Software User Manual

JMM-5000
PC/104-Plus DC/DC Power Supply

Revision A.0 MAY 2016

Revision	Date	Comment
A.0	02/05/2016	Initial release

FOR TECHNICAL SUPPORT PLEASE CONTACT:

support@diamondsystems.com

Diamond Systems Corporation
555 Ellis Street
Mountain View, CA 94043 USA
Tel 1-650-810-2500
Fax 1-650-810-2525

www.diamondsystems.com

CONTENTS

1. INTRODUCTION

This user manual contains all essential information about JMM-5000 demo application, programming guidelines and usage instructions. This manual also includes the Driver API descriptions with usage examples.

Jupiter-MM-5000 is a family of PC/104-Plus and PC/104 DC/DC power supplies with wide input range and high output power. The nominal output power is 5VDC @ 100 Watts at 25oC (5VDC @ 20A), with additional power provided on 12V rails. The output power may be de-rated at the high end of the temperature range based on the cooling capacity of the thermal solution, as well as de-rated at low input voltages based on the current capacity of the input circuit and connector.

Key Features

- Up to 196W total output power at 25^{o}C
- +5VDC at 20A max output (100W)
- +12VDC at 8 A max output (96W)
- Optional +3.3VDC at 5A maximum (minimum order quantities apply)
- Extreme load stability: 0.35% maximum output voltage droop at 5V output, 0-20A load, VIN = 12V, TA = 25ºC
- Extremely low ripple: 12mV peak-to-peak ripple at 5V output, 0-20A load, VIN = 12V, TA = 25ºC
- High efficiency: 92-94% at 5V output, 0-20A load, VIN = 12V, TA = 25ºC
- Excellent transient load response: +/-72mV at 5V output, 25-75% load step, 2.5A/usec ramp rate, VIN =
- 24V, TA = 25ºC
- Extreme temperature stability: +/-0.5% at 5V output, 10A load, VIN = 24V, TA = -40ºC to +85ºC
- Input protection circuit protects from over/under voltage, reverse polarity, surges, transients, reflected noise
- Output current limit and short circuit protection
- Wide input range: +7 to +34VDC
- Remote on/off control
- Heat sink or heat spreader cooling solutions
- Dual input option with auto-cutover (minimum order quantities apply)
- PC/104 form factor: 3.550” x 3.775” (90mm x 96mm)
- PC/104 and PC/104-Plus bus connector options
- -40oC to +85oC (-40°F to +185°F) operating temperature

2. HARDWARE OVERVIEW

2.1 Description

Jupiter-MM-5000 high-efficiency, high-precision power supplies consist of a PC/104 form factor module with complete DC-DC voltage regulator circuitry, integrated thermal solution, detachable screw terminal block I/O connections, and PC/104 bus connectors. The wide input voltage range of 7 to 34VDC is compatible with industry standard 12V, 24V, and 28V inputs.

The Jupiter-MM-5000 uses a state-of-the-art design with the latest generation high efficiency components. It delivers efficiency as high as 95 percent, reducing input power requirements as well as heat generation.

Jupiter-MM-5000 was engineered for rugged applications such as automotive or on-vehicle. Extended temperature operation of -40°C to +85°C is tested and guaranteed. Low-profile, surface mount components reduce susceptibility to shock and vibration. Both a low profile heat sink and heat spreader cooling options are available. I/O connections are made with locking screw terminal blocks for the highest degree of ruggedness.

2.2 Specification

Input	
Input Voltage	7 - 34VDC
Input protection	Over / under voltage, reverse polarity, surges, transients, reflected noise
Output	
Output Voltage / Current	+5V at 20A maximum +12V at 8A maximum +3.3V at 5A maximum optional
Output Protection	Current limit and short circuit protection
Load Regulation	±0.8%, Vmin to Vmax, 0-100% load on all outputs, 40°C to +85°C 0.35% maximum output voltage droop at 5V output, 0-20A load, VIN = 12V, TA = 25°C
Output Ripple	44mV peak-to-peak maximum 12mV peak-to-peak at 5V output, 0-20A load, VIN = 12V, TA = 25°C
Efficiency	92-94% at 5V output, 0-20A load, VIN = 12V, TA = 25°C
Transient Load Response	44mV peak-to-peak maximum
Temperature Stability	+/-0.5% at 5V output, 10A load, VIN = 24V, TA = -40°C to 85°C
General	
On /Off	Remote on/off logic input
Dimensions	PC/104 form factor 3.55 " x 3.775" (90mm x 96mm) not including screw terminals
Bus Connection Options	16-bit stackthrough ISA bus 32-bit PCI bus
Operating Temperature	-40°C to +85°C (-40°F to +185°F)
Operating Humidity	5% to 95% non-condensing
Weight	6.3oz (179g) with heat sink 8.1oz (230g) with heat spreader
RoHS	Compliant

3. GENERAL PROGRAMMING GUIDELINES

3.1 Initialization and exit function calls

All demo applications begin with the following functions and should be called in a sequence to initialize the Driver and the JMM-5000. The following function should be called prior to any other JMM5000 board specific functions.

- JMM5000_Init ( ) : This function initializes the Driver

At the termination of the demo application the user should call JMM5000_Close () function to close the file handler which is opened in JMM5000_Init () function.

These function calls are important in initializing and to free the resources used by the driver. Following is an example of the framework for an application using the driver:

```
#include "JMM5000.h"

int main()
{
      JMM5000_Init ();

      /* Application code goes here */

      JMM5000_Close ();

      return 0;
}
```

3.2 Error Handling

Driver functions provide a basic error handling mechanism that stores the last reported error in the driver. If the application is not behaving properly, last error can be checked by calling the function `DSCGetLastError` (). This function takes an ERRPARAMS structure pointer as its argument.

Nearly all of the available functions in the Universal Driver API return a BYTE value upon completion. This value represents an error code that will inform the user as to whether or not the function call was successful. User should always check if the result returns a DE_NONE value (signifying that no errors were reported), as the code below illustrates:

```
ERRORPARAMS error;

if(JMM5000_PowerControl(data) != DE_NONE)
{
      DSCGetLastError(&error);
      printf("JMM5000 PowerControl error : %s",error.error_string);
      return;
}
```

Anytime a function is not successfully executed, an error code other than DE_NONE will be generated and the current API function will terminate. The function `DSCGetLastError` () provides a description of the error that occurred.

4. JMM5000 API DESCRIPTION

4.1 JMM5000_Init

Function Definition

BYTE JMM5000_Init ()

Function Description

This function opens handle to communicate with JMM5000 function then initializes the PIC and LTC2974. This function must be called prior calling any other JMM5000 functions.

Return Value

Error code or 0.

Function Parameters

Name	Description
None	

Usage Example

To opens handle to communicate with JMM5000,
JMM5000_Init ();

4.2 JMM5000_Close

Function Definition
BYTE JMM5000_Close()

Function Description

This function closes the port, hence ends the communication with the PIC.

Return Value
Error code or 0.

Function Parameters

Name	Description
None	

Usage Example

To close the handle to communicate with JMM5000,

JMM5000_Close ();

4.3 JMM5000_PowerControl

Function Definition

BYTE JMM5000_PowerControl(**unsigned char data**)

Function Description

This function turns on or off the selected power supply outputs as indicated by the data byte. 1 turns on the indicated supply, a 0 turns it off.

Function Parameters

Name	Description
data	unsigned char Bit 0: 5V output phase 1, 5V output phase 2. Bit 1: Not Used Bit 2: 12V output Bit 3: 3.3V output

Return Value

Error code or 0.

Usage Example

To turn on 12v,5V power supply outputs.
data=0x05;
JMM5000_PowerControl(data);

4.4 JMM5000_PowerCycleControl

Function Definition
BYTE JMM5000_PowerCycleControl (unsigned char data)

Function Description

This function selects the outputs that will be controlled with the power cycle timer. 1 = enable power output, 0 = do not enable power output.

Function Parameters

Name	Description		
data	unsigned char	Bit 0:	5V output phase 1, 5V output phase 2.
		Bit 1:	Not Used
		Bit 2:	12V output
		Bit 3:	3.3V output

Return Value
Error code or 0.

Usage Example
To control the power cycle of12v and 5V,

data=0x05;
JMM5000_PowerCycleControl (data);

4.5 JMM5000_PowerOnTime

Function Definition

BYTE JMM5000_PowerOnTime(int data)

Function Description

This function sets the length of time the selected outputs will remain on each time a power-on cycle occurs. The data value indicates the number of seconds of the power-on time. Maximum on time is 65535 seconds or approximately 18.2 hours.

Function Parameters

Name	Description
data	int 0-65535

Return Value

Error code or 0.

Usage Example

To configure power on time as 5 seconds,

data=5;

JMM5000_PowerOnTime(data);

4.6 JMM5000_PowerCycleTime

Function Definition
BYTE JMM5000_PowerCycleTime (int data)

Function Description

This function sets the period of the power-on timer. The PIC will turn on the selected outputs at the time interval indicated by the value written with this function. The value is in 2-second increments; the maximum cycle time is 65535 x 2 seconds = approximately 36 hours.

Function Parameters

Name	Description	
data	int	0-131070

Return Value
Error code or 0.

Usage Example
To configure power cycle time as 10 seconds,

data=5;

JMM5000_ PowerCycleTime (data);

4.7 JMM5000_OutputStatus

Function Definition
BYTE JMM5000_OutputStatus(JMM5000RegulatorValues* parameter)

Function Description
This function reads and provides the user with voltage, current, power parameters of 3.3V, 12V, 5V.

Function Parameters

Name	Description
parameter	float voltage[3] , volatge[0]-12V volatge[1]-5V volatge[2]-3.3VV float current[3], current[0] -12V current[1]-5V current[2]-3.3V float power[3], power[0]-12V power[1]-5V power[2]-3.3V

Return Value
Error code or 0.

Usage Example
To read power parameters of 3.3v.

```
JMM5000RegulatorValues  parameter;
JMM5000_OutputStatus( &parameter );

 printf("The power parameter of 12v is %f ",parameter->Power[2]);
```

4.8 JMM5000_ControlWrite

Function Definition
BYTE JMM5000_ControlWrite(int Command,int PageEnable, int page,int data)

Function Description
This function enables the user to pass write command and data to the LTC2974.

Return Value
Error code or 0.

Function Parameters

Name	Description
Command	int 0x00 -0xFF , the commands are specific to LTC2974 refer LTC data Sheet
PageEnable	int 1 - Command requires page no, 0 – Command does not requires page Number
Page	int 0 – 12V page, 1 -5V page, 2 - 3.3V page if page is enabled
Data	int 0x0000 -0xFFFF , the data are specific to LTC2974 refer LTC data sheet

Usage Example

To write data 0xD380 into the command 0x57(VIN_OV_WARN_FAULT),

```
Command=0x57;
PageEnable=1;
page=0x01;
data=0xD380;

JMM5000_ControlWrite(Command, PageEnable, page, data);
```

4.9 JMM5000_ControlRead

Function Definition
BYTE JMM5000_ControlRead(int Command,int PageEnable, int page,int* data)

Function Description
This function enables the user to pass read command to LTC2974 and read data from the LTC2974.

Return Value
Error code or 0.

Function Parameters

Name	Description	
Command	int	0x00 -0xFF , the commands are specific to LTC2974 refer LTC data Sheet
PageEnable	int	1 - Command requires page no, 0 – Command does not requires page Number
Page	int	0 – 12V page, 1 -5V page 2 - 3.3V page if page is enabled

Usage Example

To read the command 0x57(VIN_OV_WARN_FAULT),

```
Command=0x57;
PageEnable=1;
page=0x01;

JMM5000_ControlRead(Command, PageEnable, page, &data);
printf("The data read is 0x%x",data);
```

4.10 JMM5000_ControlEEWrite

Function Definition
BYTE JMM5000_ControlEEWrite(int Command,int PageEnable, int page,int data)

Function Description
This function enables the user to write command and data to the EEPROM. This function updates RAM registers first then copies all the changes to EEPROM.

Return Value
Error code or 0.

Function Parameters

Name	Description
Command	int 0x00 -0xFF , the commands are specific to LTC2974 refer LTC data Sheet
PageEnable	int 1 - Command requires page no, 0 – Command does not requires page Number
Page	int 0 – 12V page, 1 -5V page 2 - 3.3V page if page is enabled
Data	int 0x0000 -0xFFFF , the commands are specific to LTC2974 refer LTC data sheet

Usage Example

To write 0xD380 into the command 0x57(VIN_OV_WARN_FAULT),

```
Command=0x57;
PageEnable=1;
page=0x01;
data=0xD380;

JMM5000_ControlEEWrite(Command, PageEnable, page, data);
```

4.11 JMM5000_ ControlEERead

Function Definition
BYTE JMM5000_ControlEERead(int Command,int PageEnable, int page,int* data)

Function Description
This function enables the user to read data from EEPROM. This function copies EEPROM to RAM and then reads from RAM.

Return Value
Error code or 0.

Function Parameters

Name	Description	
Command	int	0x00 -0xFF , the commands are specific to LTC2974 refer LTC data Sheet
PageEnable	int	1 - Command requires page no, 0 – Command does not requires page Number
Page	int	0 – 12V page, 1 -5V page 2 - 3.3V page if page is enabled

Usage Example

To read the command 0x57,

Command=0x57
PageEnable=1
page=0x01
JMM5000_ControlEERead (Command, PageEnable, page,& data);

printf("The data read is 0x%x",data);

4.12 JMM5000_ InputConfigStore

Function Definition
BYTE JMM5000_InputConfigStore(int ControlData, int data)

Function Description

This function is used to write data to the AD5144 quad digital potentiometer.

Return Value
Error code or 0.

Function Parameters

Name	Description
ControlData data	char Bits 7-4 = control data; bits 3-0 = address refer Appendix A char Bits 7-0 = digital potentiometer data, 8 bits refer Appendix A

Usage Example

To write data 0xd0 ,ControlData 0x10,

data=0xD0;
ControlData=0x10;
JMM5000_InputConfigStore(ControlData,data)

4.13 JMM5000_InputConfig

Function Definition

BYTE JMM5000_InputConfig(int Input, float Vlow, float Vhigh)

Function Description

This function sets the primary or secondary input voltage source range. This function updates the RDAC and EEPROM.

Return Value

Error code or 0.

Function Parameters

Name	Description	
Input	int	1-primary voltage, 2- Secondary voltage
Vlow	float	minimum value 9.00V and max value 34.00V
Vhigh	float	minimum value 9.00V and max value 34.00V

Usage Example

To set 9V as minimum, 30V as maximum voltage for secondary input range,

```
Input=2;
Vlow=9;
Vhigh=30;
JMM5000_InputConfig(Input, Vlow, Vhigh)
```

4.14 JMM5000_ResetControl

Function Definition
BYTE JMM5000_ResetControl()

Function Description
This command is used to reset the LTC2974.

Return Value
Error code or 0.

Function Parameters

Name	Description
None	

Usage Example

To reset the LTC2974,

JMM5000_ResetControl()

4.15 JMM5000_ResetPIC

Function Definition
BYTE JMM5000_ResetPIC()

Function Description
This function initializes the PIC microcontroller.

Return Value
Error code or 0.

Function Parameters

Name	Description
None	

Usage Example

To reset the PIC,

JMM5000_ ResetPIC();

4.16 JMM5000_LED

Function Definition
BYTE JMM5000_LED (int LED)

Function Description

This function controls the programmable LED on the JMM-5000.

Return Value
Error code or 0.

Function Parameters

Name	Description
LED	int 0-1, 0-LED Off, 1- LED On

Usage Example

To turn on the LED

```
LED=1;
JMM5000_LED (LED);
```

4.17 JMM5000_LEDStatus

Function Definition
BYTE JMM5000_LEDStatus(int *LED)

Function Description

This function read the LED status on the JMM-5000.

Return Value
Error code or 0.

Function Parameters

Name	Description
LED	int* holds LED status value

Usage Example

To read the LED,

```
int status=0;
JMM5000_LEDStatus ( & status);
printf("The LED status is %d", status);
```

4.18 JMM5000_Status

Function Definition
BYTE JMM5000_Status(int *data)

Function Description
This function returns 2 bytes firmware revision.

Return Value
Error code or 0.

Function Parameters

Name	Description
data	int* holds the status value

Usage Example

To read the status,

```
JMM5000_Status( &data );
printf("The LED status is 0x%x ",data);
```

APPENDIX A

Command Number	Control Bits[DB15:DB12]				Address Bits[DB11:DB8][1]				Data Bits[DB7:DB0][1]								Operation
	C3	C2	C1	C0	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	NOP: do nothing
1	0	0	0	1	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	Write contents of serial register data to RDAC
2	0	0	1	0	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	Write contents of serial register data to input register
3	0	0	1	1	X	A2	A1	A0	X	X	X	X	X	X	D1	D0	Read back contents D1 D0 Data 0 0 Input register 0 1 EEPROM 1 0 Control register 1 1 RDAC
4	0	1	0	0	A3	A2	A1	A0	X	X	X	X	X	X	X	1	Linear RDAC increment
5	0	1	0	0	A3	A2	A1	A0	X	X	X	X	X	X	X	0	Linear RDAC decrement
6	0	1	0	1	A3	A2	A1	A0	X	X	X	X	X	X	X	1	+6 dB RDAC increment
7	0	1	0	1	A3	A2	A1	A0	X	X	X	X	X	X	X	0	−6 dB RDAC decrement
8	0	1	1	0	A3	A2	A1	A0	X	X	X	X	X	X	X	X	Copy input register to RDAC (software LRDAC)
9	0	1	1	1	0	0	A1	A0	X	X	X	X	X	X	X	1	Copy RDAC register to EEPROM
10	0	1	1	1	0	0	A1	A0	X	X	X	X	X	X	X	0	Copy EEPROM into RDAC
11	1	0	0	0	0	0	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	Write contents of serial register data to EEPROM
12	1	0	0	1	A3	A2	A1	A0	1	X	X	X	X	X	X	D0	Top scale D0 = 0; normal mode D0 = 1; shutdown mode
13	1	0	0	1	A3	A2	A1	A0	0	X	X	X	X	X	X	D0	Bottom scale D0 = 1; enter D0 = 0; exit
14	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	Software reset
15	1	1	0	0	A3	A2	A1	A0	X	X	X	X	X	X	X	D0	Software shutdown D0 = 0; normal mode D0 = 1; device placed in shutdown mode