VEEK-MT2S

User Manual

terasIC
www.terasic.com

CONTENTS

Chapter 1

Introduction

The VEEK-MT2S (Video and Embedded Evaluation Kit - Multi-touch, Second Edition for SoC FPGA) is a comprehensive design environment with everything embedded developers need to create processing-based systems. The VEEK-MT2S delivers an integrated platform including hardware, design tools, and reference designs for developing embedded software and hardware platforms in a wide range of applications. The fully integrated kit allows developers to rapidly customize their processor and IP to best suit their specific application. The VEEK-MT2S is assembled by the DE10-Standard FPGA main board and MTLC2 module. The DE10-Stanard FPGA board uses Cyclone® V SoC FPGA. The MTLC2 module is an 800x480 color LCD with touch panel which natively supports five point multi-touch and gestures. An 8-megapixel digital image sensor, ambient light sensor, and 3-axis accelerometer are also included in the module.

The all-in-one embedded solution offered on the VEEK-MT2S, in combination of a LCD touch panel and digital image module, provides embedded developers the ideal platform for multimedia applications with unparalleled processing performance. Developers can benefit from the use of an FPGA-based embedded processing system such as mitigating design risk and obsolescence, design reuse, lowering bill of material (BOM) costs by integrating powerful graphics engines within the FPGA.

For SoC reference design in Linux for touch-screen display, please refer to the “Programming Guide for Touch-Screen Display” document in the System CD of VEEK-MT2S.

Figure 1-1 shows a photo of VEEK-MT2S.

Figure 1-1 The VEEK-MT2S platform

1.1 Key Features

The key features of this kit are listed below:

- Cyclone V SE SoC—5CSXFC6D6F31C6N
  - Dual-core ARM Cortex-A9 (HPS)
  - 110K programmable logic elements
  - 5,140 Kbits embedded memory
  - 6 fractional PLLs
  - 2 hard memory controllers

- Configuration Sources
  - Quad serial configuration device – EPCS128 for the FPGA
  - On-board USB Blaster II (normal type B USB connector)

- Memory Devices
  - 64MB (32Mx16) SDRAM for the FPGA
  - 1GB (2x256MBx16) DDR3 SDRAM for the HPS
  - microSD card socket for the HPS

- Peripherals
  - Two USB 2.0 Host porsts (ULPI interface with USB type A connector)
  - UART to USB (USB Mini B connector)
  - 10/100/1000 Ethernet
  - PS/2 mouse/keyboard
  - IR emitter/receiver
  - I2C multiplexer

- Connectors
  - One HSMC expansion header
  - One 40-pin expansion header
  - One 10-pin ADC input header
- One LTC connector (one Serial Peripheral Interface (SPI) master ,one I2C bus, and one GPIO interface)Display
  - 24-bit VGA DAC
  - 128x64 dot Mono Graphic LCD with backlight for HPS

- Audio
  - 24-bit CODEC, line-in, line-out, and microphone-in jacks

- Video Input
  - TV decoder (NTSC/PAL/SECAM) and Video-in connector
- ADC
  - Fast throughput rate: 1 MSPS
  - Channel number: 8
  - Resolution: 12-bit

- Analog input range : 0 ~ 2.5 V or 0 ~ 5V by selecting the RANGE bit in the control register

- Switches, Buttons and LEDs
  - 5 user keys (4 for the FPGA and 1 for the HPS)
  - 10 user switches for the FPGA
  - 11 user LEDs (10 for the FPGA and 1 for the HPS)
  - 2 HPS reset buttons (HPS_RESET_n and HPS_WARM_RST_n)
  - Six 7-segment displays

- Sensor
  - G-sensor for the HPS

- Power
  - 12V DC input

- Capacitive LCD Touch Screen Module
  - Equipped with an 7-inch Amorphous-TFT-LCD (Thin Film Transistor Liquid Crystal Display) module
  - 800x480x3(RGB) Resolution
  - 24-bit parallel RGB interface
  - Supports 5-point touch
  - Ambeient light sensor
  - 8-axis sensor: accelerometer, gyroscope, magnetometer

Table 1-1 shows the general physical specifications of the touch-screen (Note*).

Table 1-1 General physical specifications of the LCD

Item	*Specification*	*Unit*
LCD size	7-inch (Diagonal)	-
Resolution	800 x3(RGB) x 480	dot
Display mode	Normally White, Transmissive	-
Dot pitch	0.0642(W) x0.1790 (H)	mm
Active area	154.08 (W) x 85.92 (H)	mm
Module size	179.4(W) x 117.4(H) x 7.58(D)	mm
Surface treatment	Anti-Glare	-
Color arrangement	RGB-stripe	-
Interface	Digital	-
Backlight power consumption	1.674(Typ.)	W
Panel power consumption	0.22(Typ.)	W

1.2 About the Kit

The kit includes everything users need to run the demonstrations and develop custom designs, as shown in Figure 1-2.

Figure 1-2 Contents of VEEK-MT2S kit package

1.3 Power On Test

The microSD card included in the kit is pre-programmed with LXDE Linux desktop. Users can perform a power on test from the microSD card. The procedures to perform the power on test are:

1. Please make sure the microSD card is inserted to the microSD card socket (J11) onboard.
2. Set MSEL[4:0] = **01010**, as shown in Figure 1-3.
3. Plug in a USB keyboard to the USB host on the DE10-Standard board. (Optional)
4. Plug in the 12V DC power supply to the DE10-Standard board.
5. Power on the DE10-Standard board.
6. The LXDE Desktop will appear on the LCD display.
7. Use the touch-screen to select the system menu, as shown in Figure 1-4.

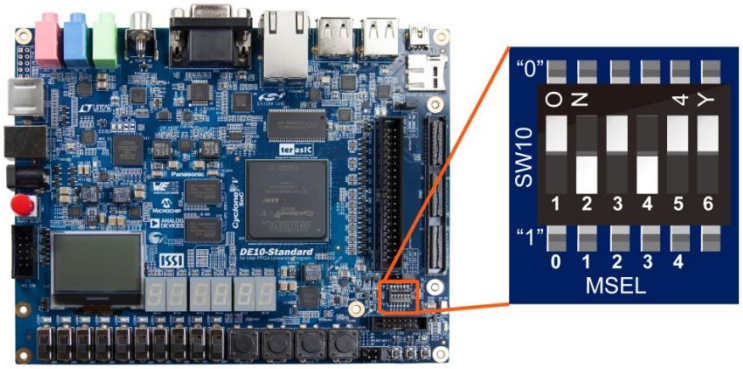

Figure 1-3 MSEL[4:0] = 01010

Figure 1-4 LXDE desktop on VEEK-MT2S platform

1.4 System CD and Linux BSP

The VEEK-MT2S System CD contains the touch-screen documentations and supporting materials, including the user manual, reference designs, and device datasheets. Users can download the System CD form the link: http://veek-mt2s.terasic.com/cd. This site also provides the Linux image files for creating a bootable microSD card. **Table 1-1** shows the contents of VEEK-MT2S System CD. For the system CD of DE10-Standard mainboard, users can download it from the link: http://de10-standard.terasic.com/cd.

Table 1-1 Contents of VEEK-MT2S System CD

Folder Name	*Description*
Datasheet	Specifications for major components on the touch-screen display module
Demonstrations	FPGA and SoC design examples
Manual	Including user manual and software programming guide
Schematic	Schematic of the touch-screen display module

1.5 Getting Help

Here is the contact information should you encounter any problem:

- Terasic Inc.
- Tel: +886-3-575-0880
- Email: support@terasic.com

Chapter 2

Architecture

This chapter provides information regarding the features and architecture of VEEK-MT2S. The kit is composed of DE10-Standard mainboard and MTLC2 (Multi-Touch LCD with Camera, second edition) module. The MTLC2 module is connected to a HSMC expansion header on DE10-Standard board through an HSMC Flex Cable. For more information about the DE10-Standard mainboard, please refer to the user manual in DE10-Standard System CD, which can be downloaded from the link:

http://de10-standard.terasic.com/cd

2.1 Layout and Components

Figure 2-1 and **Figure 2-2** show photos of VEEK-MT2S. It depicts the layout of the board and indicates the locations of connectors and key components.

Figure 2-1 VEEK-MT2S (top view)

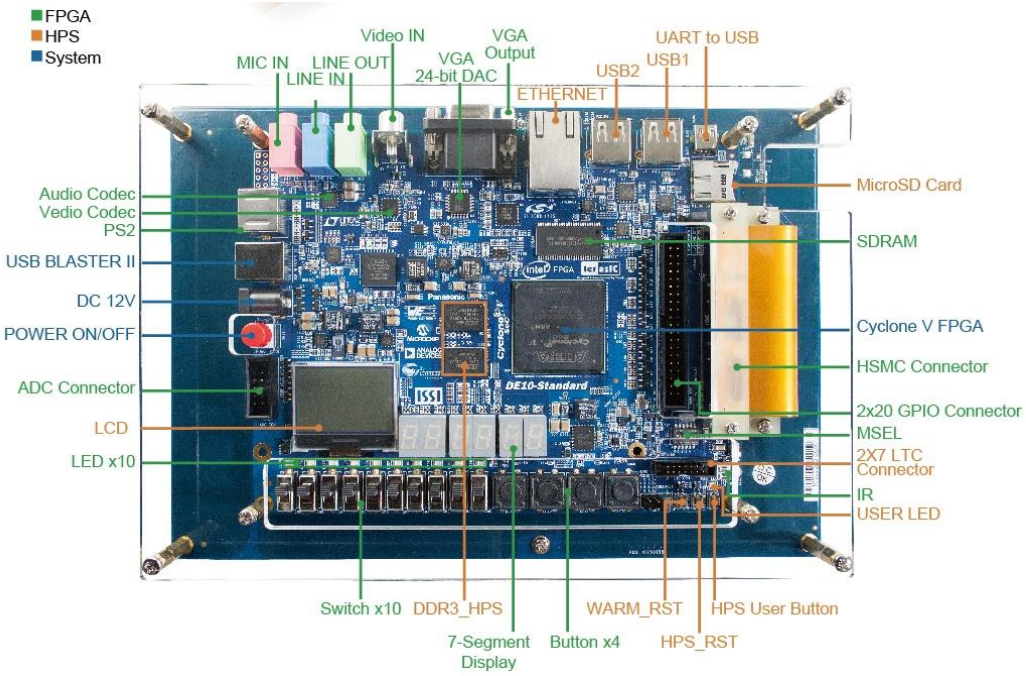

Figure 2-2 VEEK-MT2S (bottom view)

2.2 System Block Diagram

Figure 2-3 shows the block diagram of VEEK-MT2S. The HSMC connector bridges all the wires from the peripherals to the FPGA through an HSMC Flex Cable.

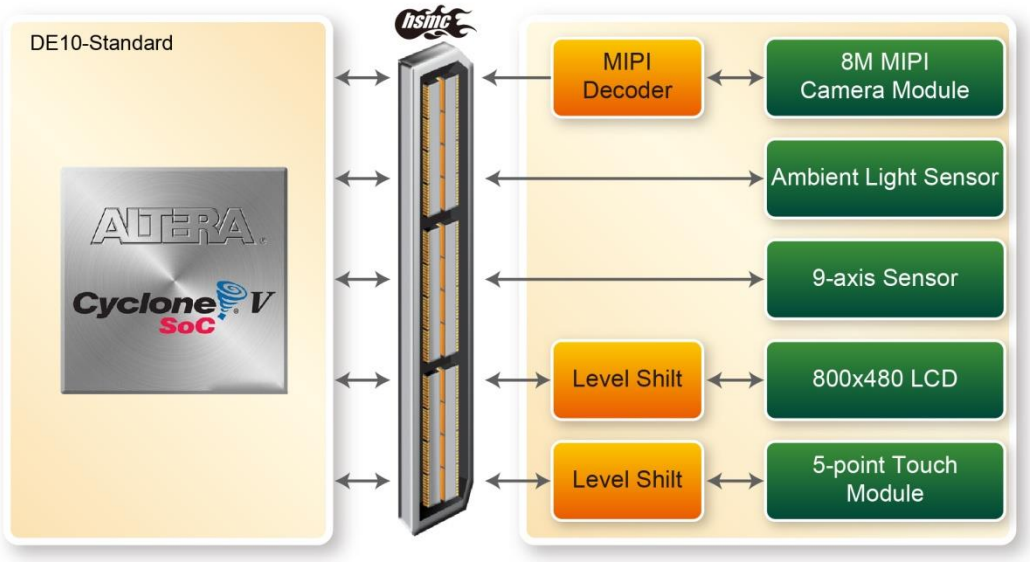

Figure 2-3 Block Diagram of VEEK-MT2S

Chapter 3

Using VEEK-MT2S

This chapter provides information on how to control the Multi-touch LCD with Camera Module Second Edition (MTLC2) hardware, which includes the definition of HSMC interface, LCD control, and multi-touch control signals, 8-mega pixel camera, ambient light sensor and 9-axis sensor.

3.1 Block Diagram

The VEEK-MT2S is composed of DE10-Standard SoC development board and 7" touch panel module MTLC2. Figure 3-1 gives the block diagram of the VEEK-MT2S board.

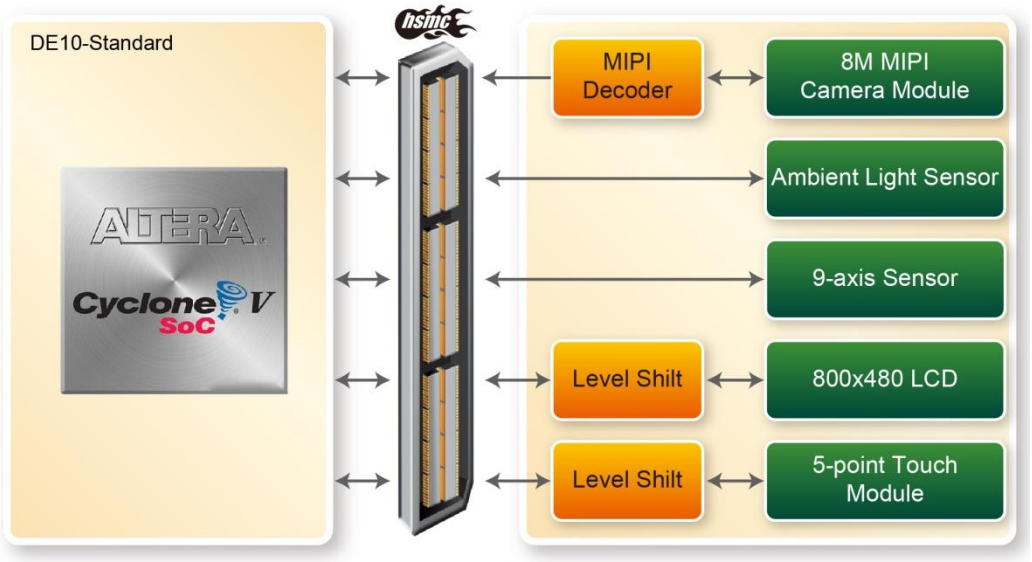

Figure 3-1 Block Diagram of VEEK-MT2S

3.2 Using FPGA

The VEEK-MT2S uses the DE10-Standard as the FPGA main board. The DE10-Standard user manual and CD are available at the following link:

http://de10-standard.terasic.com/cd

3.3 Using the 7” LCD Capacitive Touch Screen

The VEEK-MT2S features a 7-inch capacitive amorphous TFT-LCD panel. The LCD touch screen offers resolution of 800x480 to provide users the best display quality for developing applications. The LCD panel supports 24-bit parallel RGB data interface.

The 5-point touch chip used in the VEEK_MT2S is I2C interface. Developers can use generic I2C master controller to communicate with the touch chip. For details information about the touch chip, please refer to the control panel datasheet located in the folder Datasheet/lcd in the VEEK-MT2S System CD. For a Nios II project, a c-code library is provided in the multi_touch2.c files.

To display images on the LCD panel correctly, the RGB color data along with the data enable and clock signals must act according to the timing specification of the LCD touch panel as shown in **Table 3-1** and

Table 3-2**.**

Table 3-3 gives the pin assignment information of the LCD touch panel.

Table 3-1 LCD Horizontal Timing Specifications

Item	*Symbol*	*Typical Value*			*Unit*
		Min.	*Typ.*	*Max.*	
Horizontal Display Area	thd	-	800	-	DCLK
DCLK Frequency	fclk	26.4	33.3	46.8	MHz
One Horizontal Line	th	862	1056	1200	DCLK
HS pulse width	thpw	1		40	DCLK
HS Blanking	thb	46	46	46	DCLK
HS Front Porch	thfp	16	210	354	DCLK

Table 3-2 LCD Vertical Timing Specifications

Item	*Symbol*	*Typical Value*			*Unit*
		Min.	*Typ.*	*Max.*	
Vertical Display Area	tvd	-	480	-	TH
VS period time	tv	510	525	650	TH
VS pulse width	tvpw	1	-	20	TH
VS Blanking	tvb	23	23	23	TH
VS Front Porch	tvfp	7	22	147	TH

Table 3-3 Pin assignment of the LCD touch panel

Signal Name	FPGA Pin No.	Description	I/O Standard
LCD_B0	P28	LCD blue data bus bit 0	2.5V
LCD_B1	P27	LCD blue data bus bit 1	2.5V
LCD_B2	J24	LCD blue data bus bit 2	2.5V
LCD_B3	J23	LCD blue data bus bit 3	2.5V
LCD_B4	T26	LCD blue data bus bit 4	2.5V
LCD_B5	T25	LCD blue data bus bit 5	2.5V
LCD_B6	R26	LCD blue data bus bit 6	2.5V
LCD_B7	R25	LCD blue data bus bit 7	2.5V
LCD_DCLK	V24	LCD Clock	2.5V
LCD_DE	H23	Data Enable signal	2.5V
LCD_DIM	P21	LCD backlight enable	2.5V
LCD_DITH	L23	Dithering setting	2.5V
LCD_G0	P26	LCD green data bus bit 0	2.5V
LCD_G1	P25	LCD green data bus bit 1	2.5V
LCD_G2	N26	LCD green data bus bit 2	2.5V
LCD_G3	N25	LCD green data bus bit 3	2.5V
LCD_G4	L22	LCD green data bus bit 4	2.5V
LCD_G5	L21	LCD green data bus bit 5	2.5V
LCD_G6	U26	LCD green data bus bit 6	2.5V
LCD_G7	U25	LCD green data bus bit 7	2.5V
LCD_HSD	U22	Horizontal sync input.	2.5V
LCD_MODE	L24	DE/SYNC mode select	2.5V
LCD_POWER_CTL	M25	LCD power control	2.5V
LCD_R0	V28	LCD red data bus bit 0	2.5V
LCD_R1	V27	LCD red data bus bit 1	2.5V
LCD_R2	U28	LCD red data bus bit 2	2.5V
LCD_R3	U27	LCD red data bus bit 3	2.5V
LCD_R4	R28	LCD red data bus bit 4	2.5V
LCD_R5	R27	LCD red data bus bit 5	2.5V
LCD_R6	V26	LCD red data bus bit 6	2.5V
LCD_R7	V25	LCD red data bus bit 7	2.5V
LCD_RSTB	K22	Global reset pin	2.5V
LCD_SHLR	H24	Left or Right Display Control	2.5V
LCD_UPDN	K21	Up / Down Display Control	2.5V
LCD_VSD	V22	Vertical sync input.	2.5V
TOUCH _I2C_SCL	T22	touch I2C clock	2.5V
TOUCH _I2C_SDA	T21	touch I2C data	2.5V
TOUCH _INT_n	R23	touch interrupt	2.5V

3.4 Using 8-megapixel Digital Image Sensor

Terasic VEEK-MT2S board equips with an 8M pixel MIPI camera module named OV8865 (See **Figure 3-2**). The OV8865 color image sensor is a high performance, 8 megapixel RAW image sensor that delivers 3264x2448. It provides options for multiple resolutions while maintaining full field of view. Users can program image resolution, frame rate, and image quality parameters. Camera functions are controlled via I2C bus (CAMERA_I2C_SDA and CAMERA_I2C_SCL). The I2C device address is 0x6C. For more hardware description and register information about this camera module, please refer to the datasheet named OV8865 Data Sheet.pdf in the VEEK-MT2S System CD.

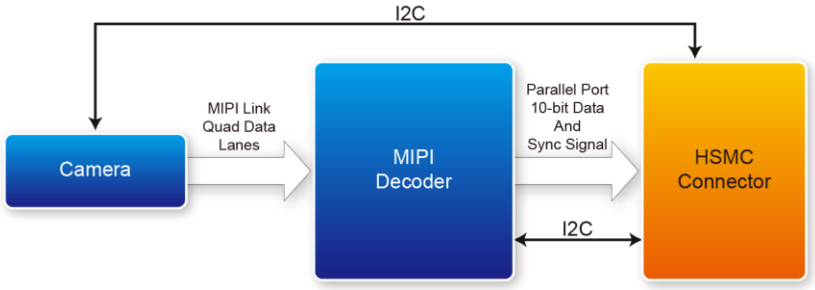

Figure 3-2 Block Diagram of the Bus Controller

■ Voice Coil Motor (VCM)

There is a Voice Coil Motor (VCM) driver chip named VCM149C on the MIPI camera module. Users can use the same I2C bus (I2C device address is 0x18) to modify the DAC value in the VCM driver chip that can allow the VCM to move its lens to the desired position for getting a sharp image and realizing the Auto Focus (AF) feature. Terasic also provides an AF demonstration and IP in the System CD, see **section 5.25.2** 如下 **RTL Camera** for details. The datasheet of this VMC driver IC named VM149C VCM Driver IC.pdf also can be found in the System CD.

■ MIPI Decoder

The MIPI camera module output interface is MIPI interface, which can not directly connect to the Terasic FPGA board; therefore, a MIPI Decoder (TC358748XBG) is added to convert MIPI interface to a parallel port interface. Decoder users can quickly obtain the image data and process it. MIPI Decoder can convert MIPI Interface up to 24-bit data. The Camera module used on the D8M can only output 10 bit data, MIPI_PIXEL_D[9:0] the HSMC connector is the camera image output data bus.

FPGA also can read/write MIPI Decoder through a I2C bus (MIPI_I2C_SDA / MIPI_I2C_SCL ; I2C device address is 0x1C), which is different from the camera module I2C bus. On the VEEK-MT2S board, MIPI Decoder can output clocks to the MIPI camera and FPGA board. So in the demonstrations, most of them show how to control IC PLL parameters as well as others. Detailed clock functions are described as below.

■ Clock Tree

Figure 3-3 is the VEEK-MT2S board's camera clock tree block diagram. MIPI Decoder PLL receives FPGA Reference Clock (MIPI_REFCLK) and outputs Clock to Camera sensor (MCLK), at the same time, MIPI Decoder PLL will also output a parallel port clock (MIPI_PIXEL_CLK) and feedback to the FPGA to deal with parallel data.

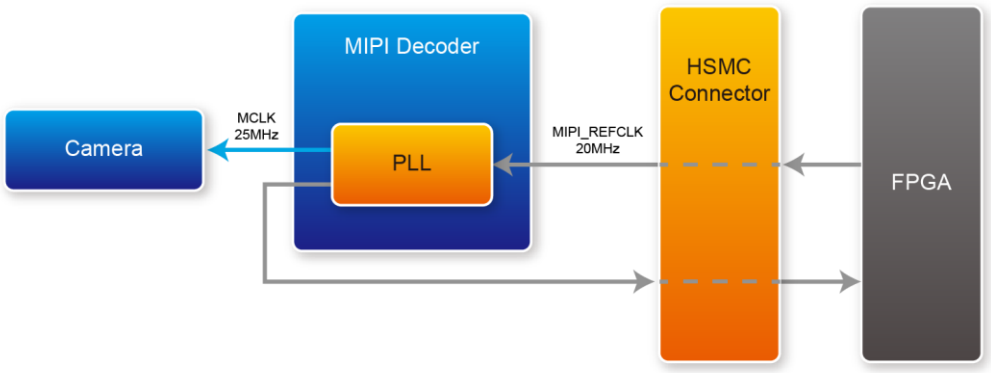

Figure 3-3 Block Diagram of the Bus Controller

In the provided demonstrations, MIPI_REFCLK is set to 20MHz. The FPGA transmits this clock to the VEEK-MT2S's MIPI Decoder PLL through the HSMC connector. No matter how much the camera resolution is, the MCLK fixed output is 25MHz. According to the output resolution, MIPI_PIXEL_CLK can be set as 25MHz for 640x480@60fps and 50MHz for 1920x1080@15fps.

For more MIPI Decoder PLL setting details, please refer to TC358746AXBG_748XBG_rev09.pdf "Chapter 5: Clock and System" or refer to Terasic camera or vip_camera demonstrations.

■ Pin Assignment

Table 3-4 Pin assignment of the Camera sensor

Signal Name	FPGA Pin No.	Description	I/O Standard
MIPI_PIXEL_D[0]	F24	Parallel Port Data	2.5V
MIPI_PIXEL_D[1]	F25	Parallel Port Data	2.5V
MIPI_PIXEL_D[2]	D26	Parallel Port Data	2.5V
MIPI_PIXEL_D[3]	C27	Parallel Port Data	2.5V
MIPI_PIXEL_D[4]	F26	Parallel Port Data	2.5V
MIPI_PIXEL_D[5]	E26	Parallel Port Data	2.5V
MIPI_PIXEL_D[6]	G25	Parallel Port Data	2.5V
MIPI_PIXEL_D[7]	G26	Parallel Port Data	2.5V
MIPI_PIXEL_D[8]	H25	Parallel Port Data	2.5V
MIPI_PIXEL_D[9]	H26	Parallel Port Data	2.5V
MIPI_PIXEL_D[10]	M26	Reserve	2.5V
MIPI_PIXEL_D[11]	M25	Reserve	2.5V
MIPI_PIXEL_D[12]	AF27	Reserve	2.5V
MIPI_PIXEL_D[13]	AE28	Reserve	2.5V
MIPI_RESET_n	D27	Master Reset signal for MIPI camera and bridge device	2.5V
MIPI_PIXEL_CLK	J27	Parallel Port Clock signal	2.5V
MIPI_PIXEL_HS	K26	Parallel Port Horizontal Synchronization signal	2.5V
MIPI_PIXEL_VS	K25	Parallel Port Vertical Synchronization signal	2.5V
MIPI_CS_n	F28	Chip Select	2.5V
MIPI_REFCLK	G23	Reference Clock Input of bridge device	2.5V
MIPI_I2C_SCL	AE26	I2C Clock for bridge device	2.5V
MIPI_I2C_SDA	AE27	I2C Data for bridge device	2.5V
CAMERA_PWDN_n	R22	Power Down signal of MIPI camera	2.5V
CAMERA_I2C_SCL	R21	I2C Clock for MIPI camera	2.5V
CAMERA_I2C_SDA	F27	I2C Data for MIPI camera	2.5V
MIPI_MCLK	G24	MIPI camera system clock (Reserve)	2.5V

3.5 Using the Gyroscope/Accelerometer/Magnetometer

The VEEK-MT2S is equipped with a Motion-Tracking device named MPU-9250. The MPU-9250 is a 9-axis Motion-Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer and

3-axis magnetometer. Detail features of these sensors are listed below:

■ Gyroscope

The MPU-9250 consists of three independent vibratory MEMS rate gyroscopes, which detect rotation about the X-, Y-, and Z- Axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis. The full-scale range of the gyro sensors may be digitally programmed to ±250, ±500, ±1000, or ±2000 degrees per second (dps). The ADC sample rate is programmable from 8,000 samples per second, down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

■ Accelerometer

The MPU-9250's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-9250's architecture reduces the accelerometers' susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0g on the X- and Y-axes and +1g on the Z-axis. The accelerometers' scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2g, ±4g, ±8g, or ±16g.

■ Magnetometer

The 3-axis magnetometer uses highly sensitive Hall sensor technology. The magnetometer portion of the IC incorporates magnetic sensors for detecting terrestrial magnetism in the X-, Y-, and Z-Axes, a sensor driving circuit, a signal amplifier chain, and an arithmetic circuit for processing the signal from each sensor. Each ADC has a 16-bit resolution and a full scale range of ±4800 μT.

Communication with all registers of the device is performed using either I2C at 400kHz or SPI at 1MHz. For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz. For more detailed information of better using this chip, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD. **Table 3-5** gives the pin assignment information of the LCD touch panel. For more detailed information of better using this chip, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.

Table 3-5 contains the pin names and descriptions of the MPU-9250.

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
MPU_AD0_SDO	K27	I2C Slave Address LSB (AD0); SPI serial data output (SDO)	2.5V
MPU_CS_n	F28	Chip select (SPI mode only)	2.5V

MPU_FSYNC	G28	Frame synchronization digital input	2.5V
MPU_INT	G27	Interrupt digital output	2.5V
MPU_SCL_SCLK	M27	I2C serial clock (SCL); SPI serial clock (SCLK)	2.5V
MPU_SDA_SDI	K28	I2C serial data (SDA); SPI serial data input (SDI)	2.5V

3.6 Using the Ambient Light Sensor

The APDS-9300 is a low-voltage digital ambient light sensor that converts light intensity to digital signal output capable of direct I2C communication. Each device consists of one broadband photodiode (visible plus infrared) and one infrared photodiode. Two integrating ADCs convert the photodiode currents to a digital output that represents the irradiance measured on each channel. This digital output can be input to a microprocessor where illuminance (ambient light level) in lux is derived using an empirical formula to approximate the human-eye response. For more detailed information of better using this chip, please refer to its datasheet which is available on manufacturer's website or under the /datasheet folder of the system CD.

Table 3-6 contains the pin names and descriptions of the ambient light sensor module.

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
LSENSOR_ADDR_SEL	J25	Chip select	2.5V
LSENSOR_INT	L28	Interrupt output	2.5V
LSENSOR_SCL	J26	Serial Communications Clock	2.5V
LSENSOR_SDA	L27	Serial Data	2.5V

Chapter 4

Linux BSP

This chapter describes how to use the Linux BSP (Board Support Package) provided by Terasic. Users can develop touch-screen GUI program easily with the BPS including QT 5.3.1 library.

4.1 Board Support Package

Figure 4-1 shows the block diagram of Linux BSP for the VEEK-MT2S kit. The BPS includes three major parts:

- Linux image files
- Quartus project
- QT library with touch-screen function included

The Linux image files are implemented on HPS/ARM and the Quartus project is implemented on FPGA/Qsys. The Linux image files include the pre-built Linux system. Users can create a Linux bootable microSD card with the image files. The Quartus project includes the controller for VGA display and the touch-screen controller for touch-screen panel.

The BSP includes not only precompiled QT library and touch-screen library in the Linux image files, but also the document that show how to cross-compile these libraries, as well as to develop touch-screen GUI program based on these libraries.

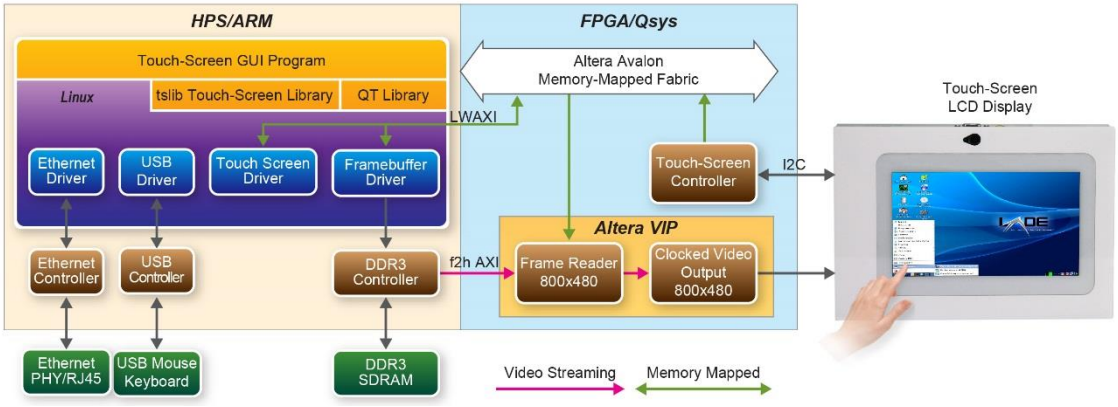

Figure 4-1 Block Diagram of Linux BSP for VEEK-MT2S kit

4.2 Linux Image Files

LXDE desktop Linux image file is provided for VEEK-MT2S. It is available from the link: http://veek-mt2s.terasic.com/cd. Developers can use a tool named **Win32 Disk Imager** to write the image file into a microSD card. For details about how to create a bootable microSD card or booting

Linux from the DE10-Standard board, please refer to chapter 5 of DE10-Standard_Getting_Started_Guide.pdf, which is included in the DE10-Standard System CD, which is available from the link: http://de10-standard.terasic.com/cd.

Figure 4-2 shows a screenshot of LXDE desktop after booting. The LXDE desktop is displayed on the LCD touch panel. This image file also includes the QT library and touch-screen library. To perform these demos, users need to double click the icons of the demos on desktop.

Figure 4-2 Screenshot of LXDE desktop

4.3 Quarts Project

The Quartus project is designed based on Altera Qsys tool. There are three major parts:

- VGA display
- Touch-screen
- HPS component

The VGA display part is designed to display the Linux console or desktop on the LCD touch panel. Altera Video and Image Processing (VIP) suite is used to implement this function. The Linux frame buffer driver fills up the DDR3 with data to be displayed, and the VIP frame-reader component reads the data from the DDR3 in a DMA manner. The video data is streamed into the VIP Clocked Video Output component. Finally, the VIP Clocked Video Output component drives the VGA DAC chip to display the video data.

An I2C master controller in Qsys is used to communicate with the touch-screen panel. The component interfaces with the touch-screen panel through I2C protocol.

The HPS communicate with the FPGA through AXI bridge. The components in FPGA are mapped into user memory of the linux system through memory-mapped interface. Then the user software can access the IPs in FPGA portion. The Quartus project is located under the folder "Demonstration/SoC_FPGA/ControlPanel/Quartus" in the VEEK-MT2S system CD.

4.4 QT Libraries

Users can develop touch-screen GUI program based on the QT library. For more information, please refer to the document "**VEEK-MT2S_Control_Panel.pdf**" included in the VEEK-MT2S system CD. The precompiled QT libraries can be found from

"Demonstration/SoC_FPGA/ControlPanel/lib/qt5.5.1_for_intel_soc.tar.gz"

in the VEEK-MT2S system CD.

Chapter 5

FPGA Demonstration

This chapter gives detailed description of the provided bundles of exclusive demonstrations implemented on VEEK-MT2S. These demonstrations are particularly designed (or ported) for VEEK-MT2S, with the goal of showing the potential capabilities of the kit and showcase the unique benefits of FPGA-based systems such as reducing BOM costs by integrating powerful graphics and video processing circuits within the FPGA. The camera demonstration is a RTL only code, and the other demonstrations are incorporated with VIP (Video and Image and Nios II processor).

5.1 Painter

This section shows how to implement a painter demo on the Multi-touch LCD module based on Altera Qsys tool and the Video and Image Processing (VIP) suite. It demonstrates how to use multi-touch gestures and resolution. The GUI of this demonstration is controlled by the Nios II processor.

■ Operation Description

Figure 5-1 shows the Graphical User Interface (GUI) of Painter demo. The GUI is classified into four separate areas: Painting Area, Gesture Indicator, Clear Button, and Color Palette. Users can select a color from the color palette and start painting in the paint area. If a gesture is detected, the associated gesture symbol will be shown in the gesture area. To clear the painting area, click the "Clear" button.

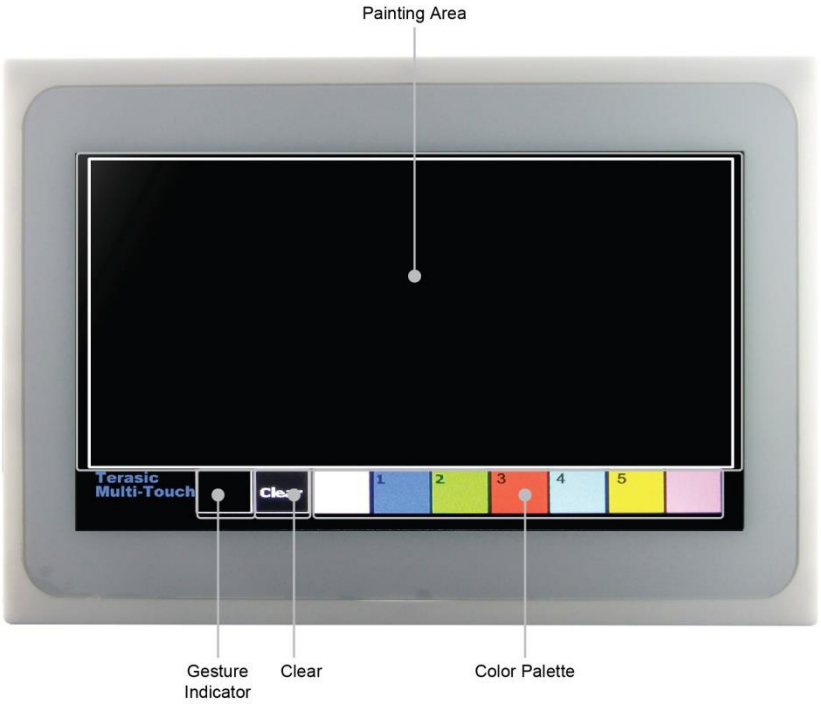

Figure 5-1 GUI of Painter Demo

Figure 5-2 shows the single-finger painting of canvas area.

Figure 5-2 Single-finger painting

Figure 5-3 shows the zoom-in gesture.

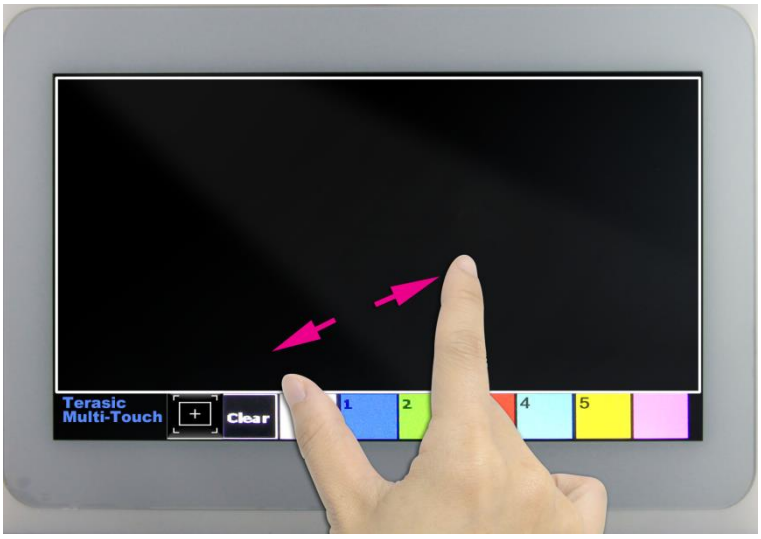

Figure 5-3 Zoom-in gesture

Figure 5-4 5-Point painting of canvas area.

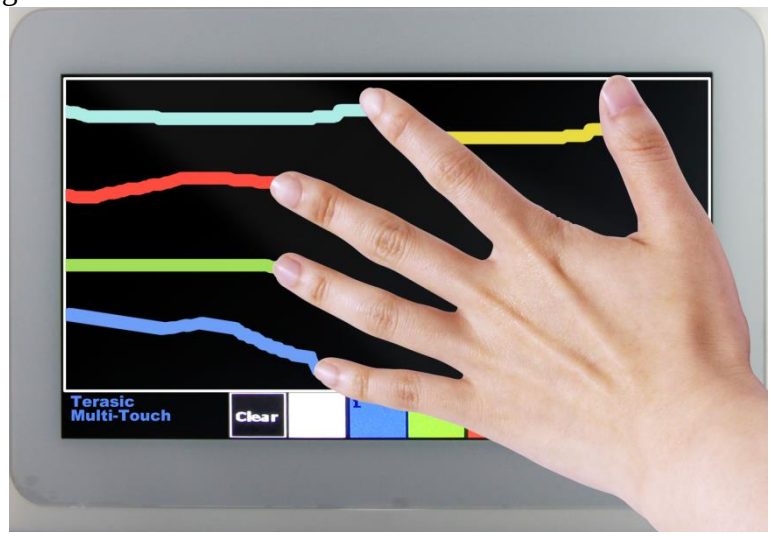

Figure 5-4 5-Point painting

■ System Description

For LCD display processing, the reference design is developed based on Altera's Video and Image Processing (VIP) suite. The Frame Reader VIP is used for reading data to be displayed from the associated video memory, and the VIP Video Out is used to display the video data. The data is drawn by the Nios II processor according to user input.

For multi-touch processing, when touch activity occurs, an I2C Controller IP is used to retrieve serial data from the I2C interface, the associated touch information including multi-touch gestures and 5 Point touch coordinates can be calculated through the data in NIOS II. Note: the license for this IP must be installed before compiling the Quartus II project including this encrypted component.

Figure 5-5 shows the system generic block diagram of painter demonstration.

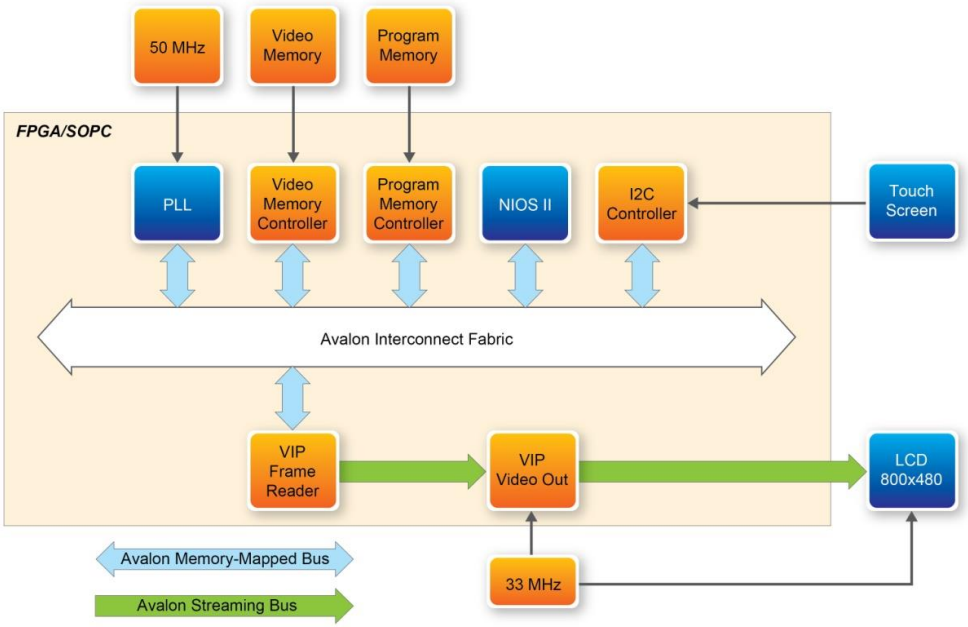

Figure 5-5 System block diagram of painter demonstration

■ Demonstration Setup

Please follow the procedures below to setup the demonstration:

1. Connect the VEEK-MT2S USB-Blaster II USB port to the PC USB Port with a USB Cable.
2. Power on the VEEK-MT2S.
3. Please make sure Quartus II v16.1 has been installed on the host PC.
4. Copy the folder \Demonstrations\FPGA\Painter\demo_batch.
5. Execute "test.bat".
6. The painter GUI will show up on the LCD panel.

■ Project Information

Item	Description
Tool	Quarts II Prime 16.1.1 Standard Edition
Quartus Project directory	Demonstration\FPGA\painter
FPG bit stream file	DE10_Standard.sof
Nios II Workspace	painter\software
Nios binary file	painter.elf
Demo batch folder	Painter\demo_batch

5.2 RTL Camera

This section provides instructions on how to store Camera capturing image (800x600@60Mhz) in a memory (Frame-Buffer), this Memory is expected to store up to one Frame image, and how to extract Frame-Buffer address data to convert RAW data to RGB data, and output the RGB data to LCD with 800x480@60 Hz timing.

■ Function Block Diagram

Figure 5-6 shows the Function block diagram of Camera demonstration. This design block is one Sdram_Control module that can control general external SDRAM Memory and read/write image data. Camera raw data will be written in SDRAM first. After finishing writing a Frame, Sdram_Control module will read out the data from SDRAM to RAW2RGB_J module to convert RAW data to RGB data. The RGB data will output along with the signal timing generated by VGA_Controller to the LCD. In the block, other module (for example, FOCUS_ADJ,MIPI_BRIDGE_CAMERA_Config) function instructions and KEY/SW operation. All module functions are described below:

- MIPI_BRIDGE_CAMERA_Config: the MIPI BRIDGE I2C and Camera I2C setting controller, such as set to output 800X480@60Hz timing. It mainly writes I2C corresponding parameters to MIPI-BRIDGE IC register and Camera Sensor IC register respectively through their own I2C buses. MIPI_I2C bus is used to write MIPI BRIDGE I2C (I2C Slave Address = 0x1c.), CAMERA_I2C bus is used to write Camera Sensor (IC Slave Address = 0x6c).

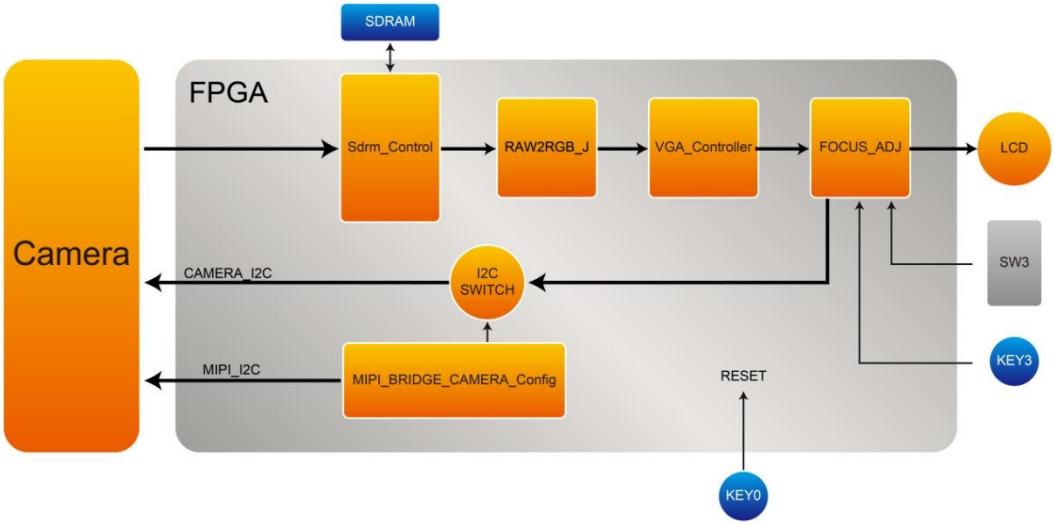

Figure 5-6 Function block diagram of Camera demonstration

- Sdram_Control: This module is one Sdram_Controller can control general external SDRAM Memory and read/write image data.
- RAW2RGB_J: This module is to convert RAW data to RGB data.
- VGA_Controller: the LCD signal timing generator can generate 800x480@60 Hz signal timing.
- FOCUS_ADJ: This module provides two main functions.

The first function is using I2C bus to write D8M Voice Coil Motor (VCM) driver IC register, and control the camera lens' movements to perform image focusing. VCM driver IC register (I2C Slave Address =0x18) shares I2C bus with camera module. The other function is doing the current image high frequency component statistic. When the VCM drives the camera lens' movement, a real-time statistics of image high-frequency sum will be done in every step of the moving. Finally, the lens will move to a position which has the largest number of high frequency to complete the automatic focus operation. Focus area can be selected by SW3. There are two options：

- Select focusing the whole screen area (set SW3 to 0)
- Select focusing the middle area (set SW3 to 1).

Once you set SW3 to a value (0 or 1) and press KEY3 one time, the automatic focus operation will be performed in the selected area.

■ The Default Camera Settings

In this demonstration, the default camera settings are:

- Resolution: 800x480

- Pixel Data: RAW10
- Frame Rate: 60 fps
- Bin Mode: 1 (no bin mode: x2, x4 will out of full-size)

■ Project Information

Item	Description
Tool	Quarts II Prime 16.1.1 Standard Edition
Quartus Project directory	Demonstration\FPGA\Camera
FPG bit stream file	Camera.sof
Demo batch folder	Camera\demo_batch

■ Demonstration Setup

- Connect the DE10-Stanrad board (J13) to the host PC with a USB cable and install the USB-Blaster II driver if necessary
- Plug the 12V adapter to VEEK-MT2S Board.
- Power on the VEEK-MT2S board.
- Load the bit stream into FPGA by executing the batch file 'test.bat' under Camera\demo_batch\ folder
- The system enters the FREE RUN mode automatically (See **Figure 5-7**).
- LED0~1 light up, stand the settings of MIPI-BRIDGE I2C and Camera I2C are completed.
- LED7~9 blink in 1Hz, stand MIPI-PIXEL-CLOCK, MIPI-REF-CLOCK and MT2-PIXEL-CLOCK are generated correctly.
- HEX1~0 decimal number "60" stands Camera capturing frame rate is 60Hz
- Camera capturing image displays on LCD, if the LCD image is fuzzy, please press Key3 one time again (will perform the focus operation again). Users can switch SW3 to "1" (there will be a yellow box on image, see **Figure 5-8**), then, press SW3 one time again, the middle area focus operation will be performed.
- **Table 5-1** summarizes the functional keys and details of each LED status.

Table 5-1 The functional keys of the digital camera demonstration

Component	***Function Description***
LED0	Lights up when MIPI-BRIDGE I2C setting is successful
LED1	Lights up when CAMERA I2C setting is successful
LED7	Blink in 1HZ (MIPI-PIXEL-CLOCK/50M)
LED8	Blink in 1HZ (MIPI-REF-CLOCK/20M)
LED9	Blink in 1HZ (LCD-PIXEL-CLOCK/33M)
KEY0	SYSTEM RESET
KEY3	Image auto focus(area based on SW3 selection)
SW3	0: No yellow box (focus on whole screen area) 1: There is a yellow box(focus on yellow box)
HEX[1:0]	fps (frames per sencond)

Figure 5-7 Screen shot of the VEEK-MT2S RTL camera demonstration

Figure 5-8 Area Focus Mode of the VEEK-MT2S RTL camera demonstration

Note: Executing the test.bat under Camera\demo_batch will automatically download the .sof file.

5.3 VIP Camera

The VIP (Video and Image Processing) Camera Example Design demonstrates how to implement a camera display system based on Altera V VIP(Video and Image Processing) Suite. The Nios II processor is used to configure the I2C devices. There is a Camera IP, from Terasic in Qsys, which translates the 8-bits Bayer Pattern from the camera to the 24-bit RGB video stream format and feeds it to the Altera VIP. The other IP developed by Terasic for auto-focus is used to find the optimized focus settings of the user-defined image area. The Altera Frame Buffer VIP and Clocked Video Output VIP are used to display the 800x480 video.

■ System Design Description

The example design demonstrates a framework for rapid development of video and image processing systems using the parameterizable MegaCore® functions that are available in the Video

and Image Processing Suite. Compilation of this demonstration requires Altera VIP suite license.

These functions allow you to fully integrate common video functions with video interfaces, processors, and external memory controllers. The example design uses an Altera Cyclone® V SE SoC 5CSXFC6D6F31C6N featured on the VEEK-MT2S.

A video source is input through the CMOS sensor on VEEK-MT2S which generates a digital output in RGB format. A number of common video functions are performed on this input stream in the FPGA. These functions include clipping, chroma resampling, motion adaptive deinterlacing, color space conversion, picture-in-picture mixing, and polyphase scaling.

The input and output video interfaces on the VEEK-MT2S are configured and initialized by software running on a Nios® II processor. Nios II software demonstrates how to control the clocked video input, clocked video output, and mixer functions at run-time is also provided. The video system is implemented using the QSYS system level design tool. This abstracted design tool provides an easy path to system integration of the video processing data path with a NTSC or PAL video input, VGA output, Nios II processor for configuration and control. The Video and Image Processing Suite MegaCore functions have common open Avalon-ST data interfaces and Avalon Memory-Mapped (Avalon-MM) control interfaces to facilitate connection of a chain of video functions and video system modeling. In addition, video data is transmitted between the Video and Image Processing Suite functions using the Avalon-ST Video protocol, which facilitates building run-time controllable systems and error recovery.

For the objective of a better visual effect, the CMOS sensor is configured with autofocus function. The autofocus function works at first when the demonstration starting up.

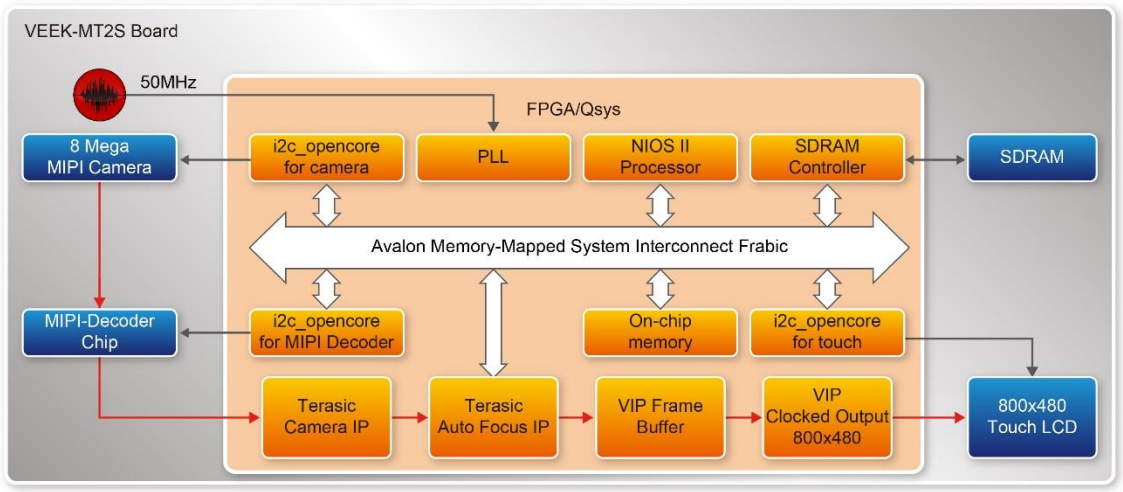

Figure 5-9 shows the Video and Image Processing block diagram. There are three I2C master controllers in the system. The first one is used to configure the OV8865 image sensor and VCM149C motor in MIPI Camera Module. The second is used to configure the MIPI-decoder TC358748XBG. The third is used to retrieve touch information from the 7" LCD touch screen. The I2C master controllers are controlled by the Nios II processor. The Nios II program runs on on-chip memory.

Note: The focus driver IC (VCM149C) in the camera module is also configured by the Terasic auto-focus IP through its own I2C master controller. Users must make sure there is only one I2C master used one at a time.

In this demonstration, the OV9965 Image sensor is configured to generate a serial 800x480 pixel Bayer pattern. The MIPI-decoder translates the serial Bayer pattern data into parallel Bayer pattern data. The Terasic CAMERA IP is used to convert the 800x480 Bayer Pattern into an 800x480 RGB Altera VIP video stream. The Terasic Autofocus is used to automatically find the proper focus position for the camera lens. The VIP Frame Buffer is used to store the complete video frame for display later. The VIP Frame Buffer uses the external SDRAM as video memory. The Clock Video Output IP will translate the video stream to an 800x480 VGA format so the video can be displayed on the VGA monitor.

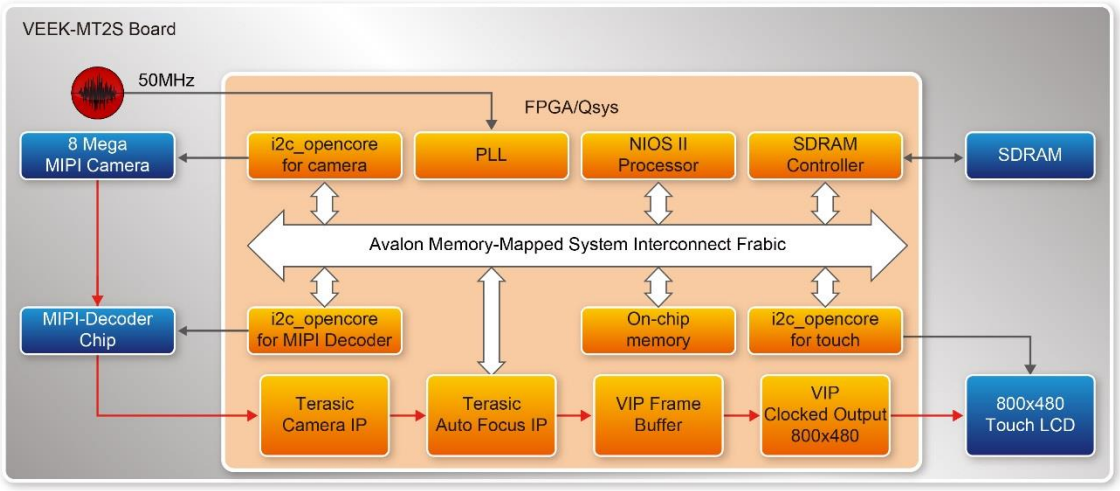

Figure 5-9 VIP Camera Example Qsys Block Diagram

■ Quartus Project

Item	Desciption
Tool	Quarts II Prime 16.1.1 Standard Edition
Quartus Project directory	Demonstration\FPGA\vip_camera
FPG bit stream file	DE10_Standard.sof
Nios II Workspace	vip_camera\software
Nios binary file	vip_camera.elf
Demo batch folder	vip_camera\demo_batch

■ Demonstration Setup

1. Load the bit stream into FPGA (note*)
2. Run the Nios II and choose VIP_Camera\Software as the workspace. Click on the Run button (note *)
3. The system enters the FREE RUN mode automatically (See **Figure 5-10**).
4. Single digit touch LCD anywhere on the screen will initiate the Camera autofocus (See **Figure 5-11**).
5. Use the Zoom-in /out touch gesture to switch the camera capture area.

Figure 5-10 The VIP_Camera demonstration running result

Figure 5-11 Touch to Initiate Auto Focus

Note: (1).Executing VIP_Camera*demo_batch*\VIP_CameraA.*bat will download .sof and .elf files.*

5.4 G-sensor and Light-sensor Demonstration

This demonstration shows a bubble level implementation based on a digital accelerometer. There are two I2C controllers used in this demonstration. One is used to communicate with MPU9250 to retrieve the gravity information, and another is used to communicate with the APDS-9300 Miniature Ambient Light Photo Sensor. The demonstration using the gravity information retrieved from the MPU9250 to implement a bubble level on the LCD panel. When tilting the VEEK-MT2, the NIOS II program reads the acceleration of gravity from the MPU9250. Based on the gravity information, the NIOS II program can compute the change of angle in the x-axis and y-axis, and

shows the angle data in the LCD display. The demonstration also displays two ambient light levels, measured by APDS-9300, on the LCD panel.

■ System Design Description

Figure 5-12 shows the hardware system block diagram of this demonstration. The system is clocked by an external 50MHz Oscillator. Through the internal PLL module, the generated 100MHz clock is used for Nios II processor, SDRAM, SSRAM, and other high speed controllers. The PLL also generates at 20MHz for low-speed peripherals, such as LEDs and buttons. The NIOS II program is running on the SRAM; the SDRAM is used as video frame buffers in the Altera VIP block. The I2C_OpenCore controllers are used to communicate with the MPU9250 chip and the APDS-9300 chips. The VIP block is used to implement video streaming for displaying graphic on the LCD panel. The video steaming is:

Nios II Processor → SDRAM → VIP: Frame Reader → VIP: Clocked Video Output → LCD Panel..

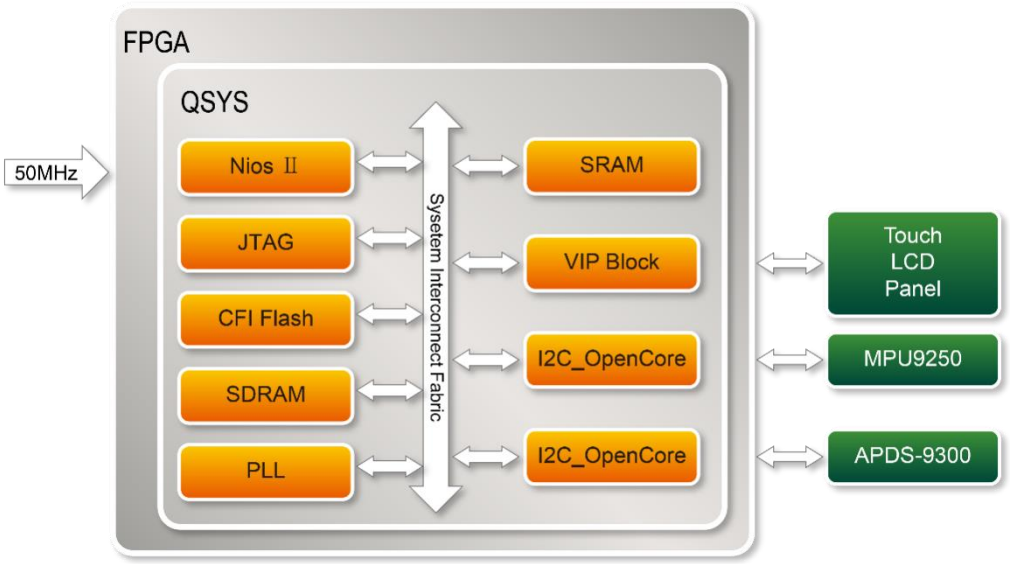

Figure 5-12 Block diagram of the digital accelerometer demonstration

In the NIOS II program, the GUI (graphic user interface) is implemented in the gui_gsensor.cpp. The MPU9250 class, implemented in the MPU9250.cpp/.h is used to retrieve accelerometer information from the MPU9250. The member function **initialize** of the MPU9250 class should be called first before calling any other member function. In this demonstration, the member function **getMotion9** is called to get the gravity value of X/Y/Z. The light_sensor.cpp/h includes the library which allows it to communicate with the APDS-9300 Miniature Ambient Light Photo Sensor. Before reading the light level information, call function Light_Init first to initialize the sensor APDS-9300. The function **Light_GetID** is designed to get the chip ID of APDS-9300. Function **Light_Get_ADCData0** and **Light_Get_ADCData1** are designed to get the two ADC values in APDS-9300

■ Project Information

Item	Description
Tool	Quarts II Prime 16.1.1 Standard Edition
Quartus Project directory	Demonstration\FPGA\G_Sensor
FPGA bit stream file	G_Sensor
Nios II Workspace	G_Sensor\software
Nios binary	GUI_APP.elf
Demo batch folder	G_Sensor\demo_batch

■ Demonstration Setup

1. Execute the test.bat under the demo_batch folder to configure FPGA and launch the Nios II program (Note*).
2. Tilt the VEEK-MT2S to all directions, and you will find that the angle of the g-sensor and value of light sensor will change. When turning the board from -80º to -10º and from 10º to 80º in Y-axis, or from 10ºto 80º and from -80º to -10º in Y-axis, the image will invert. **Figure 5-13** shows the demonstration in action. The values measured from the ambient light sensor is displayed on the top left of the GUI.

Figure 5-13 Digital Accelerometer demonstration

Note: Executing G_sensor \demo_batch\test.bat to download .sof and .elf files.

5.5 E-Compass Demonstration

This demonstration shows an e-compass on the LCD panel based on information measured by accelerometer and magnetometer in the MPU9250. There are two I^2C controllers are used in this demonstration. One is use to communicate with touchscreen chip in the LCD module, and another is used to communicate with the MPU9250 to retrieve the measured gravity and magnetic field information. The compass angel is calculated based on the gravity and magnetic field measured by the MPU9250. The calculated compass angle will be displayed on the LCD panel. A bubble level is also displayed on the LCD panel to indicate the level surface. Note, the magnetometer should be calibrated before the e-compass will work accurately. The calibration process is also included in this demonstration.

■ Operation Description

Figure 5-14 shows the graphic interface of the e-compass demonstration. The left area show the gravity information measured by accelerometer in the MPU9250. Also, a bubble level is shown on the LCD. The center of LCD panel shows the e-compass. The north direction and magnetic field information are shown on the LCD panel. The Calibration button is used to start the calibration process for the e-compass.

Figure 5-14 Graphic Interface of e-compass demonstration

■ System Design Description

Figure 5-15 shows the hardware system block diagram of this demonstration. The system is clocked by an external 50MHz Oscillator. Through the internal PLL module, the generated 100MHz clock is used for Nios II processor, SDRAM, SSRAM, and other high speed controllers. The PLL also generate a 20MHz for low-speed peripherals, such as LEDs and buttons. The NIOS II program is running on the SRAM and the SDRAM is used as video frame buffers in Altera VIP block. The I2C_OpenCore controllers are used to communicate with the touch screen chip and the MPU9250 chip. The VIP block is used to implement video streaming for displaying graphic on the LCD panel.

The video steaming is:

Nios II Processor → SDRAM → VIP: Frame Reader → VIP: Clocked Video Output → LCD Panel.

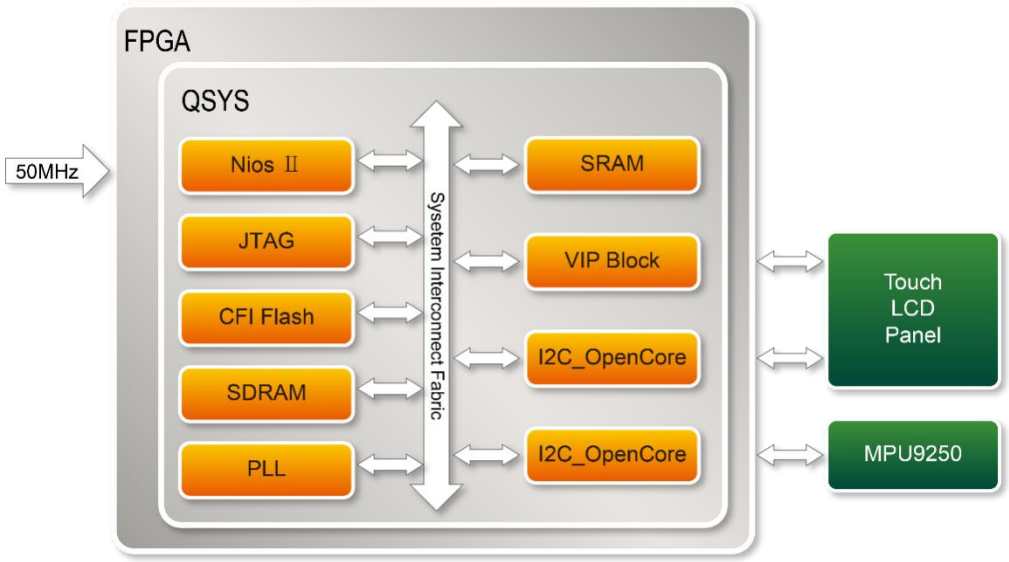

Figure 5-15 Block diagram of the e-compass demonstration

In the NIOS II program, the graphic interface is implemented in the gui_gsensor.cpp. The MPU9250 class (implemented in MPU9250.cpp.h) is used to retrieve accelerometer information from the MPU9250. The member function initialize should be called before calling any other member functions. In this demonstration, the member function getMotion9 is called to get the gravity value of X/Y/Z. The function Light_Init is designed to initialize the sensor APDS-9300. The function Light_GetID is designed to get the chip ID of APDS-9300. The Function Light_Get_ADCData0 and Light_Get_ADCData1 are designed to get the two ADC values in APDS-9300.

The section introduces the method to calculate the heading angle based on the values measured by accelerometer and magnetometer in this demonstration. If the compass is on a flat surface, then heading angle can be calculated with the following formula:

$$\arctan(Xh/Yh)$$

where X_h and Y_h are the magnetic field in X and Y directions. If the compass is not on a flat surface, the X_h and Y_h can be calculated with the following formula:

$$X_h = mx * \cos(pitch) + my * \sin(roll) * \sin(pitch) - mz * \sin(pitch) * \cos(roll)$$

$$Y_h = my*soc(roll) + mz*size(roll)$$

Where mx/my/mz are the measured valued by the magnetometer in the MPU9250. The pitch and

roll can be calculated by the gravity valued measured by the accelerometer in the MPU9250.

The measurement of magnetic field will be subjected to distortion. There are two categories of these distortions: the hard iron distortions and the soft iron distortions. The hard iron errors refer to the presence of magnetic fields around the sensor (magnets, power supply wires) and are related to the measurement offset errors, while the soft iron errors refer to the presence of ferromagnetic materials around the sensor, which skew the density of the Earth's magnetic field locally and are related to scaling offset errors. In other words, to get the correct magnetometer data you should calibrate it first. In this demonstration, we corrected the magnetic field with the following formula:

$$X_{calibrated} = X - X_{offset}$$

$$Y_{calibrated} = Y - Y_{offset}$$

$$Z_{calibrated} = Z - Z_{offset}$$

Where X_{offset}, Y_{offset} and Zoffset are the average value of X/Y/Z magnetic field, defined as:

$$X_{offset} = (X_{max} - X_{min})/2$$

$$Y_{offset} = (Y_{max} - Y_{min})/2$$

$$Z_{offset} = (Z_{max} - Z_{min})/2$$

Where $X_{max}/X_{min}/Y_{max}/Y_{min}/Z_{max}/Z_{min}$ are the maximal and minimal value of X/Y/Z values reported by magnetometer in the MPU9250 while the VEEK-MT2S is rotated in its X/Y/Z axes as shown in **Figure 5-16**.

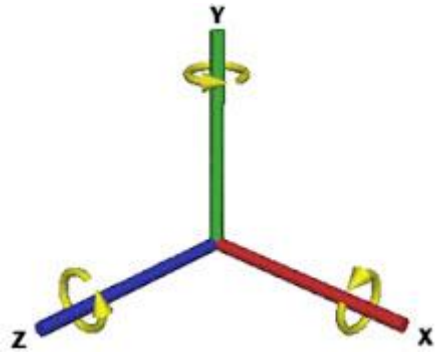

Figure 5-16 Rotate VEEK-MT2S around each of its 3 aces

■ Project Information

Item	Desciption
Tool	Quarts II Prime 16.1.1 Standard Edition
Quartus Project directory	Demonstration\FPGA\E_Compas
FPG bit stream file	DE10_Standard.sof
Nios II Workspace	E_Compass\software
Nios binary file	GUI_APP.elf
Demo batch folder	E_Compass\demo_batch

■ Demonstration Setup

1. Execute the test.bat under the demo_batch folder to configure FPGA and launch the NIOS II program (Note*).
2. The E-Compass will be displayed on the LCD panel.
3. Touch the Calibration button to start the calibration process. When the button is touched, the LCD content will be changed as shown in **Figure 5-17**.
4. Rotate VEEK-MT2S in X/Y/Z axis individually. Touch the Finished button when rotation is done.
5. When the Finished button is touched, the LCD content is changed as shown in **Figure 5-18**. The minimal, maximal, and average values are displayed in the LCD. If the value is valid, please touch Yes button to apply the calibration result.
6. Keep the VEEK-MT2S on as flat a surface as possible by watching the bubble level on the LCD. The e-compass will show the accuracy North direction on the LCD.

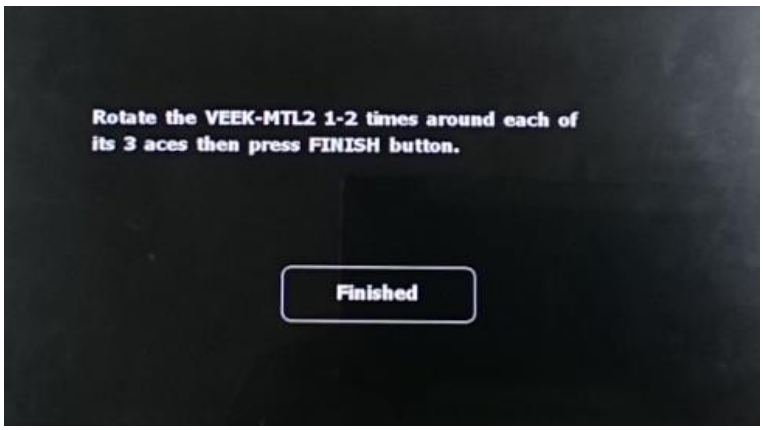

Figure 5-17 Hint for rotate VEEK-MT2

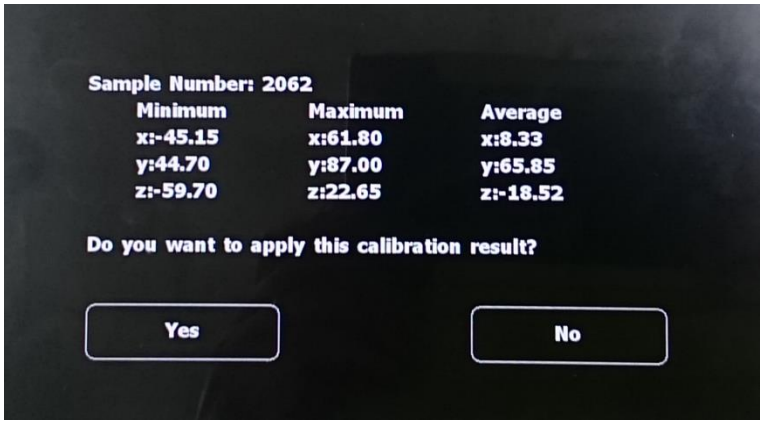

Figure 5-18 Magnetic Calibration Information

Note:

Execute E_Compass *demo_batch*\test.*bat* to *download .sof and .elf files.*

Chapter 6

HFP-FPGA Demonstration

Although HPS and FPGA can operate independently, they are tightly coupled via a high-bandwidth system interconnect built from high-performance ARM AMBA® AXITM bus bridges. Both FPGA fabric and HPS can access to each other via these interconnect bridges. This chapter provides demonstrations on how to achieve superior performance and lower latency through these interconnect bridges when comparing to solutions containing a separate FPGA and discrete processor.

The ADC9300 demo shows how to use the Linux application and to use the I2C driver to communicate with the ADC9300 chip on the FPGA sit. The MPU9250 demo shows how the Linux application is used via the SPI driver to communicate with the MPU9250 chip on FPGA sit. The System CD also controls a more complex demo, called “Control Panel”. The Control Panel is a QT based GUI application. It is a collection for controlling various hardware on VEEK-MT2S. For details about the demo, please refer to the document VEEK-MT2S_Contrl_Panel.pdf.

6.1 HPS_FPGA_ADC9300

This section shows how to implement a Linux console application to access the ADC9300 light sensor on the FPGA side. In the Linus/HPS site, I2C driver is used. In the FPGA side, the I2C opencore controller is used.

■ System Description

Figure 6-1 shows the system block diagram of hps_fpga_adc9300 demonstration. In the FPGA side, the Qsys component **i2c_opencore** is used to control the ADC9300 sensor through the I2C interface. The I2C driver drives the **i2c_opencore** controller through the Lightweight HPS-to-FPGA bridge. The hps_fpga_adc9300 uses I2C driver to access the registers in the ADC9300 to get measured lighting values.

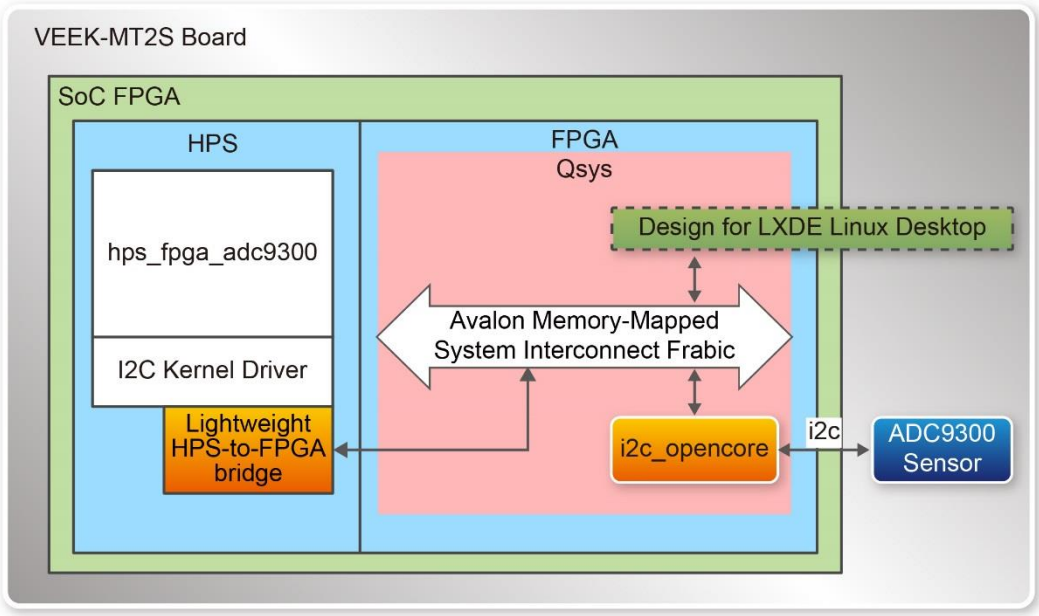

Figure 6-1System block diagram of adc9300 demonstration

The hps_fpga_adc9300 software application is implemented by C++ code. **Figure 6-2** shows the software stack of the software application. The main program is implemented in the main.cpp file. ADC9300 class is designed to provide relative functions for ADC9300 chip and it is implemented in adc9300.cpp.

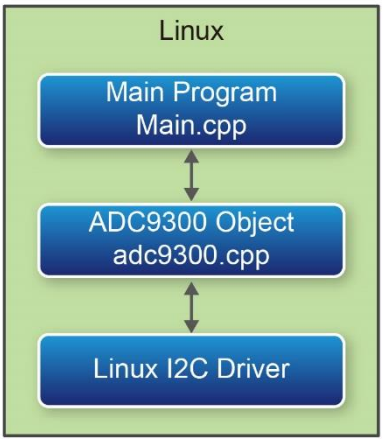

Figure 6-2 Software Stack of ADC9300 Application

Figure 6-3 shows the declaration of the ADC9300 class. In the constructor, a file name "/dev/i2c-5" should be given. In the VEEK-MT2S Desktop Linux Image, the I2C driver file "/dev/i2c-5" is mapped to the i2c_opencore controller for ADC9300 chip. Before reading the measured value, developers should call **Set_PowerSwitch** function to turn the power of ADC9300. Then, call **Get_ADCData0** and **Get_ADCData1** function to get measured lighting value for Visible+IR and IR, in respectively.

```
class ADC9300 {
public:
    ADC9300(const char *filename = "/dev/i2c-5");
    ~ADC9300();
    bool IsReady(void);
    bool Get_ADCData0(uint16_t *pData16);
    bool Get_ADCData1(uint16_t *pData16);
    bool Get_ID(uint8_t *pData8);
    bool Set_PowerSwitch(bool bSwitch);


protected:
        int m_file;
        bool REG_READ(int file, uint8_t CommandCode,uint8_t *value);
        bool REG_WRITE(int file, uint8_t CommandCode, uint8_t value);
        bool Get_ADCData(uint8_t RegLow,uint8_t RegHigh, uint16_t *pData16);

};
```

Figure 6-3 Declaration of ADC9300 Class

In the ADC9300 class, the ADC9300 register accessing functions are implemented as protected member functions **REG_READ** and **REG_WRITE**. Figure 6-3**Figure 6-4** shows the code body of these two function. The standard file **read** and **write** are used to implement these two functions.

```
bool ADC9300::REG_READ(int file, uint8_t CommandCode,uint8_t *value){
        uint8_t Value[1];
        bool bSuccess = false;

        // write to define register
        if (write(file, &CommandCode, sizeof(CommandCode)) == sizeof(CommandCode)){
                // read back value
                if (read(file, &Value, sizeof(Value)) == sizeof(Value)){
                        bSuccess = true;
                }
        }
                                *value=Value[0];
        return bSuccess;
}

bool ADC9300::REG_WRITE(int file, uint8_t CommandCode, uint8_t value){
        bool bSuccess = false;
        uint8_t szValue[2];
        // write to define register


        szValue[0] = CommandCode;
        szValue[1] = value;
        if (write(file, &szValue, sizeof(szValue)) == sizeof(szValue)){
                        bSuccess = true;
        }
        return bSuccess;
}
```

Figure 6-4 REG_READ and REG_WRITE Member Functions of ADC9300 Class

■ Demonstration Setup

Please follow the procedures below to setup the demonstration:

1. Verify the FPGA Configuration Mode Switch MSEL[4:0] = **01010**.
2. Check if the microSD card included in this kit is properly inserted.
3. Power on the VEEK-MT2S.
4. In LXDE desktop, tap the fpga_hps_adc9300 Icon twice to execute fpga_hps_adc9300.

5. When the Dialog pops up, as shown in Figure 6-5, press the "Execute in Terminal" button.
6. The measured light values will be displayed in the terminal window as shown in Figure 6-6.

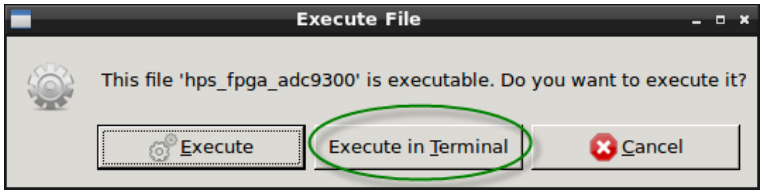

Figure 6-5 hps_fpga_adc9300 execution query dialog

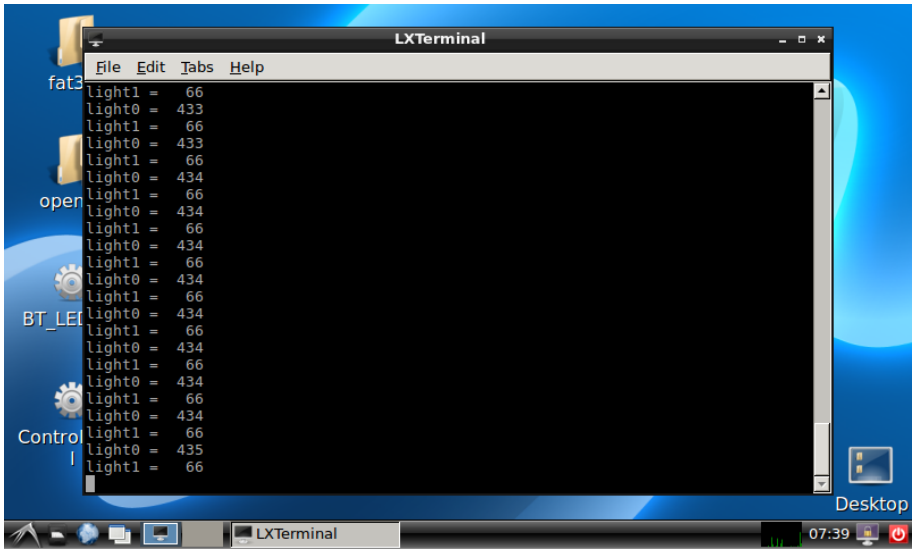

Figure 6-6 screenshot of hps_fpga_adc9300 demo

■ Project Information

Linux C++ project:

Item	Description
Compile Tool	SoC Embedded Design Suite (ESD) 16.1.0.196
Project directory	Demonstration\SoC_FPGA\hps_fpga_adc9300
Bin file	hps_fpga_adc9300
Main file	Main.cpp

Quartus project:

Item	Description
Compile Tool	Quartus Prime 16.1.2 Standard Edition
Project directory	Demonstration\SoC_FPGA\ContorlPanel\Quartus
Bin file	soc_system.rbf

6.2 HPS_FPGA_MPU9250

This section shows how to implement a Linux console application to access the MPU9250 9-axis sensor on the FPGA side. In the Linus/HPS site, SPI driver is used. In the FPGA side, the Qsys build-in SPI controller is used.

■ System Description

Figure 6-1 shows the system block diagram of hps_fpga_mpu9250 demonstration. In the FPGA side, the Qsys component **i2c_opencore** is used to control the MPU9250 sensor through the I2C interface. The I2C driver drives the **i2c_opencore** controller through the Lightweight HPS-to-FPGA bridge. The hps_fpga_mpu9250 uses I2C driver to access the registers in theMPU9250 chip.

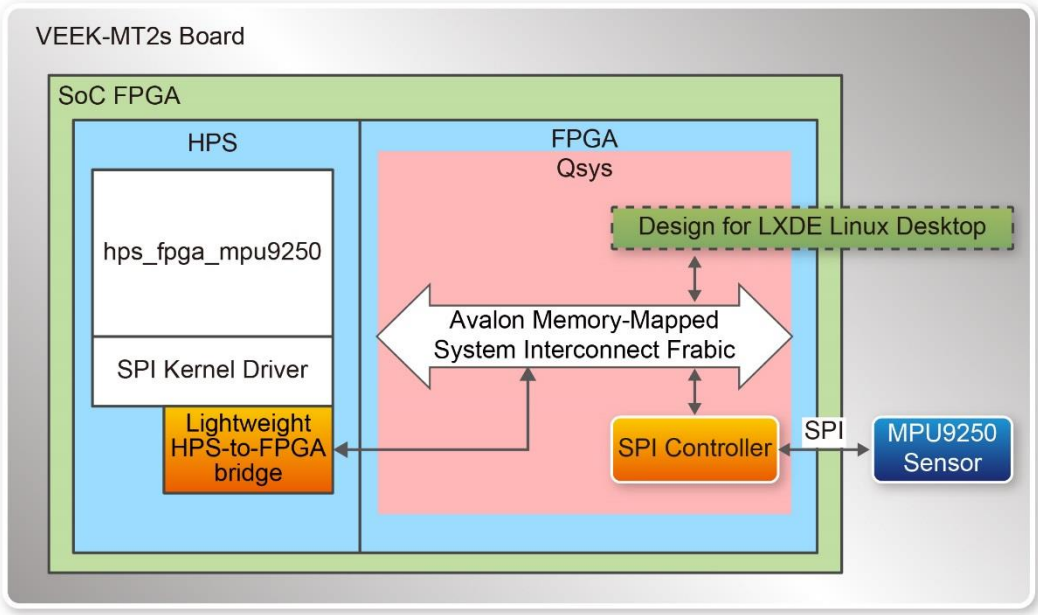

Figure 6-7System block diagram of mpu9250 demonstration

The hps_fpga_mpu9250 software application is implemented by C++ code. **Figure 6-8** shows the software stack of the software application. The main program is implemented in the hps_fpga_mpu.cpp file. MPU9250 class is designed to provide relative functions for MPU9250 chip and it is implemented in MPU9250.cpp.

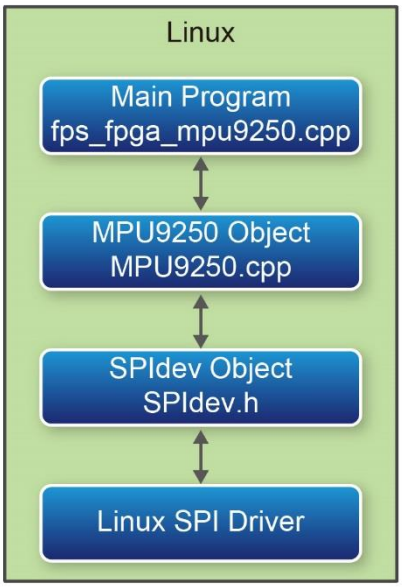

Figure 6-8 Software Stack of MPU92500 Application

Figure 6-3 shows the member function declaration of the MPU9250 class. Before reading the measured value, developers should call **initialize** function to initialize the MPU9250 chip. Then, call **GetMotion9** function to get measured 9-asix value.

```
class MPU9250 {
public:
    MPU9250();

    bool initialize(int sample_rate_div = 1, int low_pass_filter = 0x01);
    bool testConnection();

    unsigned int WriteReg( uint8_t WriteAddr, uint8_t WriteData );
    unsigned int ReadReg( uint8_t WriteAddr, uint8_t WriteData );
    void ReadRegs( uint8_t ReadAddr, uint8_t *ReadBuf, unsigned int Bytes );

    unsigned int set_gyro_scale(int scale);
    unsigned int set_acc_scale(int scale);

    void calib_acc();
        void calib_mag();

    void read_temp();
    void read_acc();
    void read_gyro();
    void read_mag();
    void read_all();

    unsigned int whoami();
        uint8_t AK8963_whoami();
        void getMotion9(float *ax, float *ay, float *az, float *gx, float *gy
    void getMotion6(float *ax, float *ay, float *az, float *gx, float *gy, fl
```

Figure 6-9 Declaration of MPU9250 Class

In the MPU9250 class, the MPU9250 register accessing functions are implemented as public member functions **WriteReg** and **ReadReg**. Figure 6-10 shows the code body of these two functions. In the code body of these two functions, a SPI driver file name "/dev/spidev32766.0" is used. In the VEEK-MT2S Desktop Linux Image, the file name is mapped to the SPI controller for MPU9250 chip. The **SPIdev** class is used to implement low-level translation. The SPIdev class is

implemented in the SPIdev.h file.

```
void MPU9250::ReadRegs( uint8_t ReadAddr, uint8_t *ReadBuf, unsigned int Bytes )
{
    unsigned int  i = 0;

    unsigned char tx[255] = {0};
        unsigned char rx[255] = {0};

        tx[0] = ReadAddr | READ_FLAG;

        SPIdev::transfer("/dev/spidev32766.0", tx, rx, Bytes + 1);

    for(i=0; i<Bytes; i++)
        ReadBuf[i] = rx[i + 1];

    usleep(50);
}
unsigned int MPU9250::WriteReg( uint8_t WriteAddr, uint8_t WriteData )
{
     unsigned int temp_val;

     unsigned char tx[2] = {WriteAddr, WriteData};
         unsigned char rx[2] = {0};

         SPIdev::transfer("/dev/spidev32766.0", tx, rx, 2);

     return rx[1];
}
```

Figure 6-10 ReadReg and WriteReg Member Functions of MPU9250 Class

■ Demonstration Setup

Please follow the procedures below to setup the demonstration:

1. Verify the FPGA Configuration Mode Switch MSEL[4:0] = 01010.
2. Check if the microSD card included in this kit is properly inserted.
3. Power on the VEEK-MT2S.
4. With your finger, double click the fpga_hps_mpu9250 icon on the LXDE desktop.
5. Click the **Execute in Terminal** button when a query dialog appears as shown in **Figure 6-11**.
6. The measured light value will be displayed in the terminal window as shown in **Figure 6-12**.

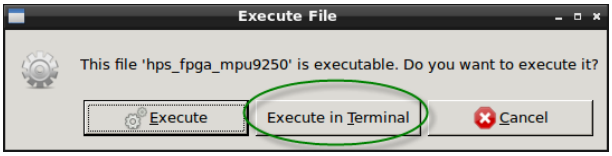

Figure 6-11 hps_fpga_mpu9250 execution query dialog

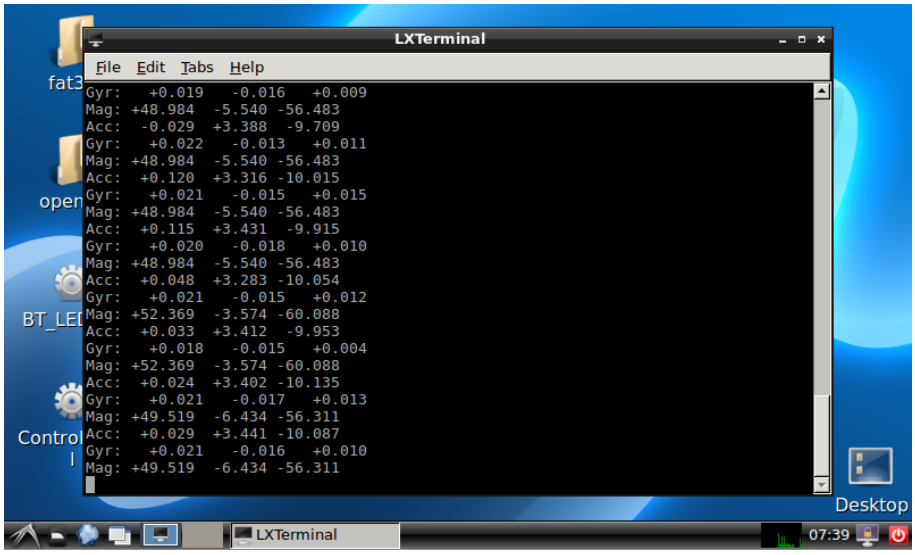

Figure 6-12 screenshot of hps_fpga_mpu9250 demo

■ Project Information

Item	Description
Compile Tool	SoC Embedded Design Suite (ESD) 16.1.0.196
Project directory	Demonstration\SoC_FPGA\hps_fpga_mpu9250
Bin file	hps_fpga_mpu9250
Main file	hps_fpga_mpu9250.cpp

Chapter 7

Appendix

7.1 Revision History

Version	*Change Log*
V1.0	Initial Version (Preliminary)

7.2 Copyright Statement

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Terasic:

K0161 K0161-EDU