|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | SPS-DDS | | |
| Deliver to | Smart Parking System  Software Detail Design  Revision: 0.1 | | 2023/02/28 | | |
|  | Ban Vien Corporation.  Embedded System Department | | |
| Approved | Checked | Issued |
| Status | Initializing / Reviewing / Approved | Name  dd/MM/yyyy | Name  dd/MM/yyyy | Name  dd/MM/yyyy |

**Table of Contents**

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

**List of tables**

￼

￼

￼

**List of figures**

￼

# Introduction

## Scope

This document describes software detail design specification of the Ban Vien Smart Parking System.

## References

|  |  |  |
| --- | --- | --- |
| **No.** | **Title (File Name)** | **Version** |
| [1] |  |  |
| [2] |  |  |
| [3] |  |  |
| [4] |  |  |
| [5] |  |  |
| [6] |  |  |
| [7] |  |  |
| [8] |  |  |
| [9] |  |  |
| [10] |  |  |
| [11] |  |  |
| [12] |  |  |

Table 1: Reference Table

## Acronyms and Abbreviations

The following acronyms and abbreviations are used throughout this document:

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| SPS | Smart Parking System |
| e.g. | [lat.] exempli gratia = [eng.] for example |
| x | applicable |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table 2: Acronyms and Abbreviations

# Architecture

## Overview of the SPS - HCM Sequence Diagram

The Sequence Diagram is shown below in Figure 1. In the diagram below there are:

* Four objects: Guards, Bvers, SPS and HCM.
* 4 APIs: Get data from employees, Report CI/CO, Upload files and Notify guards

The processing flow of the graph can be expressed as follows:

1. API “Report CI/CO”:
   * Bver swipe their employee card into the SPS.
   * SPS detects and processes images of Bver.
   * After processing, SPS posts the images to HCM through API “Upload files”.
   * HCM will return the result.
     + If HCM returns an error result, SPS will reupload files again.
     + If HCM returns multiple file names, SPS will post CI/CO info along with those that HCM just returned.
   * API example: obj.upload\_files(Path\_img\_1, Path\_img\_2, Path\_img\_3);
2. API “Get data employees”:
   * At a specific time of the day, SPS requests employee data from HCM through API “Get data employees.”
   * After HCM receives the request from SPS, HCM encrypts data and sends it back to HCM.
   * SPS decrypts data just returned from HCM and checks whether there is any data loss.
     + If the checksum is not ok, SPS will request it again.
   * API example: obj.get\_employee\_data();
3. API “Notify guards”:
   * Bver press the button to call guards at the SPS.
   * The SPS sends a notification to HCM.
   * HCM will return the result.
     + If HCM returns success result, HCM will notify both guards and Bver.
     + If HCM returns an error result, SPS will resend the notification to HCM.

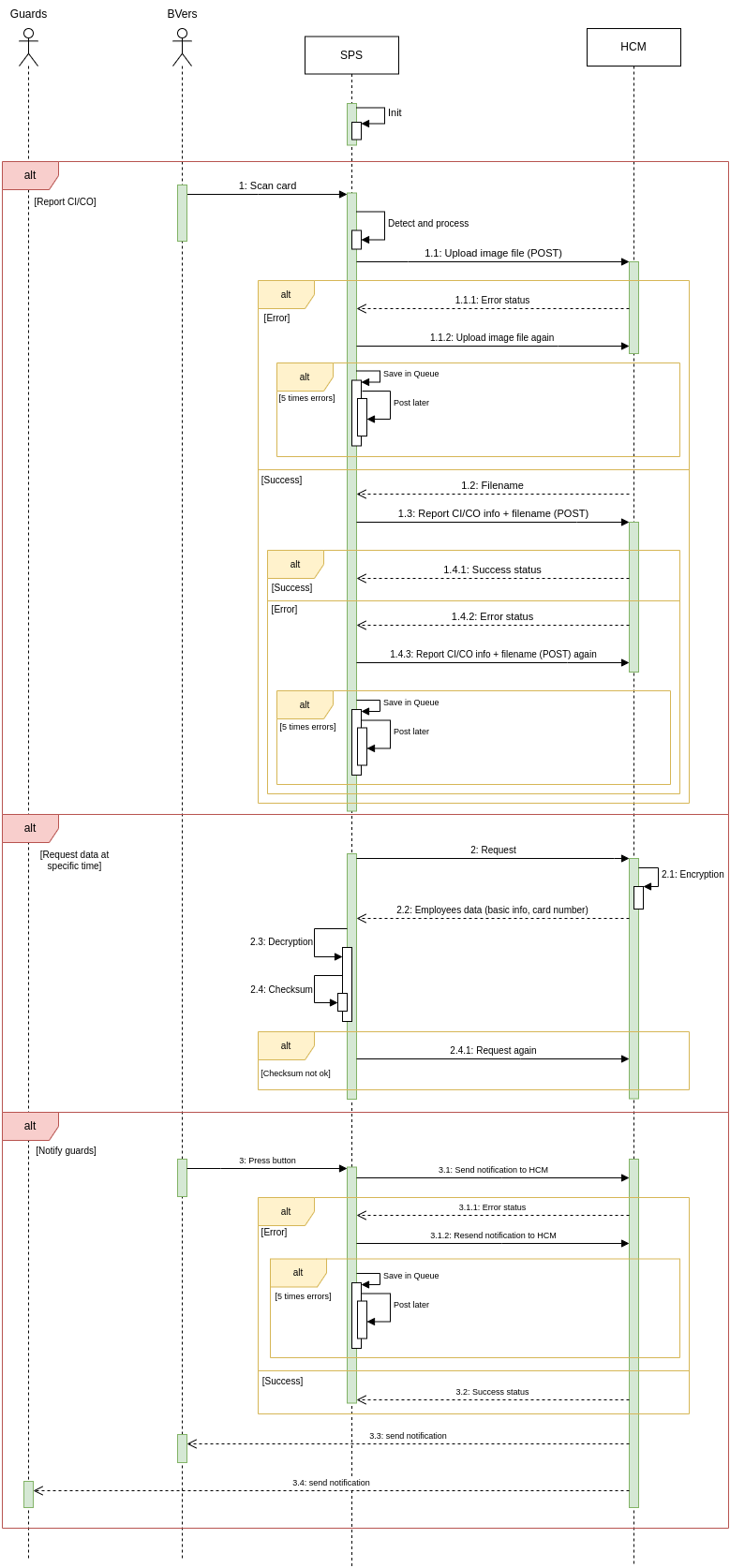


Figure 1: SPS - HCM Sequence Diagram

## File Structure

### Code File Structure

The following source code file is the driver.

**MSN\_DDD\_ARC\_001:**

|  |  |
| --- | --- |
|  |  |
|  |  |

{Ref: [2] RRS\_GEN\_NR\_0020}

### Header File Structure

The header-file structure shall include the following files:

**MSN\_DDD\_ARC\_002:**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

## H/W Function Mapping

### Main Supported H/W Features

**MSN\_DDD\_HWU\_001:**

H/W function provides for:

• Read a value from a digital I/O

• Write a value to a digital I/O

These functions do not depend on R-CarGen3 devices.

{Ref to UM of PortPin(Port Pin Specifications): [[4](#_References)] Chapter 7, [11] Chapter 6}

### Unsupported H/W Features

None

# Internal Function

## Interrupt Handler

None

## High-Level Driver Function

None

## Low-Level Driver Function

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Function Name** | **Description** | **Safety-related/**  **Non-Safety Related** |
| (1) | get\_employee\_data | This function is used to request data from HCM. | Safety-related |
| (2) | post\_data | This function is used to post CI/CO info to HCM. | Safety-related |
| (3) | upload\_files | This function is used to upload images to HCM. | Safety-related |

1. get\_employee\_data

**SPS\_DDD\_INF\_001:**

|  |  |  |
| --- | --- | --- |
| **Function Name:** | get\_employee\_data | |
| **Syntax:** | static FUNC()get\_employee\_data | |
| **Parameters (In):** | None |
|
| **Parameters (In-Out):** | None. | |
| **Parameters (Out):** | None. | |
| **Return Value:** | Std\_ReturnType | E\_OK - The checksum is ok.  E\_NOT\_OK - The checksum is not ok. |
| **Description:** | This function is used to request data from HCM. | |
| **Preconditions:** | None | |

|  |  |  |
| --- | --- | --- |
| **Function Name:** | post\_data | |
| **Syntax:** | static FUNC()post\_data | |
| **Parameters (In):** | Object |  |
| **Parameters (In-Out):** | None. | |
| **Parameters (Out):** | None. | |
| **Return Value:** | Std\_ReturnType | E\_OK - post successfully.  E\_NOT\_OK - Post fails. |
| **Description:** | This function is used to post CI/CO info to HCM. | |
| **Preconditions:** | Must have a file name from API “upload\_files.” | |

|  |  |  |
| --- | --- | --- |
| **Function Name:** | upload\_files | |
| **Syntax:** | static FUNC() upload\_files | |
| **Parameters (In):** | 3 Images |  |
| **Parameters (In-Out):** | None. | |
| **Parameters (Out):** | None. | |
| **Return Value:** | Std\_ReturnType | E\_OK - return with three file names.  E\_NOT\_OK - Upload fails. |
| **Description:** | This function is used for exclusive access to the registers from both CPUs (CA5x and CR7). | |
| **Preconditions:** | None | |

# State Transition

## Driver Status

N/A

# Data Flow

N/A

# Data Type

## General

### Msn\_ChannelType

**DIO\_DDD\_DAT\_001:**

|  |  |
| --- | --- |
| **Name:** | Msn\_ChannelType |
| **Type:** | unit8 |
| **Range:** | 0...(Total number of DIO channels-1) |
| **Description:** | Numeric ID of a DIO channel. |
| ***Location:*** | Dio.h |

{Ref: [1] SWS\_Dio\_00182, SWS\_Dio\_00015, SWS\_Dio\_00180}

## Structure

### API\_Mode

**DIO\_DDD\_DAT\_005:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:** | API\_Mode | | | |
| **Type:** | enum API\_Mode | | | |
| **1st Element:** | NONE\_REQUEST | This element represents the NONE mode in SPS. | | |
| Type: int | | |
| Range: 0 | | |
| **2nd Element:** | GET\_REQUEST | This element represents the GET mode in SPS. | | |
| Type: int | |  |
| Range: 1 | |  |
| **3rd Element:** | POST\_REQUEST | This element represents the POST mode in SPS. | | |
| Type: int |  | |
| Range: 2 |  | |
| **4th Element:** | UPLOAD\_REQUEST | This element represents the UPLOAD mode in SPS. | | |
| Type: int |  |  |
| Range: 3 |
| **Description:** | This enum is used to let SPS know what mode it is in. For example, at the finish state of each mode wil be treated differently. | | | |
| ***Location:*** | hcm\_api.cpp | | | |

### Data\_post

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:** | Data\_post | | | |
| **Type:** | enum Data\_post | | | |
| **1st Element:** | CARD\_ID | This element represents the CARD\_ID data in POST. | | |
| Type: int | | |
| Range: 0 | | |
| **2nd Element:** | CICO\_TYPE | This element represents the CICO\_TYPE data in POST. | | |
| Type: int | |  |
| Range: 1 | |  |
| **3rd Element:** | DEVICE\_ID | This element represents the DEVICE\_ID data in POST. | | |
| Type: int |  | |
| Range: 2 |  | |
| **4th Element:** | IMG\_BACK | This element represents the IMG\_BACK data in POST. | | |
| Type: int |  |  |
| Range: 3 |
| **5th Element:** | IMG\_FRONT | This element represents the IMG\_FRONT data in POST. | | |
| Type: int | | |
| Range: 4 | | |
| **6th Element:** | IMG\_PLATE | This element represents the IMG\_PLATE data in POST. | | |
| Type: int | | |
| Range: 5 | | |
| **7th Element:** | IN\_OUT | This element represents the IN\_OUT data in POST. | | |
| Type: int | | |
| Range: 6 | | |
| **8th Element:** | LICENSE\_PLATE | This element represents the LICENSE\_PLATE data in POST. | | |
| Type: int | | |
| Range: 7 | | |
| **9th Element:** | OFFICE\_ID | This element represents the OFFICE\_ID data in POST. | | |
| Type: int | | |
| Range: 8 | | |
| **10th Element:** | TIME | This element represents the TIME data in POST. | | |
| Type: int | | |
| Range: 9 | | |
| **11th Element:** | TYPE | This element represents the TYPE data in POST. | | |
| Type: int | | |
| Range: 10 | | |
| **Description:** | This enum is used to let POST know what data it is in. | | | |
| ***Location:*** | hcm\_api.cpp, hcm\_api\_employee.cpp | | | |

{Ref: [1] SWS\_Dio\_00184, SWS\_Dio\_00021, SWS\_Dio\_00022}

## Enumeration

### Dio\_ModuleName

**DIO\_DDD\_DAT\_008:**

|  |  |  |
| --- | --- | --- |
| Name: | Dio\_ModuleName | |
| Type: | Enumeration | |
| Range: | DIO\_GPIO\_MODULE | Selected module is the GPIO module. |
| Description: | This is the Module type for input parameter of Dio\_ExclusiveControl functions in the Dio module. | |
| ***Location:*** | Dio.h | |

{Ref: [5] SWS\_BSW\_00209, SWS\_BSW\_00146}

# Global Variable

The global variable is as follows. The set processing to each global variable, please refer to chapter [9 Activity](#_Activity_([6]_Section).

{Refer: [2] RRS\_GEN\_FR\_0030}

## Global Pointer Variable

### Dio\_GpPortGroup

**DIO\_DDD\_GBL\_001:**

|  |  |
| --- | --- |
| **Parameter Name:** | Dio\_GpPortGroup |
| **Type:** | P2CONST(Dio\_PortGroup, DIO\_VAR\_INIT, DIO\_CONFIG\_CONST)Dio\_GpPortGroup |
| **Generated Condition:** | None |
| **Description:** | Global variable to store pointer to Configuration of DIO port group. |
| **Referenced by:** | All DIO Driver function.  Dio\_ReadPort  Dio\_WritePort  Dio\_ReadChannel  Dio\_WriteChannel  Dio\_FlipChannel  Dio\_ReadChannelGroup  Dio\_WriteChannelGroup  Dio\_MaskedWritePort |
| **Init Value** | Address of configuration structure Dio\_GstPortGroup which is input of Dio\_ReadPort |
| **Range** | Same as Init value |

{Ref: [1] SWS\_Dio\_00133, SWS\_Dio\_00134}

{Ref: [2] RRS\_GEN\_FR\_0030}

## GlobalVariables

### Dio\_GstChannelGroupData

**DIO\_DDD\_GBL\_003:**

|  |  |
| --- | --- |
| **Parameter Name:** | Dio\_GstChannelGroupData |
| **Type:** | CONST(Dio\_ChannelGroupType, DIO\_CONST) Dio\_GstChannelGroupData[]; |
| **Generated Condition:** | None |
| **Description:** | Global variable to store data configuration of DIO Port Channel Group. |
| **Referenced by:** | Dio\_ReadChannelGroup  Dio\_WriteChannelGroup |
| **Init Value** | Dependance on the config value |
| **Range** | Dio\_GstChannelGroupData[0]->Dio\_GstChannelGroupData[7](R-Car H3/M3/M3N)  Dio\_GstChannelGroupData[0]->Dio\_GstChannelGroupData[5](R-Car V3M/V3H/V3Hv2)  Dio\_GstChannelGroupData[0]->Dio\_GstChannelGroupData[9](R-Car V3U) |

{Ref: [1] SWS\_Dio\_00137, SWS\_Dio\_00138}

{Ref: [2] RRS\_GEN\_FR\_0030}

## StatusVariables

N/A

# Activity Diagram

Refer to the following file:

# Relation of Requirement and Target Devices

| **Requirement ID** | **Target Devices** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **H3** | **V3M** | **M3** | **V3H** | **M3N** | **V3Hv2** | **V3U** |
| **Architecture** | | | | | |  |  |
| DIO\_DDD\_ARC\_001 | x | x | x | x | x | x | x |
| DIO\_DDD\_ARC\_002 | x | x | x | x | x | x | x |
| DIO\_DDD\_ARC\_003 | x | x | x | x | x | x | x |
| DIO\_DDD\_ARC\_004 | x | x | x | x | x | x | x |
| **H/W Resource Usage** | | | | | |  |  |
| DIO\_DDD\_HWU\_001 | x | x | x | x | x | x | x |
| DIO\_DDD\_HWU\_002 | x | x | x | x | x | x | x |
| **Internal Function** | | | | |  |  |  |
| DIO\_DDD\_INF\_001 | x | x | x | x | x | x | x |
| **State Transition** | | | | |  |  |  |
| - | - | - | - | - | - | - | - |
| **Data Flow** | | | | |  |  |  |
| - | - | - | - | - | - | - | - |
| **Data Type** | | | | | |  |  |
| DIO\_DDD\_DAT\_001 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_002 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_003 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_004 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_005 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_006 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_007 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_008 | x | x | x | x | x | x | x |
| DIO\_DDD\_DAT\_009 | x | x | x | x | x | x | x |
| **Global Variables** | | | | | |  |  |
| DIO\_DDD\_GBL\_001 | x | x | x | x | x | x | x |
| DIO\_DDD\_GBL\_002 | x | x | x | x | x | x | x |
| DIO\_DDD\_GBL\_003 | x | x | x | x | x | x | x |

Table 5: Relation of Requirement and Target Devices

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Ver.** | **Change** | **Changed By** |
| 2022-01-09 | 1.0.0 | - Initial Version. | Nhat Nguyen |