|  |
| --- |
| **Worksheet** |
| LED Display Controller |

**Table of Contents**

[1 Abstract 2](#_Toc502392328)

[2 Concept 2](#_Toc502392329)

[3 Hardware Basis 6](#_Toc502392330)

[3.1 LED Modules 6](#_Toc502392331)

[3.2 Controller 6](#_Toc502392332)

[3.3 HDMI Input Converter 7](#_Toc502392333)

[3.4 Interface Board 8](#_Toc502392334)

[4 Results 9](#_Toc502392335)

[4.1 System’s Block Diagram 9](#_Toc502392336)

[4.2 Supported Functionality 10](#_Toc502392337)

[4.3 Device Register’s Map 10](#_Toc502392350)

[4.4 UART Protocol 11](#_Toc502392351)

[4.5 Considering about refresh rate 12](#_Toc502392352)

[5 Optional functionality to be supported 12](#_Toc502392353)

# Abstract

This document describes the results of the work on the controller for LED Matrix screen. The work was performed during September - October 2017. This report does not describes the works that were done for supporting MBI5153 (memory-based) LED driver and TMDS decoder.

# Concept

The proposed target application – transportation’s signage.

The basic characteristics were defined as following:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Parameter** | **Value** | **Note** |
| 1 | LED Matrix type | Full Color, 2 mm pitch |  |
| 2 | Video Input | Parallel RGB VGA 640x480, 800x480  Optionally:  - LVDS RGB  - HDMI/DVI | DPI 40-pin 0.5mm pith FFC |
| 3 | Color Depth | 24 bit |  |
| 4 | Refresh Rate | 100 Hz minimum | Adjustable |
| 5 | Processing | * ROI selection * Brightness control |  |
| 6 | Control Interface | UART (RS232) |  |

To realize what resources are needed to support the required parameters, the first bundle of experiments was run. It included the monochromatic LED matrix (Fig. 1) connected to an evaluation board with mid-capacity FPGA. Then, using the FPGA and embedded CPU (NIOS) programming the most important technical parameters were tested: a usage of the internal dual-port memory (to eliminate the triple buffering with a conventional RAM), scanning modes, dynamic image streaming, refresh rate results, and brightness control.

|  |  |
| --- | --- |
|  | **Fig. 1. The 1st LED Matrix prototype** |

In parallel, I requested the pricings for the LED Modules and separate LEDs. Surprisingly (but expectable) the LED cost within the complete LED module is twice (or even more) cheaper than by buying them separately. But additionally the Module is comprised of: 6-x layers PCB, driver’s ICs, frame, and the LEDs themselves.

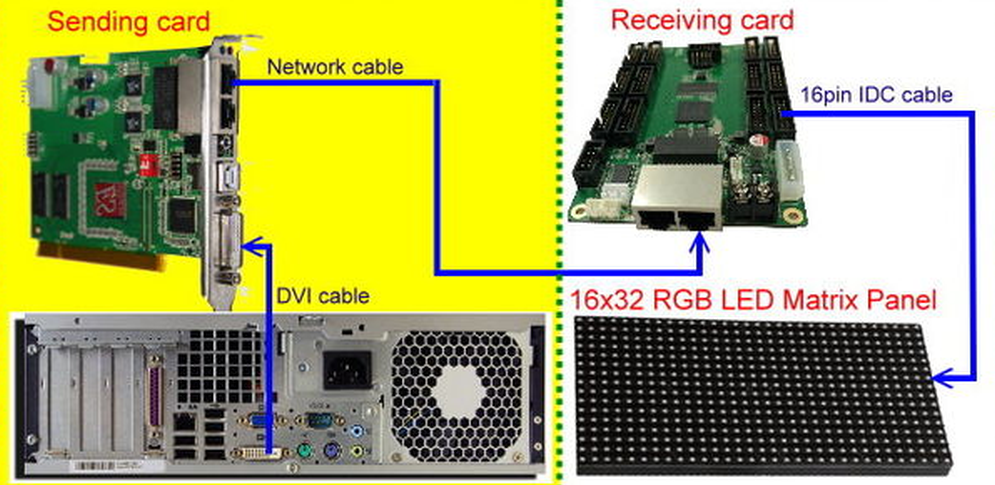
So, I decided that the optimal way to get a working display is to order the complete modules and then to connect them to our controller. At the moment of the order, I had no enough information how to work with the modules and had a couple of the questions about a system as well:

* How to control the modules to achieve the maximum brightness and refresh rate?
* What the difference between controlling the modules with the different drivers ICs?
* Power consumption of 1 module at the maximum brightness (I made the estimations later)
* Visible difference of the parasitic effects within the modules based on different drivers
* Pixel-to-pixel correction technique
* Module-to-module correction technique
* The functionalities of the testing and calibration software

All the questions above led us to order not just the modules, but a complete (Chinese) system for prototyping. Fortunately, the cost of the additional components was not high:

* So called “receiving card”, 2pcs: $50 each
* “sending card”: $150
* Cables and software: complimentary

It is important to understand that “chinese” control cards can be used only in a “pair” configuration: [Sending Card + Receiving Card] (Fig. 2). As to the throughput, 1 “Receiving” card can support 6 LED modules (64x64) maximum.



**Fig. 2. Typical configuration of the Chinese LED control hardware**

Eventually, I ordered:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Materials ordered** | **Supplier** | **Price** |
| 1 | 2.5 mm modules, 5 pcs.  (Driver IC: MBI5124)  Power and data cables | Aliexpress | LED Modules, 5 pcs: $300 |
| 2 | 3 mm modules, 2 pcs.  (Driver IC:  MBI5124 promised, ICN2038 received)  Sending Card,  Receiving card (2 pcs)  Power and data cables | Linsn (Aliexpress) | LED Modules, 2 pcs: $100  Rcv Card, 2 pcs: $80  Snd Card: $150 |

When the orders were placed, I began the works for the target FPGA programming.

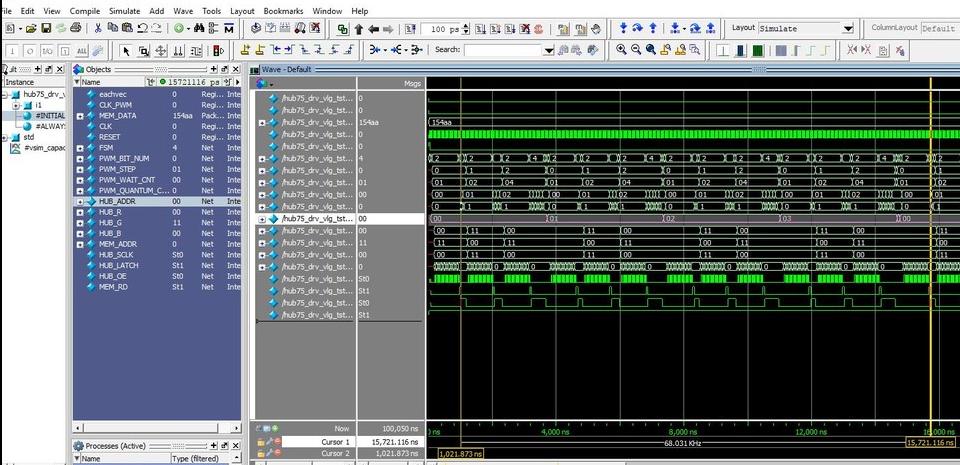
An initial concept implied the structure that is shown at Fig.3:



**Fig. 3. The initial concept of the LED Display controller and Matrix**

The works on programming the LED modules controller began on September 1st 2017. During the work the main blocks were written (Fig.4) and the necessary simulating was performed (Fig.5).

|  |  |
| --- | --- |
|  | **Fig. 4. The modules (firmware blocks) that were written** |



**Fig. 5. The simulation diagrams of the LED Modules driving circuit**

As a result, a prototype got working when the modules were received, 26th September 2017, (Fig.6):

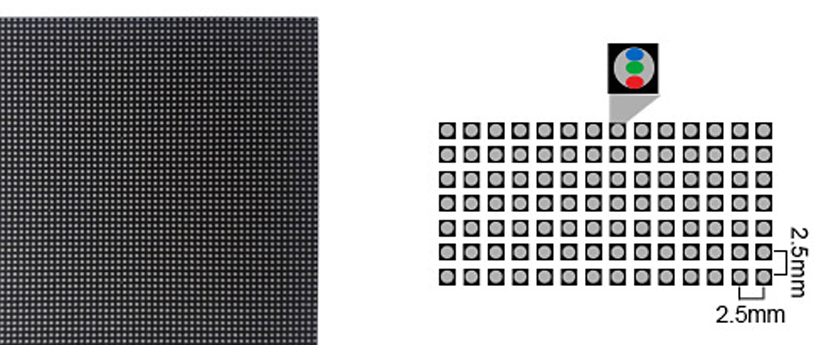


**Fig. 6. The 1st run of the LED modules**

After the first positive results about 2 weeks were spent for optimizing the design resources and make the firmware stable and platform-independent.

# Hardware Basis

## LED Modules



**Fig. 7. The LED modules that I ordered from China**

For the prototyping, the MBI5124-based modules were utilized.

MBI5124 is a well-documented driver IC, so there were no difficulties to support it in the firmware.

## Controller

Initially, for the hardware I have chosen the DE0 Terasic Cyclone iV board which I bought recently (Fig.8).

|  |  |
| --- | --- |
| http://www.terasic.com.tw/attachment/archive/593/image/image_60_thumb.jpg | **Fig. 8. FPGA Evaluation board used for prototyping** |

But a bit later I got a Terasic DE10 MAX10 board and ported the project for use on that board (Fig.9).

This recent developed “All-In-One” FPGA (Intel FPGA, formerly Altera) was chosen due to opportunity to reduce a number of components on the PCB. Using this FPGA, I could eliminate:

* External configuration memory ($7)
* External Video-buffer memory ($5) and the appropriated cost-valued resources on FPGA

So, an evaluation board from Terasic DE10-Lite is used for prototyping now. In case of need to migrate to another FPGA, there will no difficulties because the design is almost device-independent.

|  |  |
| --- | --- |
|  | **Fig. 9. FPGA Evaluation board used for “recalling” the project in 2017/10** |

## HDMI Input Converter

Unfortunately, I had no any “Free Parallel RGB Generator” and I decided to make an iron-made PCB for HDMI converter using TI HDMI receiver:

|  |  |
| --- | --- |
|  |  |

**Fig. 10. An HDMI converter made by laser printer and iron 😊**

## Interface Board

For connecting the DE1-Lite board to the LED modules, the signals’ level converter is necessary. Also, the parallel RGB input has to be built somewhere. For that purposes I made the schematics and simple PCB (Fig. 11). The Interface board comprises of:

* Parallel RGB (DPI) input
* Connector to the FPGA processing board
* Signal’s level converter
* Connector to the LED modules

|  |  |
| --- | --- |
|  |  |

**Fig. 11a. The interface board**

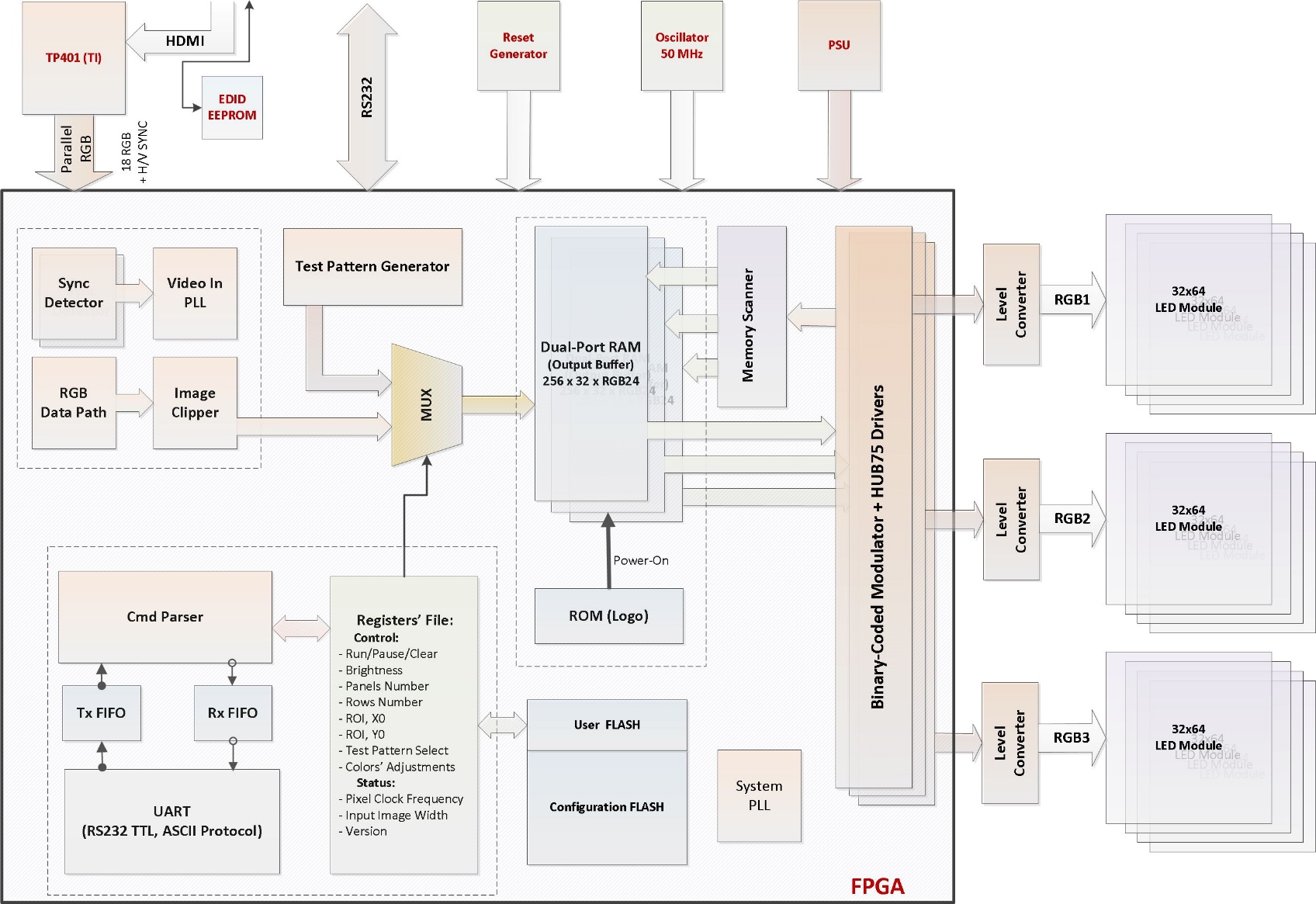


**Fig. 11b. All connected together**

# Results

## System’s Block Diagram

During the prototyping, it became clear that some blocks need a longer time for realizing, and other approaches may make the whole design simpler and easier for implementing. Thus, the initial block diagram was changed (Fig 12):

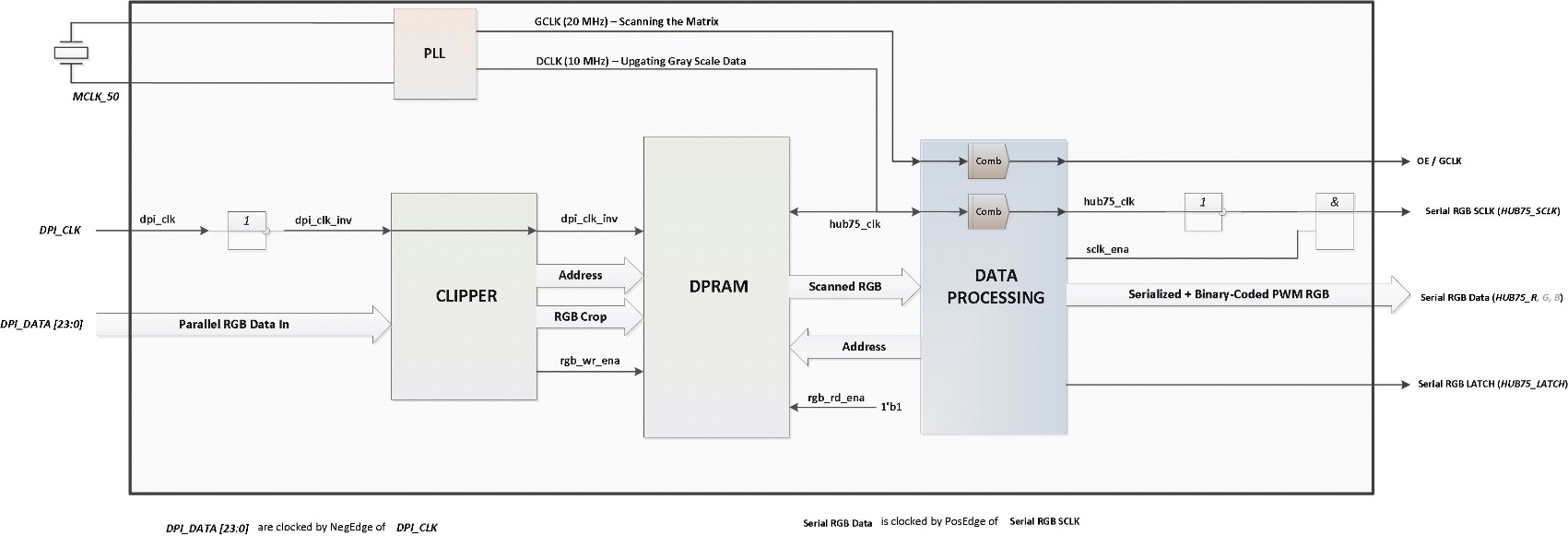


**Fig. 12. A block diagram of the current prototype**

## Constraints for the project

For stable compiling and running of the FPGA configuration the synchronous clock-source approach was used. Appropriated SDC file was written for setting the limitations of the propagation delays for the Parallel RGB bus, and for output serialized RGB data.

A rough clocks network of the project is shown on the Fig. 13, below.



**Fig. 13. A clocks-tree diagram**

## Supported Functionality

For now, the following functionalities are supported:

|  |  |  |
| --- | --- | --- |
|  | **Functionality** | **Note** |
| 1 | MBI5124 driver IC support | The most popular driver IC |
| 2 | Up to 8 (selectable)  64x64 modules support | Limited by on-chip memory |
| 3 | Low-Level Configuration is: Up to 4 x32-lines rows  1-4 rows (selectable) | 1 row comprises of 32 lines,  thus the scan ratio is 1/32 |
| 4 | DPI (Parallel RGB) input:  640x480 or 800x480 resolution  up to 24 bits per pixel (bpp) | External HDMI/DVI receiver  Currently, 18 bpp |
| 5 | Automatic Synchro Detection | Automatic recognizing of the input image format |
| 6 | Setting of the Region Of Interest | Left-Top corner coordinates |
| 7 | Image Clipper | Crop the image data according to the ROI |
| 8 | Global Brightness Control | 1. Controlled by PWM Quantum   of the binary-coded modulation   1. PWM over #OE line |
| 9 | Test Pattern Generator | Test images |
| 10 | VGA output | For test and debug purposes |
| 11 | UART Control | ASCII-based protocol: setting configuration, reading back the registers and firmware version |
| 12 | Converter Tool for creating Power-On Logo | .m files (Matlab or Octave) |

## Device Register’s Map

For setting the operating modes and getting status of the controller, the register’s file is implemented

in the FPGA.

These are 16 32-bits registers; first 8 registers (0x00..0x00) are accessible in Read/Write mode, the upper 8 registers (0x09..0x0F) are Read-only and represent status information.

|  |  |  |
| --- | --- | --- |
| **Addr** | **Purpose** | **Default (RST) value** |
| **0x00** | GCTRL Reg  Bit-field:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | | aresp | rsrv | rsrv | rsrv | rsrv | rsrv | clr1 | clr0 |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | rsrv | tpg1 | tpg0 | rsrv | rsrv | rsrv | rsrv | rsrv |   scan\_ena: enables memory scan and output to LED Modules  rfs\_ena: enables refresh video-RAM by DPI or TPG stream  self\_test: runs the self-test  led\_test: runs the test of the connected LED Modules Chain  src\_dpi\_tpg: selects the video image stream source (DPI or TPG)  tpg[1:0]: test pattern selection  00: vertical bars  01: plain color  10: black screen  11: grey gradient  aresp: auto-response enable  Bits 31..9 – reserved | 0x08003 |
| **0x01** | Brightness (OE signal modulation factor)  Allowable values: 0x0000..0x000F | 0x0002 |
| **0x02** | Image width (in pixels) | 0x00C0 |
| **0x03** | Scan ratio  Allowable values: 0x0000..0x001F | 0x0010 |
| **0x04** | Number of lanes (sub-panels that are scanned in parallel) | 0x0002 |
| **0x05** | PWM quantum  A smallest time-value for LED lighting during the binary-coded PWM  Allowable values: 0x0001..0x000F | 0x0002 |
| **0x06** | ROI\_X0  11-bits X coordinate of the ROI’s left-top corner | 0x0080 |
| **0x07** | ROI\_Y0  10-bits Y coordinate of the ROI’s left-top corner | 0x0080 |
| **0x08** | HPD/TRIG (bit 0)  Selects Trigger function for the HPD signal | 0x0000 |
| **0x19** | DPI Frequency (kHz)  Indicates the actual pixel clock frequency at the DPI (Parallel RGB) Input |  |
| **0x1A** | DPI H-Size  Indicates the actual visible image width at the DPI Input (e.g., 640 or 800) |  |
| **0x1E** | Firmware Version, maj. Format: 0xYYMM  e.g.: 0x1310 |  |
| **0x1F** | Firmware Version, min. Format: 0xDDHH  e.g.: 0x1416 |  |

## UART Protocol

To access the Register’s File the UART (RS232) interface is added to the Controller’s code.

The settings of the UART are typical: **9600 8N1** . For sending and receiving the data and commands,

an ASCII-based protocol is used.

The commands are split by two types: *Set Commands* and *Get Commands*.

The Set Command comprises of 10 ASCII symbols:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **<** | **A** | **x** | **=** | **y** | **y** | **y** | **y** | **/** | **>** |
| Start of  Packet | ‘A’ symbol  (means address) | Register’s  Address  ( HEX: 0..F ) | ‘=’ symbol | Data to set  ( 4-digits HEX value ) | | | | End of  packet | End of  packet |

Examples of the Set Command:

<00=8000/> - Color Bands Pattern

<00=8020/> - Grey Flat Pattern

<00=8120/> - Red Flat Screen

<00=8220/> - Green Flat Screen

<00=8320/> - Blue Flat Screen

<00=8040/> - Black Screen

<05=0000/> - stop scan

<05=000F/> - max quantum val (max brightness)

<08=0001/> - turn the Trigger function on the HPD pin On

The Set command may be appended with CR/LF symbols, but anyway the controller does not parse them.

The Get Command comprises of 7 ASCII symbols:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **<** | **A** | **x** | **=** | **?** | **/** | **>** |
| Start of  Packet | ‘A’ symbol  (means address) | Register’s  Address  ( HEX: 0..F ) | ‘=’ symbol | ‘?’ symbol | End of  packet | End of  packet |

Examples of the Get Command:

<01=?/> - get the actual value of the brightness register

<1E=?/> - get the major Version Data (YYMM)

<1F=?/> - get the minor Version Data (DDHH)

The controller returns the requested data in the same format as the Set command. For example:

<05=0002/> - the PWM Quantum value is ‘2’ (100 ns at 20 MHz SCLK)

<1E=1710/> - the version major is 1710, that means October 2017

## Considering about refresh rate

Increasing color depth will decrease the refresh rate:

A time T for processing of one full-color line :

where:

*M* – a number of the ICs in the chain

*N* – a number of bits per color

*Fsclk* – serial data clock frequency

*Q* – a PWM quantum (assuming, a binary-coded PWM is used)

then the refresh rate will be

where *Rm* – a multiplex ratio (16 or 32)

Refresh rate measurements on the running prototype:

|  |  |  |  |
| --- | --- | --- | --- |
| **PWM Quantum Value** | **Number of Panels** | **Color Depth** | **Timing Data** |
| **1** (50 ns @ 20MHz SCLK) | 5 (64x64) | 8 bit @ color | 1-bit Line WR Cycle: 16 us @ bit  Full Color Line WR Cycle: 144 us  Frame Cycle: 4.6 ms  Refresh Rate: **217**Hz |
| **15** (760 ns @ 20MHz)  (maximum brightness,  scan frequency may be increased by 20%) | 5 | 8 bit @ color | Full color Line WR Cycle: 322 ust  Frame Cycle: 10 ms  Refresh Rate: **100**Hz |

Thus, to achieve the higher frame rates, we have to increase a number of the driving ports (the number of the FPGA pins) or to use another LED driver ICs that implement the embedded PWM technique.

# Optional functionality to be supported

|  |  |  |
| --- | --- | --- |
|  | **Functionality** | **Note** |
| 1 | Pixel-to-Pixel correction | To make the image more “flat” |
| 2 | Module-to-Module correction | Must be done for normal mass-production displays |
| 3 | PC Software for calibrating | I have to review the prototypes  to decide what the functions are most important |
| 4 | Gamma-Correction | To control the contrast and color temperature |
| 5 | Increase the color depth (up to 14 bpp) | Connected with increasing the FPGA resources |
| 6 | On-Chip HDMI Receiver  (up to 1280x720) | To eliminate external TFP401;  External CML converters are needed |
| 7 | On-Chip HDMI Transmitter  (up to 1280x720) | To make the modules cascadable |
| 8 | More flexible Image Width/Height Control | To connect easily the modules with the size that differs from 64x64 |
| 9 | I2C Master | Should be supported for HDMI EDID |
| 10 | MBI drivers with display memory | done |
|  |  |  |