Clock and DOW Temperature Probe Example

A Clock and Dallas One-Wire example using OK202-25-USB

Application Note

Revision 1.0

Figure 1: The program Clock face running on an OK202-25-USB display

Introduction

The purpose of this example is to demonstrate how to access the Customer data of the display, communicate with a Dallas One-Wire device, as well as initialize and use Medium digits. We will be using the Alpha Library demo in C to easily access and send commands to the display. Within his or her Code the user will set the Com/TTY values creating the interface. The interface writes commands from the driver to the port and reads data from the port to the driver. The Alpha library can be found on our website at http://matrixorbital.ca/s/anotes. The library consists of the commands that are listed in the alphanumeric manuals, making it easier for the user to write higher level C Language code for their display.

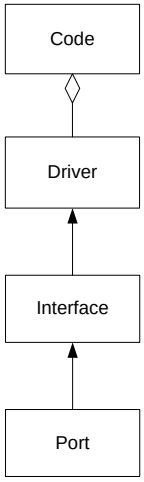

Figure 2: C Library Flow Chart

Included in the demo are the “AlphaDriver.c”, “AlphaDriver.h”, “AlphaInterface.c” and “AlphaInterface.h, which are the driver and interface respectively. The main program code can be found in the “ClockandTemp.c” and “ClockandTemp.h” files. Any changes to port settings are made in the “ClockandTemp.c” file.

Instructions for building the project in Windows and Linux environments are found within README.txt.

Port Settings

Interfacing to any Matrix Orbital display begins with the correct selection of the serial port and associated attributes. In C, a serial port object can be defined as an interface using the port name. For the sake of the demo please change the following line to reflect your port desired name in the code:

```
For Windows:
	#define PORTNAME "\\\\.\\COM54"
For Linux:
	#define PORTNAME "/dev/ttyUSB0"
```

Customer Data

One of the features that will be implemented in this application is the ability to store data directly on the display. Data can be written and stored on the display's memory using the Write Customer Data command. In this application, we will be taking temperature measurements using a Dallas One-Wire temperature probe. An all-time max temperature reading can be saved on the display, so that it can be read and displayed even after the program has ended, and the display has been power cycled.

If the display was used in a different application, different values may have been stored in Customer Data. Previous data can be cleared by writing 0's to the on board memory of the display. A function has been included that writes 0's to Customer data on the display.

```
#define ClearMaxTemperature 0
if (ClearMaxTemperature)
{
	ClearCustomerData();
}
```

You can enable the function to clear customer data by defining the ClearMaxTemperature macro as 1. By default, the ClearMaxTemperature macro is defined as 0, to ensure that customer data isn't cleared out unless permitted by the user. By enabling the function, customer data will be erased upon start up. The ClearCustomerData function is only intended to be used on the first run of the program to erase any previously stored data.

The WriteCustomerData command will be used to store the all-time max temperature reading on the display as well. It should be noted that although other data can be written on the onboard memory, there are only ~100,000 write cycles that the non-volatile memory can undergo.

The ReadCustomerData command is used to read the previously stored all-time max temperature.

```
ReadCustomerData(storedAllTimeMaxTemperature, 16);
```

Customer data is read and all 16 bytes of data is stored in a 16 byte storedAllTimeMaxTemperature array. From this array we can extract the all-time max temperature.

If this is the first time running the code, no all-time max temperature would have been stored on the display, and therefore, reading the customer data will return 0. Once the program begins taking temperature readings, it will store the max temperature reading(s) in customer data, so that it can be accessed the next time the program runs.

Dallas One-Wire

Certain Matrix Orbital USB model LCD's come with a Dallas One-Wire header that will allow a user to communicate to up to 32 devices. In this example, we will use the DOW functionality of the OK202-25-USB to communicate to a single Temperature probe. Depending on the DOW device you are using, you may have to send and receive different sets of data to attain the information you want. In this case, we will be communicating with a DS18S20 Temperature probe.

Before communicating to anything on the Dallas One-Wire line, we will need to obtain the address of each device. To obtain each device address, we need to search for one wire devices. Searching for a Dallas One-Wire device will return 14 bytes. A description of each byte returned can be found in the OK202-25 manual (www.matrixorbital.ca/ok202-25-manual).

```
SearchforaOneWireDevice(searchData, 14);
```

The SearchforaOneWireDevice command retrieves the DOW information, and stores it in the 14 byte searchData array, to be accessed later in the program. Now that we have the address of each device, we can communicate and read the temperature provided by the probe. In order to do this, the DallasOneWireTransaction command will be used.

As stated by the DS18S20 datasheet, by sending a Skip Rom command, followed by a convert temperature command, all DS18S20 temperature probes on the bus will perform simultaneous temperature conversions. In this case, we only have one probe that will convert temperature.

Using the Dallas One-Wire Transaction command, the initialSendBits array, containing both the Skip rom Command and Convert Temperature Command, is sent to the display.

```
uint8_t initialSendBits[2] = {204, 68};
DallasOneWireTransaction(initialSendBits, 2, 0, 16, 1);
```

Once the conversion is complete, you can begin a transaction to read the temperature value. This is done by sending the 10 byte tempCommand array containing the match rom command, DOW device address and the read rom command, to the DS18S20.

```
uint8_t tempCommand[10] = {85, 0, 0, 0, 0, 0, 0, 0, 0, 190};
DallasOneWireTransaction(tempCommand, 10, 72, 80, 3);
```

When the transaction is complete, the display will return 15 bytes. These 15 bytes get read, and stored in the 15 byte temperatureResponse array.

```
Read(temperatureResponse, 15);
```

The temperature value is the 6th index in the temperatureResponse array. The 7th index determines if the temperature is positive (0) or negative (0xFF). In order to display the temperature value, you will have to shift the value by 1.

A check max temperature function can be added simply by adding a variable that the current temperature is compared to. If the current temperature is currently more than the previously stored max temperature, it will be saved as the new max temperature. If a new max temperature is detected, an alert will be displayed on screen. If the new max temperature is greater than the all-time maximum temperature, it will be written to the customer data.

Medium Digit Clock

Your LCD can be made into a clock using data taken from a PC. The time_t function can be used to get the time as well as the day and date. This data can then be display on the LCD. Using medium numbers, the time can be displayed with 2 row height, rather than 1 row.

Once the data has been taken, medium numbers can be initialized and placed on the display. Medium Numbers only need to be initialized once, so they can be initialized upon the program startup.

```
InitializeMediumNumbers();
```

Medium numbers are two characters high and one character wide, which means anything written below the specified row will either be overwritten, or will overwrite the digit.

```
PlaceMediumNumbers(1, 1, Htens);
PlaceMediumNumbers(1, 2, Hones);
```

With the time displayed on screen, semi colons can be created and placed to separate the hours, the minutes, and the seconds on screen. This is done by using the SetCursorPosition command and writing periods in between the hours and minutes, and the minutes and seconds.

Data like the date, ante meridiem, and day will need to be converted into strings. All of this data will be written as regular characters.

Since each day is spelt with a different amount of letters, a quick calculation must be done to ensure that the day will always be placed on the far right, while also making sure that the full day is displayed.

Using the variables from the clock, a timer can be set to cycle through the clock face and the temperature display. At a certain time, the current temperature can be displayed, alerting the user of the current temperature. This is done by calling the temperature function at a certain time. By inserting the code into the while loop, the temperature screen will be displayed every 20 seconds.

Conclusion

With this program, your OK202-25-USB can display the current time using medium numbers, as well as day and date. In addition, it can take temperature measurements using a Dallas One-Wire device and display an alert whenever the probe detects a new max temperature. Temperature values can also be stored on the display for access in a separate session.

Contact

Sales
Phone: 403.229.2737
Email: sales@matrixorbital.ca

Support
Phone: 403.204.3750
Email: support@matrixorbital.ca

Online
Purchasing: www.matrixorbital.com
Support: www.matrixorbital.ca