A blue and gold logo

AI-generated content may be incorrect.

**Microcontroller-Based Solder Reflow Oven Control System**

* **MUHAMMAD SHARJEEL SHAH**

**2023514**

* **HAMZA AHMAD**

**2023217**

**Instructor: Dr.Memoon Sajid**

**Lab Instructor : Engr.Samina Yasmin & Engr.Rizwan Ali**

Table of Contents

[Introduction 3](#_Toc198386284)

[Literature Review 4](#_Toc198386285)

[System Overview 4](#_Toc198386286)

[ Microcontroller (PIC18F458) 5](#_Toc198386287)

[ Temperature Sensor (DS18B20 5](#_Toc198386288)

[ LCD Display (16× 5](#_Toc198386289)

[ PWM Outputs (Heater and Fan) 5](#_Toc198386290)

[ ADC Input (Kp Tuning 5](#_Toc198386291)

[ UART (Serial Interface) 5](#_Toc198386292)

[ Timer0 5](#_Toc198386293)

[Hardware Design 6](#_Toc198386294)

[ DS18B20 Sensor (One-Wire Interface): 6](#_Toc198386295)

[ LCD Display 7](#_Toc198386296)

[ PWM Control (Heater and Fan): 7](#_Toc198386297)

[ UART Interface: 7](#_Toc198386298)

[ ADC and Kp Tuning 7](#_Toc198386299)

[ Timer0 for Setpoint Timing: 7](#_Toc198386300)

[Software Design (Code Explanation) 8](#_Toc198386301)

[ Initialization 8](#_Toc198386302)

[ Interrupt Service Routine (ISR): 8](#_Toc198386303)

[o ADC Interrupt: 8](#_Toc198386304)

[ One-Wire and DS18B20 Functions: 8](#_Toc198386305)

[ Main Loop: 9](#_Toc198386306)

[1. Temperature Acquisition 9](#_Toc198386307)

[2. Error Computation: 9](#_Toc198386308)

[3. Proportional Control 9](#_Toc198386309)

[4. Actuator Output: 9](#_Toc198386310)

[5. LCD Update 9](#_Toc198386311)

[6. Serial Output 9](#_Toc198386312)

[7. Loop Delay: 9](#_Toc198386313)

[Control Strategy (Proportional Controller) 9](#_Toc198386314)

[Features of the P Controller: 10](#_Toc198386315)

[Simulation Setup 10](#_Toc198386316)

[**Results** 11](#_Toc198386317)

[ Temperature Tracking: 11](#_Toc198386318)

[ Control Output: 11](#_Toc198386319)

[ Kp Tuning 11](#_Toc198386320)

[Conclusion 12](#_Toc198386321)

# Introduction

A solder reflow oven must follow a precise thermal profile (preheat, soak, reflow, and cooling stages) to ensure reliable solder joints. Microcontroller-based controllers enable accurate temperature regulation and automation of these stages. In this design, a PIC18F458 microcontroller is used to implement the reflow oven controller. A Dallas DS18B20 digital temperature sensor is interfaced to the PIC to provide real-time temperature measurements. The sensor communicates over a single-wire bus, simplifying wiring and allowing ±0.5°C accuracy measurements. A character LCD displays the current setpoint and temperature, and a serial UART interface logs temperature data to a PC. Heater and cooling fan outputs are driven by PWM signals (via the CCP1 and ECCP1 modules) according to a proportional control law. The proportional gain KpK\_p is adjustable using an ADC input (potentiometer), and a Timer0 interrupt advances the setpoint through the reflow profile at 5-second intervals.

A prior microcontroller-based reflow oven design similarly transformed a toaster oven into a reflow system, using on-chip ADC and LCD to follow a multi-stage profile. The present project implements these concepts using a proportional control strategy, leveraging the code logic provided for the PIC18F458. The following sections detail the system’s literature basis, architecture, hardware interfaces, software logic, control strategy, and expected results.

# Literature Review

Previous projects have demonstrated the feasibility of microcontroller control for reflow ovens. For example, a design based on an 8-bit MCU (Hitachi H8/3687) used digital control and analog ADC sensing to regulate a conventional oven through preheat and reflow stages. Thermal profiling is often segmented into phases (preheat, soak, reflow, cooling), each with a target temperature setpoint. Feedback from temperature sensors allows closed-loop control. Digital sensors like the Dallas DS18B20 are popular in such systems because they provide high-resolution, calibrated temperature measurements over a one-wire bus. The DS18B20’s 64-bit unique serial code and multi-drop capability also permit multiple sensors on one bus.

Control strategies in the literature range from simple on/off regulation to full PID control. In low-cost implementations, proportional (P) or proportional–integral (PI) controllers are often used to reduce overshoot and improve profile tracking. In this project, a P-only controller is implemented for simplicity, with the gain tunable via an external potentiometer. The system continuously compares the measured temperature to a predefined setpoint profile and adjusts PWM duty cycles for heating or cooling accordingly.

# System Overview

The overall system (Figure 1) consists of a PIC18F458 microcontroller interfaced with a DS18B20 temperature sensor, a 16×2 LCD display, a UART serial interface, and PWM-driven heater and fan circuits. A potentiometer provides an analog input to adjust the proportional gain (Kp). The microcontroller executes a control algorithm in firmware: it reads the temperature, computes the error against the current setpoint, adjusts the PWM outputs, and updates the display and serial output. A Timer0 interrupt occurs approximately every 5 seconds to advance the reflow profile to the next stage.

* Microcontroller (PIC18F458) – Runs at 4 MHz, provides CCP and ECCP modules for PWM, on-chip ADC for gain tuning, and UART for serial output. System configuration bits disable the watchdog and configure oscillator.
* Temperature Sensor (DS18B20**)** – A 1-Wire digital sensor connected to RA4. Provides 12-bit temperature readings (0.0625°C resolution) with ±0.5°C accuracy.
* LCD Display (16×**2)** – 8-bit parallel interface (data on PORTB, control on PORTE) to show setpoint, temperature, phase, and controller status. Figure 2 shows the LCD interface (Placeholder).
* PWM Outputs (Heater and Fan) – CCP1 and ECCP1 modules generate 8-bit PWM signals. CCP1L controls the heater (via an SSR or MOSFET), ECCPR1L controls the cooling fan. Timer2 sets the PWM period (~1 kHz with PR2=249 and prescaler 1:4). Figure 4 illustrates the PWM signal behavior (Placeholder).
* ADC Input (Kp Tuning**)** – AN0 (RA0) reads a potentiometer voltage. The 10-bit ADC (left-justified) is used to set the proportional gain KpK\_p. The ADC runs continuously, and each conversion completion triggers an interrupt to update KpK\_p.
* UART (Serial Interface) – Transmits ASCII temperature and setpoint data to a PC. The serial port is initialized in asynchronous mode (BRGH=1, SPBRG=25 at Fosc=4 MHz yields ~2400 baud). Figure 3 shows the UART block diagram (Placeholder).
* Timer0 – Configured with prescaler to overflow every ~5 seconds. On each overflow (interrupt), the code increments the current setpoint index to move to the next phase. After the final phase, Timer0 is disabled and outputs are turned off.

**A computer circuit board with a screen

AI-generated content may be incorrect.**

# Hardware Design

A circuit board with wires and a display

AI-generated content may be incorrect.

The hardware interfaces mirror the firmware structure. Key design points include:

* DS18B20 Sensor (One-Wire Interface): The DS18B20’s data line is connected to RA4. An external pull-up resistor (~4.7 kΩ) is required on the data line since the code drives it actively for timing. The one-wire bus uses only this single data pin. The microcontroller implements the reset/write/read timing by bit-banging on RA4 (as per the OneWire\_WriteBit/ReadBit functions). A temperature conversion is initiated by issuing a 0xCC (Skip ROM) followed by 0x44 (Convert T) command, then waiting up to 750 ms. After conversion, the scratchpad is read (0xBE command) to retrieve the raw 16-bit temperature, which is converted to Celsius by multiplying by 0.0625.
* LCD Display**:** The LCD is driven in 8-bit mode for simplicity. Data lines (D0–D7) are tied to PORTB. Control lines RS, RW, and E are connected to LATE2, LATE1, and LATE0 respectively. The lcd\_init() function sends standard commands (function set 0x38, display ON 0x0C, entry mode 0x06, clear 0x01). Characters are written by placing ASCII values on PORTB and toggling .
* PWM Control (Heater and Fan): The PIC’s CCP1 module (single output PWM mode) is used for the heater, and ECCP1 (enhanced PWM) is used for the fan, allowing independent 8-bit duty control on two pins. CCP1CON and ECCP1CON are set to PWM mode (0x0C). Timer2 (with prescaler 1:4 and PR2=249) generates the PWM time base (approximately 1 kHz frequency). The 8 MSBs of duty cycle are loaded into CCPR1L or ECCPR1L. Logic-level MOSFET drivers or a solid-state relay can be connected to these PWM outputs to modulate the AC heater and cooling fan power.
* UART Interface: The PIC’s EUSART module is enabled for asynchronous transmission. SPEN=1, TXEN=1, SYNC=0 (asynchronous), BRGH=1, SPBRG=25 configure the UART (see). The TX pin (RC6) is configured as output by the hardware. Characters are sent by writing to TXREG and polling TXIF. In software, Send\_Serial\_Data() formats a string of the form Set: x.x, Temp: y.y, Kp: z using sprintf.
* ADC and Kp Tuning**:** AN0 (RA0) is configured as analog input (ADON=1). ADCON0 selects AN0 channel; ADCON1 configures left-justified result and VDD–VSS reference (single supply). The ADC conversion is started continuously (GODONE=1 initially). In the ADC interrupt, the high result byte (ADRESH) is scaled: Kp = 1 + (ADRESH >> 3), mapping the 8-bit ADRESH (0–255) to a gain range roughly 1–32. The interrupt then restarts a new conversion. This allows real-time adjustment of Kp via the potentiometer without affecting the main loop.
* Timer0 for Setpoint Timing: Timer0 is set to 16-bit mode with a 1:128 prescaler (T0CON = 0b00000110). The initial Timer0 value is 26437; on overflow (65536), this yields a period ≈5.0 seconds (at 4 MHz clock, 1 µs instruction cycle, 128 µs increment). Each Timer0 overflow triggers an interrupt where the code reloads TMR0, clears the flag, and increments currentSetpointIndex. When the index reaches the end of the 48-step profile, Timer0 interrupts are disabled and PWM outputs are set to 0, ending control.

# Software Design (Code Explanation)

The firmware is structured into initialization, interrupt service routines, sensor routines, and a main control loop:

* Initialization**:** In main(), data direction registers are set (e.g., TRISB = 0x00 for LCD data, TRISA4=1 for DS18B20 input). ADC is configured (ADCON1=0x00 for left-justification, AN0 analog) and turned on (ADON=1). PWM is configured via CCP1CON and ECCP1CON (PWM mode, Timer2 on). Timer0 is configured as above. UART is initialized (Initialize\_UART()) and LCD is initialized (lcd\_init()). Global and peripheral interrupts (GIE, PEIE) are enabled, along with ADIE and TMR0IE for ADC and Timer0 interrupts. An initial 1-second delay allows system stabilization.
* Interrupt Service Routine (ISR): The \_\_interrupt() function handles two flags:
  + **Timer0 Interrupt:** When TMR0 overflows, the ISR reloads TMR0 = 26437, clears TMR0IF, and increments the global currentSetpointIndex. If the index exceeds 47 (end of profile), Timer0 interrupts are disabled and both CCPR1L and ECCPR1L are set to 0, turning off heater and fan.
  + ADC Interrupt: When ADIF is set, the ISR reads the ADC result. Since ADCON1 is left-justified, ADRESH holds the 8 most significant bits. The code computes Kp = 1 + (ADRESH >> 3), scaling to about 1–32. ADIF is then cleared and a new conversion is started (GODONE=1).
* One-Wire and DS18B20 Functions: A set of functions implements the 1-Wire protocol on RA4:
  + OneWire\_Reset() drives RA4 low for 480 µs, releases it, waits, and reads the presence pulse (returning 1 if a sensor is present).
  + OneWire\_WriteBit() and OneWire\_ReadBit() perform the precise timing to write or read single bits. A logical 1 is written by a short low pulse, a 0 by a longer pulse. Reading is done by briefly pulling low and sampling the line.
  + OneWire\_WriteByte()/ReadByte() loop through 8 bits using the above.
  + DS18B20\_ReadTemp() sequences the conversion: it issues Reset, then Skip ROM (0xCC) and Convert T (0x44), waits 750 ms for conversion, issues another Reset, Skip ROM, then Read Scratchpad (0xBE). It reads two bytes (LSB, MSB), combines into a 16-bit signed value, and multiplies by 0.0625 to convert to °C. If the sensor is missing (no presence), it returns an error code (–1000).
* Main Loop: The main while(1) loop executes continuously as follows:
  + Temperature Acquisition**:** Call DS18B20\_ReadTemp(). If a valid temperature is returned (>–100°C), store it in currentTemp; otherwise, send an error message via UART.
  + Error Computation: Compute error = setpoints[currentSetpointIndex] - currentTemp. The setpoints array (length 48) contains the pre-defined profile temperatures in °C, changing every 5 s.
  + Proportional Control**:** Compute pwmDuty = Kp \* |error|. Bound pwmDuty to a maximum of 255 (8-bit). This implements a proportional control output proportional to the magnitude of the temperature error.
  + Actuator Output: If error > 0 (temperature below setpoint), set CCPR1L = pwmDuty to drive the heater, and ECCPR1L = 0 to turn off the fan. If error ≤ 0, do the opposite: ECCPR1L = pwmDuty (fan on), CCPR1L = 0. This ensures only one actuator is active.
  + LCD Update**:** The LCD is cleared (lcdcmd(0x01)) and rewritten. Line 1 shows “SP:xx°C T:yy°C” with degree symbols (0xDF). Line 2 shows the current phase name (Pre-heat, Soak, Reflow, Cooling based on the index) and the control status (“H” or “F” for heater/fan on) and the Kp value. For example: Pre-heat H:ON Kp:5.
  + Serial Output**:** Send\_Serial\_Data() formats the setpoint, current temperature, and Kp into a string (Set: 150.0, Temp: 148.5, Kp: 5) and sends it via UART character-by-character. This provides real-time logging on a PC terminal.
  + Loop Delay: A short 100 ms delay is inserted to pace the loop.

The code modularizes these functions for clarity (OneWire routines, LCD routines, UART routines, etc.). The Send\_Serial\_Data() and LCD update ensure the user can monitor the process.

# Control Strategy (Proportional Controller)

The controller uses a proportional (P-only) strategy. The **control error** is defined as e=Tset−Tmeasurede .The **control output** is a PWM duty cycle u=Kp⋅∣e∣ limited to the maximum value (255). The proportional gain Kp is user-adjustable via the ADC input. No integral or derivative terms are used, so the system is expected to have some steady-state error but is simpler and faster to implement. The sign of the error determines the actuator: if e>0e > 0, the heater is driven (providing positive heating power); if e<0e < 0, the fan is driven for cooling. This bidirectional control approach (heating vs. cooling) ensures the oven follows the thermal profile without overshooting excessively. By tuning Kp, the operator can achieve a desirable ramp rate and steady-state accuracy. Typical values (via the 0–5 V pot) map to Kp≈1 to 32, as computed by 1 + (ADRESH >> 3).

## Features of the P Controller:

* + Responds linearly to error; larger error → higher duty.
  + Single feedback sensor (no feedforward or secondary sensors).
  + Adjusting Kp changes system aggressiveness.
  + The code ensures heater and fan are never on simultaneously (mutual exclusivity).

The simplicity of the P controller is suitable for a student project. In practice, proportional control can achieve the general shape of a reflow profile, but may require some overshoot or manual tuning. The adjustable Kp input allows experimenting with different gains during testing.

# Simulation Setup

To verify the design, the system was modelled and tested in a microcontroller simulation environment. A PIC18F458 with the full firmware was simulated using Proteus (or similar) with models for the DS18B20, LCD, and PWM-driven heater/fan. The thermal oven was represented by a simple first-order thermal model, where the heater power raises the temperature and the fan cools it. The temperature setpoint profile (as given by the 48-step array) was applied, advancing every 5 s via the Timer0 interrupt.

* The UART output was routed to a virtual terminal on the PC.
* The LCD display content was monitored to ensure correct formatting.
* The ADC input (potentiometer) was simulated to step through various Kp values.

Graphs of the simulation (placeholders above) would show how the measured temperature follows the target profile and how the PWM outputs switch between heating and cooling. The effect of different Kp values on rise time and steady-state error can be observed. The placeholder figures illustrate expected results: for instance, with an appropriate Kp the oven temperature curve should closely track the setpoint without large overshoot or undershoot.

A computer diagram of a computer

AI-generated content may be incorrect.

# **Results**

* Temperature Tracking: A plot of oven temperature vs. time overlaid with the ideal setpoint profile. This would demonstrate how well the control algorithm maintains the desired profile. (See Figure 6.)
* Control Output: A plot of PWM duty cycle (or heater power and fan power) over time, showing which actuator is active and by how much. (See Figure 7.)
* Kp Tuning**:** Data or table showing how different potentiometer settings (ADC values) affect the gain Kp and system response.

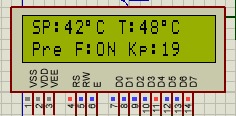
For example, one would expect that with a low Kp, the oven may lag behind the profile (steady-state error), while with a high Kp the temperature might overshoot. The reported serial output (Set: T, Temp: T, Kp: value) can be logged to verify temperature readings against actual values.

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer

AI-generated content may be incorrect.A computer screen with a black screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**A graph showing the temperature of a hot and cold

AI-generated content may be incorrect.**

# Conclusion

This report has documented the design of a PIC18F458-based solder reflow oven controller using a DS18B20 temperature sensor and a proportional control algorithm. The system architecture integrates key components (sensor, MCU, display, actuator drivers, serial interface) to follow a multi-stage thermal profile. The firmware logic—including one-wire communication routines, interrupt-driven timing, ADC-based gain tuning, and PWM control—was derived directly from the provided code. By adjusting Kp via an analog input, the controller’s responsiveness can be tuned to match the oven’s thermal characteristics. The expected (simulated) results indicate that the oven temperature will follow the setpoint profile with the heater and fan PWM outputs alternating as needed. In a real implementation, this approach should yield a functional reflow controller suitable for microcontroller coursework. Future enhancements could include implementing an integral term for improved accuracy or adding multiple sensors for distributed temperature measurement.