MANHATTAN PROGRAMMING GUIDE
IR38060/IR38062/IR38063/IR38064

IMPORTANT: THE IC MUST BE TRIMMED BEFORE ATTEMPTING TO PROGRAM USER AND MFR SECTIONS

There are several ways to program (configure) the Manhattan family of devices. Customers are advised to use methods #1 and #2 only. Method #3 is intended for Programming Station manufacturers.

TABLE 1: PROGRAMMING OPTIONS

	Programming Option	Hardware Required	Benefit	Typical Application	Documentation
#1	**Multi-Device Programmer (in PowIRCenter GUI)**	3-pin connection to IR USB005 Dongle	• Single Push-button via easy graphical interface • Program all IR parts on a board at once	Low volume programming when parts are already soldered onto a board	PowIRCenter User Guide
#2	**PMBus command**	Connector to PMBus	• Re-usable PMBus code that can be used for programming and run-time telemetry/updates • Simple programming command for USER section	Boot time on-board programming	This document, Section 1
#3	**Custom I2C code**	Connector to I2C bus	• Customizable for automated programming station using text-based config files for USER & MFR section	High volume programming station prior to assembly	This document, Section 2

For option #1, please refer to the PowIRCenter GUI User Guide.

Although, Manhattan has only 1 set of bus inputs (SCL/SDA), it responds to 2 separate addresses: one is used to accept PMBus protocol and the other is used to accept I2C protocol. Manhattan has default base PMBus address of 40h and I2C default base address of 10h. The value of the R_{ADDR} resistor offsets the PMBus and I2C base address to form the final addresses for communication.

Manhattan has a limited programmable user Non Volatile Memory (NVM), also known a Multiple times Programmable (MTP). Parts from the factory are typically programmed 1 time, so customers usually have 7 programmings available.

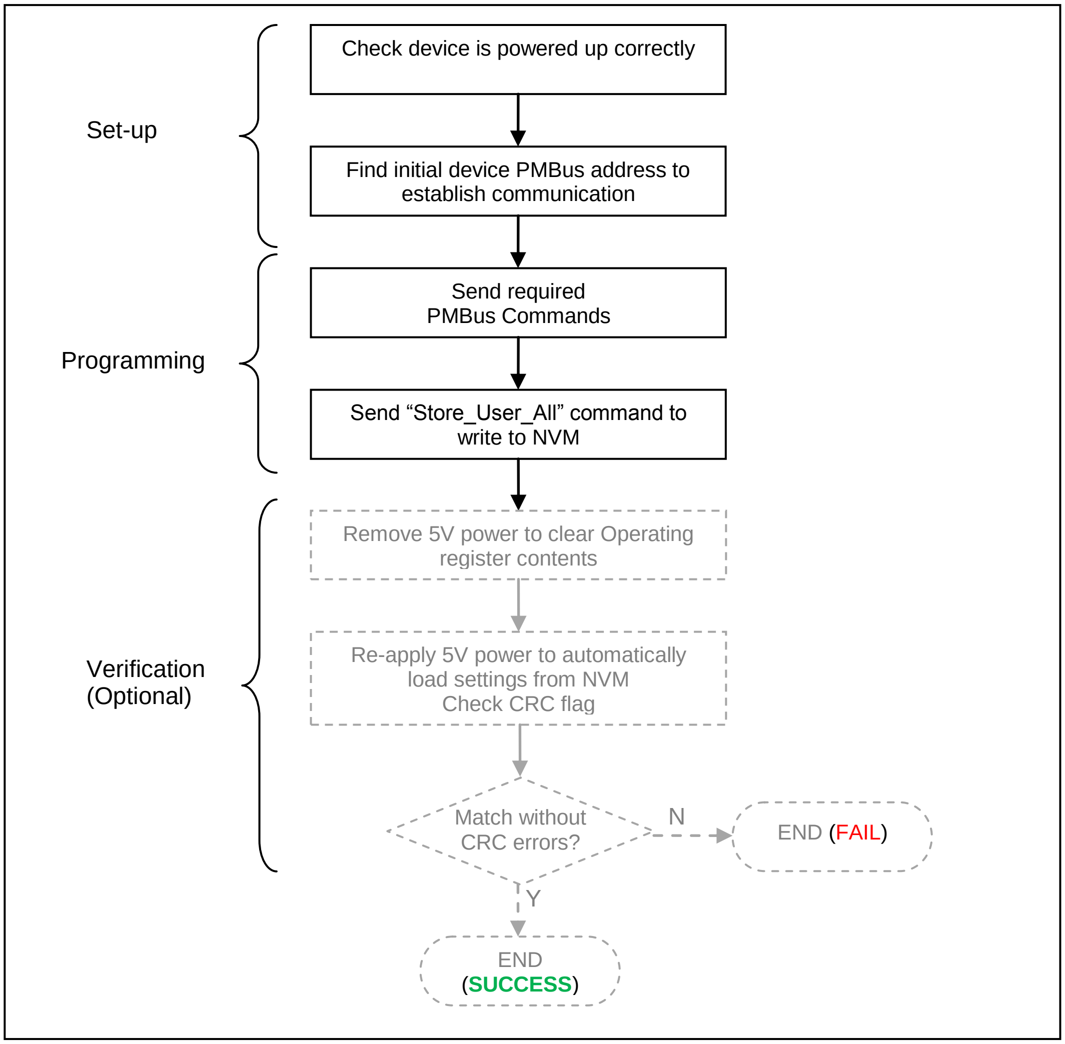
Section 1: Programming USER section by PMBus Command

Manhattan operates primarily by PMBus command and this is the recommended method for customers to configure parameters due to its simplicity. Figure 1 shows the PMBus programming flow.

Note, standard PMBus commands are used to set up parameters and write the NVM. Optionally, to access special functions such as the NVM count or the CRC flag, requires the use of manufacturer specific PMBus commands using the Process Call format:

MFR_READ_REG: use PMBus command D0h as shown in Figure 6
MFR_WRITE_REG use PMBus command D1h as shown in Figure 5

Figure 1: PMBus Programming Flow


Set-up
Programming
Verification
(Optional)
Check device is powered up correctly
Find initial device PMBus address to establish communication
Send required PMBus Commands
Send "Store_User_All" command to write to NVM
Remove 5V power to clear Operating register contents
Re-apply 5V power to automatically load settings from NVM
Check CRC flag
Match without CRC errors?
N
END (FAIL)
Y
END (SUCCESS)

Programming Procedure via PMBus

Use the PMBus read/write protocols in Figure 5 and Figure 6 to read and write registers. Refer to the "*Manhattan PMBus commandset*" for a detailed list of valid commands and formats.

Set-up & Check

1. Check Vcc is 5V.
2. Check P1V8 pin voltage is 1.8V +/- 5%.

Determine PMBus address

3. Find out what is the PMBus address of the device (if not known).
   a) Send an innocuous PMBus command such as CAPABILITY from address 08h to 0Bh and 0Dh to 77h until the device ACKs the command.
   b) Using the address obtained in step 3a), send the command DEVICE_ID and check if the data is 30h, 32h, 33h, 34h (IR38060/2/3/4 respectively). If yes, the device is a Manhattan device. Record this PMBus address and go to step 4. If not, it is either the I2C address of the Manhattan or another device on the bus, thus continue sending next address in step 3a).

Send required PMBus commands

3. Send any required PMBus commands (e.g. VOUT_COMMAND, FREQUENCY_SWITCH …)

Write New Configuration into MTP Memory

4. Send the STORE_USER_ALL command

Verification (Optional)

5. Remove the 5V power to clear the operating memory.
6. Re-apply 5V power. The device will initialize and transfer its NVM contents into the operating memory
7. Read the CRC flags by sending MFR_READ_REG with Register-Address=97h (refer to Figure 6)
8. Check the CRC error flag 97h[2:0] = 000 (successful) – see Table 10.

Section 2: Programming by I2C Custom Code

MTP Register Space

There are three MTP sections in each device, namely the trim section, user section, and manufacturer section. Each section can be programmed a finite number of times as shown in table 2. Registers are provided to indicate the number of MTP programming times that are left for each section. The trim section must be programmed first before attempting to program the other two sections. Devices are always trimmed at the factory and should **never** be changed by users. *Throughout this document, the small "h" after a number indicates that the number is displayed in "hex" format.*

TABLE 2: MTP SECTION ADDRESS RANGES

Section	Register Address Range	Max # of Programming Attempts
Trim	00h – 1Dh	3
User	20h – 72h*	9
Manufacturer (MFR)	80h – 82h	3

***In the User section, never write registers 6Eh, 6Fh, 71h, 72h. Best practice is to write only registers 20h-6Ch.**

TABLE 3: MTP PROGRAMMING TIMES LEFT REGISTER (PTR)

Section	Register Address
Trim	98h[5:3]
Manufacturer (MFR)	98h[2:0]
User	99h[3:0]

TABLE 4: PTR VALUE DEFINITIONS FOR TRIM AND MANUFACTURER SECTION

98h[5:3] or 98h[2:0]	Remaining Programming Times	Set Next Programming Pointer to
7	3	0
0	2	1
1	1	2
2 - 6	0	none left

TABLE 5: PTR VALUE DEFINITIONS FOR USER SECTION

99h[3:0]	Remaining Programming Times	Set Next Programming Pointer to
15	9	0
0	8	1
1	7	2
2	6	3
3	5	4
4	4	5
5	3	6
6	2	7
7	1	8
8 - 14	0	none left

MTP Programming commands

The MTP read/write Command register is located at address B0h, which also serves as a "Return" register for the results of the command. After the write command is issued, the command register should be polled periodically until B0[7:5] is changed to 000 ("IDLE" state - see Table 8) *OR* the worst case programming time is exceeded.

Table 6: User Write Command Structure (Register B0h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit1	Bit0
0	1	0	0	Next Programming Pointer 0 - 8			

Table 7: Manufacturer Write Command Structure (Register B0h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit1	Bit0
0	1	0	1	1	Next Programming Pointer 0 - 3		

Table 8: "Return" IDLE State Structure (Register B0h)

Bit[7:5]	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
000 = operation complete XXX = operation still processing	Bit[7:5] from MTP write/load command			0=success 1=fail	0

e.g. returns 08h for if user/mfr write is successful, returns 0Ah if user/mfr write fails, returns 04h if load command is successful.

Table 9: LOAD MTP to Operating Register Command Structure (Register B0h)

Bit[7:5]	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
001	00000				

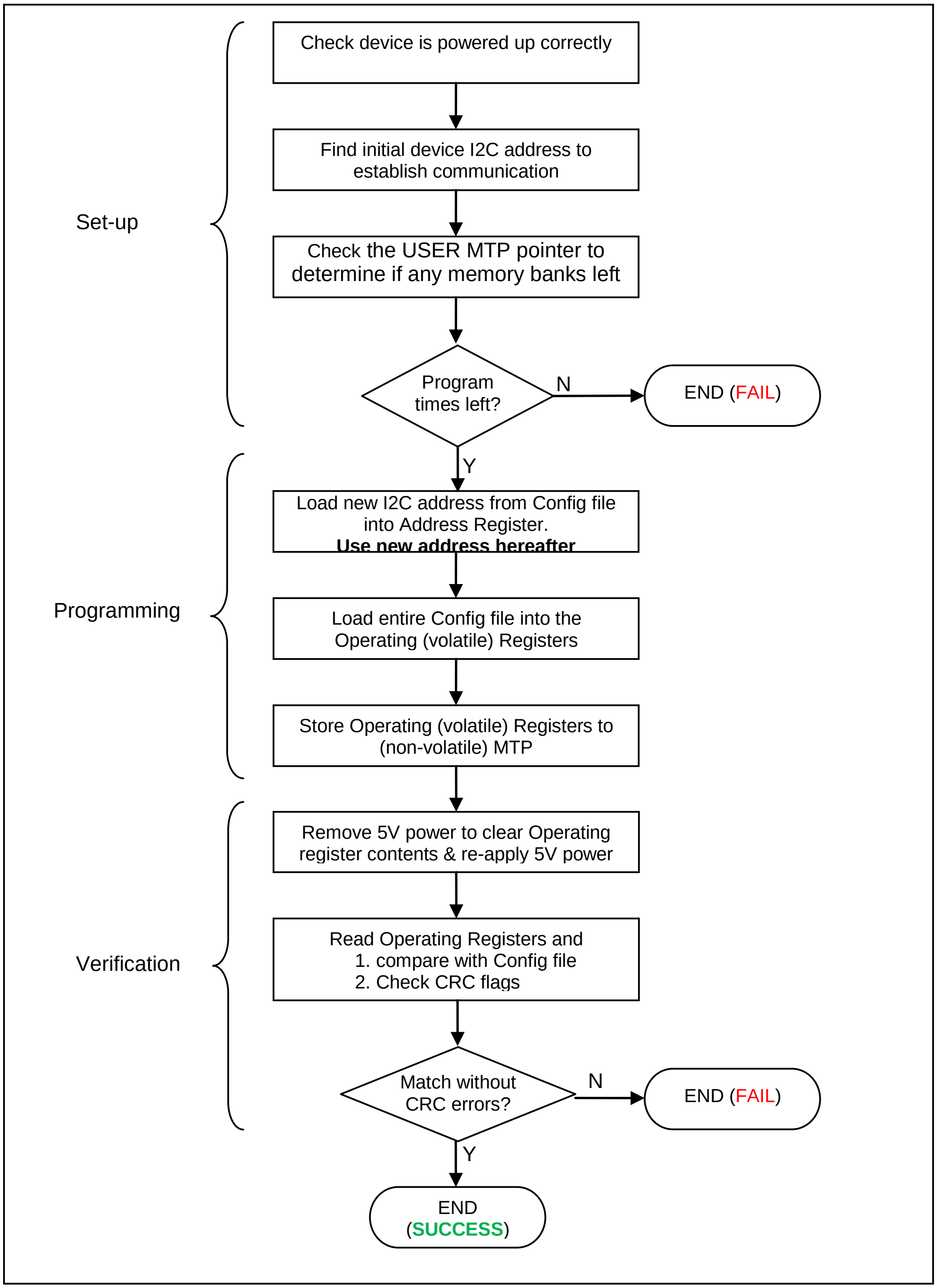
Table 10: CRC Error Flags (Register 97h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit1	Bit 0
xxxxx					Trim CRC Flag 0 = no error 1 = error	User CRC Flag 0 = no error 1 = error	MFR CRC Flag 0 = no error 1 = error

Programming Flow

Figure 2 provides an overview of the programming flow. A verification of the "store" is highly recommended.

Figure 2: Programming Flow


Set-up
Check device is powered up correctly
Find initial device I2C address to establish communication
Check the USER MTP pointer to determine if any memory banks left
Program times left?
N
END (FAIL)
Y
Programming
Load new I2C address from Config file into Address Register.
Use new address hereafter
Load entire Config file into the Operating (volatile) Registers
Store Operating (volatile) Registers to (non-volatile) MTP
Verification
Remove 5V power to clear Operating register contents & re-apply 5V power
Read Operating Registers and
1. compare with Config file
2. Check CRC flags
Match without CRC errors?
N
END (FAIL)
Y
END (SUCCESS)

MTP Programming Procedure via I2C

Use the I2C read/write protocols in Figure 4 to read and write registers.

Set-up & Check

1. Power on Vcc and Vin to 5V.
2. Check P1V8 pin voltage is 1.8V +/- 5%.

Determine current I2C address & Programming times left

3. Find out what is the i2c address of the device.
   c) Send 7-bit i2c address from 08h to 0Bh and 0Dh to 77h until the device ACKs the address.
   d) Read register 16h (IC_DEVICE_ID) and check if the data is 30h, 32h, 33h, 34h (IR38060/2/3/4 respectively). If yes, the device is a Manhattan device. Record this i2c address and go to step 4. If not, continue sending next address in step a).
4. To determine if there are any programming times left, use the i2c address to read MTP pointer register 99h (user section).
   - *Section_ptr=reg 99[3:0]h*
   - *If section_ptr>=8, quit*
   - *else if section_ptr= 15, new_section_ptr=0*
   - *else new_section_ptr=section_ptr+1*

Determine new I2C & PMBus addresses and Write new Configuration data into Working Register

5. Read the value of register 0x9B[3:0] which contains the detected address offset and save this for later use.
6. Read register 20h (PMBus address) in the Config file and write the config file data to register 20h only – note this may change the PMBus address
7. Read register 21h (i2c address) in the Config file and write the config file data to register 21h only – note this may change the i2c address!
8. Record the values sent to registers 20h and 21h which determine the new PMBus and i2C base addresses.
9. Since the new configuration may change the i2c register data at 21h, re-calculate the new i2c address as 21[6:0]h (base) + 9B[3:0]h (offset) and the new pmbus address as 20[6:0]h (base) + 9B[3:0]h (offset)
10. Start using the new i2c and pmbus address for further read/write communications.
11. Read the entire config file and Write all USER section config file data to the device's operating register. **Note, never write registers 6Eh, 6F, 71h, 72h. Best practice is write only registers 20h-6Ch (i.e do not write registers 6Dh-72h).**

Write New Configuration into MTP Memory

12. Enable the MTP programming clock by setting register 80h to 01h.
13. Write a user section programming command based on Table 5 i.e set register B0h = (*40h + new_section_ptr*)
14. Wait 50ms.
15. Read register B0h and check if register B0h = 08h. If yes, the programming is successful; if not, time out if >200ms has elapsed otherwise go back to step 14. After 200ms, the programming is considered a failure.

Verification

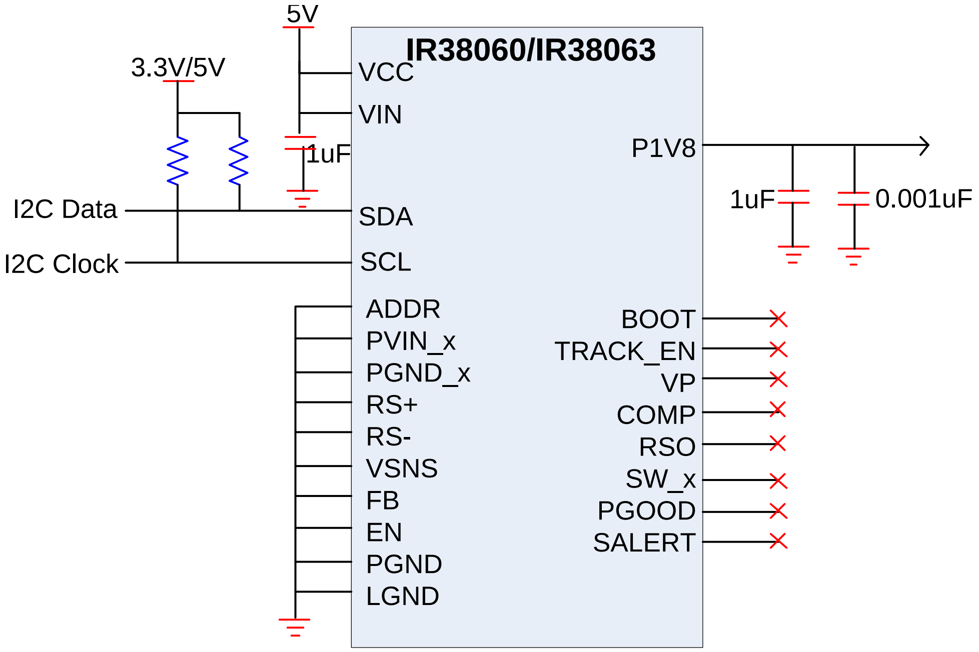
16. Remove the 5V power to clear the operating memory.
17. Re-apply 5V power. The device will initialize and transfer its NVM contents into the operating memory
18. For the User section (reg 20h-6Ch), read each register and compare contents with the Config file
19. If all register contents match and CRC error flag 0x97[2:0] = 000 – see Table 10. Then the programming is successful.

Note:

1. The quoted i2c address is 7-bit and becomes 8-bit by appending the R/W bit the LSB
2. Only the User section should be programmed. The Trim and MFR section should not be changed by customers.

I/O Terminations for a Programming Station

Figure 3: I/O Terminations



P1V8 = cap to GND (will self-bias to 1.8V)
Vin/Vcc = 5V. A better option is to separate Vin and Vcc and apply 12Vonly to Vin. Vcc (with 1uF cap to GND) will self-bias to ~5V.

I2C Read/Write Protocol

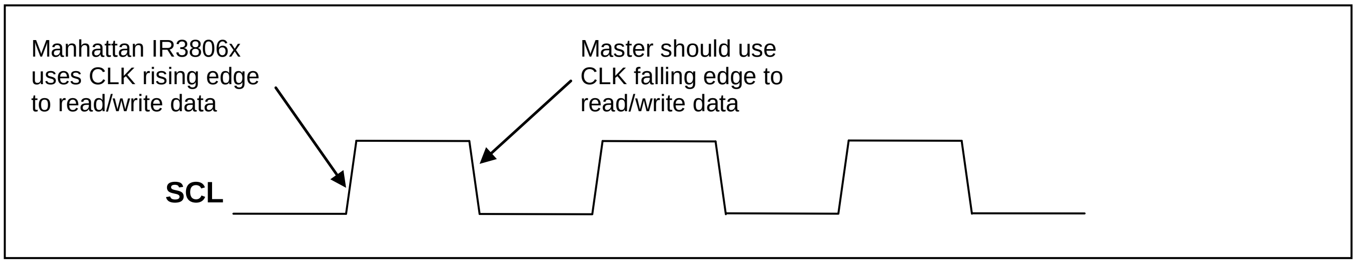
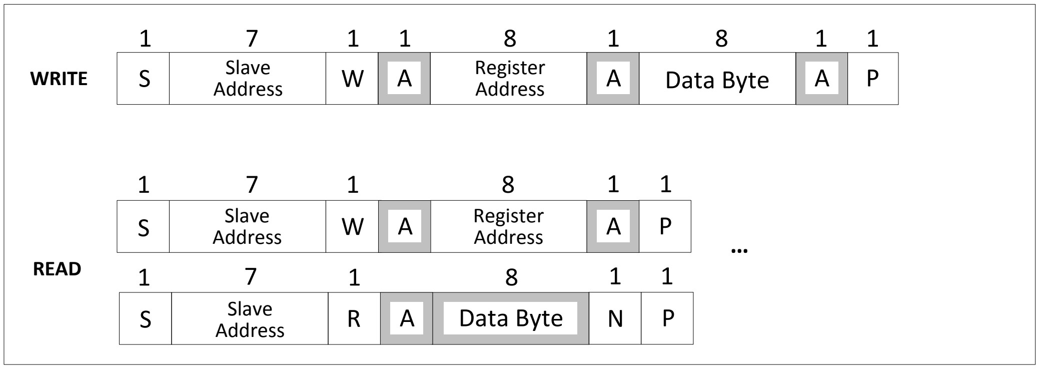

Manhattan IR3806x uses CLK rising edge to read/write data
Master should use CLK falling edge to read/write data
SCL

Figure 4: I2C Protocols to read or write a register



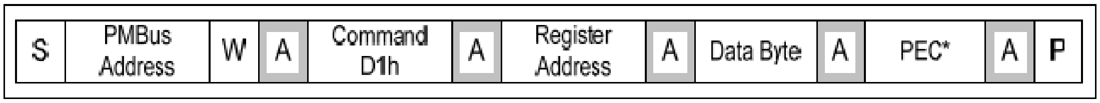
	1	7	1	1	8	1	8	1	1
WRITE	S	Slave Address	W	A	Register Address	A	Data Byte	A	P

	1	7	1		8	1	1	
READ	S	Slave Address	W	A	Register Address	A	P	…
	1	7	1		8	1	1	
	S	Slave Address	R	A	Data Byte	N	P	

S: Start Condition
A: Acknowledge (0')
N: Not Acknowledge (1')
Sr: Repeated Start Condition
P: Stop Condition
R: Read (1')
W: Write (0')
PEC: Packet Error Checking
***:** Present if PEC is enabled
□ : Master to Slave
▨ : Slave to Master

PMBus Commands to Read/Write registers

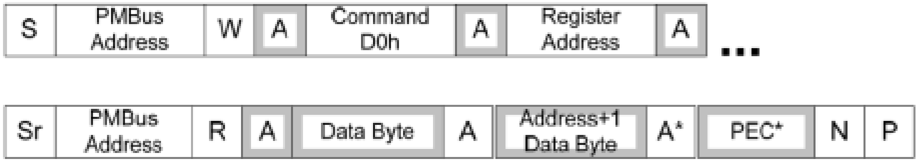
Figure 5: PMBus Protocol to Write 1 byte to a register



S	PMBus Address	W	A	Command D1h	A	Register Address	A	Data Byte	A	PEC*	A	P

*PEC is optional

Figure 6: PMBus Protocol to Read 2 bytes from a register



S	PMBus Address	W	A	Command D0h	A	Register Address	A	…		
Sr	PMBus Address	R	A	Data Byte	A	Address+1 Data Byte	A*	PEC*	N	P

*PEC is optional

The first byte returned is the contents of the specified register address. The 2nd byte returned is the contents of register address + 1.

Appendix A: The Configuration File

The program that determines IR controller operation is called a ‘configuration file’ (also called ‘Config File’).
It is a three-column, 134 line, space delimited text document as shown below.

CONFIG FILE LINES 1-134		
RegAddr	ConfigData	ConfigMask
10	25	FF
11	03	FF
12	AA	FF
13	80	FF
14	C0	FF
15	00	FF
16	00	FF
17	00	FF
18	00	FF
19	E4	FF
---	---	--
---	---	---

The first column contains the register address, the second column contains the register data, and the third column contains the register mask. Each line of the file is provided in ascending order of the register address, ranging from 10h to 83Fh. Data is read line by line. For example, row 1 indicates that data-value 25h is targeted for register address 10h with a mask of FFh. As seen from this example, the register address and data fields are self explanatory. However, the register mask column requires some explanation.

If the mask bit is zero, the corresponding register bit should not be verified after programming. If the mask bit is one, the corresponding bit must match the value from the config file after programming.

Data and specifications subject to change without notice.

International Rectifier

IR WORLD HEADQUARTERS: 233 Kansas St., El Segundo, California 90245, USA Tel: (310) 252-7105
TAC Fax: (310) 252-7903
Visit us at www.irf.com for sales contact information.
www.irf.com