**Software DESIGN DESCRIPTION (SDD)**

**For**

**Uninterruptible Power System**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Number** | **Version** | **Issue Date** | **No. Of Pages** |
| SDD/006 | 1.1 | 18-04-2013 | 92 |

|  |  |
| --- | --- |
| **Owner(s) of this document** | **List of authorized users** |
| AuthorName1 | 1. Project Team members  2. Customers and external agencies on approval from Management |

|  |  |  |  |
| --- | --- | --- | --- |
| **Verified by:** | | **Approved by:** | |
| **Name and**  **Designation** | **Signature** | **Name and**  **Designation** | **Signature** |
| Name2 |  | Name3 |  |

**Amendment History:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Issue no.** | **Brief Description of Amendment** | **Change Request**  **Ref.** | **Owner(s)** | **Verified**  **By** | **Approved by** | **Eff. From**  **Date** |
| 1.0 | Initial Issue | NA | AuthorName1 | Name2 | Name3 | 18-04-2013 |

**Table Of Contents**

[1 INTRODUCTION 5](#_Toc354265947)

[1.1 PURPOSE 5](#_Toc354265948)

[1.2 SCOPE 5](#_Toc354265949)

[1.3 Definitions, acronyms and abbreviations 5](#_Toc354265950)

[1.4 References 6](#_Toc354265951)

[1.5 Overview 6](#_Toc354265952)

[2 references 7](#_Toc354265953)

[3 decomposition description 8](#_Toc354265954)

[3.1 Module Decomposition 8](#_Toc354265955)

[3.2 Concurrent process decomposition 8](#_Toc354265956)

[3.3 DATA DECOMPOSITION 8](#_Toc354265957)

[4 DEPENDENCY DESCRIPTION 9](#_Toc354265958)

[4.1 INTERMODULE DEPENDENCIES 9](#_Toc354265959)

[4.2 INTERPROCESS DEPENDENCIES 9](#_Toc354265960)

[4.3 DATA DEPENDENCIES 9](#_Toc354265961)

[5 INTERFACE DESCRIPTION 9](#_Toc354265962)

[5.1 MODULE INTERFACE 9](#_Toc354265963)

[5.2 PROCESS INTERFACE 9](#_Toc354265964)

[6 DETAILED DESIGN 10](#_Toc354265965)

[6.1 MODULE DETAILED DESIGN 10](#_Toc354265966)

**LIST OF FIGURES**

Table No. Description Page No.

No table of figures entries found.

List of Tables

Table No. Description Page No.

[Table 1: Abbreviations 6](#_Toc354065399)

# INTRODUCTION

## PURPOSE

The Software Design Description Document is the translation of requirements into the description of the Software Structure, Software Components, Interfaces and the Data necessary for the Implementation Phase. In the complete Software Design Description Document, each requirement will be traced to one or more design entities and the important properties and relationships among these entities have been clearly defined. In essence, the Software Design Description Document becomes the detailed blue print for the implementation activity.

## SCOPE

The Software Design Description Document shows how the software system will be structured to satisfy all the requirements identified in the Software Requirements Specifications document.

## Definitions, acronyms and abbreviations

AC Alternating Current

Support Time: Time during which the UPS can supply the rated load with nominal-quality power while the primary power is down.

Battery: A device that converts chemical energy into electrical energy by means of an electrochemical reaction. Usually defined as consisting of two or more cells, but commonly used to refer to one cell.

Brownout: Conditions under which power is available but not sufficient to fully meet the needs (voltage, current) of the load; partial power failure.

Capacity (battery): The number of ampere-hours (Ah) a fully charged cell or battery can deliver under specified conditions of discharge.

DC Direct Current

Static Switch: Synchronous Bypass Facility. The combination static (semiconductor) and electromagnetic switch that bypasses the UPS to connect the critical load to the bypass power source.

System Control: The control circuits that provides total UPS system control, alarm annunciation and instrumentation metering.

UPS: Uninterruptible Power System. An electrical device or system providing an interface between the mains power supply and sensitive loads (computer systems, instrumentation, etc.). The UPS supplies sinusoidal AC power free of disturbances and within strict amplitude and frequency tolerances. If input power is removed from the UPS, it will continue to supply the load without interruption. Refer AS62040.

Useful Life (battery): The time over which a battery can deliver a useful amount of power (normally defined as 80 percent or more of the battery’s capacity).

## Overview

The SDD brings out the functionalities that the Application Software shall perform. The Section 2 of this document briefs the Overall description about Product perspective, Product functions, User characteristics, Constraints, Assumptions & dependencies of the Application Software. The Section 3 of this document briefs the Specific Requirements like External Interface Requirements, Functional Requirements, Performance Requirements and Software system attributes. The Section 4 of this document provides the Traceability Matrix from System Requirements Specifications document to SDD document.

# references

(a) IEEE 12207.1 standard Volume 4 for Software Design Description

# decomposition description

In the subsequent sections, the preliminary design details of each of the modules shown in the Figure 1 are described.

## Module Decomposition

* initialisation()
* channel\_select(0)
* mainscheck() //mains high/low cut off checking
* disp() //display load on mains/invertor.
* bvcheck() //display battery low on screen.
* solarconditioncheck() //check the solar condition ok/not.

* display() //display the output voltage.
* display() //display the input voltage.
* display() //display the solar voltage.
* display() //display the batt. voltage.
* display() //display the output current.

## Concurrent process decomposition

Concurrent process decomposition shows the concurrency among the modules depending on mode of operation

## DATA DECOMPOSITION

NA

# DEPENDENCY DESCRIPTION

Dependency description shows the relationship among the modules.

## INTERMODULE DEPENDENCIES

NA

## INTERPROCESS DEPENDENCIES

NA

## DATA DEPENDENCIES

NA

# INTERFACE DESCRIPTION

NA

## MODULE INTERFACE

NA

## PROCESS INTERFACE

NA

# DETAILED DESIGN

This section describes the internal details of the Software Application.

## MODULE DETAILED DESIGN

UPS\_DESIGN\_REQ001

[Covers : UPS\_SRS\_REQ001]

Function Name: read\_ADC

**(a) Prototype:** int read\_ADC ()

**(b)** **Description**:

To read the analog equivalent digital counts.

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

Assign 0 to k;

Assign 0 to CLK;

for(i=0; i<12; i++)

{

k<<=1;

CLK =1;

CLK = 0;

if(P1&4) k |=1;

else k &= ~1;

}

CS=1;

return k&=0xfff;

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ002

[Covers : UPS\_SRS\_REQ002, UPS\_SRS\_REQ015]

Function Name: bvcheck

**(a) Prototype:** void bvcheck ()

**(b)** **Description**:

To check battery voltage.

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

if(bv<2149) //less than 10.5v it will show battery low

{

bvf = 0;

display(str9 ,LINE2);

t1 = 0;

}

if(bv>2252)

{

bvf = 1;

t1 = 1;

}

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ003

[Covers : UPS\_SRS\_REQ003]

Function Name: solarconditioncheck

**(a) Prototype:** void solarconditioncheck ()

**(b)** **Description**:

To check solar charger, solar is greater than 13v and batt.less than 11v, mains charger OFF, solar charger OFF

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

if(orc==1)

{

if(sv>2662 & bv<2252) //solar is greater than 13v and batt.less than 11v.

{

cc = 0; //solar charger ON

mcr = 1;

}

}

else

{

if(sv>2662 & bv<2457) //solar is greater than 13v and batt.less than 11v.

{

cc = 0; //solar charger ON

mcr = 1;

}

} //mains charger OFF

if(sv>2662 & bv>2968)

{

cc = 1; //solar charger OFF

mcr = 1; //mains charger OFF

}

// if(sv<2048 & (ipv>2000 & ipv<2700))

if(sv<2458 & (ipv>2000 & ipv<2700))

{

cc = 1; //solar charger OFF

mcr = 0; //mains charger ON

}

if(sv>2662) //solar is greater than 13v.

{

mcr = 1; //mains charger OFF

}

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ004

[Covers : UPS\_SRS\_REQ004]

Function Name: channel\_select

**(a) Prototype:** void channel\_select(char )

**(b)** **Description**:

Channel selection for serial adc.

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

char i,channel;

CS = 0;

if(n==0) channel= 0x0d;

else channel= 0x0f;

for(i=0; i<4; i++)

{

CLK =0;

if(channel&8) DIn =1;

else

DIn = 0;

CLK =1;

channel <<=1;

}

P1 |= 4;

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ005

[Covers : UPS\_SRS\_REQ005]

Function Name: timer0

**(a) Prototype:** void timer0()

**(b)** **Description**:

To check whether 10 sec completed after mains is normal

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

tcount++;

TH0 = 0x4b;

TL0 = 0xff;

if(tcount==200)

{

orc = 0;

tcount=0;

EA = 0;

TR0= 0;

}

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ006

[Covers : UPS\_SRS\_REQ006]

Function Name: mainscheck

**(a) Prototype:** void mainscheck()

**(b)** **Description**:

To check mains supply

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

if(!t2)

{

orc = 1;

mf = 1;

EA = 0;

TR0 = 0;

}

if(t2)

{

if(ipv<1600) //MAINS LOW CUT OFF less than 160Volts

{

orc = 1; //output relay OFF

mf = 1;

EA = 0;

TR0 = 0;

}

if(ipv>2000 & ipv<2500) //MAINS NORMAL less than 245V & greater than 200volts.

{

if(mf)

{

EA = 1;

TR0 = 1;

mf = 0;

}

}

if(ipv>2700) //MAINS HIGH CUT OFF greater than 270Volts.

{

orc = 1; //output relay OFF

mf = 1;

EA = 0;

TR0 = 0;

}

}

if(ipv<200)

ipv = 0;

if(opv<200)

opv = 0;

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ007

[Covers : UPS\_SRS\_REQ007, UPS\_SRS\_REQ008, UPS\_SRS\_REQ009, UPS\_SRS\_REQ010, UPS\_SRS\_REQ011, UPS\_SRS\_REQ013, UPS\_SRS\_REQ014]

Function Name: display\_on\_screen

**(a) Prototype:** void display\_on\_screen (int,bit)

**(b)** **Description**:

To display the values on the screen.

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | Num | Int | To store intermediate values | Input/Output |

**(e) Algorithm:**

BEGIN

num=c/10;

If s is equal to 1

sprintf(res,"%3.1f",(float)num/100);

Else

sprintf(res,"%d",num);

display(res,0x8A);

END

**(f) Data Structures**

NIL

UPS\_DESIGN\_REQ008

Function Name: interrupt\_subroutine

**(a) Prototype:** interrupt\_subroutine ()

**(b)** **Description**:

interrupt\_subroutine.

**(c) Input/Output Data Elements:**

NIL

**(d) Local data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Name** | **Type** | **Description** | **Remarks** |
| 1 | K | Int | To store intermediate values | Input/Output |
| 2 | I | Char | Counter | Input/Output |

**(e) Algorithm:**

BEGIN

Assign 0 to k;

Assign 0 to CLK;

for(i=0; i<12; i++)

{

k<<=1;

CLK =1;

CLK = 0;

if(P1&4) k |=1;

else k &= ~1;

}

CS=1;

return k&=0xfff;

END

**(f) Data Structures**

NIL

**Macro Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.No. | Macro name | Expansion | Value |
| 1 | CNT |  | 10 |
| 2 | TIMER |  | 100 |

**Global variables**

The table below lists out all the Global Variables, which are accessible to all the functions.

| **Sl.No** | **Name** | **Type** | **Description** |
| --- | --- | --- | --- |
| 1 | Sample\_Var1 | enum | used to identify message ID’s. |
| 2 | Sample\_Var2 | HANDLE | Used to store the handle of the COM port. |