Application Report
SLVA802–August 2016

TLC6C5712-Q1 Application Reference Guide

Anda Zhang

ABSTRACT

The TLC6C5712-Q1 device is an advanced constant-current LED driver with full LED diagnostics. This report provides a guide on how to use the TLC6C5712-Q1 device on both hardware configuration and software configuration.

This report introduces the TLC6C5712-Q1 features and application configurations. It also discusses a software guide on how to realize TLC6C5712-Q1 diagnostics. The report provides a software flow chart and fault handling routine as a guide of the software configuration. A pseudocodes example on how to configure the TLC6C5712-Q1 is included at the end of the report.

Contents

List of Figures

List of Tables

1 Introduction

The TLC6C5712-Q1 device is a 12-channel constant-current LED driver with a maximum 75-mA output current per channel. The precision output current with 8-bit dot correction makes the TLC6C5712-Q1 device a perfect solution to correct LED brightness and color temperature variation. Advanced protection and diagnostics on the TLC6C5712-Q1 device improve the system-level robustness and safety. Six PWM inputs with programmable mapping support different LED color-dimming configurations and provide a high dimming ratio. A 16-bit serial-peripheral interface (SPI) with diagnostics supports multiple devices in a daisy chain and eases system-level design.

The device is typically used in applications such as: automotive cluster tell-tale indicators, HVAC panel indicators, shift-by-wire PRNDL indicators, ambient lighting, and sequential turn indicators.

2 Device Function

2.1 SPI Protocol

The serial port is used to write data to, read diagnostic status from, and configure the settings of the TLC6C5712-Q1 device. During normal operation, an 8-bit address and 8-bit data are written into the 16-bit shift register. The input data length should be 16 bits, otherwise the TLC6C5712-Q1 device does not recognize the data. Therefore, when a 16-bit SPI frame is broken for whatever reason, the device will discard the input data. On each SCK rising-edge input, data is sampled. The clock frequency $f_{(SCK)}$ can be up to 10 MHz. The device converts the 16 inputs to the registers on the LATCH rising edge.

Figure 1 shows a typical register write pattern, each register write command contains an 8-bit address and 8 bits of data. The 8-bit address should be shifted into the device first, then the 8 bits of data are shifted into the device.

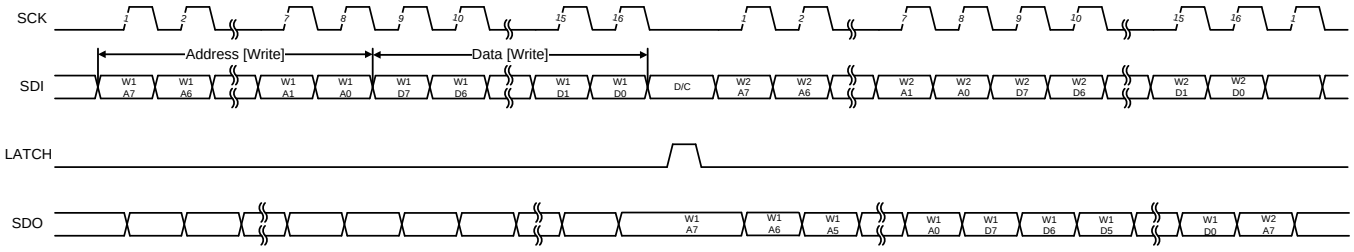

Figure 1. Register Write Protocol

Figure 2 shows a typical register read pattern, each register read command contains an 8-bit address, 8 bits of data, and 16 bits of dummy data. The 8-bit address should be shifted into the device first, then the 8 bits of data are shifted into the device (usually this data has no requirement when reading data from registers, 0x00 is typically used), the 16 bits of dummy data are used to shift out the read register information after the latch rising edge. For example, writing [0x92 0x00] + [0x00 0x00] gets the channel on mask information of CH0-CH5 from the SDO.

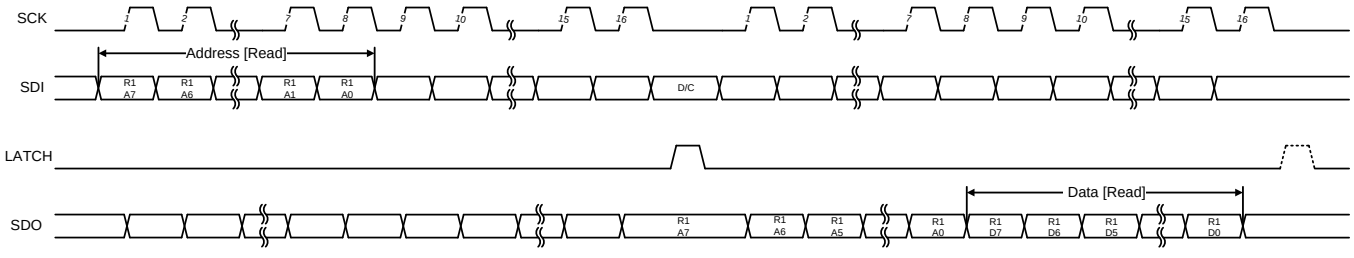

Figure 2. Register Read Protocol

The TLC6C5712-Q1 device supports multiple devices in cascaded daisy-chain mode. Each communication sequence only requires one LATCH rising edge. For a number, N, of daisy-chained devices, a communication cycle comprises 16 × N SCK cycles with corresponding data transferred to shift registers on the rising edge of LATCH.

Example 1, which follows, shows a data write and read example (MSP430G2553 codes).

```
SPI_Write (Address, Data);                          // Register Write
SPI_Read (Address, Data);                           // Register Read, return 8 bits read data
void SPI_Write(u8 DataIn1, u8 DataIn2)
{
    LATCH_LOW;
    Delay();
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF = DataIn1;                            //Send out 1st 8 bits data
    while (UCB0STAT & BUSY);                        //Wait until send complete
    UCB0TXBUF = DataIn2;                            //Send out 2nd 8 bits data
    while (UCB0STAT & BUSY);                        //Wait until send complete
    Delay();                                        //Delay for LATCH
    LATCH_HIGH;
    Delay();
    LATCH_LOW;
    Delay();
}
int SPI_Read(u8 DataIn1, u8 DataIn2)
{
    int RXData = 0;
    LATCH_LOW;
    Delay();
    while (!(IFG2 & UCB0TXIFG));
    UCB0TXBUF = DataIn1;                          //Send out 1st 8 bit data
    while (UCB0STAT & BUSY);                      //Wait until send complete
    UCB0TXBUF = DataIn2;                          //Send out 2nd 8 bit data
    while (UCB0STAT & BUSY);                      //Wait until send complete
    Delay();                                      //Delay for LATCH
    LATCH_HIGH;
    Delay();
    LATCH_LOW;
    Delay();
    while (!(IFG2 & UCB0TXIFG));                  //16 bits dummy data send
    UCB0TXBUF = 0x00;                             //Send out 1st 8 bit dummy data
    while (UCB0STAT & BUSY);                      //Wait until send complete
    UCB0TXBUF = 0x00;                             //Send out 2nd 8 bit dummy data
    while (UCB0STAT & BUSY);                      //Wait until send complete
    RXData |= (UCB0RXBUF);                        //Read LSB 8 bit back
    while (UCB0STAT & BUSY);                      //Wait until send complete
    Delay();                                      //Delay for LATCH
    LATCH_HIGH;
    Delay();
    LATCH_LOW;
    Delay();
    return(RXData);
}
```

2.2 Channel On and Off Control

Controlling the channel on and off can happen in two ways: using the [WRITE_CH_ON_MASKx] registers and using the PWM inputs.

2.2.1 Channel On and Off Register

The [WRITE_CH_ON_MASK0] and [WRITE_CH_ON_MASK1] registers are the channel-activation mask registers which control each channel output activated and deactivated. A bit value of 0 stands for channel activated status. A bit value of 1 stands for channel deactivated status. The bit value can be configured through SPI. Table 1 lists the detailed register information.

Table 1. CH_ON_MASK Register

Register Name	Addr	D7	D6	D5	D4	D3	D2	D1	D0	Default
WRITE_CH_ON_MASK0	52h	RESERVED	RESERVED	CH_ON_MASK5	CH_ON_MASK4	CH_ON_MASK3	CH_ON_MASK2	CH_ON_MASK1	CH_ON_MASK0	3Fh
WRITE_CH_ON_MASK1	53h	RESERVED	RESERVED	CH_ON_MASK11	CH_ON_MASK10	CH_ON_MASK9	CH_ON_MASK8	CH_ON_MASK7	CH_ON_MASK6	3Fh

Example 2, which follows, shows a channel on or off register configuration example.

```
SPI_Write (0x52; 0x3E);        // Turn on Channel 0
SPI_Write (0x53; 0x3E);        // Turn on Channel 6
```

2.2.2 PWM Control

PWM dimming is typically used for LED brightness control which greatly reduces LED color changes compared with output DC-current control. The TLC6C5712-Q1 device has six PWM inputs with independently configurable mapping to any of the 12 channels for PWM dimming. Each of 12 output channels has a 3-bit register [PWM_MAP_CHx] to assign to one PWM input. All output channels are assigned to $\overline{PWM0}$ by default, which means $\overline{PWM0}$ controls all of the 12 outputs by default. Table 2 lists the PWM mapping table and Table 3 lists the PWM mapping registers.

Table 2. PWM Mapping Table

BIT2	BIT1	BIT0	$\overline{PWMx}$
0	0	0	$\overline{PWM0}$
0	0	1	$\overline{PWM1}$
0	1	0	$\overline{PWM2}$
0	1	1	$\overline{PWM3}$
1	0	0	$\overline{PWM4}$
1	0	1	$\overline{PWM5}$
1	1	0	$\overline{PWM0}$
1	1	1	$\overline{PWM0}$

Table 3. PWM Mapping Registers

Register Name	Addr	D7	D6	D5	D4	D3	D2	D1	D0	Default
WRITE_MAP0	40h	RESERVED	RESERVED	PWM_MAP_CH1[2:0]			PWM_MAP_CH0[2:0]			00h
WRITE_MAP1	41h	RESERVED	RESERVED	PWM_MAP_CH3[2:0]			PWM_MAP_CH2[2:0]			00h
WRITE_MAP2	42h	RESERVED	RESERVED	PWM_MAP_CH5[2:0]			PWM_MAP_CH4[2:0]			00h
WRITE_MAP3	43h	RESERVED	RESERVED	PWM_MAP_CH7[2:0]			PWM_MAP_CH6[2:0]			00h
WRITE_MAP4	44h	RESERVED	RESERVED	PWM_MAP_CH9[2:0]			PWM_MAP_CH8[2:0]			00h
WRITE_MAP5	45h	RESERVED	RESERVED	PWM_MAP_CH11[2:0]			PWM_MAP_CH10[2:0]			00h

The TLC6C5712-Q1 PWM input signal is active low. To achieve LED fault diagnostics, the PWM input requires a 5-µs minimal on time and 40-µs minimal off time. If the PWM input is constant high or low, the LED short-to-GND and open-load fault diagnostics would not be distinguished under the channel activated state. Because behavior is the same, both output voltages are low level. If PWM dimming is not required, TI does not recommend connecting the PWM inputs directly to GND. Instead, use a low duty-cycle PWM input (for example, minimum 0.8% positive duty cycle at 200 Hz) to realize LED fault diagnostics. Table 4 lists the available status of the LED-fault diagnostics under different channel status.

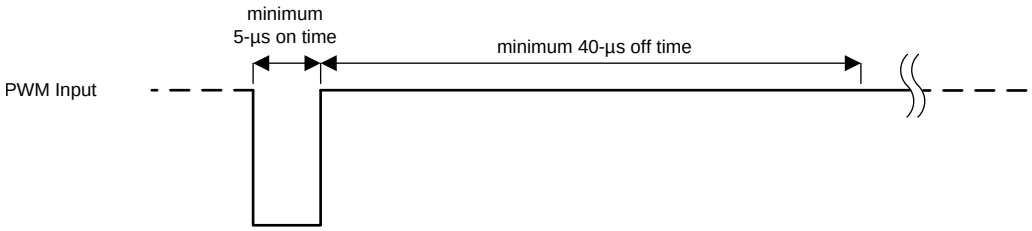

Figure 3. PWM Input Requirement for LED Diagnostics

Table 4. LED Fault Diagnostics Available Status

PWM	CH-ON-MASKx	LED Short-to-Supply	Short-to-GND	Open Load
Constant Low	High (CH deactivated)	Yes	Yes	Yes
	Low (CH activated)	Yes	No	No
Constant High	High (CH deactivated)	Yes	Yes	Yes
	Low (CH activated)	Yes	No	No
Normal PWM	High (CH deactivated)	Yes	Yes	Yes
	Low (CH activated)	Yes	Yes	Yes

Example 3, which follows, shows a PWM mapping register configuration example.

```
SPI_Write (0x40; 0x00);     // Mapping PWM0 to CH0 and CH1
SPI_Write (0x41; 0x09);     // Mapping PWM1 to CH2 and CH3
SPI_Write (0x42; 0x12);     // Mapping PWM2 to CH4 and CH5
SPI_Write (0x43; 0x1B);     // Mapping PWM3 to CH6 and CH7
SPI_Write (0x44; 0x24);     // Mapping PWM4 to CH8 and CH9
SPI_Write (0x45; 0x2D);     // Mapping PWM5 to CH10 and CH11
PWM0(Dutycycle, Freq);      // Set PWM0 input frequency and duty cycle(connect to GND if not use)
PWM1(Dutycycle, Freq);      // Set PWM1 input frequency and duty cycle(connect to GND if not use)
PWM2(Dutycycle, Freq);      // Set PWM2 input frequency and duty cycle(connect to GND if not use)
PWM3(Dutycycle, Freq);      // Set PWM3 input frequency and duty cycle(connect to GND if not use)
PWM4(Dutycycle, Freq);      // Set PWM4 input frequency and duty cycle(connect to GND if not use)
PWM5(Dutycycle, Freq);      // Set PWM5 input frequency and duty cycle(connect to GND if not use)
```

2.3 LED Output Current Control

Designers have long struggled to ensure even brightness among LEDs. Typically a customer had to bin LEDs according to brightness level which is a time-consuming and expensive task. Now, however, the TLC6C5712-Q1 device has the ability to program the LED current which eliminates the need for binning LEDs. By allowing LED current adjustment, the TLC6C5712-Q1 compensates for brightness variations during manufacturing. The system can use the external resistor on the IREF pin and dot-correction registers to change the output DC current which can realize the LED binning purpose.

2.3.1 Global Current Setting

The output current of each channel can be set by an external resistor connected to the IREF pin. Use Equation 1 to calculate the value of this resistor ($R_{(IREF)}$). Figure 4 shows the curve between the output current and the reference resistor.

$$R_{(IREF)} = \frac{V_{(IREF)}}{IOUT_{MAX}} \times K_{(OUT)} \quad (1)$$

where

- $V_{(IREF)}$ is the reference voltage which is 1.229 V.
- $IOUT_{MAX}$ is the output current under dot correction equal to 255.
- $K_{(OUT)}$ is the ratio of output current to IREF current ($I_{(OUTx)}$ / $I_{(IREF)}$) which is typically 500.

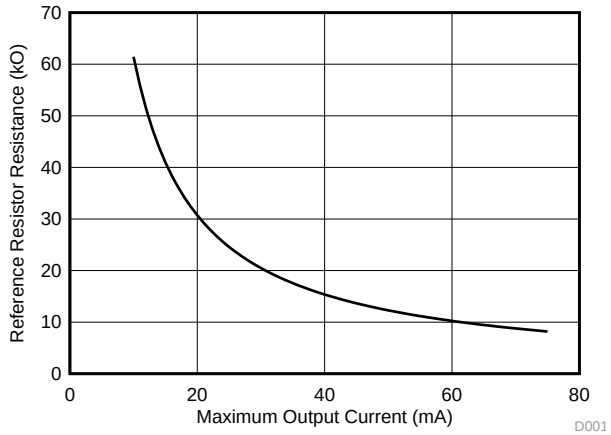

Figure 4. Output Current vs Reference Resistor Curve

2.3.2 Dot Correction

Each output channel of the device has an internal 8-bit linear current digital-analog-converter (DAC) for individual dot-correction control. The 8-bit registers [OUTPUT_DC_CHx] are used to control the DAC output current as shown in Equation 2.

$$I_{(OUT)} = I_{(OUT,MAX)} \times \frac{\text{Dot Correction} + 1}{256} \quad (2)$$

The minimal current is 1/256 of $I_{(OUT,MAX)}$. If zero current is required in some applications, it can be realized by deactivate the output channel by setting the corresponding channel enable register [CH_ON_MASKx] HIGH. Table 5 lists the detailed information for dot correction registers.

Example 4, which follows, shows a dot correction configuration example.

```
SPI_Write (0x46; 0xFF);      // Set CH0 DC current to maximum output current
SPI_Write (0x47; 0x7F);      // Set CH1 DC current to 50% max output current
…
SPI_Write (0x50; 0x00);      // Set CH10 DC current to 1/256 maximum output current
SPI_Write (0x51; 0xFF);      // Set CH11 DC current to maximum output current
```

Table 5. Dot Correction Registers

Register Name	Addr	D7	D6	D5	D4	D3	D2	D1	D0	Default
WRITE_CORR0	46h	OUTPUT_DC_CH0[7:0]								00h
WRITE_CORR1	47h	OUTPUT_DC_CH1[7:0]								00h
WRITE_CORR2	48h	OUTPUT_DC_CH2[7:0]								00h

Table 5. Dot Correction Registers (continued)

Register Name	Addr	D7	D6	D5	D4	D3	D2	D1	D0	Default
WRITE_CORR3	49h	OUTPUT_DC_CH3[7:0]								00h
WRITE_CORR4	4Ah	OUTPUT_DC_CH4[7:0]								00h
WRITE_CORR5	4Bh	OUTPUT_DC_CH5[7:0]								00h
WRITE_CORR6	4Ch	OUTPUT_DC_CH6[7:0]								00h
WRITE_CORR7	4Dh	OUTPUT_DC_CH7[7:0]								00h
WRITE_CORR8	4Eh	OUTPUT_DC_CH8[7:0]								00h
WRITE_CORR9	4Fh	OUTPUT_DC_CH9[7:0]								00h
WRITE_CORR10	50h	OUTPUT_DC_CH10[7:0]								00h
WRITE_CORR11	51h	OUTPUT_DC_CH11[7:0]								00h

2.3.3 Output Parallel Configuration

If larger current is required for some applications, outputs can be paralleled to achieve a higher output current. Figure 5 shows a high-current configuration by paralleling the outputs together.

If adjacent pins are tied together to get a higher output current, the device should ignore the adjacent pin short fault. If adjacent-pin short-fault detection is required, pins with an even number should be tied together and pins with an odd number should be tied together.

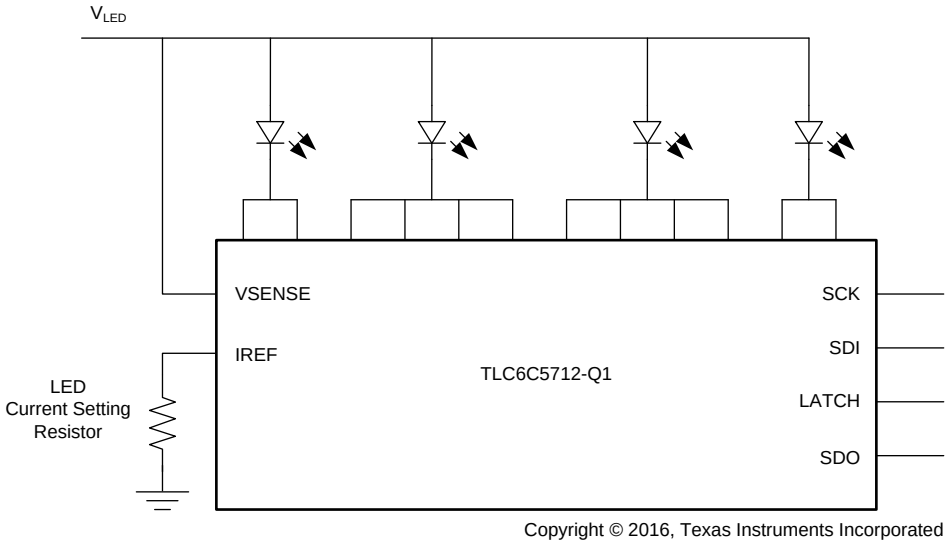

Figure 5. Parallel Output Configuration

2.4 Thermal Consideration

The selection of the LED supply voltage (V_{LED}) is influenced by the parameters that follow:

- The minimum voltage drop across current outputs ($V_{(OUT,min)}$) which should be high enough to help ensure the desired output current.
- The maximum LED forward voltage ($V_{F,max}$)
- The maximum power that can be dissipated by the device package under the real application conditions
- The maximum output voltage rating, $V_{OUT,max}$ (7 V)
- The accuracy of the LED supply voltage (V_{LED} can vary in range and the minimum value should be considered)

To realize constant current operation and safe operation, the LED supply voltage should be in the range of $V_{OUT,max} \geq V_{LED} \geq V_{OUT,min} + V_{F,max}$

To keep the device power dissipation low, the LED supply voltage should as low as possible. Use Equation 3 to calculate the power dissipation on the device ($P_{T(tot)}$).

$$P_{T(tot)} = V_{CC} \times I_{CC} + \sum_{x=0}^{11}\left(V_{(OUTx)} \times I_{(OUTx)}\right) - \frac{V_{(IREF)}}{R_{(IREF)}^{2}} \quad (3)$$

where

- $P_{T(tot)}$ is the total power dissipation of the device.
- $V_{(OUTx)}$ is the voltage drop for channel x.
- $I_{(OUTx)}$ is the average LED current for channel x.
- $V_{(IREF)}$ is the reference voltage, 1.229 V.
- $R_{(IREF)}$ is the reference resistor.

To make sure the device is in normal operation, the total power dissipation should not be larger than the maximum total power dissipation. Based on the real application conditions, use Equation 4 to calculate the maximum power dissipation ($P_{T,max}$) of the device.

$$P_{T,max} = \frac{\left(T_J - T_A\right)}{R_{\theta JA}} \quad (4)$$

where

- $P_{T,max}$ is the maximum total power dissipation of the device.
- T_J is the maximum device junction temperature, 150°C typically.
- T_A is the maximum application ambient temperature.
- $R_{\theta ja}$ is the device junction-to-ambient thermal resistance.

The proper power supply must be a trade-off between the correct value assuring the desired LED current and low power dissipation.

In some applications, if different LED colors are used, the LED supply voltage should be selected high enough to help ensure that all the LEDs are working normally. But the LED forward voltage of different colors varies a lot. For example, in RGB applications, the forward voltage of the red LEDs is 2 V (typical) which is lower than the green and blue LEDs. Therefore, the channel of the red LEDs has larger power dissipation. To minimize the power dissipation, two different approaches are available which are described as follows:

- Figure 6 shows an application with only one voltage rail (V_{LED}). A resistor in series with each red LED is added. In this way, the voltage excess drops across the resistor instead of dropping across the current generators. This solution implies a significant reduction of the power dissipated by the chip. However the total power dissipation does not change and a large portion of the power is still wasted on the series resistor.

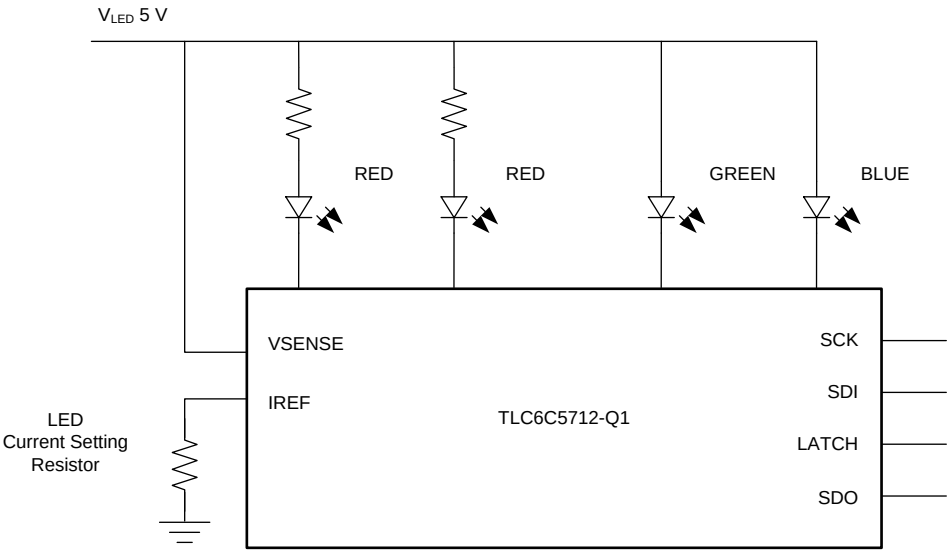

Figure 6. Application With Resistor In Series With LED

- Figure 7 shows a solution with two separate voltage rails: one for blue and green LEDs and one for red LEDs which can be derived from a voltage regulator. The voltage rails are tailored to the type of LEDs they drive and the wasted power is significantly reduced as well as the heat produced.

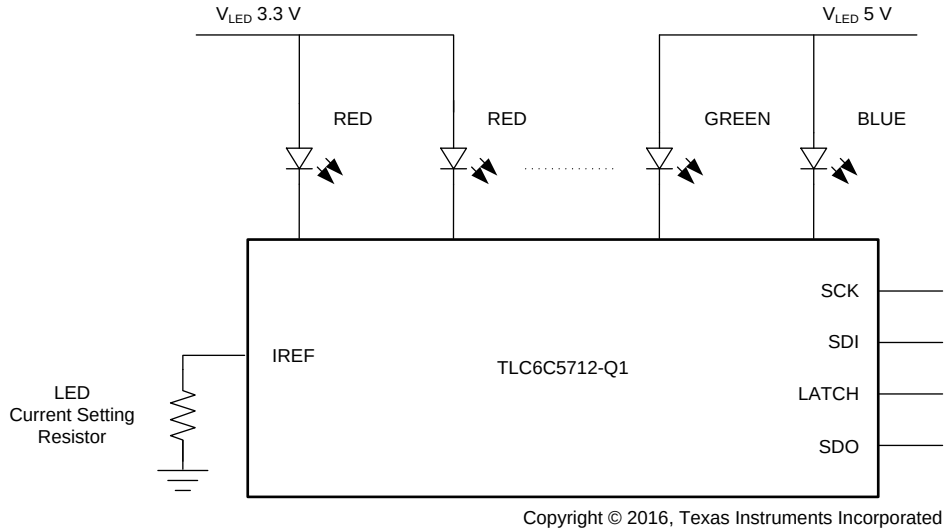

Figure 7. Application With Two Separate LED-Supply Rails

2.5 Device Diagnostics and Protections

2.5.1 LED Fault

The TLC6C5712-Q1 device can detect different kinds of LED faults, such as LED short-to-supply, short-to-GND, and open-load faults. The device can detect the faults in the channel in both the activated and deactivated state which is useful for applications that require LED off-state diagnostics. To achieve LED full diagnostics, a minimum 5-µs LED on time and minimum 40 µs LED off time must pass as shown in Figure 3. Table 4 lists the available status of the LED-fault diagnostics under different channel status.

When a fault is on an LED, the corresponding LED-fault register bit changes to HIGH. The register state remains latched even when the fault is removed. To clear the fault register, issue the RESET_STATUS command after the fault is removed.

Example 5, which follows, shows an LED fault detection example.

```
SPI_Write (0x54, 0x00);                    // Unmask CH0-CH5 LED short fault
SPI_Write (0x55, 0x00);                    // Unmask CH6-CH11 LED short fault
SPI_Write (0x56, 0x00);                    // Unmask CH0-CH5 Short to GND fault
SPI_Write (0x57, 0x00);                    // Unmask CH6-CH11 Short to GND fault
SPI_Write (0x58, 0x00);                    // Unmask CH0-CH5 LED open fault
SPI_Write (0x59, 0x00);                    // Unmask CH6-CH11 LED open fault
STATUS0 = SPI_Read(0xA2; 0x00)             // Read STATUS0 register value

If (STATUS0>>4)&&Bit0 == 1;                // There is a short to GND or LED short fault
//Identify the detailed short fault channel
Short_0 = SPI_Read (0x9A; 0x00);           // Check CH0-CH5 short fault register value
Short_1 = SPI_Read (0x9B; 0x00);           // Check CH6-CH11 short fault register value
SG_0 = SPI_Read (0x9C; 0x00);              // Check CH0-CH5 short to GND fault register value
SG_1 = SPI_Read (0x9D; 0x00);              // Check CH6-CH11 short to GND fault register
                                           value

If (STATUS0>>5)&&Bit0 == 1;                // There is an open fault
//Identify the detailed open fault channel
Open_0 = SPI_Read (0x9E; 0x00);            // Check CH0-CH5 open fault register value
Open_1 = SPI_Read (0x9F; 0x00);            // Check CH6-CH11 open fault register value
```

2.5.2 Adjacent Pin Short

When the device is soldered on the application board, undesired short-circuits between adjacent pins can occur. The TLC6C5712-Q1 device implements an adjacent pin short-detection function which can be used to detect the adjacent pin short fault. This feature requires offline diagnostics which means the outputs should be in the channel off state when performing the adjacent pin-short detection.

To start the adjacent pin-short detection, set the [ADJ_DIAG_START] bit HIGH. This bit automatically returns to LOW when the adjacent pin diagnostic procedure finishes. If any two adjacent pins are shorted, the [ADJ_FLAG_CHx] bit for the faulty channel is set HIGH.

Example 6, which follows, shows an adjacent pin short detection example.

```
// Start the adjacent pin short detection, the bit will return to 0 after the detection finishes.
SPI_Write (0x67, 0x08);
AD0 = SPI_Read (0XA8, 0x00);               // Check the CH0-CH5 adjacent pin short fault
AD1 = SPI_Read (0XA9, 0x00);               // Check the CH6-CH11 adjacent pin short fault
If (AD0>>1) &&Bit0 == 1 and AD0&&Bit0 == 1 // CH0 and CH1 are shorted together
```

2.5.3 Thermal Prewarning and Shutdown

When the junction temperature exceeds the pre-thermal-warning threshold T(PTW) (135C typical), [PRE_TSD_FLAG] in the <READ_STATUS0> register is set HIGH to signal the pre-thermal warning. The $\overline{\text{ERR}}$ open-drain output is also pulled down. The microcontroller should respond to the fault warning and take actions to prevent junction temperature rising.

If junction temperature continues to rise and exceeds thermal-shutdown threshold T(TSD) (165C typical), the over temperature fault bit [TSD_FLAG] in the <READ_STATUS0> register is set HIGH to signal thermal shutdown, the $\overline{ERR}$ open-drain output is pulled down, and all output channels are turned off for protection. [PRE_TSD_FLAG] and [TSD_FLAG] will remain latched even the device returns to normal operation state. To clear fault, issue the RESET_STATUS command after the device returns to normal operation.

Example 7, which follows, shows a thermal prewarning example.

```
STATUS0 = SPI_Read(0xA2; 0x00)          // Read STATUS0 register value
If (STATUS0>>1) && Bit0 == 1;           // There is a thermal pre-warning fault.
PWM0(99%, 200Hz);                       // Reduce the turn on time to reduce the heat
SPI_Write (0x46; 0x00);                 // Reduce channel output current the reduce the
                                        heat
```

2.5.4 LED Weak Supply

The TLC6C5712-Q1 device provides weak-LED-supply detection to avoid reporting false faults because of a supply failure. Implementation of weak-LED-supply detection is by monitoring the $V_{(SENSE)}$ voltage using the internal threshold voltage $V_{(WLS)}$ (4.2 V typical) as a reference. The default threshold $V_{(WLS)}$ is set for a 5-V supply. If a 3.3-V LED supply is used, the threshold voltage can be set to $V_{(WLS\_OPT)}$ (2.77 V typical) by setting the [WLS_TH] bit HIGH.

When a fault is detected, the [WLS_FAULT_FLAG] bit is set to HIGH. The [WLS_FAULT_FLAG] bit remains latched even if the voltage recovers. To clear the fault, issue the RESET_STATUS command after the fault is removed.

Example 8, which follows, shows a 3.3-V LED power supply example.

```
SPI_Write (0x67, 0x01);                 // Set the weak supply threshold to 2.77V for 3.3V
                                        supply
STATUS0 = SPI_Read(0xA2; 0x00)          // Read STATUS0 register value
If (STATUS0>>2) && Bit 0 == 1;          // There is a weak supply fault.
```

2.5.5 REF Open and Short

The device integrates a failsafe mode. When a short or open fault occurs on the reference resistor, the device enters the failsafe mode. In failsafe mode, the maximum output current is defined as $I_{(OUTx\_default)}$ which has a typical value of 10 mA.

When a fault is detected, the [REF_FAULT_FLAG] bit is set to HIGH. The [REF_FAULT_FLAG] bit remains latched even if the fault is removed. To clear the fault, issue the RESET_STATUS command after the fault is removed.

Example 9, which follows, shows a REF open and short detection example.

```
SPI_Write (0x67, 0x01);                 // Set the weak supply threshold to 2.77V for 3.3V
                                        supply
STATUS0 = SPI_Read(0xA2; 0x00)          // Read STATUS0 register value
If (STATUS0>>2) && Bit 0 == 1;          // There is a weak supply fault.
```

2.5.6 PWM Input Monitor

The TLC6C5712-Q1 PWM input can monitor the input PWM signal. Each PWMx input channel has an independent rising-edge triggered timer. The timer starts counting from 0, when the timer length reaches the threshold t_{PWM} 20 ms, the [PWM_FAULTx] register bit is set to HIGH. The PWM rising edge resets the timer and restarts counting from 0. So, if the input PWM frequency is less than 50 Hz, or if the PWM input duty cycle is 0% or 100%, the PWM fault register reports a PWM fault.

Example 10, which follows, shows a PWM input monitor example.

```
SPI_Write (0x60, 0x00);                  // Unmask the PWM input fault
STATUS0 = SPI_Read(0xA2; 0x00)           // Read STATUS0 register value
If (STATUS0>>3) && Bit0 == 1;            // There is a fault on PWM input
PWM_Fault = SPI_Read(0xA1; 0x00)         // Read PWM Fault register value
If (PWM_Fault>>1) && Bit0 == 1;          // There is a fault on PWM1 input
SPI_Write (0x62, 0x66);                  // Reset status to clear register bits when fault is
                                         removed
```

2.5.7 Register Lock and Unlock

To avoid an unintended change of critical registers, register locking and unlocking functions are provided. When the registers are locked, the registers cannot be overwritten until an unlock command is issued. But, when the registers are locked, the registers are still available for reading.

Example 11, which follows, shows a example of a register lock and unlock.

```
SPI_Write (0x68, 0xA5);    // Lock PWM mapping register write function
SPI_Write (0x6C, 0xCC);    // Unlock PWM mapping register write function
SPI_Write (0x69, 0x55);    // Lock dot correction register write function
SPI_Write (0x6D, 0x33);    // Unlock dot correction register write function
SPI_Write (0x6A, 0xAA);    // Lock mask register write function
SPI_Write (0x6E, 0x3C);    // Unlock mask register write function
SPI_Write (0x6B, 0x5A);    // Lock misc command register write function
SPI_Write (0x6F, 0XC3);    // Unlock misc command register write function
```

2.6 ERROR Flag

2.6.1 Fault Mask

The $\overline{\text{ERR}}$ pin can be used to monitor the fault on the device. The fault mask registers are implemented in the TLC6C5712-Q1 device which can prevent undesired faults reported to $\overline{\text{ERR}}$ pin. If the fault must be reported to $\overline{\text{ERR}}$ pin, the corresponding fault mask registers should be set to LOW. Figure 8 shows the detailed relationship between the fault register and the fault mask register.

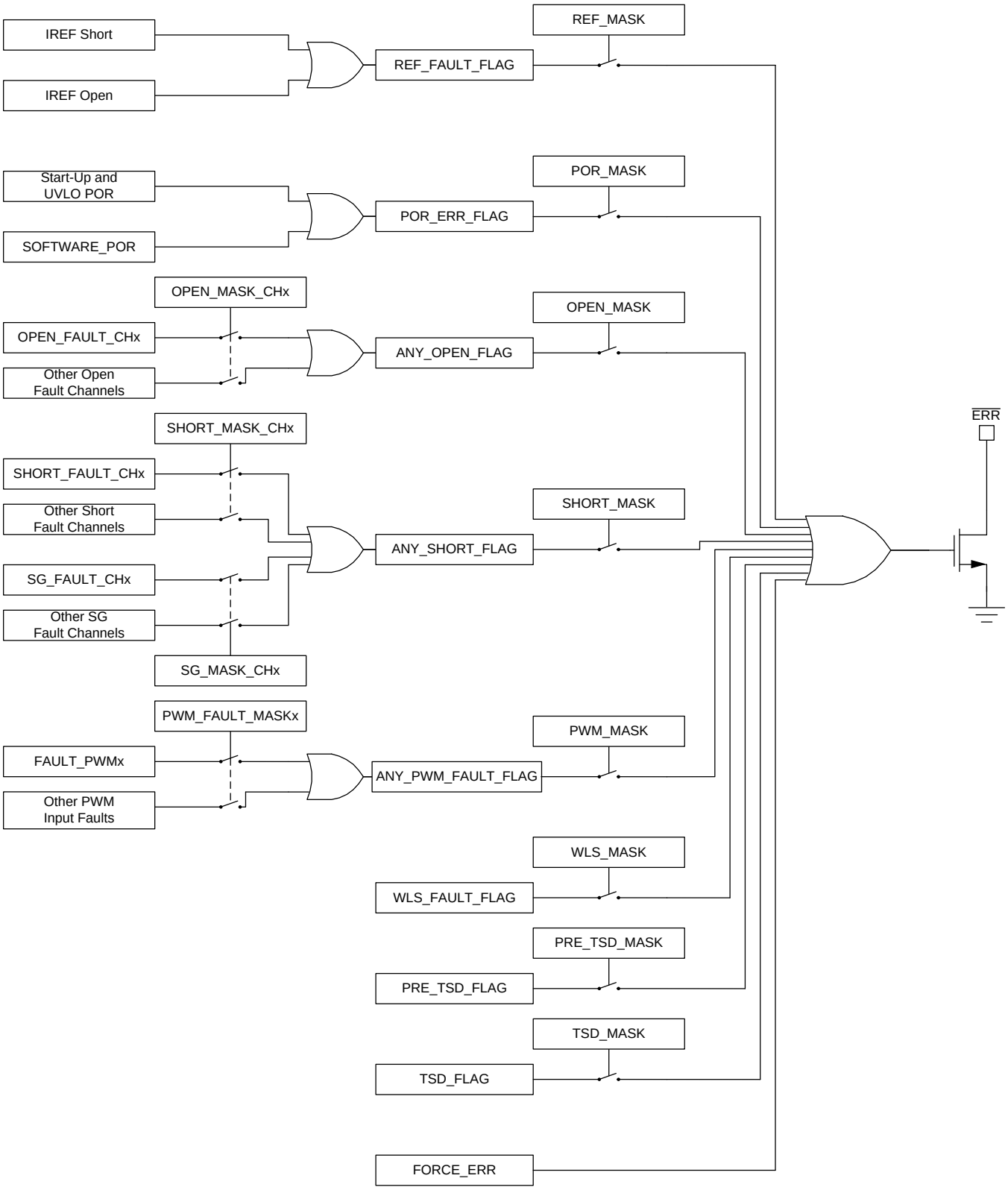

Figure 8. Fault Mask Diagram

2.6.2 Fault Capture

The $\overline{\text{ERR}}$ pin is an open drain structure, typically a pullup resistor (a value of 10 kΩ is typically sufficient) is used to set the $\overline{\text{ERR}}$ pin to high level. A microcontroller (MCU or µC) can capture a TLC6C5712-Q1 error by detecting the voltage on the $\overline{\text{ERR}}$ pin. When the TLC6C5712-Q1 device is in the fault state, the $\overline{\text{ERR}}$ pin is pulled down to low, and the MCU can capture the fault and then execute the fault handling routine. Figure 9 shows the hardware connection between the MCU and the TLC6C5712-Q1 device.

NOTE: Set the corresponding fault mask bits low to make sure the corresponding faults report to the $\overline{\text{ERR}}$ pin. For the channel-activated state (CH_ON_MASKx register bits low) fault detection, the PWM duty cycle should have at least 5-µs LED on time and 40-µs LED off time for LED full diagnostics.

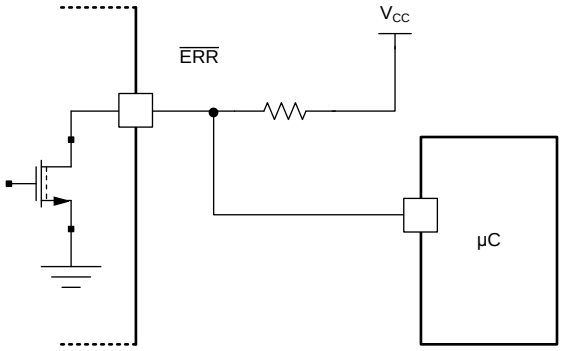

Figure 9. $\overline{\text{ERR}}$ Pin Configuration

2.6.3 Error Recovery

When any fault occurs, all FAULT information can be read from the corresponding fault registers. When the error is removed, the register information is still latched and the $\overline{\text{ERR}}$ pin remains low until the fault is masked or the RESET_STATUS command has been issued. However, if the error condition still exists after issuing the RESET_STATUS command, the $\overline{\text{ERR}}$ pin pulls low after the deglitch time and the corresponding FAULT register is set HIGH again.

Example 12, which follows, shows a $\overline{\text{ERR}}$ recovery example.

```
SPI_Write (0x62, 0x66);          // Reset status to clear register bits when fault is removed
```

2.6.4 Force ERROR

To check the connection between the MCU and the TLC6C5712-Q1 device, a force-error function is implemented in the TLC6C5712-Q1 device. The [FORCE_ERR] register is provided to enable an $\overline{\text{ERR}}$ state to simulate a fault. When [FORCE_ERR] is set HIGH, the $\overline{\text{ERR}}$ open-drain output is pulled down. Change the [FORCE_ERR] bit to low will remove the force error.

Example 13, which follows, shows a force error example.

```
SPI_Write (0x67, 0x02);          // Generate an error to check the error feedback connection
SPI_Write (0x67, 0x00);          // Clear the force error to check the error feedback connection
```

3 Software Guidance

This section describes a software guide when using the TLC6C5712-Q1 device. A flow chart is discussed in this section. The pseudocodes about the device initialization and configuration are provided at the end of the section.

3.1 Flow Chart

This section describes a flow chart of the TLC6C5712-Q1 configuration. After power up, the MCU must initialize the TLC6C5712-Q1 registers, which includes checking the faults, performing the power-on reset (POR), initializing the registers, and checking the $\overline{ERR}$ feedback circuit by force $\overline{ERR}$. Then MCU enables an interrupt to capture the fault on the TLC6C5712-Q1 device. Adjacent pin short-detection function is optional. If adjacent pin short detection is needed, it can be realized by writing register [0x67 0x08]. Finally a main routine occurs which includes the general registers settings such as the LED current setting, LED channel on or off, PWM dimming, and other LED configurations. When the MCU captures an $\overline{ERR}$ pin interrupt, the main routine enters the interrupt service routine. The fault is handled in this routine. After the fault handling routine finishes, the MCU goes back to main routine.

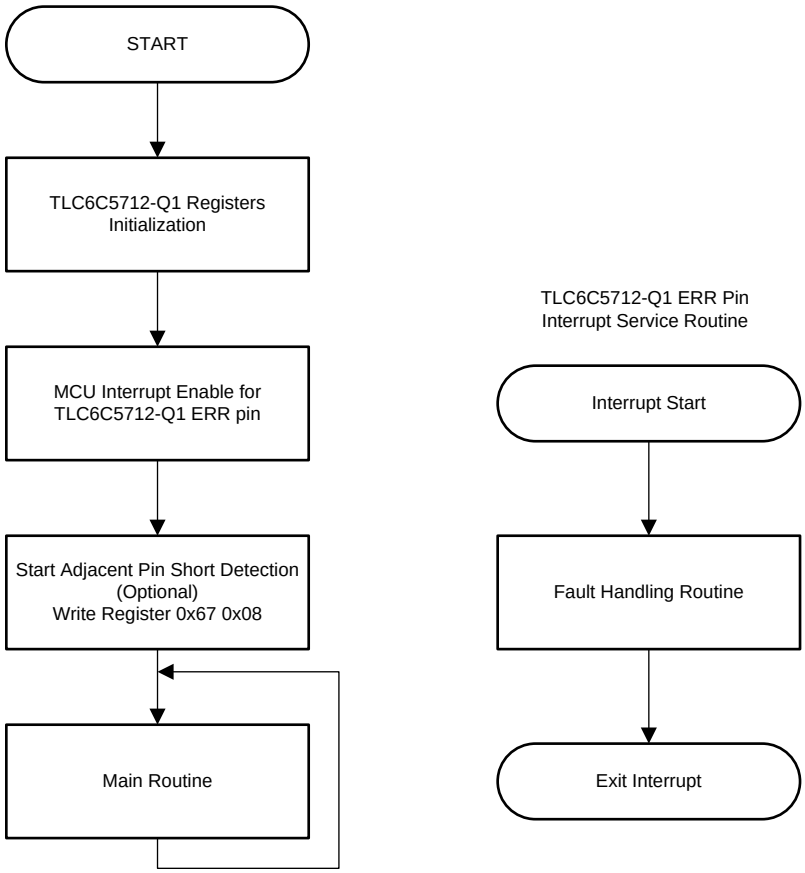

Figure 10. Flow Chart Using Interrupt

Aside from using the MCU interrupt to capture the error, another way to detect the fault is checking the $\overline{ERR}$ pin voltage periodically, as shown inFigure 11. This method requires more time to detect the error compared to using the MCU interrupt method.

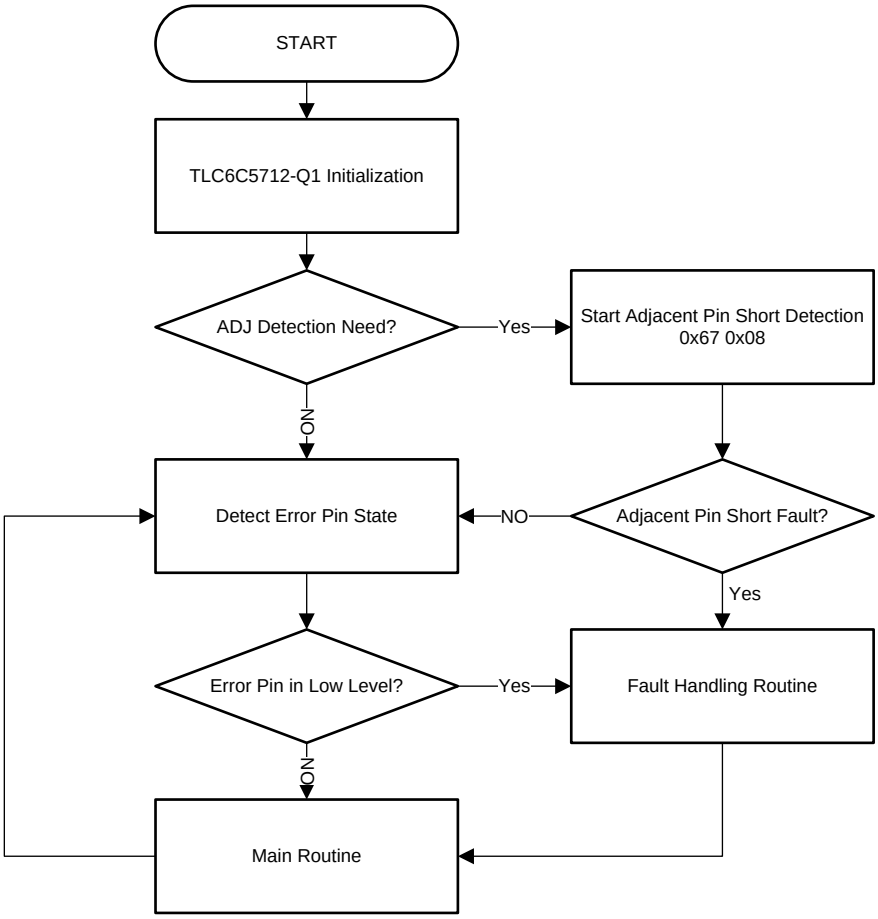

Figure 11. Flow Chart by Checking Error Pin Status Periodically

3.2 Start-Up Sequence

When the TLC6C5712-Q1 device is powered up, all the internal registers have default values. The following lists the typical TLC6C5712-Q1 initialization sequence after power-up:

1. Power up the system.
2. Set all fault mask registers value HIGH to mask all faults.
3. Write the [FORCE_ERR] register to check the $\overline{\text{ERR}}$ feedback circuit connection.
4. Set the corresponding fault mask registers bit LOW to make the required faults report to the $\overline{\text{ERR}}$ pin. (Mask the PWMx fault if it is not used in the application.)
5. Clear the [RESET_POR] POR flag.
6. Reset the fault status [RESET_STATUS] registers.
7. Read the [READ_STATUS] registers or $\overline{\text{ERR}}$ pin state confirm fault information, if a fault exists, then enter the fault handling routine.
8. Set the [ADJ_DIAG_START] bit HIGH to start the adjacent pin short detection.
9. Read the adjacent pin short-fault register to confirm the status of the adjacent pin short fault, if a fault exists, enter the fault handling routine.
10. Set the output current by writing the [WRITE_CORRx] registers.
11. Verify the current dot correction by reading the [READ_CORRx] registers.
12. Configure the PWM mapping by writing the [WRITE_MAPx] registers.
13. Verify the PWM mapping by reading the [READ_MAPx] registers.
14. Write the [LOCK_MAP] command to lock the PWM mapping settings if required.
15. Configure the PWM input signal for LED fault diagnostics.
16. Turn on or turn off output channels by writing the [WRITE_CH_ON_MASKx] registers.
17. Verify channel on or channel off states by reading the [READ_CH_ON_MASKx] registers.
18. Adjust the PWM duty cycle for dimming if required.
19. Enter the main routine.

3.3 Fault Handling Rountine

During normal operation, the MCU monitors the state of the TLCC65712-Q1 $\overline{\text{ERR}}$ pin, when a fault occurs on the TLC6C5712-Q1 device, the MCU can detect the fault. The MCU enters the fault handling routine and finds the corresponding fault by checking the status of the registers. The MCU then acts to handle the faults.

For example, when the TLCC65712-Q1 reports a PRE-TSD fault, the controller can reduce device power dissipation such as reducing PWM duty cycle, reduce dot correction settings, or turn off channels to prevent thermal shutdown. Figure 12 shows the detailed flow chart of the fault handling routine.

When the fault status is removed, issue the RESET_STATUS command to clear the fault.

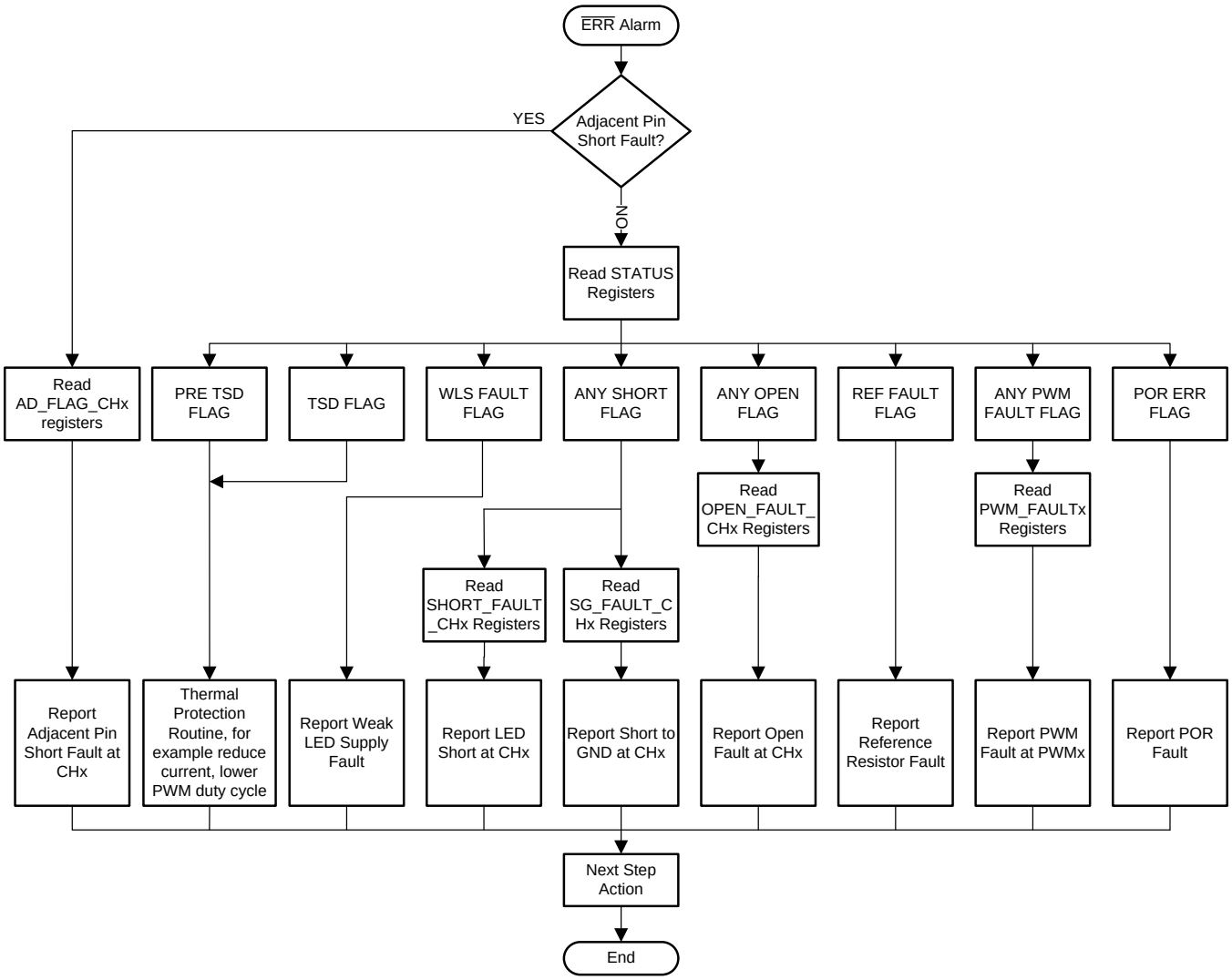

Figure 12. Fault Handling Routine

3.4 Pseudocodes

This section describes the pseudocodes on how to realize the full diagnostics of the TLC6C5712-Q1 device.

MCU initialization and I/O configuration	// MCU configuration
Enable the ERR pin interrupt	// MCU configuration
SPI_Write (0x66 0x3F)	// Mask all faults
SPI_Write (0x67 0x02)	// Set force ERR register value to HIGH
SPI_Write (0x67 0x00)	// Set force ERR register value to LOW ERR pin level checking Make sure the ERR pin and MCU I/O connection is good.
SPI_Write (0x61 0x69)	// Power on Reset
SPI_Write (0x62 0x66)	// Reset Status to clear pre-exiting faults
SPI_Read (0xA2 0x00)	// Check the status register to confirm no faults
SPI_Read (0xA3 0x00)	// Check the status register to confirm no faults
SPI_Write (0x54 0x00)	// Unmask CH0-CH5 SHORT Fault

```
SPI_Write (0x55 0x00)          // Unmask CH6-CH11 SHORT Fault
SPI_Write (0x56 0x00)          // Unmask CH0-CH5 SHORT GND Fault
SPI_Write (0x57 0x00)          // Unmask CH6-CH11 SHORT GND Fault
SPI_Write (0x58 0x00)          // Unmask CH0-CH5 OPEN Fault
SPI_Write (0x59 0x00)          // Unmask CH6-CH11 OPEN Fault
SPI_Write (0x60 0x3E)          // Unmask PWM0 Fault, if other PWM inputs are used, unmask
                               corresponding fault mask
SPI_Write (0x66 0x00)          // Unmask ERROR_MASK
SPI_Write (0x67 0x08)          // Adjacent pin short pin start
SPI_Read (0xA8 0x00)           // Check if there is any adjacent pin short fault on CH0-CH5
SPI_Read (0xA9 0x00)           // Check if there is any adjacent pin short fault on CH6-CH11
SPI_Write (0x46 0x7F)          // Set CH0 output current to half scale
SPI_Read (0x86 0x00)           // Check if the CH0 dot correction registers value is correct
SPI_Write (0x47 0x7F)          // Set CH1 output current to half scale
SPI_Read (0x87 0x00)           // Check if the CH1 dot correction registers value is correct
…
SPI_Write (0x51 0xFF)          // Set CH11 output current to full scale
SPI_Read (0x91 0x00)           // Check if the CH11 dot correction registers value is correct
SPI_Write (0x69 0x55)          // Lock dot correction registers
SPI_Write (0x40 0x08)          // Map PWM0 to CH0, Map PWM1 to CH1
SPI_Read (0x80 0x00)           // Check if PWM Mapping register values are correct
SPI_Write (0x41 0x00)          // Map PWM0 to CH2, Map PWM0 to CH3
SPI_Read (0x81 0x00)           // Check if PWM Mapping register values are correct
…
SPI_Write (0x68 0x35)          // Lock Map registers
Configure the PWM input
frequency & duty cycle
SPI_Write (0x52 0x00)          // Turn on the CH0-CH5
SPI_Read (0x92 0x00)           // Read Channel on mask state of CH0-CH5
SPI_Write (0x53 0x00)          // Turn on the CH6-CH11
SPI_Read (0x93 0x00)           // Read Channel on mask state of CH6-CH11
Adjust PWM duty cycle for
dimming if needed
…
```

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265