A Digital Thermometer Using the AT89C2051 Microcontroller

AT89C2051 Flash Microcontroller

Application Note

Introduction

The system presented in this application note implements a simple digital thermometer that includes a built-in LCD and RS-485 communication port. It is designed around Atmel's AT89C2051 processor, a DS1620 digital thermometer/thermostat from Dallas Semiconductor, a small 8 X 2 LED backlit LCD, and an RS485 line interface. The system, shown in Figure 1, can be used as the basis for developing custom solutions for networked and stand alone data collection and control equipment. It can be centrally powered due to its low current requirement and its small size allows it to be placed almost anywhere.

Software

The LCD driver is written entirely in C and compiles under Micro-C (from Dunfield Development Systems) using the tiny memory model. Although a canonical stack-based implementation, Micro-C includes a number of special features that make it quite suitable for generating ROMable code for small systems. The overhead incurred performing stack manipulations is made up by the library functions that are all hand coded in an highly optimized assembler. As an added benefit, Micro-C comes with fully documented library source code so special modifications can be made as circumstances arise.

The first few functions contained in the LCD driver module are conventional C library implementations. PutString displays a null terminated string by merely passing characters off to PutChar until a null byte is encountered. PutChar outputs a character at a time to the LCD and handles the newline character by advancing the cursor to the beginning of the next line.

PositionLcd simply sets the cursor address to the value specified by the caller. The ClearLcd function is used to clear the entire LCD and home the cursor.

The functions that follow concern themselves with actual physical communication to the LCD. Since there is not a direct correspondence between the LSI's data RAM and the LCD's physical mapping, it is necessary to keep watch for certain boundary conditions. When a boundary is encountered, the cursor must be repositioned in order to keep the output contiguous. Since all displayable data must pass through DataWr, it makes sense to contain the corrective actions here. To handle this problem, keep track of the logical cursor position and invoke a remedial maneuver whenever discontinuity may occur. There are two ways you can accomplish this:

- Read the LCD's status register (to get the cursor address) or
- Keep a local copy just for your reference.

Not wanting to waste a pin to control the LCD's read/write line (it runs in write-only mode), the latter approach was adopted. Here, the global register (IRAM actually) variable Cursor is used for this purpose. Cursor is consulted prior to any data write operation. If a correction is necessary, a new cursor address is generated and dispatched to the LCD control register via CommandWr.

Following this, DataWr splits the data byte into nibbles (remember the LCD operates using a 4-bit bus) and falls

0597A-B–12/97

through to handle the actual physical transfer. Using Micro-C's extended preprocessor allows bit manipulation macros that expand directly to 8051 SETB and CLR instructions. Here, clearing DRS selects the LCD's data register and DEN is toggled to generate the data strobe. CommandWr operates similarly but does not have to deal with any cursor entanglements. It selects the command register as its destination by setting DRS high prior to clocking the nibbles across the interface.

The initialization function InitLcd begins at a more rudimentary nibble oriented level since no assumption can be made as to the operational status of the LCD at this time. The first three sequences ensure that the transfer mode is set to operate over a 4-bit bus. Repeating the sequence three times ensures that the command will be recognized regardless of the operational mode of the LSI. (It is wise to make no assumptions when performing any low-level initialization.) Following this, the actual operating parameters are transferred to the LCD using the standard CommandWr function. Software for this application note may be downloaded from Atmel's Web Site or BBS.

Digital Temperature

Temperature acquisition is handled using the DS1620 thermometer/thermostat IC from Dallas Semiconductor. The DS1620 contains all temperature measurement and signal conditioning circuitry on-chip and presents the processor with a 3-wire digital interface composed of a bi-directional data line DQ, a reset input \RST, and a clock input CLK. The temperature reading is provided in a 9 bit, two's complement format. The measurement range spans from -55°C to +125°C in .5°C increments.

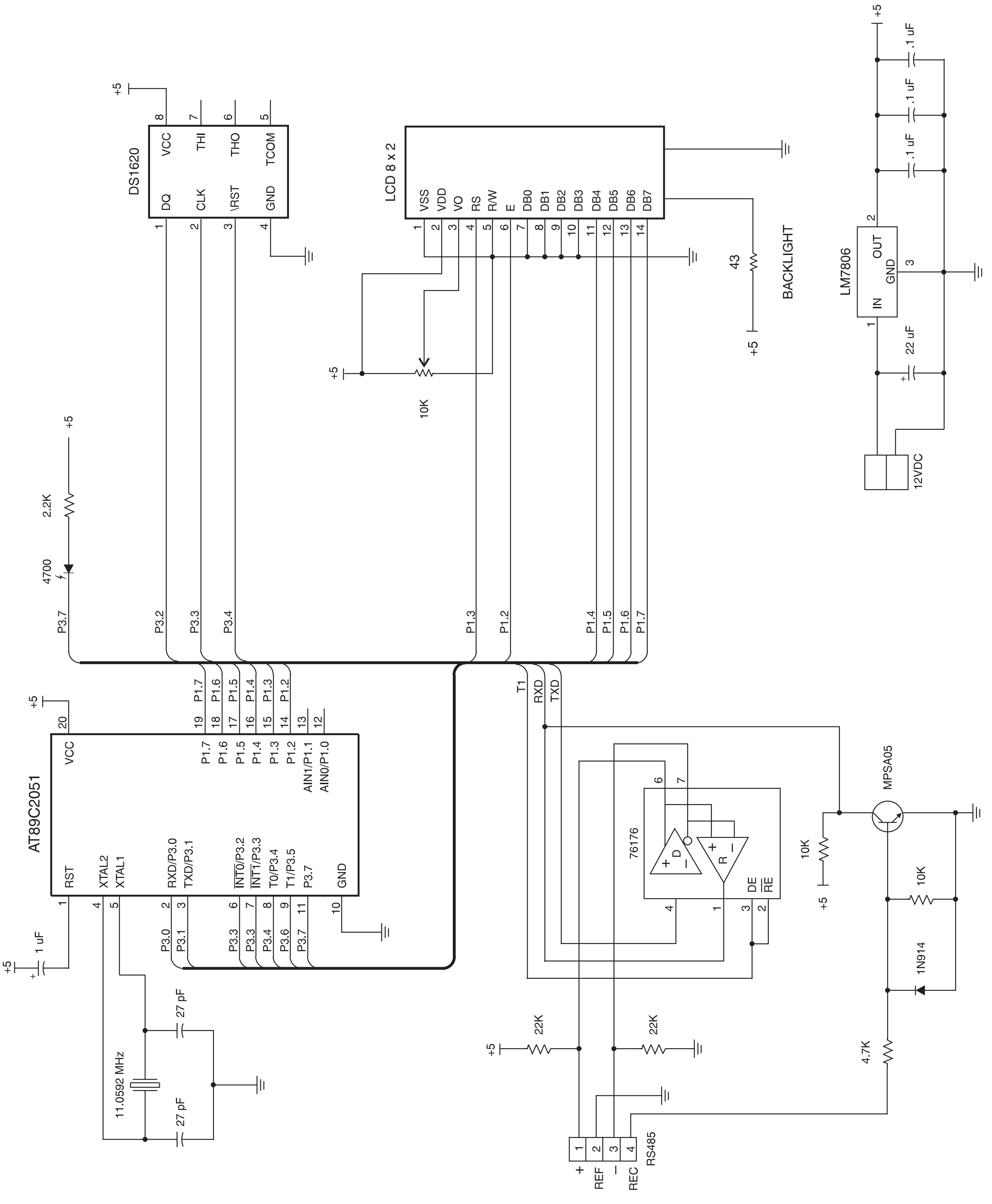
Data transfers into and out of the DS1620 are initiated by driving \RST high. Once the DS1620's reset is released, a series of clock pulses is emitted by the processor to actually transfer the data. For transmission to the DS1620, data must be valid during the rising edge of the clock pulse. Data bits received by the processor are output on the falling edge of the clock and remain valid through the rising edge. Taking the clock high results in DQ assuming a high impedance state. The sequence can be immediately terminated by pulling \RST low which forces DQ into a high impedance state and concludes the transfer. Temperature data is transmitted over the 3-wire bus in lsb first format. A total of nine bits are transmitted where the most significant bit is the sign bit. If all nine bits are not of interest, the transfer can be terminated at any time by asserting \RST.

The DS1620 support routines are coded in assembly. The DS1620 also has nonvolatile EEPROM configuration registers that hold thermostatic and operational control information. TempConfig is hardcoded to set the mode for operation under CPU control and continuous temperature conversion. Once in continuous conversion mode, the actual conversion process is started by issuing the start conversion command through TempConvert. Now the DS1620 can be read at any time and the last temperature conversion that was performed will be returned. This is accomplished by calling TempRead. The result is returned in the 16 bit accumulator as defined by Micro-C consisting of the B (msb) and ACC (lsb) registers.

Mainline Glue

Coordination of the support drivers is managed by the main module. This module takes control after the Micro-C startup routine finishes. On entry, the code initializes the serial port, instructs the DS1620 to start performing temperature conversions, initializes the LCD, displays a log on message, and enters into an endless loop. This loop continuously reads the DS1620, performs a Celsius to Fahrenheit conversion, translates the resulting binary number to an ASCII string, and displays the conversion result on the LCD. Periodically the code falls through and toggles an LED and transmits the temperature data serially.

The Celsius to Fahrenheit conversion is performed using the familiar equation: F = C * 9/5 + 32. Since the DS1620 returns temperature in .5°C increments the value is first divided by 2. Unlike the often impenetrable gyrations that result when working with numbers in assembler, the C representation of this calculation is perfectly clear in intent and function. A short sequence of divisions, modulos, and logical OR operations results in decimal ASCII values that make up the output string.


AT89C2051
RST
XTAL2
XTAL1
RXD/P3.0
TXD/P3.1
INT0/P3.2
INT1/P3.3
T0/P3.4
T1/P3.5
P3.7
GND
VCC
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
AIN1/P1.1
AIN0/P1.0
DS1620
DQ
CLK
\RST
GND
VCC
THI
THO
TCOM
LCD 8 x 2
VSS
VDD
VO
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
BACKLIGHT
43
10K
+5
2.2K
4700
1 uF
27 pF
11.0592 MHz
LM7806
IN
GND
OUT
.1 uF
22 uF
12VDC
76176
D
R
DE
RE
MPSA05
1N914
4.7K
22K
T1
RXD
TXD
RS485
REF
REC