# **Tài liệu hướng dẫn sử dụng SPI Driver theo chuẩn AUTOSAR Classic**

## **Mục lục**

1. Giới thiệu
2. Cấu trúc dữ liệu và các định nghĩa quan trọng
3. Macro và định nghĩa cần thiết
4. Các hàm API
   * **Spi\_Init**
   * **Spi\_DeInit**
   * **Spi\_WriteIB**
   * **Spi\_AsyncTransmit**
   * **Spi\_ReadIB**
   * **Spi\_SetupEB**
   * **Spi\_GetStatus**
   * **Spi\_GetJobResult**
   * **Spi\_GetSequenceResult**
   * **Spi\_GetVersionInfo**
   * **Spi\_SyncTransmit**
   * **Spi\_GetHWUnitStatus**
   * **Spi\_Cancel**
   * **Spi\_SetAsyncMode**
   * **Spi\_MainFunction\_Handling**

## **1. Giới thiệu**

**SPI Driver (Serial Peripheral Interface Driver)** theo chuẩn **AUTOSAR** cung cấp các API để khởi tạo, điều khiển, và quản lý giao tiếp SPI trên các vi điều khiển. SPI Driver quản lý quá trình truyền dữ liệu giữa vi điều khiển và các thiết bị ngoại vi như cảm biến, bộ nhớ, hay các thiết bị truyền thông khác.

Mục đích: Cung cấp một lớp trừu tượng cho phần cứng SPI, giúp mã nguồn dễ dàng được tái sử dụng và bảo trì mà không cần thay đổi khi chuyển đổi giữa các nền tảng phần cứng khác nhau.

## **2. Cấu trúc dữ liệu và các định nghĩa quan trọng**

### **Spi\_ChannelType**

* Đại diện cho một kênh SPI.
* **Kiểu dữ liệu**: uint8.

### **Spi\_JobType**

* Đại diện cho một Job SPI (tương ứng với một nhiệm vụ cụ thể).
* **Kiểu dữ liệu**: uint8.

### **Spi\_SequenceType**

* Đại diện cho một Sequence SPI (chuỗi các Job cần thực hiện).
* **Kiểu dữ liệu**: uint8.

### **Spi\_DataBufferType**

* Đại diện cho dữ liệu được truyền/nhận qua SPI.
* **Kiểu dữ liệu**: uint8.

### **Spi\_StatusType**

* Trạng thái của SPI Driver.
  + SPI\_UNINIT: Chưa được khởi tạo.
  + SPI\_IDLE: Nhàn rỗi.
  + SPI\_BUSY: Đang bận xử lý truyền/nhận.
* **Kiểu dữ liệu**: uint8.

### **Spi\_JobResultType**

* Kết quả của một Job SPI.
  + SPI\_JOB\_OK: Hoàn thành thành công.
  + SPI\_JOB\_PENDING: Đang chờ xử lý.
  + SPI\_JOB\_FAILED: Thất bại.
* **Kiểu dữ liệu**: uint8.

### **Spi\_SeqResultType**

* Kết quả của một Sequence SPI.
  + SPI\_SEQ\_OK: Hoàn thành thành công.
  + SPI\_SEQ\_PENDING: Đang chờ xử lý.
  + SPI\_SEQ\_FAILED: Thất bại.
  + SPI\_SEQ\_CANCELED: Bị hủy bỏ.
* **Kiểu dữ liệu**: uint8.

### **Spi\_BaudRateType**

* Tốc độ truyền dữ liệu qua SPI (Baud Rate).
* **Kiểu dữ liệu**: uint32.

## **3. Macro và định nghĩa cần thiết**

### **Định nghĩa các kênh SPI**

| #define SPI\_CHANNEL\_1 0 /\*\*< Kênh SPI1 \*/ #define SPI\_CHANNEL\_2 1 /\*\*< Kênh SPI2 \*/ |
| --- |

### **Định nghĩa các Job SPI**

| #define SPI\_JOB\_1 0 /\*\*< Job 1 \*/ #define SPI\_JOB\_2 1 /\*\*< Job 2 \*/ |
| --- |

### **Định nghĩa các Sequence SPI**

| #define SPI\_SEQ\_1 0 /\*\*< Sequence 1 \*/ #define SPI\_SEQ\_2 1 /\*\*< Sequence 2 \*/ |
| --- |

## **4. Các hàm API**

### **4.1 Spi\_Init**

* **Chức năng**: Khởi tạo SPI Handler/Driver với cấu hình đã cho.

**Cú pháp**:

| void Spi\_Init(const Spi\_ConfigType\* ConfigPtr); |
| --- |

* **Tham số**:
  + ConfigPtr: Con trỏ tới cấu trúc cấu hình Spi\_ConfigType chứa các thông số khởi tạo SPI.
* **Mô tả**: Hàm này khởi tạo các thành phần của SPI, bao gồm các kênh, Job, và phần cứng SPI. Cấu hình này có thể bao gồm tốc độ baud rate, cực tính CPOL, pha CPHA, chế độ Master/Slave, và các tham số khác.

**Ví dụ**:

| Spi\_ConfigType spiConfig = {  .Channel = SPI\_CHANNEL\_1,  .Job = SPI\_JOB\_1,  .Sequence = SPI\_SEQ\_1 };  Spi\_Init(&spiConfig); |
| --- |

### **4.2 Spi\_DeInit**

* **Chức năng**: Hủy khởi tạo SPI Driver và giải phóng tài nguyên liên quan.

**Cú pháp**:

| Std\_ReturnType Spi\_DeInit(void); |
| --- |

* **Giá trị trả về**:
  + E\_OK: Hủy thành công.
  + E\_NOT\_OK: Hủy thất bại.
* **Mô tả**: Hàm này hủy khởi tạo SPI Driver, tắt các kênh SPI và giải phóng các tài nguyên đã được cấp phát trong quá trình khởi tạo.

**Ví dụ**:

| Std\_ReturnType result = Spi\_DeInit(); if (result == E\_OK) {  // Hủy thành công } else {  // Hủy thất bại } |
| --- |

### **4.3 Spi\_WriteIB**

* **Chức năng**: Ghi dữ liệu vào kênh SPI.

**Cú pháp**:

| Std\_ReturnType Spi\_WriteIB(Spi\_ChannelType Channel, const Spi\_DataBufferType\* DataBufferPtr); |
| --- |

* **Tham số**:
  + Channel: ID của kênh SPI.
  + DataBufferPtr: Con trỏ trỏ tới buffer chứa dữ liệu cần ghi.
* **Giá trị trả về**:
  + E\_OK: Ghi thành công.
  + E\_NOT\_OK: Ghi thất bại.
* **Mô tả**: Ghi dữ liệu vào bộ đệm nội bộ của SPI Driver để truyền dữ liệu ra ngoài.

**Ví dụ**:

| Spi\_DataBufferType dataToSend = 0x55; // Dữ liệu 8-bit cần ghi Std\_ReturnType result = Spi\_WriteIB(SPI\_CHANNEL\_1, &dataToSend); if (result == E\_OK) {  // Ghi dữ liệu thành công } else {  // Ghi dữ liệu thất bại } |
| --- |

### **4.4 Spi\_AsyncTransmit**

* **Chức năng**: Truyền dữ liệu không đồng bộ.

**Cú pháp**:

| Std\_ReturnType Spi\_AsyncTransmit(Spi\_SequenceType Sequence); |
| --- |

* **Tham số**:
  + Sequence: ID của Sequence cần truyền.
* **Giá trị trả về**:
  + E\_OK: Truyền thành công.
  + E\_NOT\_OK: Truyền thất bại.
* **Mô tả**: Truyền dữ liệu không đồng bộ qua Sequence SPI mà không cần đợi quá trình truyền kết thúc.

**Ví dụ**:

| Std\_ReturnType result = Spi\_AsyncTransmit(SPI\_SEQ\_1); if (result == E\_OK) {  // Truyền không đồng bộ thành công } else {  // Truyền thất bại } |
| --- |

### **4.5 Spi\_ReadIB**

* **Chức năng**: Đọc dữ liệu từ kênh SPI.

**Cú pháp**:

| Std\_ReturnType Spi\_ReadIB(Spi\_ChannelType Channel, Spi\_DataBufferType\* DataBufferPtr); |
| --- |

* **Tham số**:
  + Channel: ID của kênh SPI.
  + DataBufferPtr: Con trỏ trỏ tới buffer để lưu dữ liệu nhận được.
* **Giá trị trả về**:
  + E\_OK: Đọc thành công.
  + E\_NOT\_OK: Đọc thất bại.
* **Mô tả**: Đọc dữ liệu từ buffer nội bộ của kênh SPI.

**Ví dụ**:

| Spi\_DataBufferType receivedData; Std\_ReturnType result = Spi\_ReadIB(SPI\_CHANNEL\_1, &receivedData); if (result == E\_OK) {  // Đọc dữ liệu thành công } else {  // Đọc dữ liệu thất bại } |
| --- |

### **4.6 Spi\_SetupEB**

* **Chức năng**: Cấu hình bộ đệm ngoài cho kênh SPI.

**Cú pháp**:

| Std\_ReturnType Spi\_SetupEB(Spi\_ChannelType Channel, const Spi\_DataBufferType\* SrcDataBufferPtr, Spi\_DataBufferType\* DesDataBufferPtr, Spi\_NumberOfDataType Length); |
| --- |

* **Tham số**:
  + Channel: ID của kênh SPI.
  + SrcDataBufferPtr: Con trỏ trỏ tới buffer nguồn.
  + DesDataBufferPtr: Con trỏ trỏ tới buffer đích.
  + Length: Số lượng phần tử dữ liệu cần truyền.
* **Giá trị trả về**:
  + E\_OK: Cấu hình thành công.
  + E\_NOT\_OK: Cấu hình thất bại.
* **Mô tả**: Cấu hình buffer ngoài cho kênh SPI để thực hiện truyền/nhận dữ liệu.

**Ví dụ**:

| Spi\_DataBufferType srcBuffer[10] = {0x01, 0x02, 0x03}; // Dữ liệu nguồn Spi\_DataBufferType destBuffer[10]; // Bộ đệm đích Std\_ReturnType result = Spi\_SetupEB(SPI\_CHANNEL\_1, srcBuffer, destBuffer, 10); if (result == E\_OK) {  // Cấu hình thành công } else {  // Cấu hình thất bại } |
| --- |

### **4.7 Spi\_GetStatus**

* **Chức năng**: Lấy trạng thái hiện tại của SPI Driver.

**Cú pháp**:

| Spi\_StatusType Spi\_GetStatus(void); |
| --- |

* **Giá trị trả về**: Trạng thái hiện tại của SPI Driver (SPI\_UNINIT, SPI\_IDLE, SPI\_BUSY).
* **Mô tả**: Hàm này trả về trạng thái hiện tại của SPI Driver.

**Ví dụ**:

| Spi\_StatusType status = Spi\_GetStatus(); if (status == SPI\_BUSY) {  // SPI đang bận } else {  // SPI đang nhàn rỗi } |
| --- |

### **4.8 Spi\_GetJobResult**

* **Chức năng**: Lấy kết quả của một Job SPI.

**Cú pháp**:

| Spi\_JobResultType Spi\_GetJobResult(Spi\_JobType Job); |
| --- |

* **Tham số**:
  + Job: ID của Job cần kiểm tra kết quả.
* **Giá trị trả về**: Kết quả của Job (SPI\_JOB\_OK, SPI\_JOB\_PENDING, SPI\_JOB\_FAILED, SPI\_JOB\_QUEUED).
* **Mô tả**: Hàm này trả về kết quả của một Job SPI đã được truyền.

**Ví dụ**:

| Spi\_JobResultType jobResult = Spi\_GetJobResult(SPI\_JOB\_1); if (jobResult == SPI\_JOB\_OK) {  // Job hoàn thành thành công } else {  // Job thất bại hoặc đang chờ xử lý } |
| --- |

### **4.9 Spi\_GetSequenceResult**

* **Chức năng**: Lấy kết quả của một Sequence SPI.

**Cú pháp**:

| Spi\_SeqResultType Spi\_GetSequenceResult(Spi\_SequenceType Sequence); |
| --- |

* **Tham số**:
  + Sequence: ID của Sequence cần kiểm tra kết quả.
* **Giá trị trả về**: Kết quả của Sequence (SPI\_SEQ\_OK, SPI\_SEQ\_PENDING, SPI\_SEQ\_FAILED, SPI\_SEQ\_CANCELED).
* **Mô tả**: Hàm này trả về kết quả của một Sequence SPI đã được truyền.

**Ví dụ**:

| Spi\_SeqResultType seqResult = Spi\_GetSequenceResult(SPI\_SEQ\_1); if (seqResult == SPI\_SEQ\_OK) {  // Sequence hoàn thành thành công } else {  // Sequence thất bại hoặc bị hủy } |
| --- |

### **4.10 Spi\_GetVersionInfo**

* **Chức năng**: Lấy thông tin phiên bản của SPI Driver.

**Cú pháp**:

| void Spi\_GetVersionInfo(Std\_VersionInfoType\* VersionInfo); |
| --- |

* **Tham số**:
  + VersionInfo: Con trỏ tới cấu trúc Std\_VersionInfoType chứa thông tin phiên bản.
* **Mô tả**: Hàm này trả về thông tin phiên bản của SPI Driver.

**Ví dụ**:

| Std\_VersionInfoType versionInfo; Spi\_GetVersionInfo(&versionInfo); |
| --- |

### **4.11 Spi\_SyncTransmit**

* **Chức năng**: Truyền dữ liệu đồng bộ qua SPI.

**Cú pháp**:

| Std\_ReturnType Spi\_SyncTransmit(Spi\_SequenceType Sequence); |
| --- |

* **Tham số**:
  + Sequence: ID của Sequence cần truyền.
* **Giá trị trả về**:
  + E\_OK: Truyền thành công.
  + E\_NOT\_OK: Truyền thất bại.
* **Mô tả**: Truyền dữ liệu đồng bộ qua Sequence SPI và đợi quá trình truyền hoàn thành.

**Ví dụ**:

| Std\_ReturnType result = Spi\_SyncTransmit(SPI\_SEQ\_1); if (result == E\_OK) {  // Truyền thành công } else {  // Truyền thất bại } |
| --- |

### **4.12 Spi\_GetHWUnitStatus**

* **Chức năng**: Lấy trạng thái phần cứng của đơn vị SPI.

**Cú pháp**:

| Spi\_StatusType Spi\_GetHWUnitStatus(void); |
| --- |

* **Giá trị trả về**: Trạng thái hiện tại của phần cứng SPI.
* **Mô tả**: Hàm này trả về trạng thái của phần cứng SPI.

**Ví dụ**:

| Spi\_StatusType hwStatus = Spi\_GetHWUnitStatus(); |
| --- |

### **4.13 Spi\_Cancel**

* **Chức năng**: Hủy một Sequence đang truyền.

**Cú pháp**:

| Std\_ReturnType Spi\_Cancel(Spi\_SequenceType Sequence); |
| --- |

* **Tham số**:
  + Sequence: ID của Sequence cần hủy.
* **Giá trị trả về**:
  + E\_OK: Hủy thành công.
  + E\_NOT\_OK: Hủy thất bại.
* **Mô tả**: Hủy quá trình truyền của một Sequence hiện tại.

**Ví dụ**:

| Std\_ReturnType result = Spi\_Cancel(SPI\_SEQ\_1); |
| --- |

### **4.14 Spi\_SetAsyncMode**

* **Chức năng**: Cài đặt chế độ truyền không đồng bộ cho SPI.

**Cú pháp**:

| Std\_ReturnType Spi\_SetAsyncMode(Spi\_AsyncModeType Mode); |
| --- |

* **Tham số**:
  + Mode: Chế độ truyền không đồng bộ (SPI\_POLLING\_MODE hoặc SPI\_INTERRUPT\_MODE).
* **Giá trị trả về**:
  + E\_OK: Cài đặt thành công.
  + E\_NOT\_OK: Cài đặt thất bại.
* **Mô tả**: Cài đặt chế độ không đồng bộ cho quá trình truyền dữ liệu của SPI.

**Ví dụ**:

| Std\_ReturnType result = Spi\_SetAsyncMode(SPI\_POLLING\_MODE); |
| --- |

### **4.15 Spi\_MainFunction\_Handling**

* **Chức năng**: Hàm chính xử lý các tác vụ của SPI.

**Cú pháp**:

| void Spi\_MainFunction\_Handling(void); |
| --- |

* **Mô tả**: Hàm này được gọi định kỳ để xử lý các Job và Sequence của SPI.

**Ví dụ**:

| while (1) {  Spi\_MainFunction\_Handling(); } |
| --- |