0;

}

}

last = now;

// Cập nhật LED theo trạng thái

// (Active-low → 0 = sáng)

switch (state)

{

case 0:

R\_PORT6->PODR\_b.PODR9 = 0; // LED green ON

R\_PORT6->PODR\_b.PODR10 = 1;

R\_PORT6->PODR\_b.PODR1 = 1;

break;

case 1:

R\_PORT6->PODR\_b.PODR9 = 1;

R\_PORT6->PODR\_b.PODR10 = 0; // LED red ON

R\_PORT6->PODR\_b.PODR1 = 1;

break;

case 2:

R\_PORT6->PODR\_b.PODR9 = 1;

R\_PORT6->PODR\_b.PODR10 = 1;

R\_PORT6->PODR\_b.PODR1 = 0; // LED blue ON

break;

}

}

}

# Phần B: GENERAL PWM TIMER (GPT)

## 1. Giới thiệu tổng quan về ngoại vi

GPT (General PWM Timer) là một ngoại vi bộ định thời đa năng và linh hoạt, đóng vai trò quan trọng trong việc tạo ra các dạng sóng điều chế độ rộng xung (PWM) phức tạp và thực hiện các ứng dụng đo lường. GPT là một hệ thống đa kênh, bao gồm bốn kênh bộ định thời 32-bit (GPT32n, với n=0 đến 3) và sáu kênh bộ định thời 16-bit (GPT16m, với m=4 đến 9), tổng cộng có 10 kênh hoạt động độc lập.

1. **Cấu trúc cơ bản và Nguồn Clock:**

Chức năng cơ bản của GPT là đếm định kỳ, được kiểm soát bởi thanh ghi GTCNT và thanh ghi chu kỳ GTPR. GPT hỗ trợ ba chế độ đếm chính: đếm lên (up-counting) hoặc đếm xuống (down-counting) để tạo ra dạng sóng răng cưa (saw waves), và đếm lên/xuống (up/down-counting) để tạo ra dạng sóng tam giác (triangle waves).

Nguồn xung clock cho bộ đếm GTCNT có thể được lựa chọn một cách độc lập cho từng kênh. Các nguồn này bao gồm xung clock hệ thống PCLKD được chia với nhiều tỷ lệ (từ PCLKD/1 đến PCLKD/1024). Ngoài ra, GPT còn có thể sử dụng tín hiệu kích hoạt từ bên ngoài, được cấp thông qua các chân GTETRGA, GTETRGB, GTETRGC, và GTETRGD.

Mỗi kênh GPT được trang bị hai chân đầu vào/đầu ra (I/O pins): GTIOCnA và GTIOCnB. Các chân này có thể được cấu hình cho cả việc xuất tín hiệu so sánh (output compare output) hoặc nhận tín hiệu bắt (input capture input).

1. **Hệ thống Thanh ghi và Buffer:**

GPT sở hữu một hệ thống thanh ghi so sánh/bắt tín hiệu phức tạp hơn các bộ định thời đơn giản. Mỗi kênh có hai thanh ghi so sánh/bắt tín hiệu chính là GTCCRA và GTCCRB. Ngoài ra, GPT còn cung cấp thêm bốn thanh ghi GTCCRC, GTCCRD, GTCCRE, và GTCCRF. Bốn thanh ghi này được thiết kế để hoạt động như các thanh ghi đệm (buffer registers) cho GTCCRA và GTCCRB, nhưng chúng cũng có thể hoạt động như các thanh ghi so sánh độc lập khi chức năng đệm không được sử dụng.

Trong hoạt động PWM, các thanh ghi đệm này cho phép chuyển đổi buffer tại các điểm đỉnh (crests) hoặc điểm đáy (troughs) của sóng. Khả năng này, đặc biệt trong chế độ sóng tam giác, cho phép tạo ra các dạng sóng PWM bất đối xứng ngang (laterally asymmetric PWM waveforms).

1. **Khả năng Kích hoạt và Đồng bộ:**

GPT được thiết kế để tích hợp sâu với các nguồn sự kiện phần cứng. Hoạt động đếm (GTCNT) có thể được khởi động (Start), dừng (Stop), hoặc xóa (Clear) đồng thời trên các kênh tùy ý, thông qua việc ghi vào các thanh ghi phần mềm chung như GTSTR, GTSTP, và GTCLR. Đồng thời, các thao tác này cũng có thể được kích hoạt bởi các nguồn phần cứng, bao gồm tới tám sự kiện từ ELC (Event Link Controller), trạng thái của hai chân đầu vào, hoặc tối đa bốn tín hiệu kích hoạt ngoài (external triggers) (GTETRGn).

GPT cũng hỗ trợ Chức năng Đếm Sự kiện (Event Count Operation), trong đó bộ đếm GTCNT tăng hoặc giảm dựa trên các sự kiện bên ngoài được lựa chọn trong các thanh ghi GTUPSR (đếm lên) và GTDNSR (đếm xuống), thay vì sử dụng xung clock nội được chọn trong GTCR.TPCS.

1. **Tính năng Bảo vệ và Điều khiển Đầu ra (POEG):**

Một tính năng nổi bật của GPT là cơ chế bảo vệ và điều khiển đầu ra tích hợp với module Port Output Enable for GPT (POEG). POEG cho phép vô hiệu hóa các chân đầu ra GPT (GTIOCnA/GTIOCnB) trong các trường hợp lỗi hoặc cần bảo vệ hệ thống.

Việc vô hiệu hóa đầu ra có thể xảy ra theo bốn cách:

1. Phát hiện mức tín hiệu đầu vào: POEG có thể vô hiệu hóa đầu ra khi phát hiện mức tín hiệu đầu vào trên các chân GTETRGn.
2. Yêu cầu vô hiệu hóa từ GPT: GPT tự tạo ra yêu cầu vô hiệu hóa nếu phát hiện cả chân GTIOCxA và GTIOCxB được điều khiển đến mức active (cao hoặc thấp) cùng lúc.
3. Phát hiện dừng dao động: Các chân GPT có thể bị vô hiệu hóa khi mạch tạo clock phát hiện sự dừng dao động.
4. Cài đặt Thanh ghi: Vô hiệu hóa trực tiếp bằng cách ghi vào thanh ghi phần mềm POEG (POEGGn.SSF).

POEG cũng cung cấp các tính năng lọc nhiễu (noise filtering) cho tín hiệu đầu vào GTETRGn và có thể cấu hình để tạo ra ngắt khi phát hiện các yêu cầu vô hiệu hóa đầu ra.

GPT có khả năng tự động tạo thời gian chết (dead time) trong các chế độ PWM (trừ Saw-wave PWM Mode) bằng cách sử dụng thanh ghi GTDTCR và GTDVU. Chức năng này tự động thiết lập giá trị so sánh cho dạng sóng pha âm vào GTCCRB dựa trên giá trị pha dương (GTCCRA) và giá trị thời gian chết (GTDVU). GPT còn có chức năng đếm chu kỳ (Period count function), cho phép đếm số lượng chu kỳ PWM đã hoàn thành.

## 2. Mô tả các thanh ghi liên quan đến GPT

**A diagram of a computer circuit

AI-generated content may be incorrect.**

Hình : Sơ đồ khối mô tả POEG (Port Output Enable for GPT)

Địa chỉ cơ sở của GPT phụ thuộc vào kênh được sử dụng:

GPT32n (n = 0 đến 3): 0x4016\_9000 + 0x0100 × n

GPT16m (m = 4 đến 9): 0x4016\_9000 + 0x0100 × m

|  |  |  |  |
| --- | --- | --- | --- |
| Thanh ghi | Kích thước | Địa chỉ thanh ghi | Chức năng |
| GTCR (Control Register) | 32-bit | 0x2C | Điều khiển hoạt động của bộ đếm GTCNT. Chứa các bit CST (Count Start/Stop) và MD[2:0] (Mode Select) để chọn chế độ hoạt động PWM hoặc bộ định thời. Chứa TPCS[3:0] (Timer Prescaler Select) để chọn nguồn clock. |
| GTPR (Cycle Setting Register) | 16/32-bit | 0x64 | Đặt giá trị đếm tối đa (chu kỳ) của GTCNT. Đối với sóng răng cưa (saw waves), chu kỳ là (GTPR + 1). |
| GTCNT (Counter) | 16/32-bit | 0x48 | Thanh ghi bộ đếm đọc/ghi, chứa giá trị đếm hiện tại. Kích thước hiệu dụng là 32-bit (GPT32n) hoặc 16-bit (GPT16m). |
| GTCCRA (Compare Capture Register A) | 16/32-bit | 0x4C | Thanh ghi đa chức năng, được sử dụng cho cả Output Compare (so sánh) và Input Capture (bắt tín hiệu đầu vào). |
| GTCCRB (Compare Capture Register B) | 16/32-bit | 0x50 | Thanh ghi đa chức năng tương tự GTCCRA. |
| GTIOR (I/O Control Register) | 32-bit | 0x34 | Cấu hình chức năng cho các chân GTIOCnA và GTIOCnB. Chứa OAE/OBE (Output Enable) và OADF[1:0]/OBDF[1:0] để xác định giá trị đầu ra khi có yêu cầu vô hiệu hóa (Output Disable). |
| GTUDDTYC (Count Direction and Duty Setting Register) | 32-bit | 0x30 | Đặt hướng đếm (UD) (đếm lên/xuống) và thiết lập duty cycle 0% hoặc 100% (OADTY/OBDTY) cho đầu ra GTIOCnA/B. |
| GTST (Status Register) | 32-bit | 0x3C | Cho biết trạng thái của GPT. Chứa các cờ ngắt như TCFA/TCFB (Input Capture/Compare Match Flag), TCFPO (Overflow Flag), và TCFPU (Underflow Flag). |
| GTSSR (Start Source Select Register) | 32-bit | 0x10 | Đặt nguồn kích hoạt phần cứng để Bắt đầu (Start) bộ đếm GTCNT (ví dụ: thông qua GTETRGn hoặc sự kiện ELC). |
| GTPSR (Stop Source Select Register) | 32-bit | 0x14 | Đặt nguồn kích hoạt phần cứng để Dừng (Stop) bộ đếm GTCNT. |
| GTCSR (Clear Source Select Register) | 32-bit | 0x18 | Đặt nguồn kích hoạt phần cứng để Xóa (Clear) bộ đếm GTCNT. |
| GTDTCR (Dead Time Control Register) | 32-bit | 0x88 | Thanh ghi điều khiển chức năng tự động đặt thời gian chết (Automatic Dead Time Setting). Bit TDE (Negative-Phase Waveform Setting) cho phép sử dụng GTDVU để tự động đặt GTCCRB. |
| GTDVU (Dead Time Value Register U) | 16/32-bit | 0x8C | Thanh ghi chứa giá trị thời gian chết (dead time value) được sử dụng để tạo dạng sóng PWM có thời gian chết. |

## 3. Mô tả các chế độ hoạt động của GPT

1. **Các chế độ hoạt động chính của GPT được chọn thông qua các bit MD[2:0] trong thanh ghi GTCR:**

GPT sử dụng hai loại dạng sóng đếm chính: sóng răng cưa (Saw-wave) cho chu kỳ đếm một chiều và sóng tam giác (Triangle-wave) cho chu kỳ đếm đối xứng, cùng với các cơ chế chuyển giao buffer khác nhau:

**3.1.1. Saw-wave PWM Mode (MD[2:0] = 000b)**

Trong chế độ này, bộ đếm GTCNT chỉ đếm theo một hướng (nửa sóng), có thể là đếm lên hoặc đếm xuống, tùy thuộc vào cài đặt của bit UD trong thanh ghi GTUDDTYC. Chu kỳ đếm được xác định là giá trị GTPR cộng 1. Đầu ra PWM được tạo ra khi GTCNT khớp với các giá trị so sánh được đặt trong GTCCRA hoặc GTCCRB. Chế độ này hỗ trợ linh hoạt hoạt động Buffer đơn (Single Buffer) hoặc Buffer kép (Double Buffer), cho phép cập nhật các giá trị PWM mà không làm gián đoạn chu kỳ hiện tại.

A diagram of a computer program

AI-generated content may be incorrect.

Hình : Ví dụ về hoạt độngSaw-wave PWM Mode với đếm lên, hoạt động bộ đệm, mức cao tại thời điểm so khớp GTCCRA/GTCCRB, và mức thấp tại cuối chu kỳ

**3.1.2. Saw-wave One-Shot Pulse Mode (MD[2:0] = 001b)**

Chế độ này được sử dụng để tạo ra xung một lần (one-shot pulse) và có cấu trúc hoạt động buffer **cố định**. Cơ chế truyền dữ liệu buffer trong chế độ này phức tạp hơn so với PWM thông thường; ví dụ, dữ liệu từ thanh ghi GTCCRD được chuyển vào thanh ghi tạm thời A (Temporary Register A) tại thời điểm kết thúc chu kỳ (cycle end), và sau đó được chuyển sang GTCCRA khi có so khớp giá trị GTCCRA.

A diagram of a diagram

AI-generated content may be incorrect.

Hình : Ví dụ về Saw-wave One-Shot Pulse Mode với đếm lên, đầu ra mức thấp từ chân GTIOCnA và đầu ra mức cao từ chân GTIOCnB tại thời điểm bắt đầu đếm, đầu ra được đảo trạng thái tại thời điểm so khớp GTCCRA/GTCCRB, và đầu ra được giữ nguyên tại cuối chu kỳ.

**3.1.3. Triangle-wave PWM Modes**

Trong các chế độ này, bộ đếm GTCNT thực hiện hoạt động sóng tam giác (triangle-wave) (sóng đầy đủ).

**Triangle-wave PWM mode 1 (MD[2:0] = 100b)**: Bộ đệm 32-bit được chuyển giao (buffer transfer) tại điểm đáy (trough) của sóng, tức là khi GTCNT chuyển từ 0 lên 1.

A diagram of a diagram

AI-generated content may be incorrect. Hình: Ví dụ về hoạt động của triangle-wave PWM mode 1 với hoạt động bộ đệm, đầu ra mức thấp từ chân GTIOCnA và đầu ra mức cao từ chân GTIOCnB tại thời điểm bắt đầu đếm, đầu ra được đảo trạng thái tại thời điểm so khớp thanh ghi GTCCRA/GTCCRB, và đầu ra được giữ nguyên tại cuối chu kỳ.

**Triangle-wave PWM mode 2 (MD[2:0] = 101b):** Cho phép tạo dạng sóng PWM bất đối xứng ngang vì hoạt động chuyển giao buffer 32-bit xảy ra tại cả điểm đỉnh (crest) (khi GTCNT đạt GTPR và bắt đầu đếm xuống) và điểm đáy (trough)của sóng.

A diagram of a computer

AI-generated content may be incorrect.

Hình : Ví dụ về hoạt động của triangle-wave PWM mode 2 với hoạt động bộ đệm, đầu ra mức thấp từ chân GTIOCnA và đầu ra mức cao từ chân GTIOCnB tại thời điểm bắt đầu đếm, đầu ra được đảo trạng thái tại thời điểm so khớp GTCCRA/GTCCRB, và đầu ra được giữ nguyên tại cuối chu kỳ.

**Triangle-wave PWM mode 3 (MD[2:0] = 110b):** Bộ đệm 64-bit (sử dụng các thanh ghi đệm GTCCRC/D/E/F) được chuyển giao tại điểm đáy (trough). Hoạt động bộ đệm là cố định.

A diagram of a computer system

AI-generated content may be incorrect.

Hình: Ví dụ về hoạt động của triangle-wave PWM mode 3 với đầu ra mức thấp từ chân GTIOCnA và đầu ra mức cao từ chân GTIOCnB tại thời điểm bắt đầu đếm, đầu ra được đảo trạng thái tại thời điểm so khớp GTCCRA/GTCCRB, và đầu ra được giữ nguyên tại cuối chu kỳ.

1. **Các chức năng hoạt động khác:**

GPT cũng hỗ trợ các chức năng hoạt động dựa trên cấu hình thanh ghi khác:

**3.2.1. Input Capture Function**

Giá trị hiện tại của bộ đếm GTCNT được chuyển (capture) vào thanh ghi GTCCRA hoặc GTCCRB khi phát hiện một cạnh tín hiệu được chọn (tăng, giảm, hoặc cả hai) trên chân GTIOCnA/GTIOCnB hoặc tín hiệu kích hoạt ngoài GTETRGn (qua POEG) hoặc sự kiện ELC.

**3.2.2. Event Count Operation**

GPT có thể đếm các sự kiện bên ngoài thay vì sử dụng xung clock nội.

Nếu các bit trong GTUPSR (Up Count Source Select) hoặc GTDNSR (Down Count Source Select) được thiết lập, bộ đếm GTCNT sẽ tăng hoặc giảm dựa trên các nguồn phần cứng (ví dụ: chân GTIOCnA/GTIOCnB hoặc tín hiệu GTETRGn qua POEG, hoặc sự kiện ELC). Trong chế độ này, xung clock được chọn trong GTCR.TPCS và hướng đếm (GTUDDTYC.UD) sẽ bị bỏ qua.

**3.2.3. Phase Counting Function**

GPT có thể phát hiện sự khác biệt pha giữa các tín hiệu đầu vào trên hai chân GTIOCnA và GTIOCnB. Chức năng này cho phép GTCNT đếm lên hoặc đếm xuống để xác định vị trí hoặc hướng của một bộ mã hóa vòng quay (rotary encoder).

**3.2.4. Automatic Dead Time Setting**

Chức năng này được kích hoạt thông qua thanh ghi GTDTCR và sử dụng thanh ghi GTDVU để thiết lập giá trị thời gian chết.Khi được kích hoạt (GTDTCR.TDE = 1), GPT sẽ tự động tính toán và đặt giá trị so sánh cho dạng sóng pha âm vào thanh ghi GTCCRB (sử dụng giá trị từ GTCCRA và GTDVU), đảm bảo khoảng thời gian chết giữa cặp xung PWM dương và âm. Chức năng này có sẵn trong chế độ Saw-wave One-shot Pulse Mode và tất cả các chế độ Triangle PWM (1, 2, 3).

**3.2.5. GPT\_OPS**

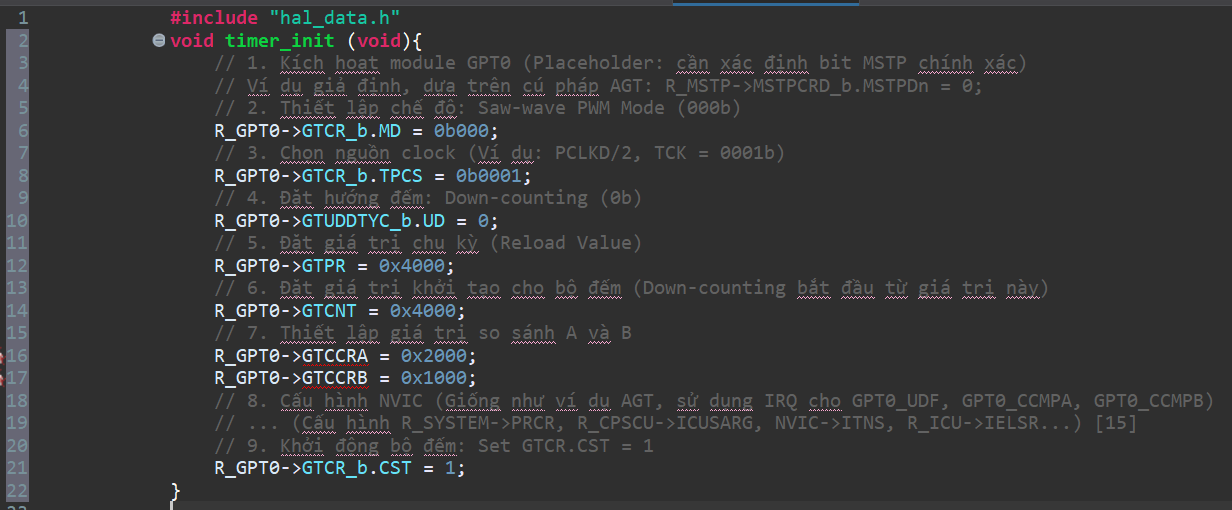
GPT\_OPS là một chức năng chuyên biệt, sử dụng kênh GPT320 để tạo ra dạng sóng PWM 6 pha (U/V/W dương/âm) cần thiết cho việc điều khiển động cơ DC không chổi than (BLDC motor control). GPT\_OPS cho phép chọn tín hiệu điều khiển đầu vào từ cảm biến Hall (GTIU/GTIV/GTIW) hoặc từ cài đặt phần mềm.

**3.2.6. Điều khiển Vô hiệu hóa Đầu ra (Output Disable Control)**

Sử dụng module POEG (Port Output Enable for GPT), các chân đầu ra của GPT (GTIOCnA/GTIOCnB) có thể bị vô hiệu hóa (Output Disable).Việc vô hiệu hóa có thể được kích hoạt bởi: phát hiện mức tín hiệu đầu vào từ chân GTETRGn, yêu cầu vô hiệu hóa từ GPT (ví dụ: khi cả hai chân GTIOCnA và GTIOCnB cùng xuất mức cao hoặc mức thấp), hoặc khi phát hiện dừng dao động của mạch tạo clock.

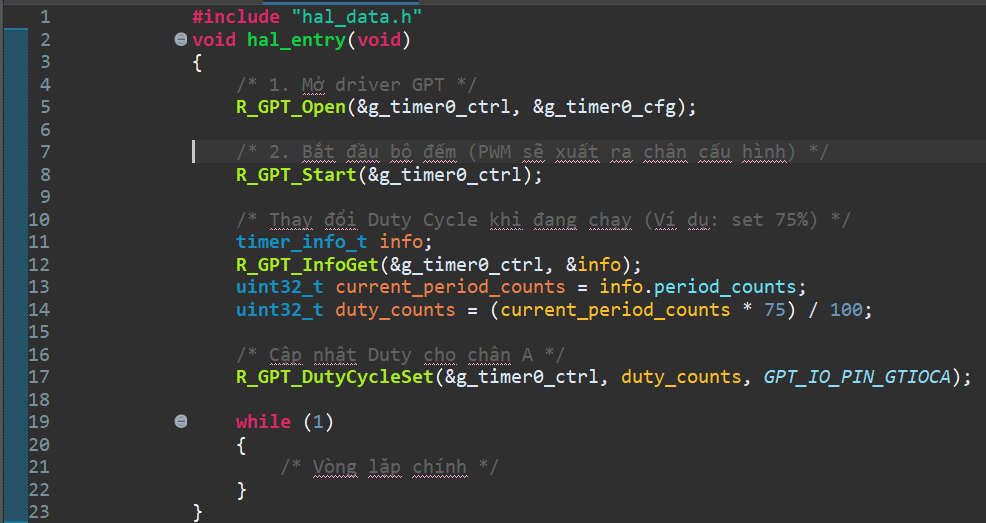
## 4. Các chương trình ví dụ về ngoại vi

Ví dụ 1: Lập trình Thanh ghi cho GPT - Chế độ Timer

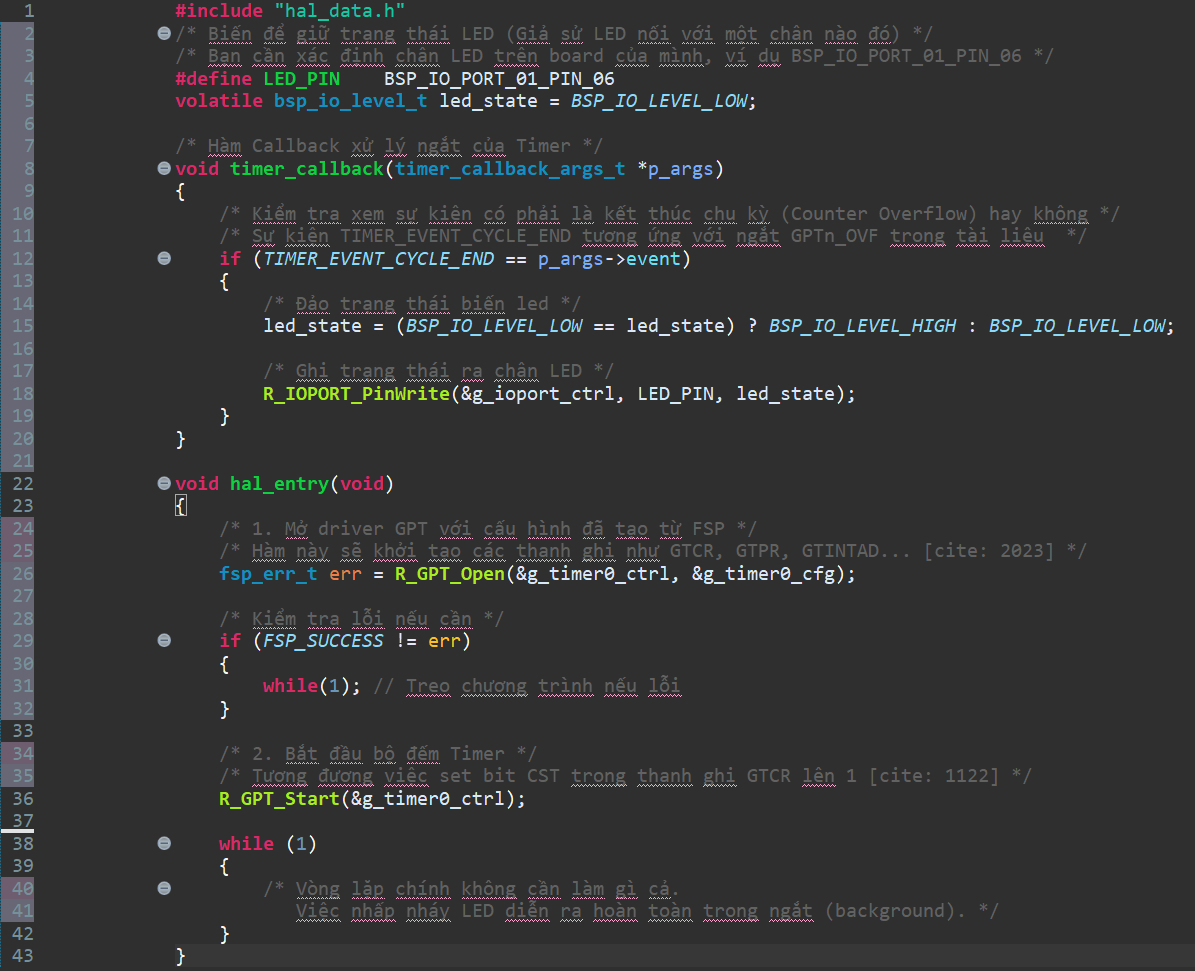


Ví dụ 2: Lập trình trên GPT bằng FSP

1. Cấu hình GPT kênh 0 (GPT320) để tạo xung PWM sóng răng cưa (Saw-wave)

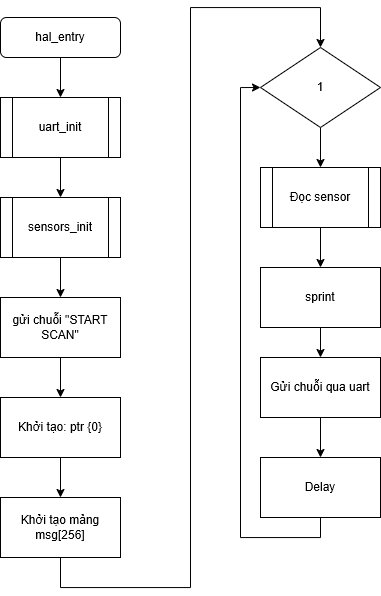


1. Sử dụng GPT làm Bộ định thời ngắt chu kỳ (Periodic Timer) để nhấp nháy LED.



# PHẦN C: GIẢI THUẬT LẬP TRÌNH CK RA6M5 ĐỌC CẢM BIẾN VÀ GỬI CHUỖI DỮ LIỆU QUA UART

## Lưu đồ giải thuật

****

uart\_init: khởi tạo các giá trị bạn đầu cho uart (chọn mode, cấu hình ngắt)

sensors\_init: khởi tạo các giá trị ban đầu: gỡ lỗi I2C, khởi tạo I2C, ICP10101, ZMOD4410.

Đọc sensor: đọc 2 giá trị của sensor.

# PHẦN D : THIẾT KẾ WEB SERVER HIỂN THỊ DỮ LIỆU ĐỌC TỪ PORT UART

## Lựa chọn framework:

Lựa chọn framework DJango lập trình bằng ngôn ngữ Python.

Django là một framework phát triển ứng dụng web mã nguồn mở, cấp cao, được viết bằng Python. Nó tuân theo mẫu kiến trúc Model-View-Controller (MVC), mặc dù Django thường mô tả kiến trúc của mình là Model-View-Template (MVT).

Có nhiều lý do khiến Django trở thành một lựa chọn phổ biến và mạnh mẽ cho việc phát triển web:

### + Tốc độ Phát triển Nhanh chóng (Rapid Development)

Django cung cấp rất nhiều tính năng "có sẵn" (built-in) giúp tăng tốc độ phát triển, bao gồm:

Hệ thống ORM (Object-Relational Mapper) mạnh mẽ: Cho phép tương tác với cơ sở dữ liệu bằng code Python thay vì SQL thô.

Giao diện quản trị viên tự động (Automatic Admin Interface): Django có thể tự động tạo giao diện quản trị viên chuyên nghiệp dựa trên các Model của bạn, giúp quản lý nội dung dễ dàng ngay từ đầu.

Hệ thống Template linh hoạt: Giúp tách biệt logic (View) và cách trình bày (Template).

* Khả năng Mở rộng (Scalability) và Tính Ổn định:

Django được thiết kế để xử lý lượng truy cập lớn và đã được sử dụng bởi các trang web có lưu lượng truy cập cao như Instagram, Pinterest, và The Washington Post. Cấu trúc mô-đun (modular) của nó giúp dễ dàng mở rộng theo chiều ngang (horizontal scaling).

Và còn nhiều lí do đặc biệt khác nữa. Vậy nên nhóm quyết định chọn Django như framework để phát triển server hiển thị dữ liệu mà nhóm đọc từ kit CK-RA6M5.

## Khởi tạo server:

2.1. Khởi tạo dự án:

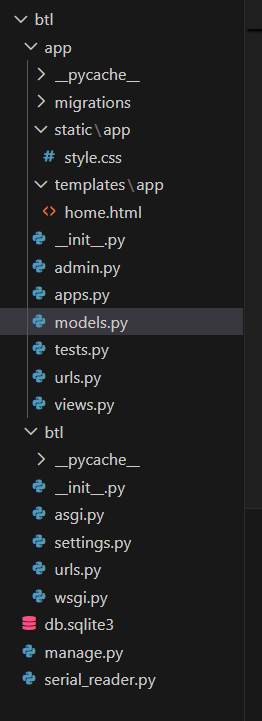
Thực hiện khởi tạo project file bằng lệnh:

>django-admin startproject btl

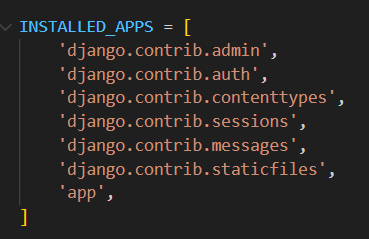
>cd btl

>python [manage.py](http://manage.py) startapp app

Sau khi khởi tạo các lệnh sau sẽ tạo ra các key và direction file theo chuẩn của Django:



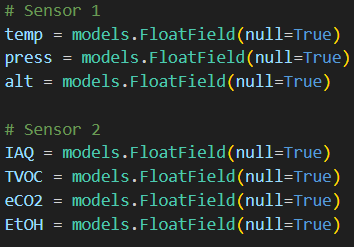
Sau khi tạo app thì chúng ta cần phải khai báo ‘app’ vào trong file [setting.py](http://setting.py):



2.2. Tạo database:

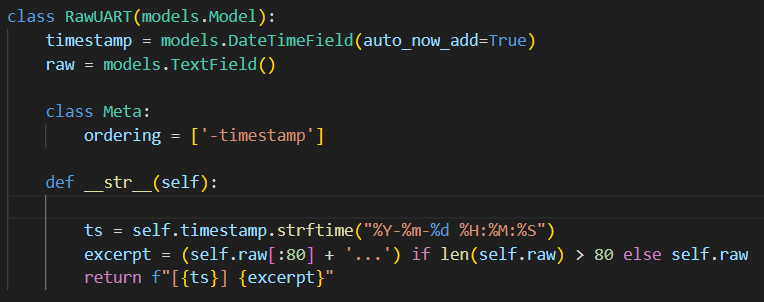
Để tạo database chúng ta cần khai báo các biến và loại của biến đó trong file [models.py](http://models.py):

2.2.1. Khai báo database cho sensors:



Gán type cho model là float để có thể lưu giá trị cho các biến dữ liệu sensors đọc được là số thực.

2.2.2. Khai báo database cho history logs:

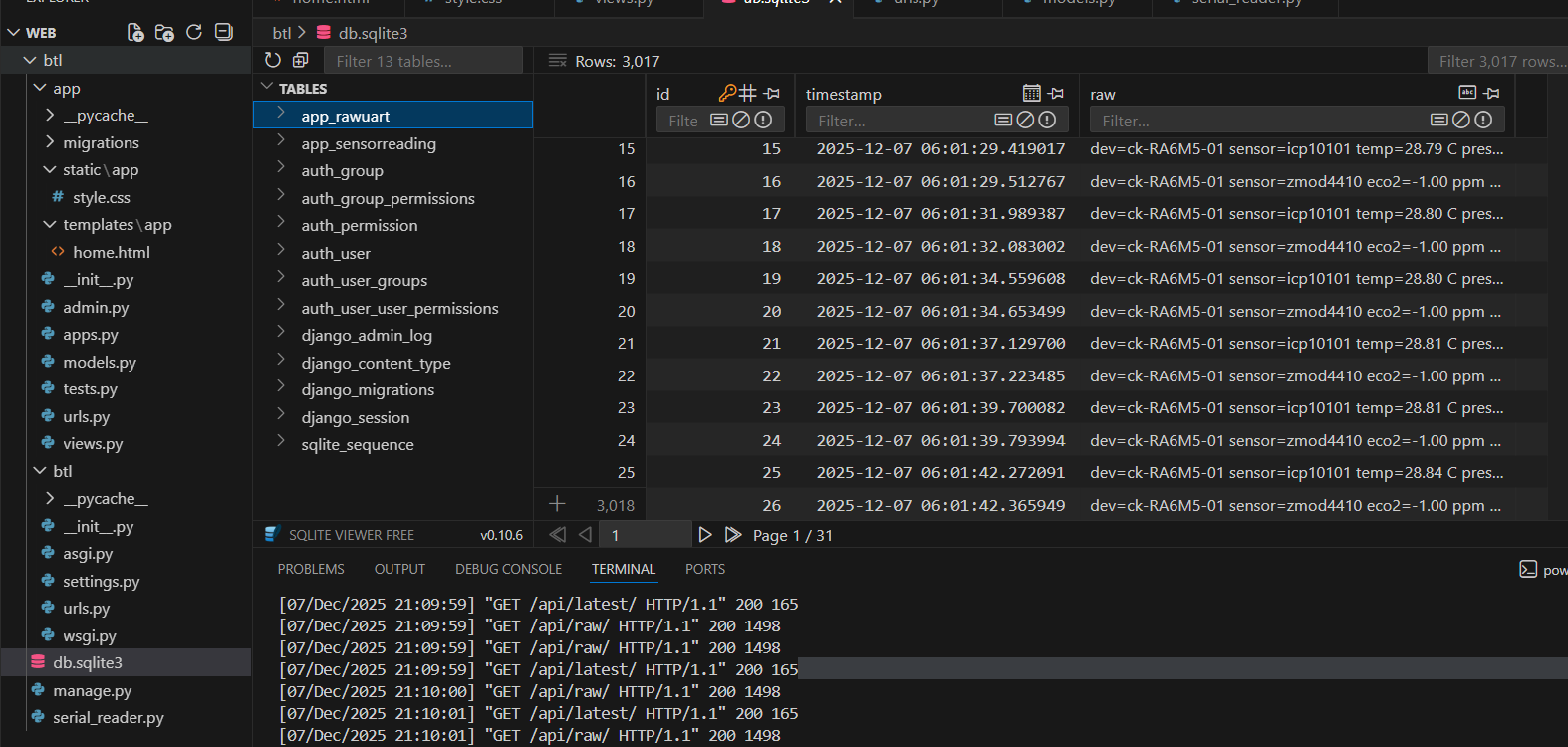


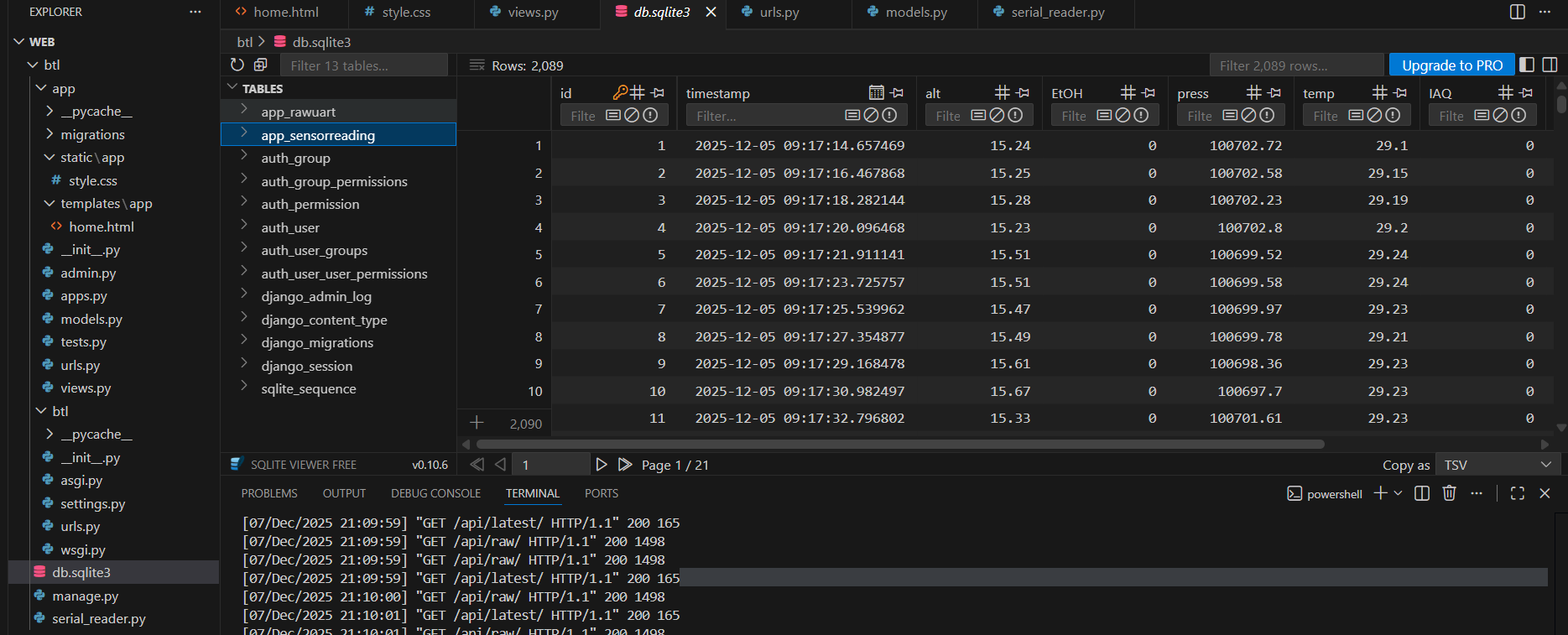
Để có thể lưu database cho history logs chúng ta cần 2 thứ:

Thứ nhất , khai báo model thời gian type Datetime để log hiển thị đúng thời gian thực khi đọc được cảm biến.

Thứ hai , khai báo model raw( dòng đọc từ port uart) type text để lưu vào database dưới dạng dòng text raw.

Sau đó , dữ liệu trong database sẽ được lưu trữ ở trong cơ sở dữ liệu cục bộ được lưu trong chính máy chủ server:

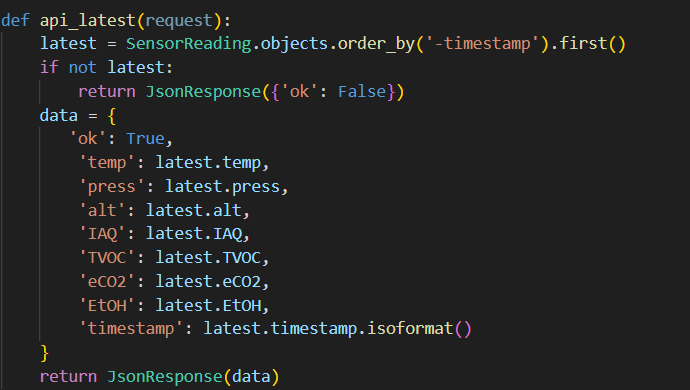


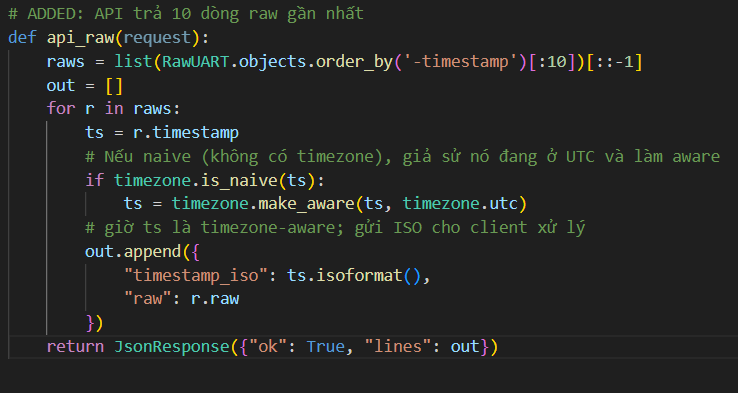


2.3. Thiết kế backend:

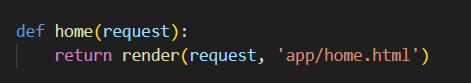
2.3.1. Khởi tạo file [views.py](http://views.py) và [urls.py](http://urls.py):

Để có thể lấy được dữ liệu từ database thì chúng ta cần gọi ra các api như api\_latest(api đọc dữ liệu sensor) và api\_raw(đọc text raw từ port uart) , và get dữ liệu được đóng gói dưới dạng json:

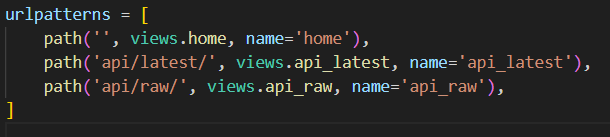




Còn lại là gọi ra hàm home để render template:



Và cấu hình các view api vào file [urls.py](http://urls.py):



Sau khi đã khai báo và cấu hình xong chúng ta cần chạy 2 lệnh

>python [manage.py](http://manage.py) makemigration

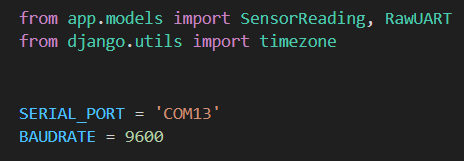
>python [manage.py](http://manage.py) migrate

Để hoàn chỉnh cập nhật cấu trúc dữ liệu ( database)

2.4. Khởi tạo app đọc dữ liệu từ port Uart:

File này phải được đặt cùng folder với file [manage.py](http://manage.py) vì khi chạy server chúng ta cần phải chạy cả 2 file cùng lúc để server có thể vừa đọc được dữ liệu từ port uart và vừa get dữ liệu để hiển thị lên template(frontend).

2.4.1. Khai báo port kết nối:



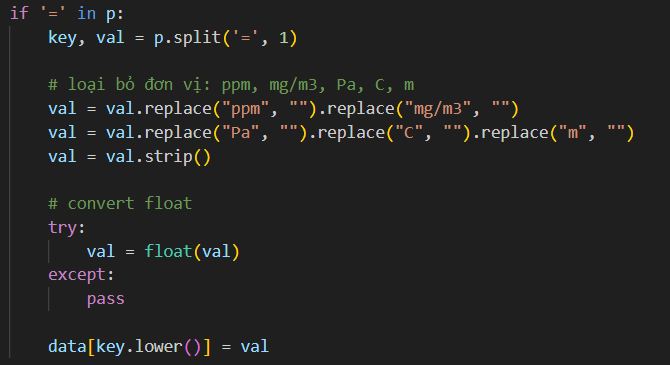
2.4.2. Tách dữ liệu:

Chúng ta đã đọc được dữ liệu UART log có dạng như sau:

dev=ck-RA6M5-01 sensor=zmod4410 eco2=-1.00 ppm etoh=-1.00 ppm iaq=-1.00 tvoc=-1.00 mg/m3

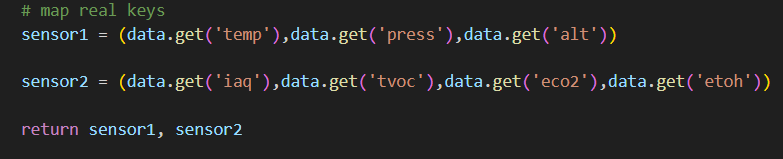
dev=ck-RA6M5-01 sensor=icp10101 temp=24.51 C press=101327.88 Pa alt=15.84 m

Và từ 2 dòng log như này , chúng ta bắt đầu tách dữ liệu từ các biến như temp , press , alt ,... và lưu dữ liệu dưới dạng float:

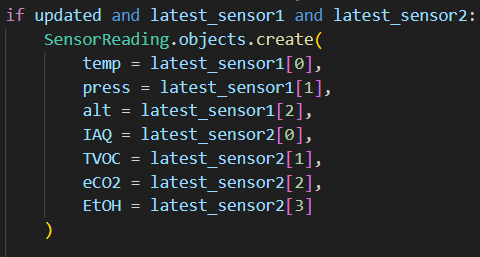


2.4.3. Đóng gói và lưu dữ liệu:

Và sau đó lấy dữ liệu lưu vào gói sensor1 và sensor2:



Cuối cùng là lấy dữ liệu lưu vào trong database đã khởi tạo là SensorReading:



Thông qua bước đọc này chúng ta đã có thể tách lấy được các giá trị cảm biến thu được từ kit CK-RA6M5 và lưu vào trong database để từ đó backend sẽ lấy dữ liệu và render lên frontend.

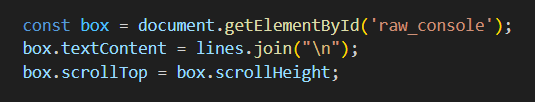
2.5. Thiết kế frontend:

2.5.1. Thiết kế template:

Tạo file template/app/home.html nhằm là nơi hiển thị dữ liệu từ backend.







Ta có thể thấy dữ liệu được lấy từ port uart nên sẽ không có khả năng streaming dữ liệu như cách kết nối wifi giao thức http,.... Nên chúng ta cần chèn 1 cái Javascript giúp reload trang mỗi 2s cho dashboard và 1s cho history logs nhằm khiến cho server hiển thị cập nhật liên tục các giá trị đọc được từ port Uart:



Và như vậy thì browser của chúng ta đã có thể hiển thị được dữ liệu được cập nhật liên tục rồi.

1. Vận hành:

Để có thể chạy server đọc được dữ liệu từ kit thông qua port UART thì chúng ta cần phải chạy cả 2 script cùng 1 lúc đó là script đọc uart và chạy server:

Tạo 2 cái terminal trong VS CODE STUDIO:

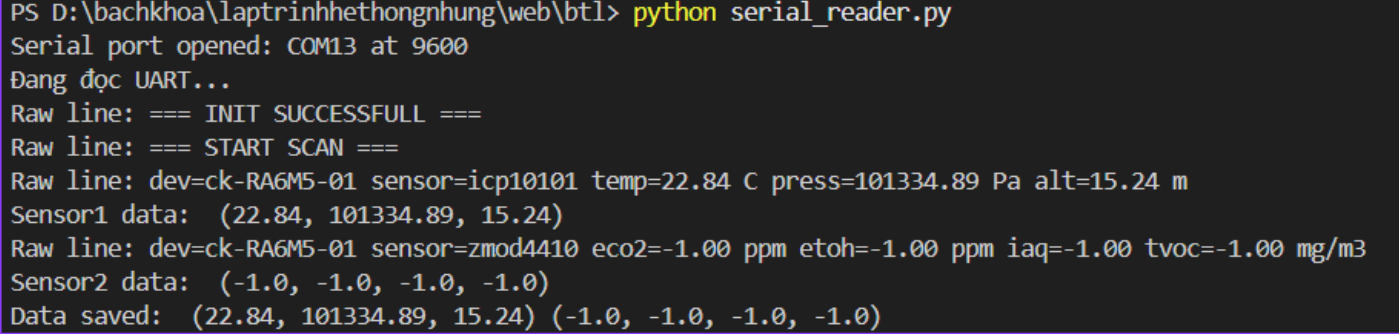
Thứ nhất chạy lệnh:

>python [manage.py](http://manage.py) runserver

Thứ nhất chạy lệnh:

>python serial\_reader.py

Sau khi lệnh serial\_reader.py chạy sẽ bắt đầu đọc và lưu dữ liệu như sau:



Và dữ liệu này sẽ được tách ra và lưu vào database sau đó sẽ được đóng gói và gửi lên frontend cho browser như sau:

