## SALEM PROGRAMMING GUIDE for IR35201/IR35203/IR35204/IR35217/ASP1405I

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

## MTP (MULTIPLE-TIME PROGRAMMABLE) PROGRAMMING OVERVIEW

There are three MTP sections in each device, namely the trim section, user section, and manufacturer section. Trim and manufacturer section can be programmed up to three times, while the user section can be programmed up to seven times. All parts are programmed by the factory so the typical number of writes remaining for customer use is 2 in the trim section, 6 in the USER section, and 2 in the MFR section. The minimum number of writes remaining for customer use is 1 in the trim section, 5 in the USER section, and 1 in the MFR section. 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. Typically, devices are trimmed at the factory and shouldn't be changed by users. *Throughout this document, "h"or "H" after a number indicates that the number is displayed in "hex" format.*

**TABLE 1: MTP SECTION ADDRESS RANGES**

| Section | Address Range | Max # of Programming Attempts |
|---|---|---|
| Trim | 00h – 23h | 3 |
| User | 24h – 96h | 7 |
| Manufacturer | 98h – A5h | 3 |

**TABLE 2: MTP PROGRAMMING TIMES LEFT REGISTER (PTR)**

| Section | Register Address |
|---|---|
| Trim | B6h[5:3] |
| Manufacturer | B6h[2:0] |
| User | B8h[3:0] |

**TABLE 3: PTR VALUE DEFINITIONS FOR TRIM AND MANUFACTURER SECTION**

| B6h[5:3] or B6h[2:0] | Remaining Programming Times | Next Programming Pointer |
|---|---|---|
| 7 | 3 | 0 |
| 0 | 2 | 1 |
| 1 | 1 | 2 |
| 2 - 6 | 0 | none left |

**TABLE 4: PTR VALUE DEFINITIONS FOR USER SECTION**

| B8h[3:0] | Remaining Programming Times | Next Programming Pointer |
|---|---|---|
| 15 | 7 | 0 |
| 0 | 6 | 1 |
| 1 | 5 | 2 |
| 2 | 4 | 3 |
| 3 | 3 | 4 |
| 4 | 2 | 5 |
| 5 | 1 | 6 |
| 6 - 14 | 0 | none left |

## DEVICE IO TERMINATIONS

Serveral IO pins of the Salem device must be set correctly as shown in Figure 1. Other IO pins should be terminated as recommended in Figure 2, 3, and 4.

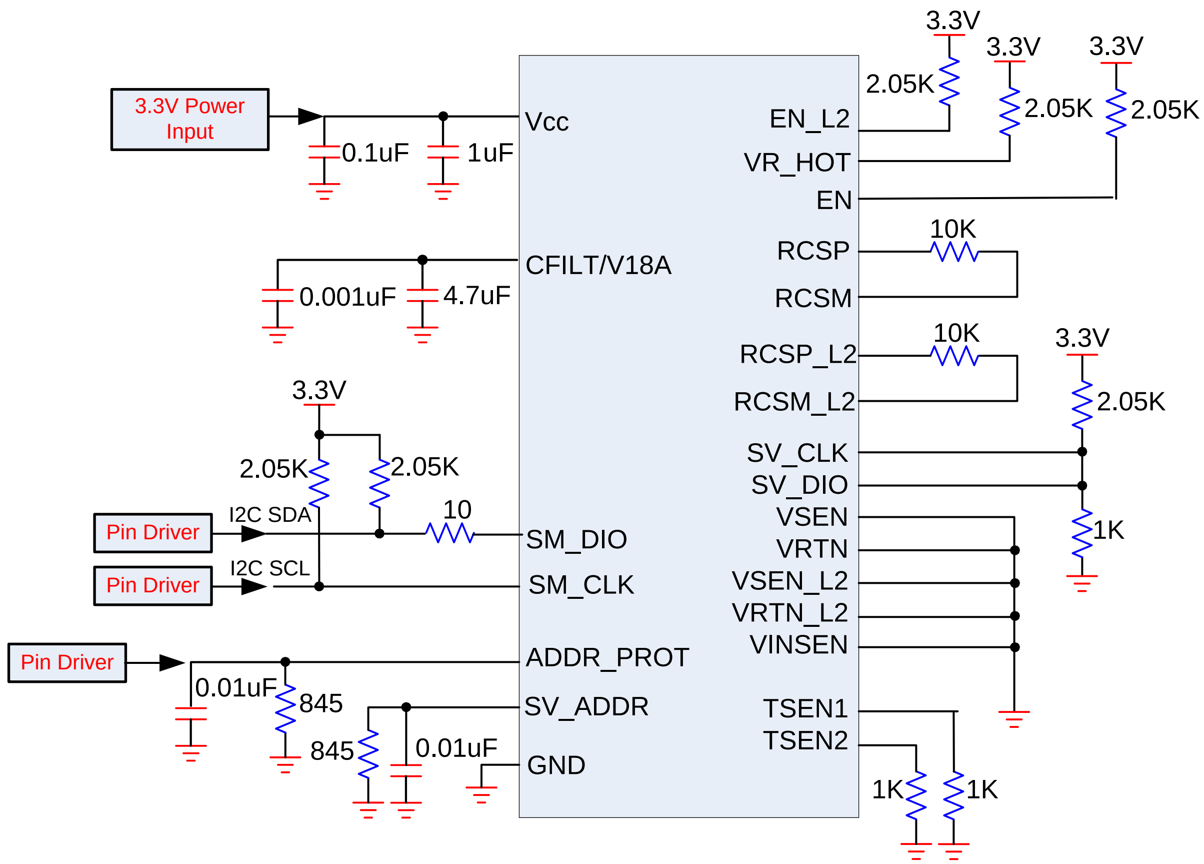


**Figure 1. Essential IO Terminations for Programming**

**Note:**

1. All pins not mentioned in the IO termination diagram can be floating.
2. CFILT/V18A is the internal 1.8V supply. Presence of 1.8V +/- 5% is an indication that the core circuitry is functioning properly..

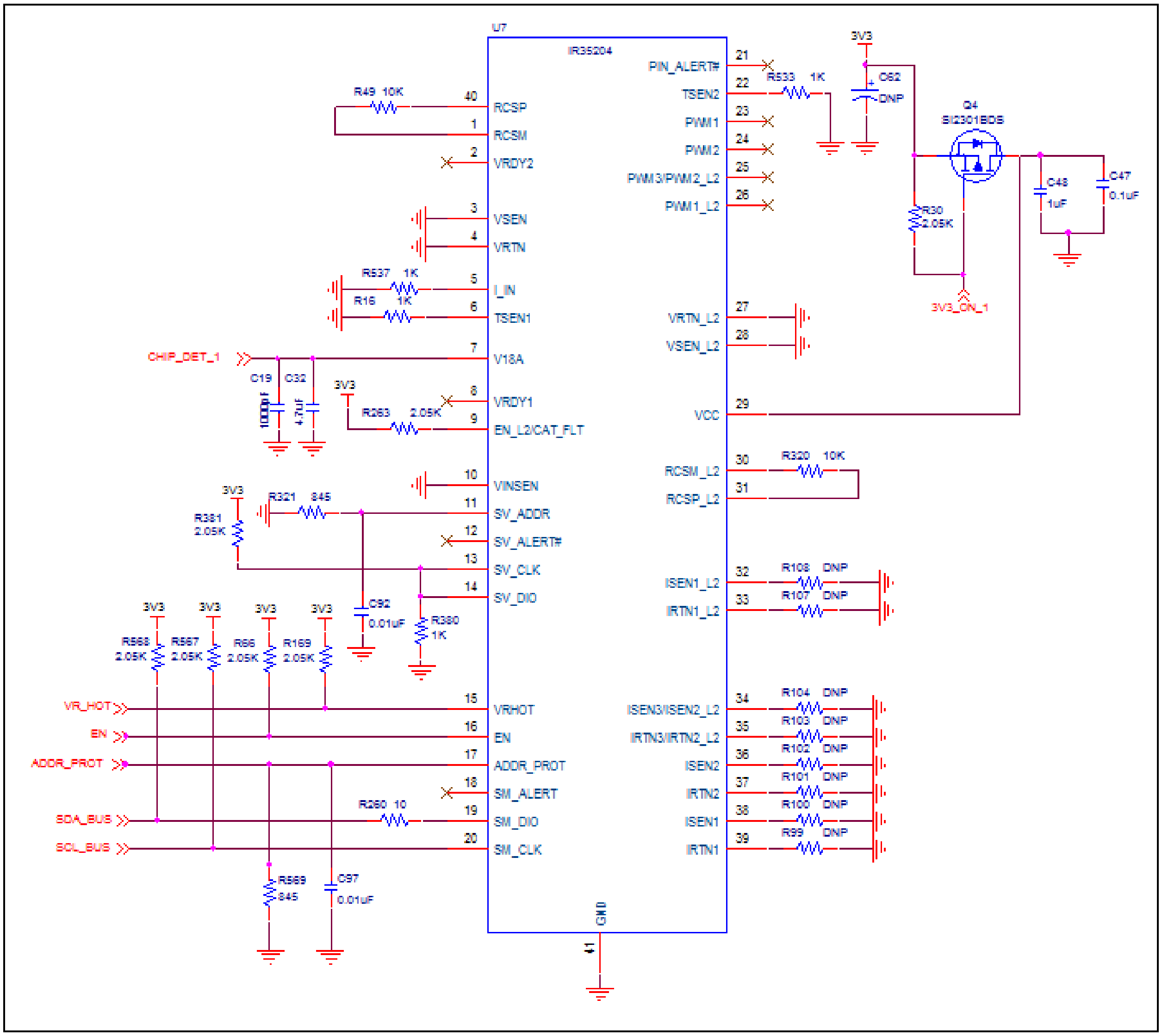


**Figure 2. IR35204 Pin Terminations**

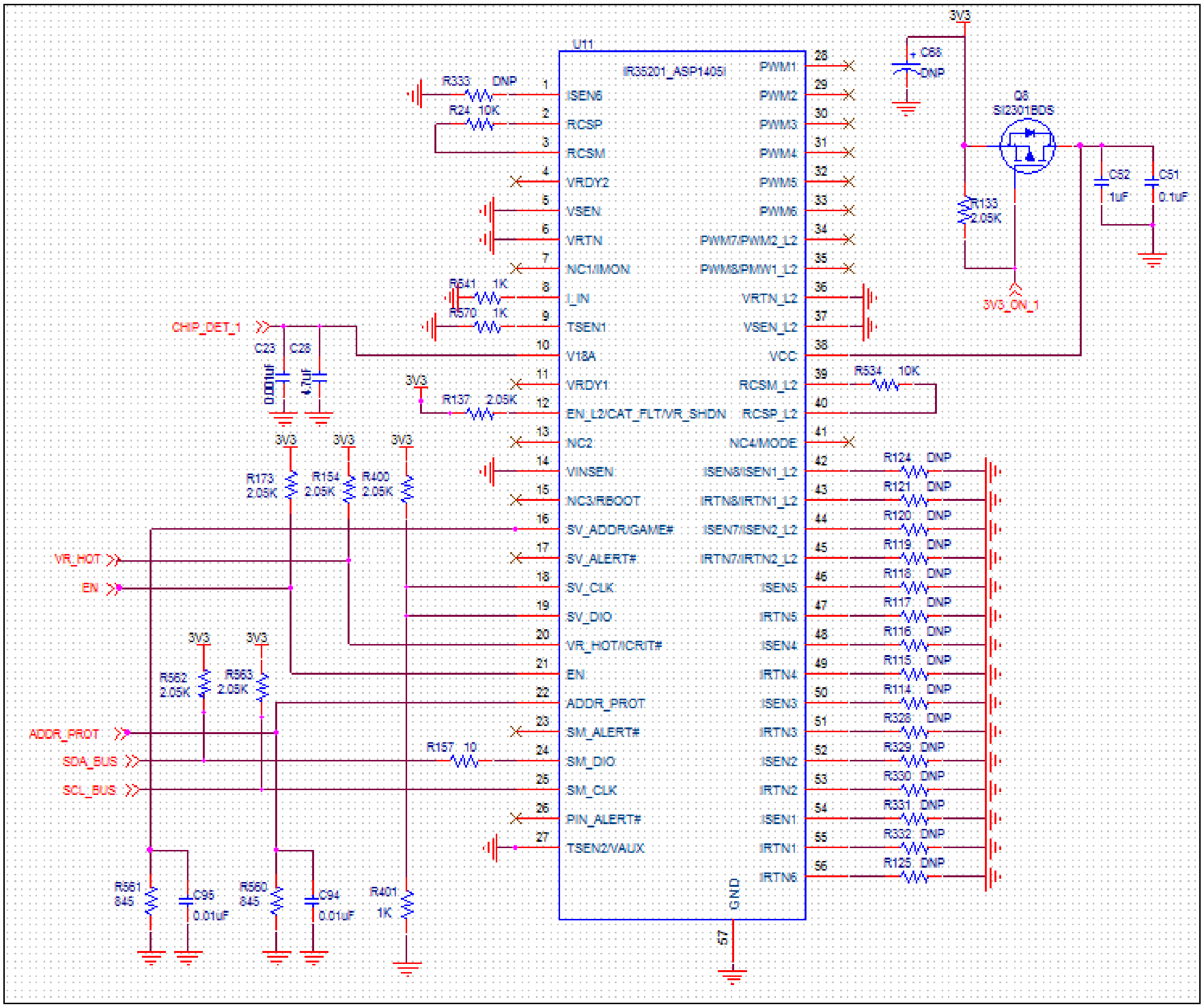

**Figure 3. IR35201/IR35217/ASP1405I Pin Terminations**

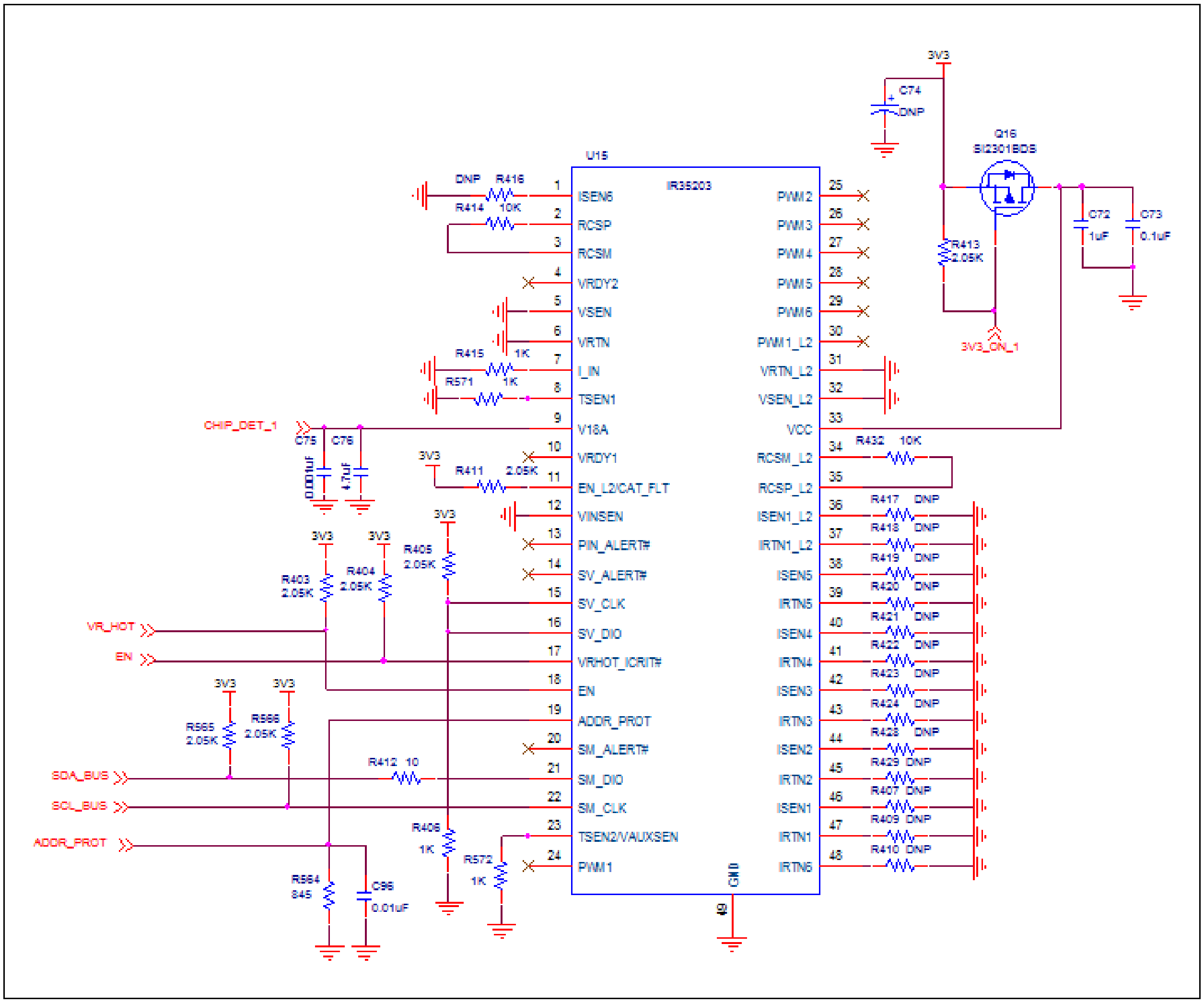


Figure 4. IR35203 Pin Terminations

## PROGRAMMING MTP SECTION

The MTP read/write Command register is located at address CEh, 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 CE[7:5] is changed to 000 ("IDLE" state - see Table 5) *OR* the worst case programming time is exceeded.

### TABLE 5: USER WRITE COMMAND STRUCTURE (REGISTER CEH)

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

### TABLE 6: MANUFACTURER WRITE COMMAND STRUCTURE (REGISTER CEH)

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | Next Programming Pointer 0 - 2 | | |

### TABLE 7: "RETURN" IDLE STATE STRUCTURE (REGISTER CEH)

| Bit[7:5] | Bit 4 | Bit 3 | Bit 2 | Bit1 | Bit 0 |
|---|---|---|---|---|---|
| 000 | Don't Care | | | | |

## MTP PROGRAMMING AND VERIFICATION METHOD 1 – USING TEST MODE I2C ADDRESS 0X0A BY SETTING ADDR_PROT PIN TO 3.3V AT POWER UP.

If register xCE[7:5] doesn't change to 000 after time out period on MTP Write in step 11. Then MTP Write is considered a failure.

| STEP | READ/WRITE REGISTER | DESCRIPTIONS |
|---|---|---|
| 1 | | Power Vcc and ADDR_PROT to 3.3V |
| 2 | | Check if V18A pin is 1.8V +/- 5% |
| 3 | | I2C address will be fixed at x0A (7-bit) in test mode |
| 4 | Write x3E to Reg xCE and wait 50ms | Read the latest OTP image |
| 5 | Read Reg xB6 and xB8 | Use Table 2,3,4 to determine next programming pointer |
| 6 | Write Reg xE8 = x00 | Unlock Vmax and Address registers |
| 7 | Write Reg x24 to xA5 | Write configuration data |
| 8 | Write Reg xA0[2] = 1 | Enable programming clock |
| 9 | Reg xCE = User Write Command | Send User section write command. Refer to Table 5. |
| 10 | Check Reg xCE[7:5] ?= 000 or<br>200 milliseconds time out | Wait until MTP programming is complete.<br>Typical programming time is about 30ms. |
| 11 – Note | Reg xCE = MFR Write Command | Send MFR section write command. Refer to Table 6. |
| 12 – Note | Check Reg xCE[7:5] ?= 000 or<br>100 milliseconds time out | Wait until MTP programming is complete.<br>Typical programming time is about 10ms. |
| 13 | | Vcc 3.3V power off |
| 14 | | Power Vcc and ADDR_PROT to 3.3V |
| 15 | | I2C address will be fixed at x0A (7-bit) in test mode |
| 16 | Write x3E to Reg xCE and wait 50ms | Read the latest OTP image |

| STEP | READ/WRITE REGISTER | DESCRIPTIONS |
|---|---|---|
| 17 | Check Reg xB5[2:0] ?= 000 | CRC flags of trim, user, and mfr section.<br>0 = CRC check pass; 1 = CRC check fail |
| 18 | Read Reg x24 to xA5 | Compare with configuration data |
| 19 | | Power off |

**Note :** Step 11 and 12 are optional only if programming Mfr section is attempted.vr

## MTP PROGRAMMING AND VERIFICATION METHOD 2 – USING NORMAL I2C ADDRESS. ADDR_PROT PIN IS CONNECTED TO GROUND WITH A 845 OHMS RESISTOR.

If register xCE doesn't change to 00 after time out period on MTP Write. Then MTP Write is considered a failure.

The PMBus command examples in blue color used in the following table are assuming:

1. No PEC is used in the read or write commands
2. The PMBus address is x70 (7-bit)
3. S means Start condition
4. Sr means Restart condition
5. P means Stop condition

| STEP | READ/WRITE REGISTER | DESCRIPTIONS |
|---|---|---|
| 1 | | ADDR_PROT pin is connected to ground with a 845 ohms resistor only |
| 2 | | Power on Vcc to 3.3V |
| 3 | | Check if CFILT (V18A) pin is 1.8V +/- 5% (optional) |
| 4 | Read Reg xB6 and B8<br>PMBus: S E0 CE B6 Sr E1 RR XX P<br>PMBus: S E0 CE B8 Sr E1 RR XX P | Use Table 2,3,4 to determine next programming pointer<br>RR is the read data byte. Ignore XX byte |
| 5 | Write Reg xE8 = x00<br>PMBus: S E0 D1 E8 00 P | Unlock Vmax and Address registers |
| 6 | Write Reg x24 to xA5 with configuration data<br>FOR AA = x24 to xA5<br>PMBus: S E0 D1 AA WW P<br>NEXT AA | Write configuration data<br>Master must change i2c address after writing register x24. If the configuration data x24[6:0] is different from the current values.<br>WW is the configuration data bytes. AA is the address from x20 to xA3 |
| 7 | Write Reg xA0[2] = 1<br>WW = REG[xA0] \| 0x04<br>PMBus: S E0 D1 A0 WW P | Enable programming clock.<br>Set register xA0[2] to 1 but not change other bits. |
| 8 | Reg xCE = User Write Command<br>PMBus: S E0 D1 CE WW P | Send USER section write command. Refer to Table 5.<br>WW is the USER section programming command. |
| 9 | Check Reg xCE[7:5] ?= 000 or<br>100 milliseconds time out<br>PMBus: S E0 D0 CE Sr E1 RR XX P | Wait until MTP programming is complete.<br>Typical programming time is about 30ms.<br>RR is the read data byte. Ignore XX byte. |
| 10 (optional) | Write Reg xA0[2] = 1<br>WW = REG[xA0] \| 0x04<br>PMBus: S E0 D1 A0 WW P | Enable programming clock.<br>Set register xA0[2] to 1 but not change other bits. |

| STEP | READ/WRITE REGISTER | DESCRIPTIONS |
|---|---|---|
| 11 (optional) | Reg xCE = MFR Write Command<br>PMBus: S E0 D1 CE WW P | Send MFR section write command. Refer to Table 6.<br>WW is the MFR section programming command. |
| 12 (optional) | Check Reg xCE[7:5] ?= 000 or<br>50 milliseconds time out<br>PMBus: S E0 D0 CE Sr E1 RR XX P | Wait until MTP programming is complete.<br>Typical programming time is about 10ms.<br>RR is the read data byte. Ignore XX byte. |
| 13 | | Vcc 3.3V power off |
| 14 | | Power on Vcc to 3.3V |
| 15 | Check Reg xB5[2:0] ?= 000 | CRC flags of trim, user, and mfr section.<br>0 = CRC check pass; 1 = CRC check fail |
| 16 | Read Reg x20 to xA3<br>FOR AA = x24 to x96<br>PMBus: S E0 D0 AA Sr E1 RR XX P<br>NEXT AA<br><br>FOR AA = x98 to xA5<br>PMBus: S E0 D0 AA Sr E1 RR XX P<br>NEXT AA | Verification: compare read data with configuration data.<br>RR is the read data byte. Ignore XX byte.<br>USER section is from x24 to x96<br>MFR section is from x98 to xA5 |

## SALEM REGISTER MAP

Trims
User
Read Only
Read Write
Mfr

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 00 | 9F | 9F | 9F | 9F | 9F | 9F | 80 | 80 |
| 1x | 20 | 82 | 82 | 00 | 04 | 00 | 00 | 00 | 00 | 82 | 00 | FF | FF | 00 | 00 | 00 |
| 2x | 22 | 08 | 10 | 08 | 90 | 22 | 10 | 08 | C8 | 11 | 00 | A1 | 55 | 55 | 50 | 00 |
| 3x | 00 | 00 | 00 | 00 | 00 | E8 | E8 | 24 | 14 | FF | 08 | 1C | 18 | 10 | 10 | 60 |
| 4x | A0 | 69 | 76 | 1D | 21 | 30 | 34 | 14 | 90 | F4 | EF | 4C | 4F | 08 | 50 | 00 |
| 5x | E0 | AA | A0 | 90 | A0 | 90 | C3 | 44 | 52 | 02 | 00 | 7D | 58 | 18 | 88 | 00 |
| 6x | C0 | 40 | 80 | 00 | 00 | 00 | BA | 00 | F0 | A0 | 08 | 01 | 01 | 30 | 10 | 32 |
| 7x | 02 | 50 | 51 | FF | AB | 68 | 02 | B7 | 01 | 39 | 30 | 88 | 00 | 00 | 00 | 00 |
| 8x | 00 | 07 | 00 | 52 | 49 | 00 | 00 | 00 | 01 | 04 | FF | FF | 00 | 00 | 00 | 00 |
| 9x | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | D4 | FC | 00 | 00 | 00 | 00 | FB | FB |
| Ax | 20 | 00 | 23 | 10 | 06 | 16 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| Bx | 33 | 01 | 00 | 30 | 5E | 00 | 3F | 02 | 0F | 55 | 18 | 63 | 31 | 00 | C8 | 00 |
| Cx | 05 | 66 | 01 | 01 | 13 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 03 |
| Dx | 02 | 40 | AD | 0C | 00 | 00 | 3F | 00 | 02 | 00 | 00 | C0 | E4 | 0F | 20 | 00 |
| Ex | 00 | 00 | 07 | 80 | 02 | 00 | 07 | 03 | 00 | 00 | 40 | 00 | 00 | 00 | 00 | 00 |
| Fx | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 01 | 00 | 00 | 01 | 00 | 49 | 52 | 64 | 32 |

# I2C REGISTER READ/WRITE PROTOCOL

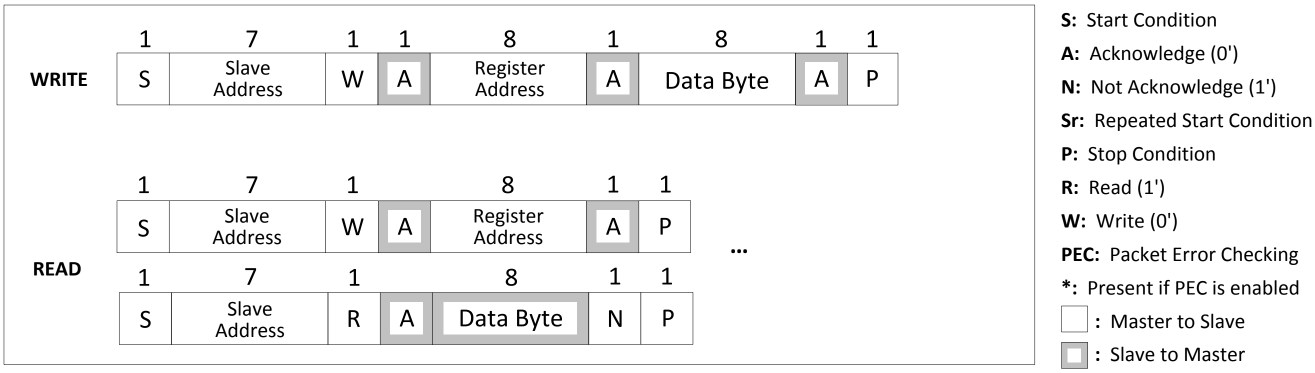


**Figure 2: I2C Protocols to Read or Write an IR Register**

## I2C READ WRITE.C

```
/*------------------------------------*
/     i2c Read
/*------------------------------------*/

void I2CReadData(unsigned char SlaveAddress, unsigned char RegAddress,
                 unsigned char * RegData)
{
  I2CStart();                    // Send start signal
  I2CSendByte(SlaveAddress);     // Slave I2C address
  I2CSendByte(RegAddress);       // Send Register address
  I2CStop();                     // Send I2C Stop; Optional
  I2CStart();                    // Send I2C ReStart
  I2CSendByte(SlaveAddress +1); // address + Read
  * RegData = I2CGetByte(1);         // Read one byte and send NACK (1)
  I2CStop();                    // Send I2C Stop Transfer
}

/*------------------------------------*
/     i2c Write
/*------------------------------------*/

void I2CWriteData(unsigned char SlaveAddress, unsigned char RegAddress,
                  unsigned char * RegData)
{
  I2CStart();                    // Send start signal
  I2CSendByte(SlaveAddress);     // Slave I2C address
  I2CSendByte(RegAddress);       // Send Register address
  I2CSendByte(RegData);          // Send Register data
  I2CStop();                     // Send I2C Stop
}
```

  

# PMBUS REGISTER READ/WRITE PROTOCOL

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | PMBus Address | W | A | Command D0h | A | Register Address | A | Sr | PMBus Address | R | A | Data Byte | A | PEC | N | P |

**Figure 3: MFR_READ_REG**

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

**Figure 4: MFR_WRITE_REG**

NOTE: PEC IS REQUIRED FOR MFR_READ_REG COMMAND

```
/*-----------------------------------*
/     PMBus Read Register
/*-----------------------------------*/

void PMBusReadRegister(unsigned char PMBusAddress, unsigned char RegAddress,
                 unsigned char * RegData)
{
  I2CStart();                    // Send start signal
  I2CSendByte(PMBusAddress);    // Slave address
  I2CSendByte(0xCE);             // Send Read Command
  I2CSendByte(RegAddress);      // Send Register address
  I2CStart();                    // Send ReStart
  I2CSendByte(PMBusAddress +1); // address + Read
  * RegData = I2CGetByte(0);         // Read one byte and ACK (0)
  I2CGetByte(1);                // Force Slave to send PEC and NACK (1)
  I2CStop();                   // Send I2C Stop Transfer
}

/*-----------------------------------*
/     PMBus Write Register
/*-----------------------------------*/

void PMBusWriteRegister(unsigned char PMBusAddress, unsigned char RegAddress,
                  unsigned char * RegData)
{
  I2CStart();                    // Send start signal
  I2CSendByte(PMBusAddress);    // Slave address
  I2CSendByte(0xD1);             // Send Write Command
  I2CSendByte(RegAddress);      // Send Register address
  I2CSendByte(RegData);          // Send Register data
  // PEC is optional in MFR_WRITE_PEC
  I2CStop();                     // Send I2C Stop
}
```

# IDENTIFY DEVICE I2C ADDRESS IN NORMAL OPERATING MODE

```
For ScanAddress = 0x08 to 0x77 Excep 0x0C
        If Read_I2C_Byte(ScanAddress, register 0xFB, Product_ID) is ACKed then
                If Product_ID = 0x4D or 0x4E or 0x4F or 0x50 or 0x5F then
                        Device Address = ScanAddress
                End If
        End If
Next
```

  

# 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, space delimited text document as shown below.

| CONFIG FILE | | |
|---|---|---|
| RegAddr | ConfigData | ConfigMask |
| 24 | 08 | FF |
| 25 | 03 | FF |
| 26 | AA | FF |
| 27 | 80 | FF |
| 28 | C0 | FF |
| 29 | 00 | FF |
| 30 | 00 | FF |
| 31 | 00 | FF |
| 32 | 00 | FF |
| 33 | E4 | FF |
| --- | --- | -- |
| --- | --- | --- |
| --- | --- | --- |
| 9F | 4C | FF |
| A0 | 03 | FF |
| A1 | 4C | FF |
| A2 | AA | FF |
| A3 | 4C | FF |
| A4 | 80 | FF |
| A5 | 16 | 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. Data is read line by line. For example, row 1 indicates that 08h is targeted for register 24h 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 doesn't need to be verified after programming. If the mask bit is one, the corresponding bit has to match the value from the data column after programming.

# APPENDIX B: SALEM CONTROLLER MEMORY TEMPERATURE LIMITS

## ABSOLUTE MAXIMUM RATINGS

| | |
|---|---|
| MTP Memory operating temperature | -40C to +125C |
| Memory Programming temperature | 0C to +55C |

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

## Revision History

| Version | ECN# | Reason for Change | Date | Modified by: | Approved by: |
|---|---|---|---|---|---|
| 1.2 | | 1. Add programming method 1<br>2. Fix register addresses for Salem R1B silicon<br>3. Add pin termination diagrams | May 1, 2014 | S. Lee | David Williams |
| 1.1 | | Add APPENDIX B | Aug 1, 2014 | Janice Lund | David Williams<br>Carl Smith |
| 1.0 | | Intitial Release.  Approved by DW email 06/10/13 | Oct 28, 2013 | L.Fraser | David Williams |