

# BÀI 9: DEVICE TREE (DTS)

## Bản đồ phần cứng cho hệ điều hành

# Mục tiêu bài học

1. **Tại sao cần DTS:** Hiểu vấn đề "Hardcoding" trong nhân Linux cũ.
2. **Cú pháp:** Đọc hiểu các Node, Property, Label và Alias.
3. **Quy trình:** Biên dịch `.dts` sang `.dtb` (Device Tree Blob).
4. **Thực hành:** Sửa file dts để kích hoạt một thiết bị ngoại vi (LED/UART).

# 1. Device Tree là gì?

“ Là một cấu trúc dữ liệu mô tả phần cứng.”

- **Trước đây (Legacy):** Thông tin phần cứng (địa chỉ RAM, chân GPIO) được viết cứng trong code C ( `arch/arm/mach-xxx` ). Mỗi lần sửa phần cứng phải biên dịch lại Kernel.
- **Hiện nay (Device Tree):** Phần cứng được mô tả trong file văn bản `.dts` . Kernel đọc file này khi khởi động để biết có những thiết bị nào.
- Giúp Kernel trở nên "Generic" (Một kernel chạy được trên nhiều board khác nhau).

## 2. Quy trình làm việc (Workflow)

1. **DTS** ( **.dts** ): Source code (Người đọc được).
2. **DTC (Device Tree Compiler)**: Trình biên dịch.
3. **DTB** ( **.dtb** ): Binary (Máy đọc được) - File này sẽ được nạp vào RAM cùng Kernel.

*# Lệnh biên dịch thủ công*

```
dtc -I dts -O dtb -o myboard.dtb myboard.dts
```

### 3. Cấu trúc một file DTS

```
/dts-v1/;

/ {
    model = "My Embedded Board";
    compatible = "ti,am33xx";

    cpus {
        cpu@0 {
            compatible = "arm,cortex-a8";
        };
    };

    memory@80000000 {
        device_type = "memory";
        reg = <0x80000000 0x10000000>; /* 256MB RAM */
    };

    leds {
        compatible = "gpio-leds";
        user_led0 {
            label = "heartbeat";
            gpios = <&gpio1 21 0>;
        };
    };
};
```

## 4. Các thuộc tính quan trọng

- **compatible** : Quan trọng nhất! Kernel dùng chuỗi này để tìm Driver phù hợp.
- **reg** : Địa chỉ vật lý và độ dài vùng nhớ (Register address).
- **status** :
  - **"okay"** : Thiết bị được kích hoạt.
  - **"disabled"** : Thiết bị bị tắt.
- **label** : Nhãn để tham chiếu (ví dụ **&gpio1** ở slide trước).

# **PHẦN THỰC HÀNH (LAB 09)**

## **Chỉnh sửa & Biên dịch Device Tree**

# Bước 1: Xác định file dts của Board

Trong thư mục source kernel: `arch/arm/boot/dts/`

- **BeagleBone Black:** `am335x-boneblack.dts`
- **Raspberry Pi 3:** `bcm2710-rpi-3-b.dts`



## Bước 2: Thử sửa đổi (Tắt bớt thiết bị)

Mở file dts (hoặc dtsti) tương ứng:

```
/* Tìm node của i2c2 hoặc uart1 */  
&i2c2 {  
    status = "disabled"; /* Thử đổi từ okay sang disabled */  
};
```

## Bước 3: Biên dịch DTB

Quay lại thư mục gốc của Kernel:

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-

# Chỉ biên dịch các file device tree
make dtbs
```

Kết quả: File `.dtb` mới sẽ nằm trong `arch/arm/boot/dts/`.



## Bài tập về nhà

1. Tìm hiểu khái niệm **Device Tree Overlay ( .dtbo )**. Tại sao nó quan trọng với các board như Raspberry Pi và BeagleBone (khi gắn thêm Cape/Hat)?
2. Đọc tài liệu: [Documentation/devicetree/bindings/](#) trong source kernel để hiểu cách viết property cho một cảm biến cụ thể.

# Q & A

## Hẹn gặp lại ở Bài 10: Biên dịch Kernel!