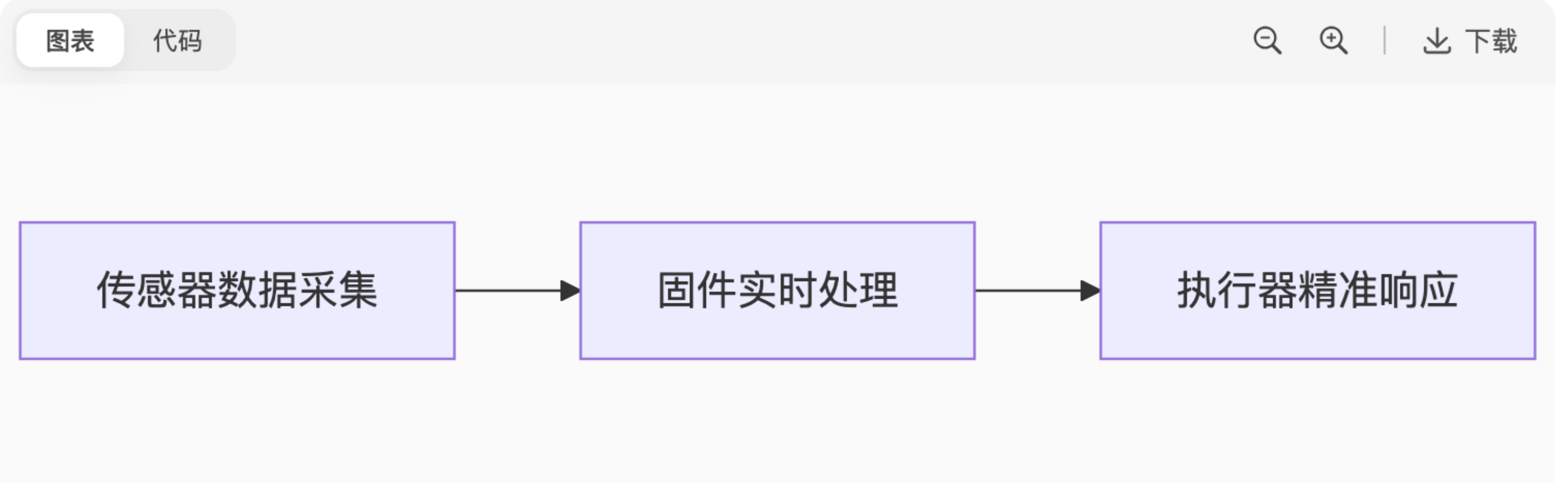
# Speeduino ECU 工作原理详解

**传感器 → 固件处理 → 执行器** 全流程技术文档



## 一、传感器数据采集阶段

### 1. 核心传感器类型与功能

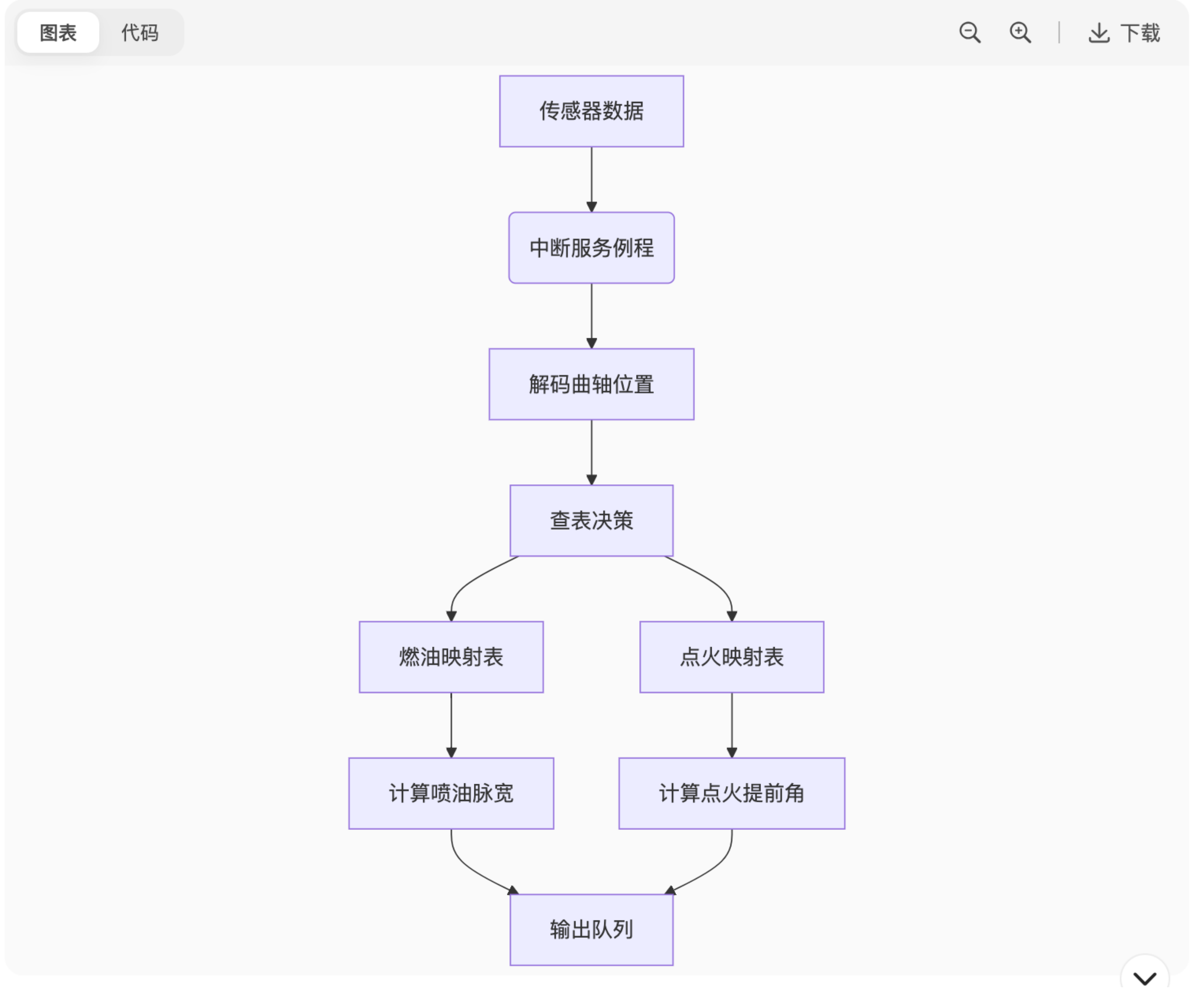
| 传感器 | 物理信号 | ECU接收信号 | 采样频率 | 关键用途 |
|---|---|---|---|---|
| **曲轴位置传感器** | 磁脉冲/方波 | 数字脉冲信号 | 每0.5°曲轴转角 | 计算转速、确定活塞位置(TDC/BDC) |
| **凸轮位置传感器** | 霍尔方波 | 单脉冲信号 | 每720°曲轴转角 | 判缸(区分压缩/排气冲程) |
| **MAP传感器** | 进气歧管压力 | 模拟电压(0-5V) | 100Hz | 计算进气量→喷油量基础 |
| **TPS传感器** | 节气门开度 | 模拟电压(0-5V) | 50Hz | 负载计算→急加速增油 |
| **O2传感器** | 排气氧浓度 | 模拟电压(0-5V) | 10Hz | 闭环控制空燃比(λ=14.7) |
| **CLT/IAT** | 温度 | 模拟电压(0-5V) | 5Hz | 冷启动补偿/空气密度修正 |

### 2. 信号预处理电路

```c
// 硬件层信号处理示例
void processCrankSignal() {
  if (rawSignal > 2.5V) {   // 通过比较器整形
    digitalSignal = HIGH;
  } else {
    digitalSignal = LOW;
  }
}

// ADC转换代码 (Arduino Mega)
int readMAP() {
  int raw = analogRead(MAP_PIN);  // 10位ADC (0-1023)
  return raw * 0.00488;           // 转换为电压值(0-5V)
}
```

## 二、固件处理阶段

### 1. 实时处理架构



### 2. 核心算法与代码实现

```c
// 每0.5°曲轴转角触发的中断服务例程
ISR(TIMER1_COMPA_vect) {
  crankAngle += 0.5;  // 更新曲轴角度

  // 1. 解码曲轴位置
  if (crankAngle == 0) {
    currentStroke = COMPRESSION; // 上止点
  }

  // 2. 查表决策 (每90°执行一次)
  if (crankAngle % 90 == 0) {
    // 获取当前工况
    int rpm = getRPM();
    float load = getLoad(); // 基于MAP或MAF

    // 查燃油映射表
    basePulseWidth = fuelTable[rpm][load];

    // 查点火映射表
    ignitionAngle = ignitionTable[rpm][load];
  }

  // 3. 实时修正
  applyCorrections(); // 温度/氧传感器修正
}
```

### 3. 三维控制映射表示例

**燃油映射表 (单位: ms)**

| RPM\负载 | 20kPa | 40kPa | 60kPa | 80kPa | 100kPa |
|---|---|---|---|---|---|
| **1000** | 2.1 | 2.3 | 2.6 | 3.0 | 3.5 |
| **3000** | 3.0 | 3.5 | 4.2 | 5.0 | 6.0 |
| **6000** | 4.5 | 5.2 | 6.0 | 7.5 | 9.0 |

**点火映射表 (单位: °BTDC)**

| RPM\负载 | 20kPa | 40kPa | 60kPa | 80kPa | 100kPa |
|---|---|---|---|---|---|
| **1000** | 12 | 14 | 16 | 18 | 20 |
| **3000** | 18 | 22 | 26 | 30 | 28 |
| **6000** | 24 | 28 | 32 | 30 | 26 |

### 4. 高级控制算法

```c
// 空燃比闭环控制
void closedLoopControl() {
  float targetAFR = 14.7;  // 理论最佳空燃比
  float actualAFR = readO2Sensor();

  // PID控制器
  float error = targetAFR - actualAFR;
  integral += error * dt;
  derivative = (error - prevError) / dt;

  float correction = KP*error + KI*integral + KD*derivative;

  // 应用修正
  currentPulseWidth *= (1.0 + correction/100.0);

  prevError = error;
}
```

## 三、执行器驱动阶段

### 1. 执行器类型与控制参数

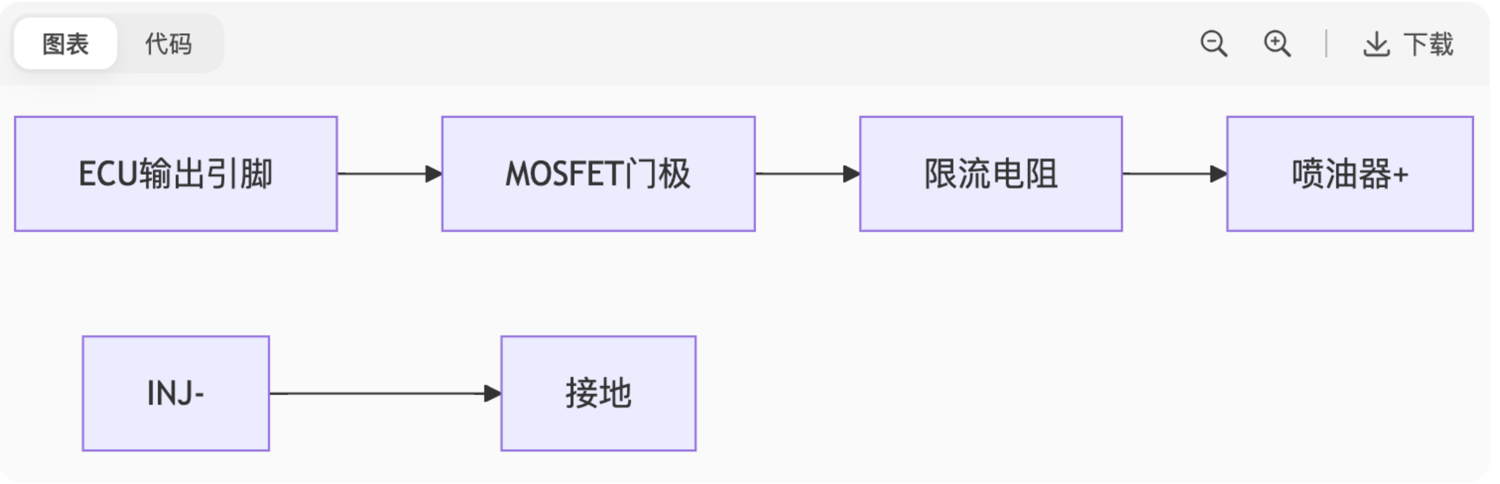
| 执行器 | ECU输出信号 | 物理动作 | 时序精度 | 驱动电路 |
|---|---|---|---|---|
| **喷油器** | 12V PWM脉冲 | 电磁阀开启→燃油喷射 | ±0.1ms | MOSFET驱动模块(VNH50 |
| **点火线圈** | 5V逻辑电平 | 初级线圈断电→高压放电 | ±0.5°曲轴转角 | BIP373达林顿管 |
| **怠速阀(IAC)** | 步进脉冲序列 | 阀芯位置调节 | 每100ms更新 | L293D步进电机驱动器 |
| **燃油泵** | 继电器控制信号 | 燃油系统加压 | 点火开关ON时启动 | 功率继电器 |

### 2. 执行器驱动代码

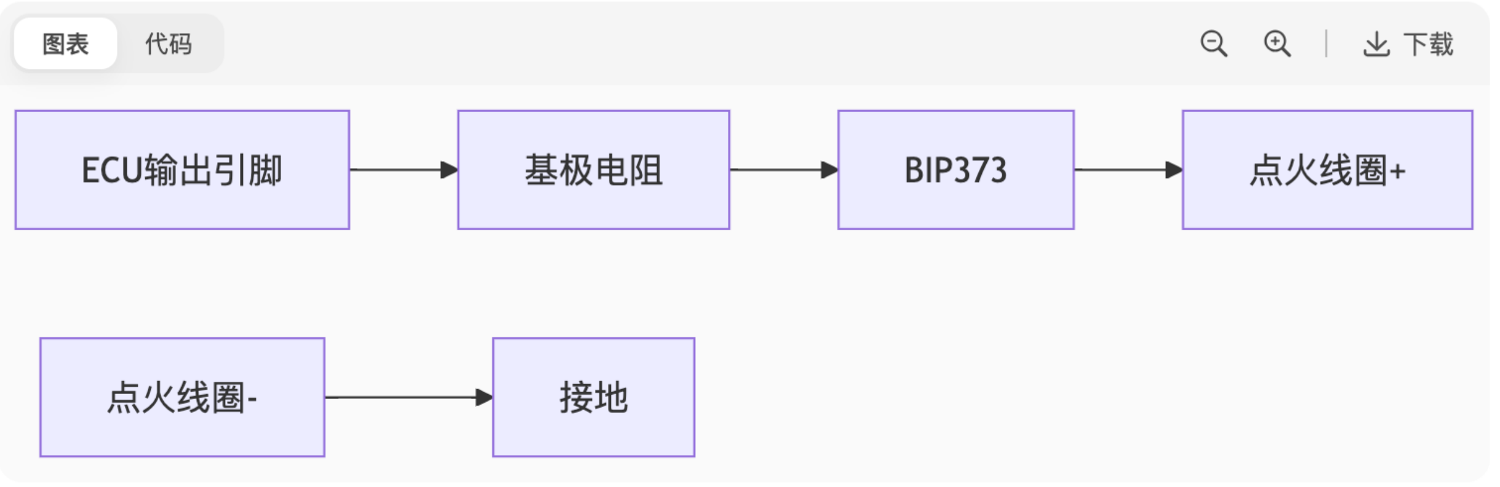
```c
// 喷油器控制示例
void triggerInjector(int cylinder, float startAngle, float pulseWidth) {
  // 计算喷油开始角度
  float currentAngle = getCrankAngle();
  float angleDiff = startAngle - currentAngle;

  if (angleDiff > 0) {
    // 设置定时器在目标角度触发
    setTimer(angleDiff, injectorOn);
  }
}

void injectorOn() {
  digitalWrite(INJ_PIN[cylinder], HIGH);
  setTimer(pulseWidth, injectorOff);
}

void injectorOff() {
  digitalWrite(INJ_PIN[cylinder], LOW);
}

// 点火控制示例
void fireIgnition(int cylinder, float advanceAngle) {
  float ignitionPoint = 360.0 - advanceAngle; // 上止点前角度

  if (getCrankAngle() >= ignitionPoint) {
    digitalWrite(IGN_PIN[cylinder], HIGH);
    delayMicroseconds(50); // 线圈充电时间
    digitalWrite(IGN_PIN[cylinder], LOW); // 产生火花
  }
}
```

### 3. 驱动电路设计

**喷油器驱动电路**



**点火驱动电路**



## 四、全流程示例：冷启动工况

1. **钥匙转到ON位置**
   - ECU通电自检(POST)
   - 燃油泵继电器激活2秒建立油压
2. **启动马达工作**
   - 曲轴传感器检测转速>200RPM
   - ECU进入启动模式：

```c
if (rpm < 800 && clt < 70°C) {
  pulseWidth *= 1.5; // 冷启动加浓
  ignitionAngle += 5; // 增加点火提前角
}
```

3. **首次点火与运行**
   - 凸轮传感器提供相位参考
   - 顺序喷射启动(各缸独立控制)
   - 氧传感器加热，准备闭环控制
4. **暖机过程**
   - 冷却液温度<70°C时维持快怠速
   - IAC阀逐步关闭进气旁通通道
   - O2传感器达到工作温度后启用闭环控制

## 五、故障安全机制

### 1. 传感器失效处理

| 故障传感器 | 降级策略 | 影响 |
|---|---|---|
| **凸轮传感器** | 切换至分组喷射模式 | 油耗增加5-10% |
| **氧传感器** | 禁用闭环控制，使用开环燃油表 | 排放超标，油耗略增 |
| **冷却液温度** | 使用默认值(80°C)并报警 | 冷启动困难，暖机性能下降 |
| **进气温度** | 使用默认值(25°C) | 高温天气可能爆震 |

### 2. 系统保护功能

```c
// 喷油器过热保护
void injectorProtection() {
  float dutyCycle = pulseWidth / (60.0/rpm*1000)*100; // 计算负载周期

  if (dutyCycle > 85%) {
    reduceFuel(10%); // 强制减少喷油量
    setErrorCode(ERR_INJ_OVERLOAD);
  }
}

// 看门狗定时器
void watchdogReset() {
  if (lastReset > 500ms) {
    systemReset(); // 强制ECU重启
  }
}
```

## 六、性能优化技术

1. **时间触发架构**
   - 硬中断保证时序精度(每0.5°曲轴转角)
   - 软定时器处理非实时任务
2. **映射表压缩算法**

```c
// 使用双线性插值减少表尺寸
float interpolateTable(float table[][], float x, float y) {
  int x1 = floor(x), x2 = ceil(x);
  int y1 = floor(y), y2 = ceil(y);

  float q11 = table[x1][y1];
  float q12 = table[x1][y2];
  float q21 = table[x2][y1];
  float q22 = table[x2][y2];

  return (q11*(x2-x)*(y2-y) + q21*(x-x1)*(y2-y) +
          q12*(x2-x)*(y-y1) + q22*(x-x1)*(y-y1));
}
```

3. **RAM优化策略**
   - 关键变量使用 `register` 关键字
   - 频繁访问的表存放在快速RAM区

---