Keywords: MAXQ, multimeter, display, I2C

APPLICATION NOTE 6315

MAXQ® TO THE RESCUE: A MULTIMETER GETS A NEW LEASE ON LIFE

By: Fons Janssen, Principal Member of the Technical Staff, and Field Application Engineer, Maxim Integrated

Abstract: This application discusses the repair of a vintage Philips multimeter using the MAXQ615. The original LCD was broken and beyond repair. A MAXQ615 is used to listen in on the data to the display driver, decode it, and display it onto a new display module.

Introduction

A while ago I purchased a second-hand Philips system multimeter that had a minor defect. The display was no longer legible, but I thought it would be easy to repair. Nothing could have been farther from the truth! No matter what I tried, there was no way to rescue the LCD. I needed a different solution using modern components. I found such a way using the MAXQ615. This application note discusses the steps I used to achieve the solution.

Standard Display Architecture

After studying the service manual for my meter, a Philips PM2535[1], I decided it must be possible to connect a different display module.

The LCD module in the meter is controlled by a Philips PCF8576 LCD driver IC. It receives its data from the main processor over an I²C bus. The idea was to tap this data stream, decode it, and then display the data on a different module.

The PCF8576 can drive up to 160 segments on an LCD module. The service manual describes exactly how the display segments are linked to the memory in the driver IC, which consists of a matrix of 40 rows of 4-bit nibbles. Each bit in the memory determines whether the corresponding segment is visible or hidden.

The data sheet for the PCF8576[2] describes the protocol for using I²C commands to fill the display memory. First, one or more command bytes are sent, followed by data bytes. With the aid of a logic analyzer, it quickly became apparent that the data is written to the display driver in two sessions. The first session is for nibbles 6 to 39, and the second session is for nibbles 0 to 5. These sessions are repeated over and over to continuously refresh the display. The data stream is shown schematically in **Figure 1**.

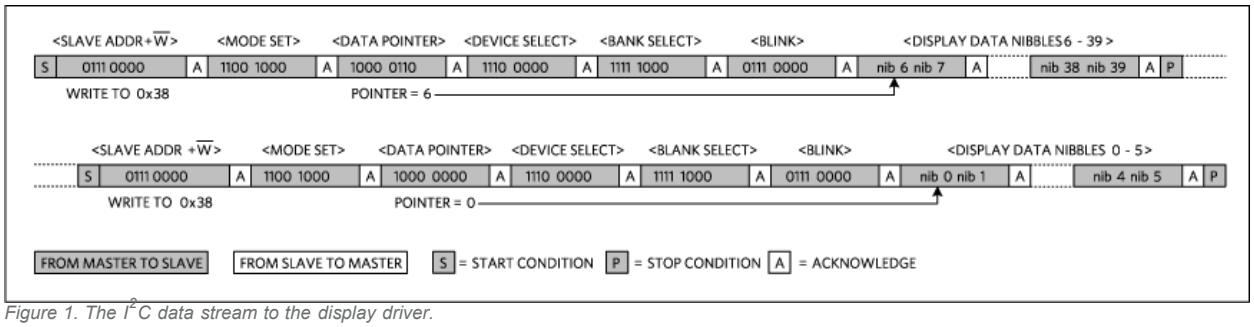

Figure 1. The I²C data stream to the display driver.

The (mode set), (device select), (bank select), and (blink) commands are configuration commands for the LCD driver. They always have the same values and are not important for our purposes. However, the (data pointer) command is important because it indicates where the data will be written in the memory. The commands are followed by the data bytes, with each byte containing two data nibbles. They are automatically written to the right locations in the display memory.

The New Display

I chose an HD44780-based display module with four rows of 20 characters to replace the original module. It has just enough room to display all the original data. A microcontroller acting as an I²C slave device can be used to tap the data stream to the PCF8576. It can also decode this data and write it to the new display module. For this task I used the MAXQ615. Along with an I²C port, the MAXQ615 has exactly enough I/O pins to drive the display module in 4-bit mode.

Figure 2 shows a simplified schematic diagram of the hardware. It is built on a MAXQ615 evaluation board, which has a perfboard prototyping area where the additional components are mounted.

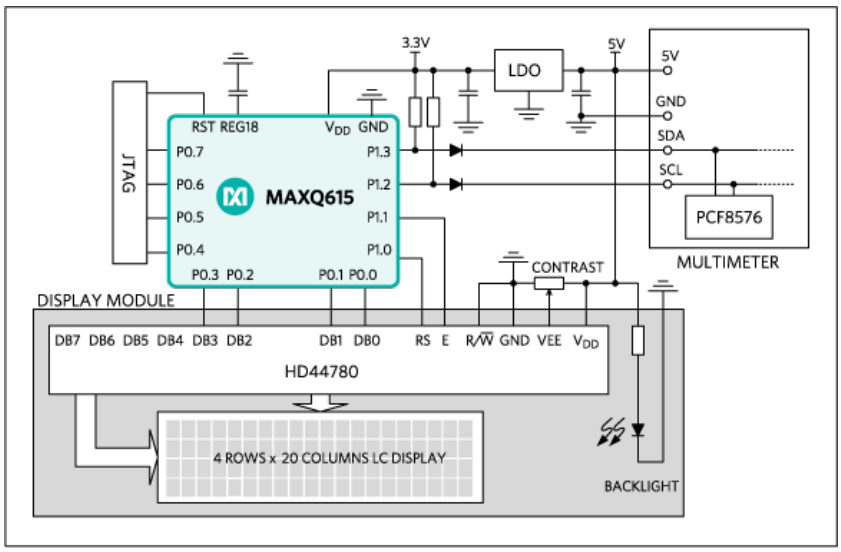

Figure 2. The hardware for driving the new display. The display module integrates an HD44780 display driver, a 4-line by 20-character LCD, and a backlight. It is connected to the multimeter by four lines: 5V and GND for power, and SDA and SCL for the data link.

The MAXQ615 operates at 3.3V and is, therefore, not compatible with the 5V logic levels used in the meter. Since the microcontroller only has to receive signals, the incompatibility issue can be solved easily by giving the SDA and SDL pins their own pullup resistors and isolation diodes. When the I C master pulls these lines low, the diodes conduct and the levels on the microcontroller inputs are also low. When the I C master sets these lines high, the diodes are reverse biased and the level is limited to 3.3V. When the MAXQ615 pulls the lines low (to acknowledge a command, for example), the diodes are also reverse biased. This prevents the MAXQ615 from "talking back" and disturbing communications between the main processor and the PCF8576.

The Firmware

The MAXQ615 has a hardware-based I C interface, so the firmware only needs to initialize a few registers to enable I C functionality and configure the IC as a slave device.

In the main routine the microcontroller waits for data on the I C interface. When the I C hardware recognizes its own slave address (the same as that of the PCF8576), the firmware knows that the next byte will be a command byte. All following command bytes are ignored except the data pointer command, which is used to determine where to store the data. The data bytes come after the last command byte in the string. The data bytes are stored in an array of 20 bytes (two nibbles per byte). At the byte level this array is a copy of the display memory in the PCF8576, with the associated pointer forming half of the data pointer at the nibble level.

As previously mentioned, the data is sent in two sessions. After both these data blocks have been received, the data can be processed. The content for the new display is held in a string of 80 characters. The firmware scrolls through the received data and determines what must be shown on the display and where it must be shown. That is easy for simple on/off segments, but it is more complicated for 7-segment and 16-segment characters. The bits belonging to a 7-segment character are grouped into one byte in the array, while the bits for a 16-segment character are grouped into two bytes. The characters can be derived from the byte values in a straightforward manner by using a lookup table. Once the final character has been determined, the display string is copied to the new LCD module in a single operation. The entire data processing architecture is shown in schematic form in **Figure 3**.

The MAXQ615 firmware is available here for download.

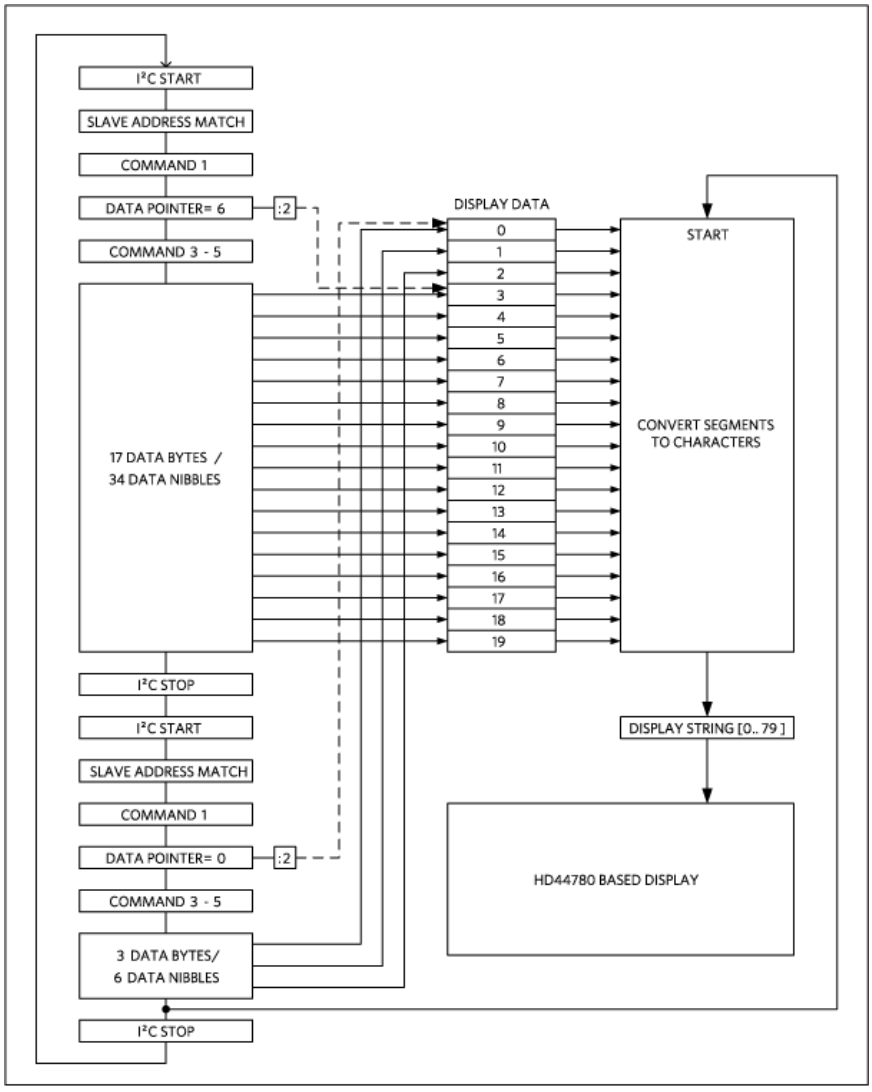

Figure 3. Schematic representation of the data processing architecture.

Installation

To make room for the new display, the original display was sawn out of the circuit board and moved to a different place in the enclosure, along with the MAXQ615 evaluation board. Then the new display was glued into the freed-up space.

As you can see from the photos of the meter with the original defective display (**Figure 4**) and the new display (**Figure 5**), the replacement display is not only legible, but also distinctly easier to read thanks to the backlighting.

Figure 4. The meter with the defective display, which was beyond repair.

Figure 5. The meter with the repaired display.

Conclusion

The meter is again fully operational, thanks in part to the MAXQ615, its evaluation board, and some creative engineering.

Footnotes:

1. Philips System Multimeter Service Manual PM2534-PM2535, No. 4822 872 35313 900205: www.download-service-manuals.com/download.php?file=Philips-6930.pdf.
2. PCF8576 data sheet: www.nxp.com.

A similar version of this application note appeared in July 2016, in *Elektor Magazine*.

Related Parts		
MAXQ615	16-Bit MAXQ Microcontroller with Hardware Multiplier	Free Samples
MAXQ615-KIT	Evaluation Kit for the MAXQ615	

More Information
For Technical Support: https://www.maximintegrated.com/en/support
For Samples: https://www.maximintegrated.com/en/samples
Other Questions and Comments: https://www.maximintegrated.com/en/contact

Application Note 6315: https://www.maximintegrated.com/en/an6315
APPLICATION NOTE 6315, AN6315, AN 6315, APP6315, Appnote6315, Appnote 6315