Volume-4, Issue-4, April-2015 • ISSN No 2277 - 8160

Research Paper | Engineering

Development of dsPIC based HMI for Real Time Monitoring, Controlling and Graphical Analysis of Industrial AC Drive

Maulik Timbadiya — Department of Electronics and Communication Engineering, Venus International college of Technology, Gandhinagar

Prof. Dimple Khant — Asst. Professor, Department of Electronics and Communication Engineering, Venus International college of Technology, Gandhinagar

ABSTRACT

AC Drive is a perfect solution for soft start, speed regulation, energy saving and intelligent control of Induction Motor. For demanding applications like crane (hoist), conveyor, stacker, centrifuge, etc. It delivers unmatched performance.

A researcher shows the Supervisory Control on drive is implemented using a Programmable Logic Controller (PLC) to monitor and control the drive performance. This paper focus on Design of Human Machine Interface (HMI) for Real Time Monitoring and Controlling Operation of AC Drive especially for Graphical Analysis of drive parameter such as torque, speed, temperature, etc., The design is basically divided in two sections out of which one is HMI controller section and second is AC Drive controller section which communicates using Modbus Protocol. HMI section includes Dot Matrix LCD and dsPIC33EP512MC506 series controller. AC Drive controller section includes TMS320F28069 DSP controller to control rectifier unit, DC bus and Inverter Unit. And it has certain controlling Algorithm and Fault Detection Algorithm. This AC Drive also includes Built-in PLC control that can be programmed by HMI.A rigorous Design and Development will be carried out using MPLAB X and dsPIC33 Development Board for PIC Controller, Code Composer studio and DSP starter kit for DSP Controller.

KEYWORDS : Modbus, Human Machine Interface, AC Drive, Graphical LCD

HMI Power Supply Design Using MC33063A

The MC33063A is easy-to-use ICs containing all the primary circuitry needed for building simple dc-dc converters. These devices primarily consist of an internal temperature-compensated reference, a comparator, an oscillator, a PWM controller with active current limiting, a driver, and a high-current output switch. Thus, the devices require minimal external components to build converters in the boost, buck, and inverting topologies.

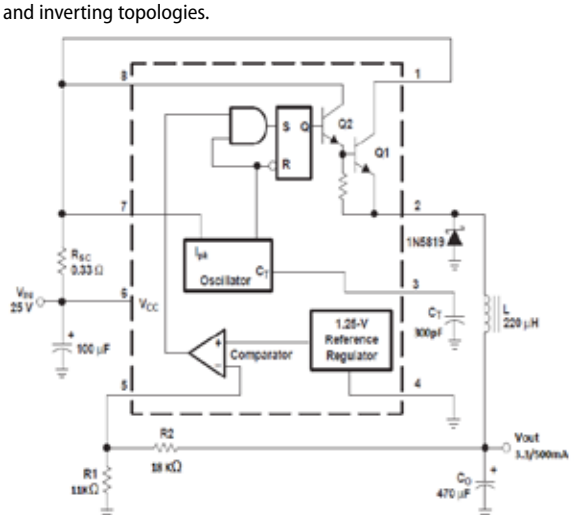

Figure 1 Functional Block Diagram of Step down DC-DC Converter

As show in figure, Here we are configured IC in buck topologies to get 3.3V DC from 24V DC. Pin no.5 of IC is comparator inverting input, make divider network so that it will give 1.25V at comparator inverting input pin. So, by comparing both this input value IC makes a close loop to maintain 3.3V continuously. For, Current limiting purpose a resistor is placed between Vcc and Ipk sense. Here we are using 0.3 Ω to limit 500mA current maximum, if current reach to 500mA then it will shut down the clock and it will further switch off RS flip-flop and final output will be zero.

Implementation Methodology

In firmware,first stage is oscillator initialization in which we are using 12 MHz external oscillator with dsPIC33EP512MC506. Before we going further, I am boosting my external clock frequency with PLL because of controller can support up to 70 MIPS. By using this feature we can achieve very high speed data processing. So, I am continuously wait in while loop until PLL is locked.Whenever phase is locked it will indicate by an acknowledgement flag.

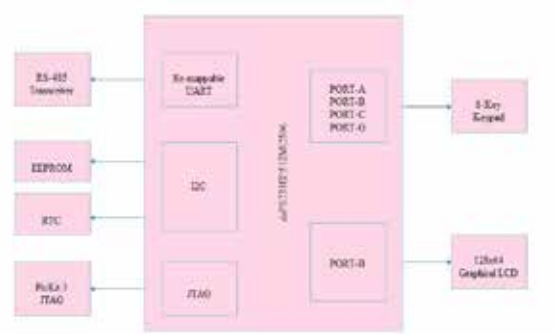

Figure 2 Simplified Block-diagram of HMI

At this stage controller working on 70 MHz, after that we are able to start further processing so first we initialize General Purpose Input Output (GPIO). After GPIO is initialized, we can now able to initialize GLCD. After that according to our need of different types of peripheral we would like to initialize UART, TIMER, I2C and RAM peripherals.

Now we are going to do our main task that is real time monitoring and controlling of AC drive. We are continuously working on timely divided process inwhile(1); statement. It will repeat until controller is restarted. In while(1);, suppose any time any peripheral required a service then it will be served according to that interrupts priority. Here whole Process of Serial and I2C communication done through the interrupt basis because of polling method is not reliable in for a large coding like this HMI. To get real time and high speed data we must go through the interrupts based process.

After successfully completion of Interrupt service routine, controller need to get back when it will cause by interrupt and need complete present activity it is done by the stack pointer.

Modbus Query-Response Model

When a message is sent from a master to a slave device the function code field tells the slave what kind of action to perform [1].

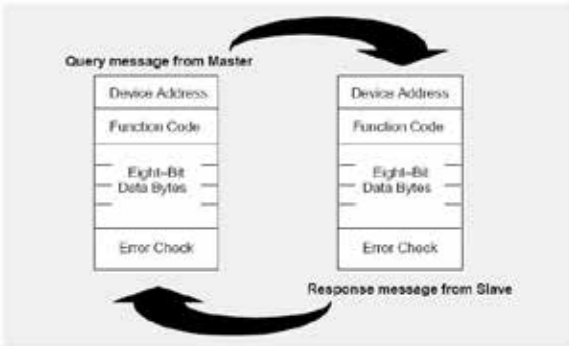

Figure 3Master–Slave Query–Response Cycle

When the slave responds to the master, it uses the function code field to indicate either a normal (error–free) response or that some kind of error occurred (called an exception response). For a normal response, the slave simply echoes the original function code. For an exception response, the slave returns a code that is equivalent to the original function code with its most–significant bit set to logic 1. For example, a message from master to slave to read a group of holding registers would have the following function code:

0000 0100 (Hexadecimal 04)

If the slave device takes the requested action without error, it returns the same code in its response. If an exception occurs, it returns:

1000 0100 (Hexadecimal 84)

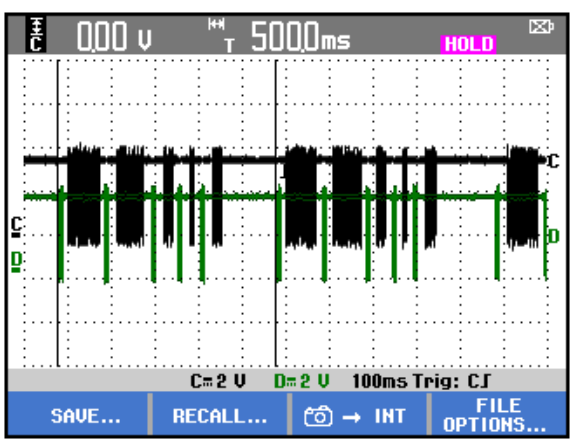

Figure 4 Query–Response Cycles at 500ms

In addition to its modification of the function code for an exception response, the slave places a unique code into the data field of the response message [1]. This tells the master what kind of error occurred, or the reason for the exception. The master device's application program has the responsibility of handling exception responses. Typical processes are to post subsequent retries of the message, to try diagnostic messages to the slave, and to notify operators. In monitor mode parameter, we are doing queries on every 500ms for real time monitoring.

dsPIC33E performance evaluation with HMI

To evaluate performance of dsPIC33EP512MC506, we should go through the firmware portion. First of all for oscillator verification we set timer of 2ms with oscillation frequency of 70MHz. And as we can show on digital oscilloscope waveform, it is clearly shows that timer works properly. For the timer verification I put one digital IO as output and ON-OFF continuously on timer interval.

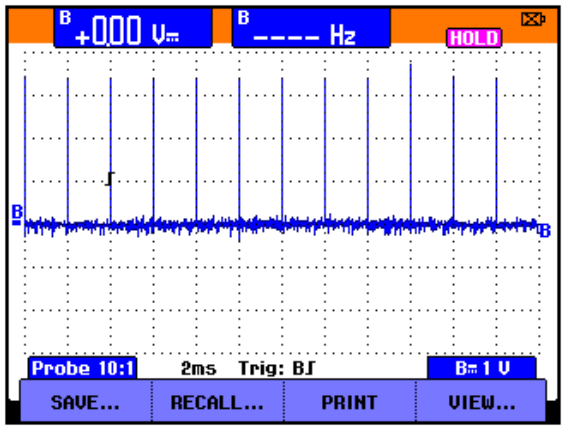

Figure 5 2ms Timer Verification

In firmware of display we have developed user friendly navigation system. In which, hundreds of parameter are divided in terms of different MODEs, GROUPs and pages. For example if parameter is M101 means that MODE-M, GROUP-1 and 1st parameter of that group. Instead of going into any parameter user can continuously monitor value of 1-8 parameter simultaneously in normal mode.

Normal mode is accessible by the user by pressing NORM key from the keypad. If there is any fault occur in AC drive it will initiate instantly with the fault name. For analysis purpose we have stored 10 fault in fault history screen with the 8 different parameter related to AC drive. Like DC bus voltage, Output current, Output frequency and Kilo-Watt etc.

Time taken from the RTC is used to check fault occurrence time and after that it is stored in fault history. We have the screen shot of LCD for firmware verification purposes.

Parameter	Value
HMI Power Supply Design (DC-DC step down)	500 mA at 3.3 Vdc (able to deliver up to 1.5A)
Ripple Frequency	<3% of Vout
Modbus Protocol	•Communication with control card through serial port (UART) using RS – 485 & Modbus RTU protocol •Baud Rate – 9600 bps •Frame Structure – 1 start bit, 8 data bits, 1 parity bit, 1 stop bit •Parity – Odd parity
I2C Protocol	100 KHz Clock (EEPROM and RTC)
LCD Access System	Row : Paging System Column : Line System

Table 1 HMI parameter and its value

Conclusion

I have successfully completed hardware design of HMI. I simulated different DC-DC step down converter and finally concluded that MC33063A IC is a perfect solution according to this application. In this DC-DC converter, I have also achieved ripple frequency less than 3% for better hardware efficiency. I have successfully interfaced GLCD, TRANSCEIVER, KEY-PAD, EEPROM and RTC with dsPIC33EP512MC506. Also I have developed firmware for HMI and conclude that it works satisfactory. I have implemented RS-485 modbus protocol on firmware and successfully communicate between HMI (dsPIC33EP512MC506) and AC drive control (TI DSP) card.

REFERENCES

[1] Modicon Modbus Protocol Reference Guide, PI–MBUS–300 Rev. J | [2] Sreejeth M.; Singh M.; Kumar P., "Monitoring, Control and Power Quality Issues of PLC Controlled Three-Phase AC Servomotor Drive", IEEEPower India Conference, Pg. no. 1-5, December 2012. | [3] He Yang; Li Kejian; CaiQizhong, "Design of ARM-based human-machine interface of plastic injection blow molding machine", IEEE International Conference on Computer Application and System Modeling (ICCASM 2010), Pg. no. 449-453, October 2010. | [4] Yanfang Wang; Wandui Mao; Jinying Li; Peng Zhang; "A Distributed Rectifier Testing System Based on RS-485",5th IEEE Conference on Industrial Electronics and Applicationsis, Pg. no. 779-781, June 2010. | [5] Karim I. A., "A Low Cost Portable Oscilloscope Based on Arduino and GLCD",3rd International Conference on Informatics, IEEE Electronics & Vision, Pg. no. 1-4, May 2014. | [6] Zhang Haibo; GaoShenyong; Li Daqing; Yu Feng, "An Efficient Chinese Character System on Embedded Systems", IEEEInternational Forum on Information Technology and Applications, Pg. no. 61-63, July 2010. | [7] Urbas L.; Doherr F., "autoHMI: a model driven software engineering approach for HMIs in process industries", CSAE, IEEE, Pg. no. 627-631, June 2011. |