# i.MX31 SOM-LV GPIO Configuration and Control

Application Note 359

Logic Product Development
Published: December 2008

## Abstract

This application note describes how to use software and hardware to configure and control GPIO signals on the i.MX31 SOM-LV.

# Revision History

| REV | EDITOR | DESCRIPTION | APPROVAL | DATE |
|---|---|---|---|---|
| A | GLJ | Initial release | GLJ | 12/12/08 |

  

# Table of Contents

**1 Introduction ..... 1**
**2 i.MX31 On-Chip GPIOs ..... 1**
2.1 IOMUX Registers ..... 1
2.1.1 GPR Register ..... 1
2.1.2 SW_MUX_CTL Register ..... 1
2.1.3 SW_PAD_CTL Register ..... 2
2.2 GPIO Registers ..... 2
2.3 Example GPIO Configuration ..... 3
2.4 i.MX31 SOM-LV LED GPIOs ..... 5
**3 Special i.MX31 SOM-LV GPIOs through the LAN9117 Ethernet Controller ..... 7**
**4 Summary ..... 8**
**Appendix A: KEYPAD GPIO Script ..... 9**
**Appendix B: LED GPIO Script ..... 10**
**Appendix C: Windows CE LED GPIO Code Example ..... 11**
**Appendix D: References ..... 13**

  

# 1 Introduction

The purpose of this application note is to describe the General Purpose Input/Output (GPIO) signals on the i.MX31 SOM-LV that can be accessed and controlled through hardware and software configurations. Many pins on the i.MX31 processor are multi-function pins (MFP) that require the developer to decide which will be available for GPIO functionality. Examples of specific GPIOs will be illustrated in this application note.

Access and control of i.MX31 on-chip GPIO functionality is shared between the IOMUX and GPIO modules. Additionally, there are GPIO signals routed through other devices on the i.MX31 SOM-LV; these GPIO signals are designated within this document as “special” since they require further configuration and software control for operation. Special GPIO signals are described in Section 3.

Examples of hardware configuration and software scripts are given in the appendices of this document. Adaptation to other GPIO signals should be possible using these examples; however, the developer is responsible for understanding the unique requirements for each GPIO signal pad and not overwriting existing functionality when reconfiguring for GPIO use.

Technical references to the i.MX31 processor registers can be found in Freescale’s *MCIMCX31 and MCIMX31L Applications Processors Reference Manual* (document number MCIMX31RM, Rev. 2.3). A list of other documents referenced in this application note can be found in “Appendix D: References.”

# 2 i.MX31 On-Chip GPIOs

On-chip GPIO signals are routed directly from the i.MX31 to the expansion bus header pins. Each external signal pad of the i.MX31 can be configured with one of ten possible modes: two hardware modes, the main function mode, six alternate modes, and the GPIO mode. The GPIO mode is the focus of this application note.

The i.MX31 IOMUX and GPIO logic blocks contain registers used in GPIO control. Other registers found in these two logic blocks are not covered within this document since they are not relevant to GPIO control.

## 2.1 IOMUX Registers

The i.MX31 IOMUX logic block contains three registers used for GPIO configuration: GPR, SW_MUX_CTL, and SW_PAD_CTL.

### 2.1.1 GPR Register

The GPR register remains at the default values (all zero) since it is only used for hardware mode pad settings.

### 2.1.2 SW_MUX_CTL Register

There are 82 SW_MUX_CTL registers. Each register describes the function and direction of four signal pads with each pad containing an 8-bit field. The bits for a read and a write are illustrated in Figure 2.1. Please refer to the *MCIMX31RM* for address locations of specific signal pad fields.

  

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | Res | SW_MUX_CTL_SIGNAL_4 | | | | | | | Res | SW_MUX_CTL_SIGNAL_3 | | | | | | |
| Write | | SW_MUX_OUT_EN | | | MUX2_IN | MUX1_IN | FUNC_IN | GPIO PSR/ ISR | | SW_MUX_OUT_EN | | | MUX2_IN | MUX1_IN | FUNC_IN | GPIO_PSR/ ISR |
| Field bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | 15 | 14 | 29 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | Res | SW_MUX_CTL_SIGNAL_2 | | | | | | | Res | SW_MUX_CTL_SIGNAL_1 | | | | | | |
| Write | | SW_MUX_OUT_EN | | | MUX2_IN | MUX1_IN | FUNC_IN | GPIO PSR/ ISR | | SW_MUX_OUT_EN | | | MUX2_IN | MUX1_IN | FUNC_IN | GPIO_PSR/ ISR |
| Field bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

***Figure 2.1: SW_MUX_CTL Register***

**Note:** The figure above comes from an example in Freescale's *MCIMX31RM*.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | SW_MUX_OUT_EN | | | SW_MUX_IN_EN | | | |

***Figure 2.2: Fields for Each Signal Pad in SW_MUX_CTL***

To configure SW_MUX_OUT_EN as a GPIO output, enter the bit value 0b000.

To configure SW_MUX_IN_EN as "no inputs" selected (disabled), enter the bit value 0b0000. This setting is used for GPIO output only.

To configure SW_MUX_IN_EN as a GPIO PSR/ISR input, enter the bit value 0b0001. This setting is used for either GPIO input only or GPIO input/output.

Each GPIO signal pad will have a unique set of valid bits for output and input configurations.

**Note:** The field SW_MUX_IN_EN shown in Figure 2.2 is an unofficial combination of the input configuration bits. Please refer to Table 4-6 in the *MCIMX31RM* for SW_MUX_CTL bit configuration descriptions.

### 2.1.3 SW_PAD_CTL Register

The SW_PAD_CTL registers do not need to be altered for GPIO signal pad configuration.

## 2.2 GPIO Registers

The i.MX31 GPIO logic block contains three registers for GPIO configuration: Data Register (DR), GPIO Data Direction Register (GDIR), and Pad Sample Register (PSR).

Each register has three GPIOn block designations: GPIO1, GPIO2, and GPIO3.

  

Every bit in the GPIO register relates to one signal pad on the i.MX31; the correlation of the specific signal pad to GPIO function is found in Table 4-8 of the *MCIMX31RM*. Within that table, the “GPIO Mode” column uses the naming scheme MCUn_x, where “n” relates to the GPIOn and the “x” relates to the pin number of the specified GPIO register (Table 2.1 provides some examples).

***Table 2.1: Examples for MCUn_x to GPIO Register Pins***

| MCUn_x* | GPIOn | Bit # | Signal Name |
|---|---|---|---|
| MCU1_0 | GPIO1 | 0 | GPIO1_0 |
| MCU1_4 | GPIO1 | 4 | GPIO1_4 |
| MCU1_27 | GPIO1 | 27 | RXD2 |
| MCU2_18 | GPIO2 | 18 | KEY_ROW4 |
| MCU2_19 | GPIO2 | 19 | KEY_ROW5 |
| MCU2_20 | GPIO2 | 20 | KEY_ROW6 |
| MCU3_4 | GPIO3 | 4 | CSI_D4 |
| MCU3_17 | GPIO3 | 17 | CSI_VSYNC |

***Note:** The “MCUn_x” column comes from Table 4-8 in Freescale’s *MCIMX31RM*.

There are three GPIO Data Registers (DR) that hold data output to the signal pad; these registers are configured as outputs.

There are three GPIO Data Direction Registers (GDIR) that control the direction of the signal pad. To configure the signal pad as an input, set the corresponding signal pad bit to “0” (zero). To configure the signal pad as an output, set the corresponding signal pad bit to “1”.

There are three GPIO Pad Sample Registers (PSR) that receive the signal pad values when configured as an input.

The base addresses of the GPIO registers are shown in Table 2.2.

***Table 2.2: GPIOn Register Base Addresses***

| Register | GPIO1 | GPIO2 | GPIO3 |
|---|---|---|---|
| DR | 0x53FC_C000 | 0x53FD_0000 | 0x53FA_4000 |
| GDIR | 0x53FC_C004 | 0x53FD_0004 | 0x53FA_4004 |
| PSR | 0x53FC_C008 | 0x53FD_0008 | 0x53FA_4008 |

## 2.3 Example GPIO Configuration

This section will provide an example of register settings required to configure a specific signal pad as a GPIO output pin, KEY_ROW5. From Table 2.1, KEY_ROW5 relates to bit 19 in GPIO2 registers.

Software examples will be presented in the form of LogicLoader “Losh>” commands. These can be translated into application code by the developer. A LogicLoader script for this example is provided in the appendices of this document.

**Step 1:** Set the proper SW_MUX_CTL register for KEY_ROW5.

  

The SW_MUX_CTL register at address 0x43FA_C068 contains the field “sw_mux_ctl_key_row5” in signal position 3 covering bits [22:16]. This register is structured as shown in Figure 2.3.

**Address:** 0x43FA_C068

**SW_MUX_CTL**

| bits: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | | sw_mux_ctl_key_row4 | | | | | | | | sw_mux_ctl_key_row5 | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| bits: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | | sw_mux_ctl_key_row6 | | | | | | | | sw_mux_ctl_key_row7 | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

***Figure 2.3: SW_MUX_CTL Register for Key_Row5***

The default value for this register (0x0101_0101) needs to be changed to 0x0100_0101 to reflect the output only for KEY_ROW5. The write command to achieve this is:

```
losh> w /w 0x43FAC068  0x01000101
```

The register value can be examined with:

```
losh> x /w 0x43FAC068
```

**Step 2:** Set the GPIO data direction to output for signal pad KEY_ROW5.

The GPIO2 GDIR register bit 19 needs to be set to “1” to configure KEY_ROW5 as an output. This register defaults to 0x0000_0000 and, for this example, needs to be changed to 0x0008_0000. The write command to achieve this is:

```
losh> w /w 0x53FD0004  0x00080000
```

The register value can be confirmed with:

```
losh> x /w 0x53FD0004
```

**Step 3:** Alter the value in the GPIO data register for the KEY_ROW5 signal pad.

The GPIO2 DR register bit 19 can be toggled and confirmed with a register read or by testing the output voltage on the i.MX31 SOM-LV breakout board. Signal tracing to the breakout board is shown in Table 2.3.

With the breakout board header pin J16.31 attached to J15.34, the output of signal pad KEY_ROW5 will provide an input for signal pad KEY_ROW6. Observe that the KEY_ROW6 signal pad keeps its default input setting with the SW_MUX_CTL configuration setting and that the direction remains as an input, from settings above. The

  

resulting input into KEY_ROW6 can be observed in the GPIO2 PSR register by inspecting pin 20 (0x0010_0000) with:

```
losh> x /h 0x53FD0008
```

The KEY_ROW5 output is toggled by the GPIO2 data register with the following command:

```
losh> w /w 0x53FD0000  0x00080000
```

and

```
losh> w /w 0x53FD0000  0x00000000
```

***Table 2.3: Hardware Access Pins***

| i.MX31 SOM-LV | | | Breakout Board | |
|---|---|---|---|---|
| **i.MX31 Register Bit** | **i.MX31 Pin** | **SOM Bus Name** | **Bus Name** | **Header.Pin** |
| GPIO2_18 | MCU2_18 | KEY_ROW4 | uP_GPIO6 | J16.75 |
| | | | | J16.32 |
| GPIO2_19 | MCU2_19 | KEY_ROW5 | uP_GPIO7 | J16.76 |
| | | | | J16.31 |
| GPIO2_20 | MCU2_20 | KEY_ROW6 | MFP_D59 | J15.34 |

**Note:** While GPIO direction is set to input (GDIR = 0), a read access to DR does not return DR data. Instead, it returns the PSR data, which is the corresponding signal pad value.

The KPP keypad registers do not need to be altered, since the GPIO configurations will override the keypad functionality.

As an exercise, repeat steps 1 through three to configure KEY_ROW4 as an output.

## 2.4 i.MX31 SOM-LV LED GPIOs

The ability to alter outputs through the onboard LEDs (GPIO0 [D3] and GPIO1 [D4]) provide a quick feedback for GPIO control. These baseboard LED GPIOs are the focus of this section.

**Step 1:** Set the proper SW_MUX_CTL register for GPIO1_7 (CAPTURE signal pad) and GPIO1_8 (COMPARE signal pad).

The SW_MUX_CTL register at address 0x43FA_C150 contains the field “sw_mux_ctl_capture” in signal position 4, bits [30:24], and contains the field “sw_mux_ctl_compare” in signal position 3, bits [22:16]. This register is structured as shown in Figure 2.4.

  

**Address:** 0x43FA_C150

**SW_MUX_CTL**

| bits: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | | sw_mux_ctl_capture | | | | | | | | sw_mux_ctl_compare | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| bits: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Read | | sw_mux_ctl_watchdog_rst | | | | | | | | sw_mux_ctl_pwm0 | | | | | | |
| Write | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

***Figure 2.4: SW_MUX_CTL Register for CAPTURE and COMPARE***

The default value for this register (0x0101_0001) needs to be changed to 0x0000_0001 to reflect the output only for the CAPTURE and COMPARE signal pads. The write command to achieve this is:

```
losh> w /w 0x43FAC150  0x00000001
```

The register value can be examined with:

```
losh> x /w 0x43FAC150
```

The GPIO1 hardware signals for GPIO_1 and GPIO_0 are traced from the i.MX31 SOM-LV and baseboard schematics and are listed in Table 2.4.

***Table 2.4: GPIO LED Hardware Access Pins***

| **i.MX31 SOM-LV** | | | **Baseboard** | **Breakout Board** |
|---|---|---|---|---|
| **i.MX31 Register Bit** | **i.MX31 Pin** | **SOM Bus Name** | **Bus Name** | **Header.Pin** |
| GPIO1_7 | MCU1_7 | CAPTURE | uP_GPIO_1 | J18.4 |
| GPIO1_8 | MCU1_8 | COMPARE | uP_GPIO_0 | J17.4 |

**Step 2:** Set the GPIO data direction to output for signal pads GPIO_1 and GPIO_0.

The GPIO1 GDIR register bits 7 and 8 need to be set to “1” to configure GPIO_1 and GPIO_0 as outputs. This register defaults to 0x0000_0000 and, for this example, needs to be changed to 0x0000_0180. The write command to achieve this is:

```
losh> w /w 0x53FCC004  0x00000180
```

The register value can be confirmed with:

```
losh> x /w 0x53FCC004
```

**Step 3:** Alter the value in the GPIO data register for the GPIO_1 and GPIO_0 signal pads.

  

The GPIO1 DR register bits 7 and 8 can be toggled and confirmed with a register read or by testing the output voltage on the i.MX31 SOM-LV breakout board.

The GPIO_1 and GPIO_0 outputs are toggled by the GPIO1 data register with variations on the following bit-set commands:

(Both ON) **`losh> w /w 0x53FCC000  0x00000180`**

and

(Both OFF) **`losh> w /w 0x53FCC000  0x00000000`**

# 3 Special i.MX31 SOM-LV GPIOs through the LAN9117 Ethernet Controller

Some GPIO signals may be routed through other devices on the SOM. This section describes software examples and hardware configurations required to access and control these signals. The ability to alter GPIO outputs for signals routed through the LAN9117 Ethernet Controller require an address for chip-select 4 (CS4) and a register offset setting in the LAN9117. The LAN9117 GPIO1 and GPIO2 signal pads (pins 98 and 99) are the focus of this section.

The LAN9117 is selected with the i.MX31 CS4 active and with address bit A23 low (see the *i.MX31 SOM-LV Hardware Specification*). This will drive the WRLAN_nMCS select signal on the i.MX31 SOM-LV. The memory map location for CS4 is located at 0xB400_0000 (see the *i.MX31 SOM-LV LoCE User Manual Addendum*).

The LAN9117 GPIO configuration register (GPIO_CFG) controls the outputs for the LAN9117 LEDs. These LEDs are located on the Zoom i.MX31 LITEKIT baseboard as LAN1 (D1) and LAN2 (D2). The offset for the GPIO_CFG register is 0x88. Bits of concern in this register are listed in Table 3.1.

***Table 3.1: LAN9117 GPIO_CFG for GPIO***

| Bit Field | Function | Set For GPIO |
|---|---|---|
| 31 | Not necessary for this example | 0 |
| 30:28 | GPIO0 – bit 28 0 = GPIO function<br>GPIO1 – bit 29 1 = LED function<br>GPIO2 – bit 30 (not used) | 000 |
| 26:19 | Not necessary for this example | All zeros |
| 18:16 | GPIO0 – bit 16 0 = Push/Pull<br>GPIO1 – bit 17 1 = Open Drain<br>GPIO2 – bit 18 (not used) | 011 |
| 15:11 | Not necessary for this example | All zeros |
| 10:8 | GPIO0 – bit 9 0 = Input<br>GPIO1 – bit 8 1 = Output<br>GPIO2 – bit 10 (not used) | 011 |
| 7:3 | Not necessary for this example | All zeros |
| 2:0 | GPIO0 – bit 0 DATA: 0 = LED On<br>GPIO1 – bit 1 1 = LED Off<br>GPIO2 – bit 2 (not used) | 011, 010,<br>001, 000 |

The GPIO0 and GPIO1 outputs are controlled by the variations on bits 1:0 and the following bit-set commands (please see page 87 in SMSC's *LAN9117 Ethernet Controller Datasheet*):

(Both LEDs ON) **`losh> w /w 0xB4000088  0x00030300`**

and

  

(Both LEDs OFF) **`losh> w /w 0xB4000088  0x00030303`**

# 4 Summary

The GPIO signals of the i.MX31 SOM-LV can be accessed and controlled with the examples given in this application note. The specific examples can be adapted to other GPIO signals as long as the developer gives proper consideration to the unique requirements of each GPIO signal pad.

  

# Appendix A: KEYPAD GPIO Script

The following script will configure the KEY_ROW5 and KEY_ROW6 signal pads as GPIOs. To use this script: save the script on a CompactFlash card, insert the CompactFlash card into the slot on the Zoom i.MX31 LITEKIT baseboard, and then run in LogicLoader. The script steps are:

1. Set the signal pad mux to GPIO for the two signal pads
2. Configure the GPIO2_19 (KEY_ROW5) as an output and GPIO2_20 (KEY_ROW6) as an input
3. Set GPIO2_19 to “1”
4. Set GPIO2_19 to “0”
5. Set GPIO2_19 to “1”

SCRIPT:

```
**************************
w /w 0x43FAC068  0x01000101
w /w 0x53FD0004  0x00180000
w /w 0x53FD0000  0x00080000
w /w 0x53FD0000  0x00000000
w /w 0x53FD0000  0x00080000
**************************
```

The resulting bit settings from the script can be observed in the Tera Term command window by adding “examine” commands as described in Section 2.3. Also, the input GPIO2_20 PSR register can be observed as reflecting the GPIO2_19 outputs when connected on the breakout board as described in Section 2.3.

  

# Appendix B: LED GPIO Script

The following script will configure the LED GPIO0 and GPIO1 signal pads as GPIOs. To use this script: save the script on a CompactFlash card, insert the CompactFlash card into the slot on the Zoom i.MX31 LITEKIT baseboard, and then run in LogicLoader. The script steps are:

1. Set the signal pad mux to GPIO for the two signal pads
2. Configure the GPIO1_7 (baseboard LED 1) and GPIO1_8 (baseboard LED 0) as outputs
3. Set GPIO1_7 and GPIO1_8 to “1”
4. Set GPIO1_7 and GPIO1_8 to “0”
5. Set GPIO1_7 to “1” and GPIO1_8 to “0”
6. Set GPIO1_7 to “0” and GPIO1_8 to “1”
7. Set GPIO1_7 to “0” and GPIO1_8 to “0”

SCRIPT:

```
**************************
w /w 0x43FAC150  0x00000101
w /w 0x53FCC004  0x00000180
w /w 0x53FCC000  0x00000180
w /w 0x53FCC000  0x00000000
w /w 0x53FCC000  0x00000080
w /w 0x53FCC000  0x00000100
w /w 0x53FCC000  0x00000000
**************************
```

The resulting bit settings from the script can be observed in the Tera Term command window by adding “examine” commands as described in Section 2.4.

  

# Appendix C: Windows CE LED GPIO Code Example

Sample Windows CE code is shown below. This code is for example purposes only and does not include any bit testing or other production code essentials. The Windows CE example requires a minimum of wrapper code (include files, etc.) not discussed here. The following script will configure the LED GPIO0 and GPIO1 signal pads as GPIOs. To use this script: save the script on a CompactFlash card, insert the CompactFlash card into the slot on the Zoom i.MX31 LITEKIT baseboard, and then run in LogicLoader.

The steps are:

1. Set the signal pad mux to GPIO for the two signal pads
2. Configure the GPIO1_7 (baseboard LED 1) and GPIO1_8 (baseboard LED 0) as outputs
3. Set GPIO1_7 and GPIO1_8 to “1”
4. Set GPIO1_7 and GPIO1_8 to “0”

   Alternating the GPIO bits (DR[8:7]) will vary the GPIO LEDs

SCRIPT:

```
*************************
//PROGRAM NAME: RW_LED_GPIO.cpp
//PURPOSE: To read and write to the GPIO baseboard LEDs from registers in the iMX31 SOM
//STATUS:  Non-supported code.
//
#include "stdafx.h"

int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPTSTR    lpCmdLine,
                   int       nCmdShow)
{
    PHYSICAL_ADDRESS phys_addr_mux, phys_addr0, phys_addr1;
    unsigned int *sw_mux_word;    // Pointer to SW_MUX_CTL register
    unsigned int *gpiodir_word;    // Pointer to GPIO1_GDIR register
    unsigned int *gpiodat_word;    // Pointer to GPIO1_DR register

    volatile int Address;
    volatile unsigned int Value;

// Step 1: Set SW_MUX_CTL as output GPIOs for baseboard GP100 & GP101 LED's
    Address = 0x43FAC150;
    Value = 0x00000101;
      RETAILMSG(1, (L"\n"));
      RETAILMSG(1, (L"   Address: %8x  Value: %8x \n \r",Address, Value));
    phys_addr_mux.LowPart = Address;
    phys_addr_mux.HighPart = 0;
```

  

```
    sw_mux_word = (unsigned int *)   MmMapIoSpace( phys_addr_mux, 4, FALSE );
    *sw_mux_word = (*sw_mux_word && 0x00000000);
    *sw_mux_word = Value;
       RETAILMSG(1, (L"  sw_mux_word: %8x  *sw_mux_word: %8x \n \r",sw_mux_word,
*sw_mux_word));


// Step 2: Configure baseboard GP100 & GP101 LED's as outputs
    Address = 0x53FCC004;
    Value = 0x00000180;
      RETAILMSG(1, (L"   Address: %8x  Value: %8x \n \r",Address, Value));
    phys_addr0.LowPart = Address;
    phys_addr0.HighPart = 0;
    gpiodir_word = (unsigned int *)   MmMapIoSpace( phys_addr0, 4, FALSE );
    *gpiodir_word = (*gpiodir_word && 0x00000000);
    *gpiodir_word = Value;
      RETAILMSG(1, (L"  gpiodir_word: %8x  *gpiodir_word: %8x \n \r",gpiodir_word,
*gpiodir_word));


// Step 3: Set Pointer and set Value for GPIO1_7 and GPIO1_8 to "1" (GP100 & GP101 LED's)
    Address = 0x53FCC000;
    Value = 0x00000180;
      RETAILMSG(1, (L"   Address: %8x  Value: %8x \n \r",Address, Value));
    phys_addr1.LowPart = Address;
    phys_addr1.HighPart = 0;
    gpiodat_word = (unsigned int *)   MmMapIoSpace( phys_addr1, 4, FALSE );
    *gpiodat_word = (*gpiodat_word && 0x00000000);
    *gpiodat_word = Value;
       RETAILMSG(1, (L"  gpiodat_word: %8x  *gpiodat_word: %8x \n \r",gpiodat_word,
*gpiodat_word);
    Sleep(1000);  // Provide a short time for observing the LED's.


// Step 4: Set Values for GPIO1_7 and GPIO1_8 to "0" (GP100 & GP101 LED's)
     Value = 0x00000000;
       RETAILMSG(1, (L"  Value: %8x \n \r",Value));
    *gpiodat_word = Value;
       RETAILMSG(1, (L"  gpiodat_word: %8x  *gpiodat_word: %8x \n \r",gpiodat_word,
*gpiodat_word));
    Sleep(1000);  // Provide a short time for observing the LED's.
    return 0;
}
```

  

# Appendix D: References

The following documents are referenced within this application note and are available on their respective company's website:

- Logic's *i.MX31 SOM-LV Hardware Specification* (1005992 Rev F)
- Logic's *i.MX31 LoCE User Guide Addendum* (1008850 Rev A)
- SMSC's *LAN9117 High Performance Single-Chip 10/100 Non-PCI Ethernet Controller Datasheet* (document number SMSC LAN9117 Rev. 1.5)
- Freescale's *MCIMCX31 and MCIMX31L Applications Processors Reference Manual* (document number MCIMX31RM, Rev. 2.3)

  