

**Operator's Manual
for
Model TP04010A ThermoStream® Systems**

Revision C

TEMPTRONIC CORPORATION
55 Chapel Street
Newton, MA 02158-1010

SAFETY PRECAUTIONS FOR OPERATING PERSONNEL

This manual contains procedures intended for use by operating personnel. Refer to the *TP04010A Service Manual* (LM00910) for corrective maintenance procedures intended for service personnel.

WARNING: Operating personnel should perform only the procedures described and recommended in this manual. Only qualified service personnel familiar with the electrical shock hazards present inside the equipment should perform any disassembly or corrective maintenance of the equipment.

WARNING: The locations of potentially dangerous voltages and other hazards are labeled and described on the equipment. Carefully observe these warnings when installing, operating, maintaining, or servicing the equipment. Observe all warnings given in this manual.

WARNING: To avoid shock hazard, the equipment must be grounded with an adequate earth ground per local electrical codes.

WARNING: During the up/down motion of the TP04010A's head, keep your fingers out of the space between the thermal cap on the head and the DUT site.

CAUTION: Observe the precautions given on the equipment and within this manual to prevent damage to the equipment.

CAUTION: Use proper handling and packaging procedures for static-sensitive circuit boards. Assume that all circuit boards are the static-sensitive type.

CAUTION: Unauthorized personnel should not remove from the equipment those panels that are provided for cooling and protection.

©Copyright 1994 - 1996 by Temptronic Corporation

All Rights Reserved

The text of this publication, or any part thereof, may not be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, storage in an information retrieval system, or otherwise, without the prior written permission of Temptronic Corporation.

The software program contained within the Model TP04010A ThermoStream® System for operation of this system is protected by copyright laws that pertain to computer software. Use of this software does not authorize the de-compiling, disassembling or reverse engineering to gain access to the program code. Temptronic Corporation does not authorize any copy, change, or other use of this software.

Notice

Patents have been granted and/or patent applications are pending or are in process of preparation on all our developments.

The material in this manual is for informational purposes and is subject to change without notice.

Temptronic Corporation assumes no responsibility for any errors which may appear in this manual.

Temptronic Corporation warrants this product as covered in the Warranty statement included at the rear of this manual.

Printed In U.S.A.

The following are trademarks of Temptronic Corporation, Newton, Massachusetts:

| | | |
|---------------|---------------|---------------|
| ThermoChuck® | ThermoSpot® | ThermoJogger™ |
| ThermoDome® | ThermoStream® | ThermoLab™ |
| ThermoSocket® | ThermoZone® | MiniZone™ |
| MiniDome™ | | |

Manual Number: LM00890



Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

Contents

Section 1 - General Data

| | | |
|-----|-------------------------------------|-----|
| 1-1 | Introduction | 1-1 |
| 1-2 | Specifications | 1-2 |
| 1-3 | Power and Air Requirements | 1-2 |
| 1-4 | User/Owner Registration Cards | 1-3 |
| 1-5 | Performance Report Forms | 1-3 |

Section 2 - Preparation for Use

| | | |
|-----|---|------|
| 2-1 | Introduction | 2-1 |
| 2-2 | Receipt of Shipment | 2-1 |
| 2-3 | Placement Requirements | 2-2 |
| | 2-3.1 AC Power Interconnections | 2-2 |
| | 2-3.2 Compressed Air Interconnection | 2-3 |
| 2-4 | Unpacking Instructions | 2-3 |
| 2-5 | Assembly Instructions | 2-5 |
| | 2-5.1 Thermal Cap Attachment | 2-5 |
| | 2-5.2 Air Dryer Mounting | 2-6 |
| | 2-5.3 Buck/Boost Transformer Mounting | 2-6 |
| 2-6 | System Interconnections | 2-6 |
| | 2-6.1 Air Connection | 2-6 |
| | 2-6.2 Power Connection | 2-7 |
| 2-7 | System Interfacing | 2-7 |
| | 2-7.1 ESD Protection | 2-7 |
| | 2-7.2 ThermoStream Interfacing | 2-7 |
| | 2-7.3 Air Purging | 2-10 |
| | 2-7.4 Direct DUT Temperature Sensing | 2-10 |
| | 2-7.5 Remote Control Communications | 2-11 |
| | 2-7.6 Peripheral I/O Panel Ports | 2-11 |
| 2-8 | Initial Start-Up | 2-11 |
| 2-9 | Repackaging | 2-12 |

Section 3 - System Operation

| | | |
|-----|---|------|
| 3-1 | Introduction | 3-1 |
| 3-2 | Local Operation Controls | 3-1 |
| | 3-2.1 Electrical and Air Controls | 3-1 |
| | 3-2.2 Head and Manipulator | 3-3 |
| | 3-2.3 Operator Control Module | 3-8 |
| 3-3 | Startup/Shutdown | 3-10 |
| | 3-3.1 Startup from Power On | 3-10 |
| | 3-3.2 System Shutdown | 3-10 |
| 3-4 | Operator Mode | 3-11 |
| | 3-4.1 Startup Sequence | 3-11 |
| | 3-4.2 Test Procedure | 3-12 |
| | 3-4.3 Test Setup | 3-16 |
| 3-5 | Variable Setup Mode | 3-21 |
| | 3-5.1 Startup Sequence | 3-21 |
| | 3-5.2 Test Procedure | 3-22 |
| | 3-5.3 Test Setup | 3-26 |

Contents (continued)

| | | |
|---|---|----------------|
| 3-6 | Top Menu Selections..... | 3-30 |
| 3-6.1 | Top Menu Screen..... | 3-30 |
| 3-6.2 | Explanation of Screen Selections..... | 3-31 |
| 3-7 | System Configuration..... | 3-32 |
| 3-7.1 | System Configuration Screen..... | 3-32 |
| 3-7.2 | Explanation of Configuration Options..... | 3-33 |
| 3-8 | Error Messages..... | 3-36 |
| Section 4 - Remote Interfaces | | |
| 4-1 | Introduction..... | 4-1 |
| 4-2 | MCT Interface..... | 4-2 |
| 4-3 | IEEE-488/RS232C Host Interface..... | 4-3 |
| 4-3.1 | Parallel/Serial Programming..... | 4-3 |
| 4-3.2 | Device Specific Commands and Queries..... | 4-4 |
| 4-4 | IEEE-488 Interface..... | 4-9 |
| 4-4.1 | Parallel Bus Interface Parameters..... | 4-9 |
| 4-4.2 | Mandatory IEEE-488.2 Common Commands and Queries..... | 4-9 |
| 4-5 | RS232C Interface (optional)..... | 4-10 |
| 4-5.1 | Serial Interface Connector..... | 4-10 |
| 4-5.2 | Serial Interface Parameters..... | 4-11 |
| 4-5.3 | Serial Interface Special Commands..... | 4-11 |
| Section 5 - Operator Maintenance | | |
| 5-1 | Introduction..... | 5-1 |
| 5-2 | Inspection and Cleaning..... | 5-1 |
| 5-2.1 | Inspection..... | 5-1 |
| 5-2.2 | Cleaning..... | 5-1 |
| 5-3 | Maintenance Log..... | 5-2 |
| 5-4 | Back-Up Battery Replacement..... | 5-5 |
| 5-5 | Air Path Maintenance..... | 5-6 |
| 5-5.1 | Front Panel Removal..... | 5-6 |
| 5-5.2 | Air Filter Servicing..... | 5-6 |
| 5-5.3 | Exhaust Muffler Replacement..... | 5-8 |
| 5-6 | Calibration..... | 5-9 |
| 5-6.1 | Verification Procedure..... | 5-9 |
| 5-6.2 | Calibration Procedure..... | 5-12 |
| 5-7 | Air Chiller Module Defrosting..... | 5-15 |
| 5-8 | BIOS Setup..... | 5-17 |
| Appendix A - IEEE-488 Demo Program..... | | A-1 |
| Appendix B - RS232C Demo Program..... | | B-1 |
| Index | | Index-1 |
| Warranty, Registration Cards, Performance Report, Reader Comments Card | | |

Section 1

General Data

1-1 Introduction

Information in this manual supports the installation and operation of the Model TP04010A ThermoStream System (see Figure 1-1). This system is a programmable temperature controlling airstream system for testing and characterizing electronic devices at temperatures from -80°C to $+225^{\circ}\text{C}$. An internal microcomputer controls the system, permitting either local or remote operation.

The TP04010A display shows the air and DUT (device-under-test) temperatures; programmed values including hot, ambient, and cold temperature setpoints, soak time; and cycle information. In addition, the complete system operations status and any error messages are displayed. All data is shown in real-time alphanumeric format, keeping the user fully informed at all times.

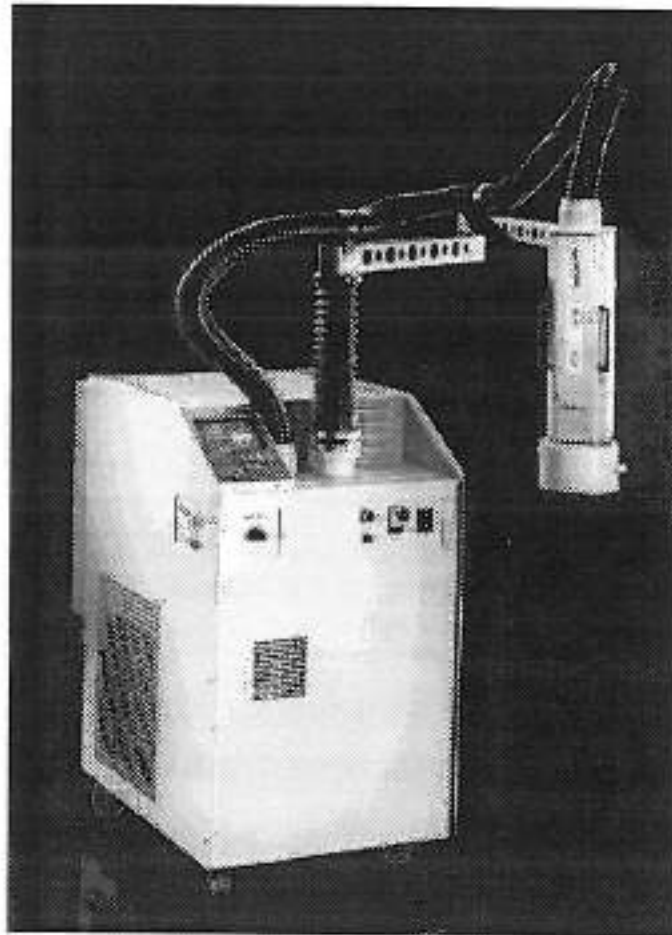


Figure 1-1. Model TP04010A ThermoStream System

1-2 Specifications

Table 1-1 lists the general specifications for the Model TP04010A ThermoStream Systems.

Table 1-1. TP04010A General Specifications

| | |
|--|---|
| Temperature Limit - high | +225 °C |
| Temperature Limit - low | (at air supply of +20 to +28 °C) |
| Standard head/no arm configuration | -85 °C |
| Standard head with arm configuration | -80 °C |
| Temperature transition (air) | 30 seconds from +125 to -55 °C |
| Air Flow | 12 SCFM maximum, manual control |
| Temperature Set and Display Resolution | 0.1 °C |
| Temperature Control | Air or direct DUT control |
| DUT sensors | Type T or Type K thermocouple inputs |
| Remote Interface Ports | |
| Standard | IEEE-488, or |
| Optional | RS232C |
| Handler Interface | Start-Test, End-of-Test, and Stop-on-First-Fail signals |
| User Interface | CRT monitor for data entry and display |

1-3. Power and Air Requirements

Table 1-2 lists ac power and compressed air requirements for installation of the Model TP04010A ThermoStream Systems.

Table 1-2. TP04010A Power and Air Requirements

| | |
|---------------------|--|
| AC Power | Single phase, 20 amperes |
| 60 Hz, or | 220 to 240V* (220V nominal)** |
| 50 Hz | 200V* (nominal)** |
| Compressed Air | |
| Supply pressure | 80 to 110 PSIG with air dryer (100 PSIG nominal)**, or 70 to 110 PSIG without air dryer (100 PSIG nominal)** |
| Supply flow rate*** | 9 to 21 SCFM with air dryer (21 to 25 SCFM recommended -- provides optimum system performance), 3 to 15 SCFM without air dryer |

* Optional buck/boost transformer is available for other supply voltages.

** Reduced performance may be encountered at operating environments less than or greater than nominal.

***Flow requirements reduced by 6 SCFM without air dryer.

1-4 User/Owner Registration Cards

Four prenumbered User/Owner Registration post cards are supplied as a single perforated sheet (P/N LMS1480) in the back portion of this manual.

Upon receipt of the system, card #1 should be completed and returned to Temptronic. When ownership changes or additional personnel become responsible for the system, the remaining cards should be used. By following this procedure, you will be assured of receiving any important information pertinent to your system. Additional cards may be obtained through the Temptronic Service Department (refer to Par. 5-1).

1-5 Performance Report Forms

A "User Interface Performance Report" is provided at the back of this manual. Use this report to submit any desired enhancements or functional discrepancies in the system. However for prompt response, you may contact the Temptronic Service Department directly whenever a problem occurs.

Also, a "Reader Comments" card is included at the back of this manual. Use this card to feed back your suggestions and opinions regarding the effectiveness of the manual.



Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

Section 2

Preparation for Use

2-1 Introduction

This section contains the instructions for unpacking, placement, assembly, and interfacing the Model TP04010A ThermoStream Systems.

2-2 Receipt of Shipment

Your Model TP04010A ThermoStream System is shipped to you in one packing carton on a skid (see Figure 2-1). The packaged system can be easily transported at your location with a single fork-lift. Overall shipping size of the packaged system is 66 inches high by 43-1/2 inches by 62 inches. Overall shipping weight is 745 pounds.

All purchased items are included inside the packing carton. When received, the carton should be examined for any signs of mishandling or damage during shipment.

NOTE: If you see any obvious signs of damage to the packing carton, contact the carrier immediately and do not proceed with the unpacking. Since the shipment is made FOB factory, you should consult your administration concerning claims for shipping damage.

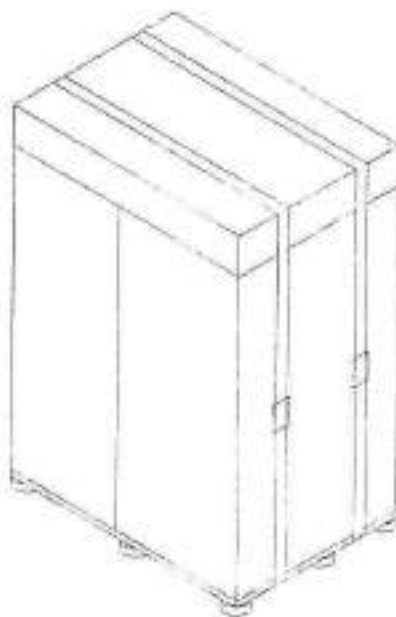


Figure 2-1. TP04010A in Packing Carton for Shipment

2-3 Placement Requirements

The proposed site for placement of the TP04010A should be one near the necessary ac power and compressed air service interconnections. Also, the proposed site should be near any tester with which the TP04010A will be interfaced. For proper cooling of the TP04010A and for affording a minimum workspace, the system should be located with minimum clearances around it as shown in Figure 2-2.

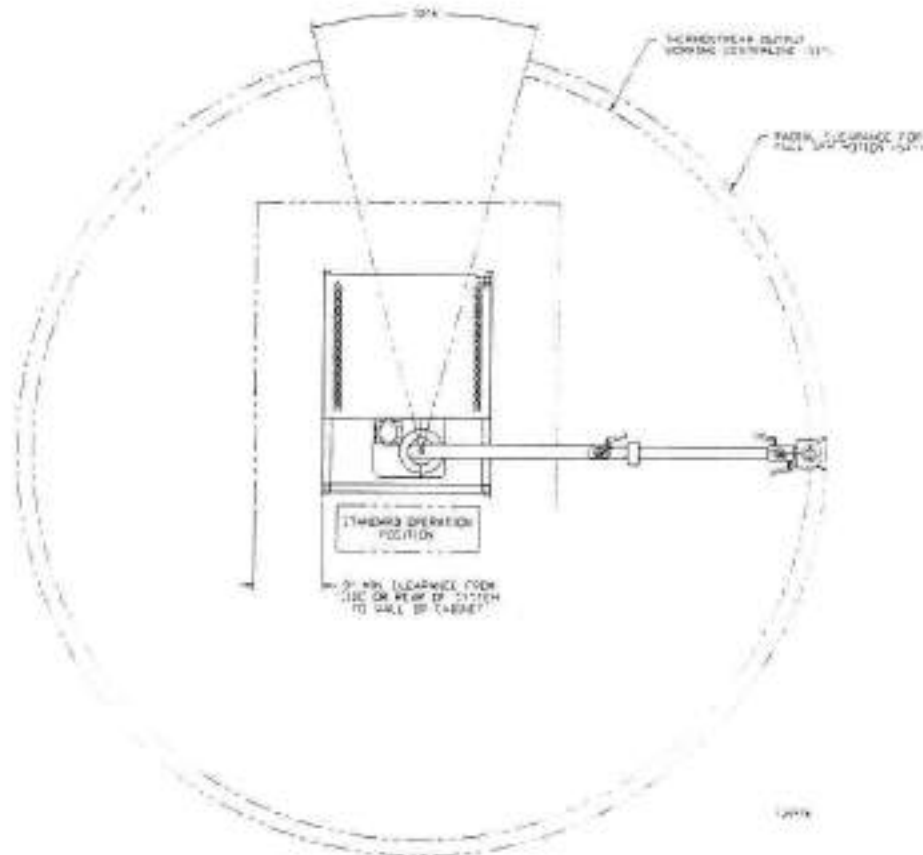


Figure 2- 2. Placement Clearances for TP04010A

2-3.1 AC Power Interconnections

The TP04010A includes an attached 8-foot long power cord with a 3-wire twist-lock grounding male connector, Bryant type 70620NP. This connector should mate with a suitable wall receptacle and wiring that comply with the applicable governmental and local codes.

As specified at the time of purchase, the TP04010A must be connected to a source of single-phase ac power at 220 to 240 volts, 60 Hz or at 200 volts, 50 Hz. An optional buck/boost transformer is available for other supply voltages. The power source must be capable of supplying 20 amperes, minimum.

2-3.2 Compressed Air Interconnection

The TP04010A is supplied with a 10-foot long, 3/8-inch ID braid reinforced, oil-resistant rubber hose and its associated 3/8-inch NPT barb fitting and two hose clamps.

The compressed air supply pressure must be from 80 to 110 PSIG when an air dryer is used or from 70 to 110 PSIG without an air dryer. The supply flow must be capable of 21 SCFM (a supply below 9 SCFM results in a reduced performance).

NOTE: The flow requirements may be reduced by 6 SCFM when an air dryer is not used. (An optional dryer orifice kit reduces the dryer purge air to 4 SCFM instead of 6 SCFM.)

2-4 Unpacking Instructions

When unpacking the system, save all packaging material in the event the TP04010A system has to be reshipped later.

1. Transport the packaged TP04010A to its proposed site when possible.
CAUTION: Some of the packaging materials of the TP04010A shipment may be a source for ESD potential. Do not unpack in the vicinity of ESD sensitive components.
2. Cut and remove the two band seals holding the packing carton on its skid. Lift off the carton cover and the carton shell. (See Figure 2-3A).
3. Cut and remove the band seal holding the ramp on the pallet and against the TP04010A. Place the ramp in position at the edge of the pallet for removal of the system (see Figure 2-3B).
4. Cut and remove the remaining band seals that secure the TP04010A on the pallet. The Temptronic air dryer will be installed in place on the back side of the TP04010A.
5. Roll the TP04010A off the pallet, down the ramp, into position on the floor where it will be used (see Figure 2-3C). If the TP04010A is equipped with optional leveling feet, screw the posts down to level and stabilize the machine.
6. For a TP04010A System with a head and manipulator, swing the manipulator so the head is in its operating position (loosen cam locks to position, refer to Figures 3-3 and 3-4).
7. Visually inspect the TP04010A after unpacking and do not proceed with its preparation for use if any signs of damage are found.
 - a. Review the packing slip to verify that all purchased items have been received.
 - b. Verify that all panel switches are in place and turned off.
 - c. Verify the integrity of all exposed cables and connections.
 - d. Verify that all visible hardware is secure.

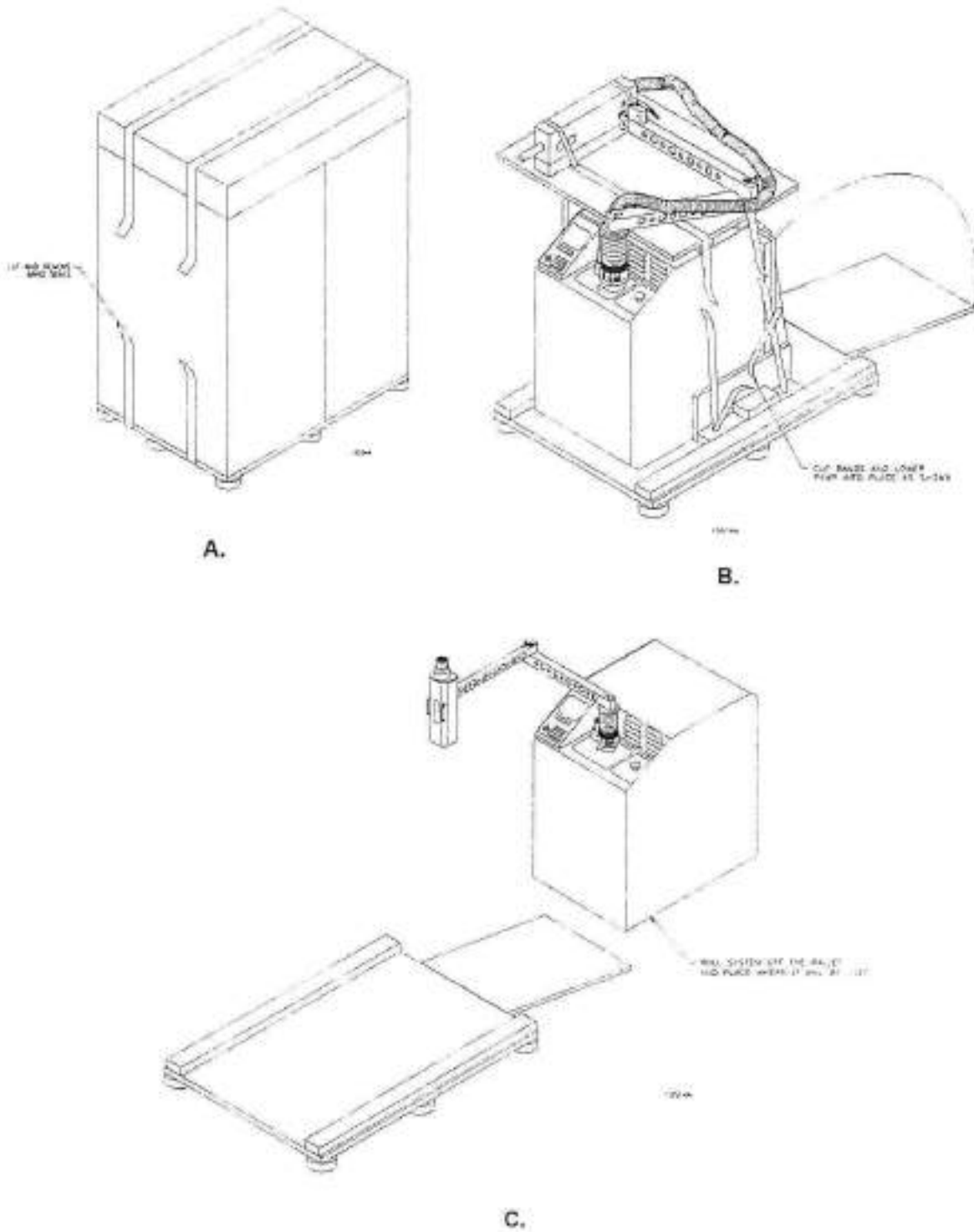


Figure 2-3. Unpacking of TP04010A System

2-5 Assembly Instructions

Each system is configured to customer specifications at the time of order and shipped complete. Typically, the assembly required is to attach the thermal cap to the head module. In some cases, it may be necessary to mount the optional buck/boost transformer.

2-5.1 Thermal Cap Attachment

Attach the standard, insulated (metal housing) thermal cap onto the bottom of the head module. No tools are required. Note that the same procedure applies for the optional, transparent-type thermal cap.

1. At initial installation of the TP04010A, install the three thumbscrews in the mounting ring at the head output (see Figure 2-4). (Thumbscrews are packed with the thermal cap in a separate box.)
2. Apply upward pressure, twist back and forth, and slip (push) the thermal cap into the mounting ring of the head module.
3. Rotate the thermal cap to locate its exhaust port away from the operator's position, and tighten the three thumbscrews to secure the thermal cap in place.
4. Slip the non-conductive thermal shroud over the end of the head outlet nozzle to concentrate the air flow around the DUT. Two different shaped shrouds are provided, a square one and a rectangular one. Use the appropriate shaped one for the type of DUT being tested (additional sizes are available from Temptronic).
5. Attach the purge air hose between the thermal cap purge input fitting and the fitting at the lower rear of the head module (refer to Figure 3-4).

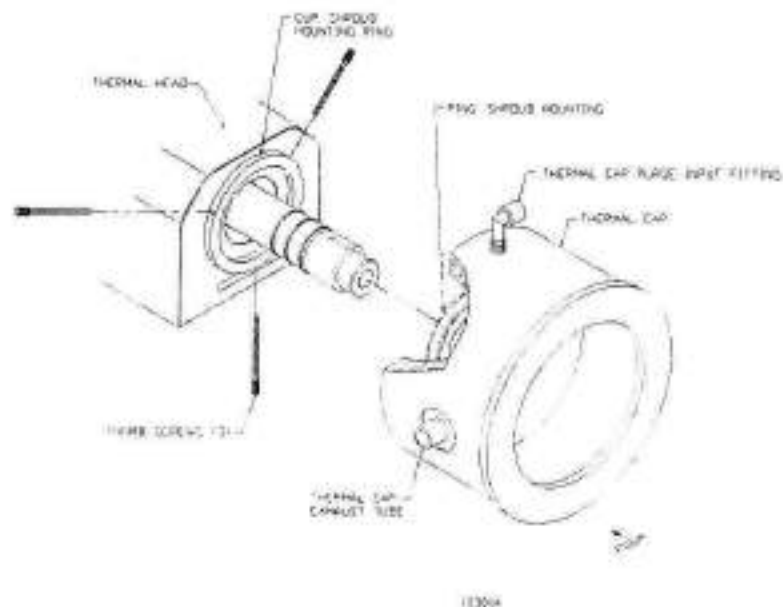


Figure 2-4. Thermal Cap Attachment

2-5.2 Air Dryer Mounting

The air dryer is shipped mounted on the rear of the TP04010A with all necessary connections made at the power and air input panel (refer to Figure 2-5).

If a system is configured without the air dryer, a hose shunt is factory installed between the two dryer ports.

2-5.3 Buck/Boost Transformer Mounting

The optional buck/boost transformer is supplied as a separate assembly for installation by the customer. It is configured to hang on the rear of TP04010A and connects between the ac power service and the ac power input to the system. Install and connect the buck/boost windings per the instructions provided with the transformer.

2-6 System Interconnections

Figure 2-5 shows the power and air input panel (rear of frame module) for making the system interconnections to energize the TP04010A.

2-6.1 Air Connection

Thread the 3/8-inch NPT barb fitting supplied with the air hose into the AIR INPUT fitting on the power and input panel. **CAUTION:** Hold the AIR INPUT fitting with a second wrench while tightening the barb fitting to prevent the AIR INPUT fitting from rotating in the panel.

Push the air hose over the barb fitting and secure with a hose clamp (supplied). Connect the other end of the air hose to your air supply and secure with a hose clamp (supplied).

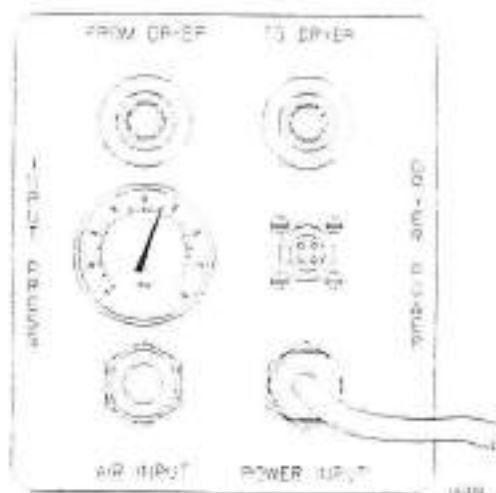


Figure 2-5. Power and Air Input Panel

2-6.2 Power Connection

CAUTION: Check that the ac voltage and frequency to be supplied to the TP04010A are correct for those listed on its data plate (located on rear system panel, upper left corner).

Plug the service cord from the power and air input panel into the supply receptacle with the correct ac power (if optional buck/boost transformer is needed, plug service cord into the transformer receptacle and plug the transformer power cord into the ac power source).

2-7 System Interfacing

2-7.1 ESD Protection

The air flow through the TP04010A is fully grounded from the air inlet to its air nozzle output. When coupled with the optional conductive shroud kit for interfacing to the DUT and the standard metal thermal cap, an ESD-free environment is assured without any added air ionizer.

2-7.2 ThermoStream Interfacing

The DUT site should be carefully prepared before the testing of devices at both elevated and reduced temperatures. You should minimize the heat transfer by thermal conductivity from the DUT site within the thermal cap to the external test equipment. In addition, components on the tester board in the vicinity of the DUT should be protected from the temperature extremes of the ThermoStream air flow. These paths of undesired heat transfer can be minimized by the insertion of insulation material.

Each TP04010A is supplied with an insulation kit (P/N ZAK101890). This kit contains one sheet of silicone foam material (12 inches square by 1/4 inch thick). The silicone material combines the property of low thermal conductivity with the added feature of elasticity. Cut to fit snugly around a given component configuration, thereby forming a seal to minimize heat transfer.

Should more insulation material be required, an optional insulation kit (P/N SA22450) contains three sheets (1/8-, 1/4-, and 1/2-inch thick) of 18-inch square silicone foam material. Another optional insulation kit (P/N ZAK40480) contains three sheets (1/8-, 1/4-, and 1/2-inch thick) of 12-inch square silicone foam material. Both of these optional kits come with RTV adhesive. For separate pieces of available foam material and adhesive, contact the Temptronic Service Department (refer to Par. 5-1).

See Figures 2-6 and 2-7 for a typical test site installation of the insulation material. For convenience, pieces and sheets of the silicone foam may be cemented together with an RTV adhesive.

AT TEST TEMPERATURES BELOW ROOM AMBIENT

1. DUT AND TEST SOCKET MUST BE MOISTURE PROTECTED BY USING THE THERMAL CAP TO SILICONE RUBBER SEAL.
2. BEST MOISTURE PROTECTION OF TESTER ELECTRONICS CAN BE ACHIEVED BY PROVIDING AN AIR GAP AROUND TEST SOCKET LEADS AND PURGING THE AREA WITH DRY AIR.
3. BY UTILIZING A THERMAL SHROUD TO CONFINED THE TEMPERATURE CONTROLLED AIR WITHIN THE IMMEDIATE AREA AROUND THE DUT AND SOCKET, BETTER HEAT TRANSFER CAN BE ACHIEVED FROM AIRSTREAM TO DUT.

NOTE: TEMPERATURE TRANSITION TIMES ARE DEPENDENT ON MASS SPECIFIC HEAT CHARACTERISTICS AND THE THERMAL COUPLING AND THERMAL CONDUCTION PATHS TO THE DUT, TEST SOCKET, TEST LEADS AND OTHER FIXTURING FEATURES.

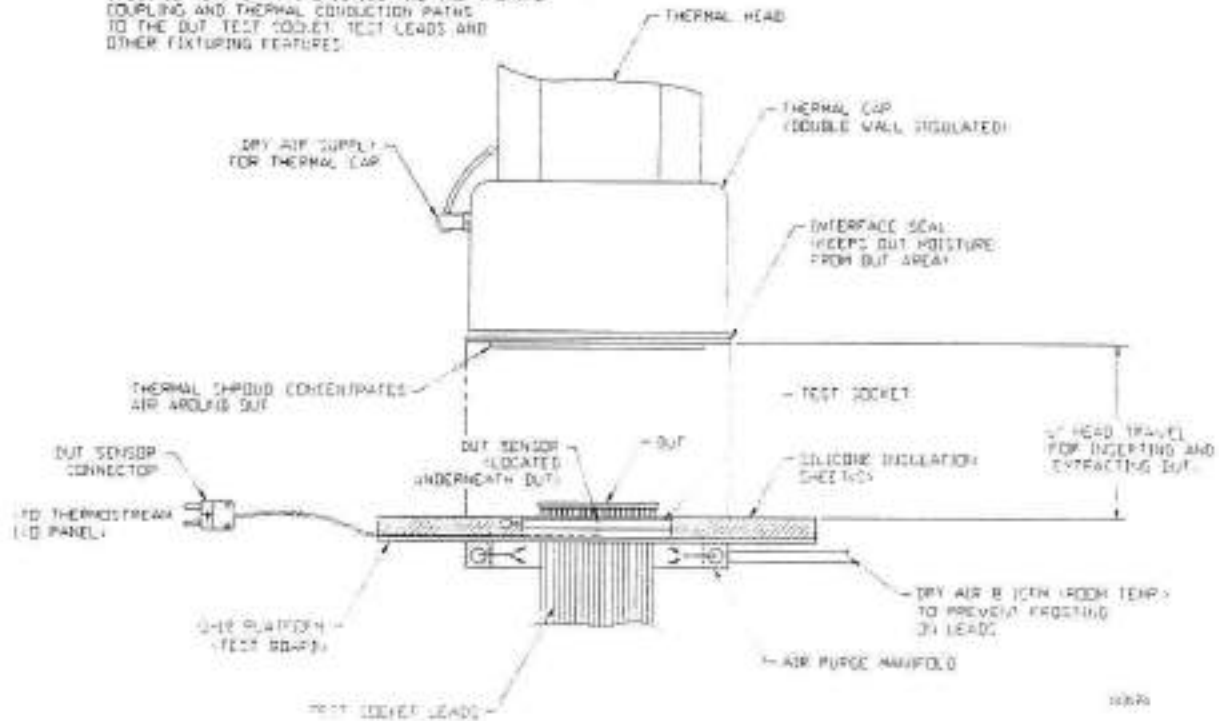
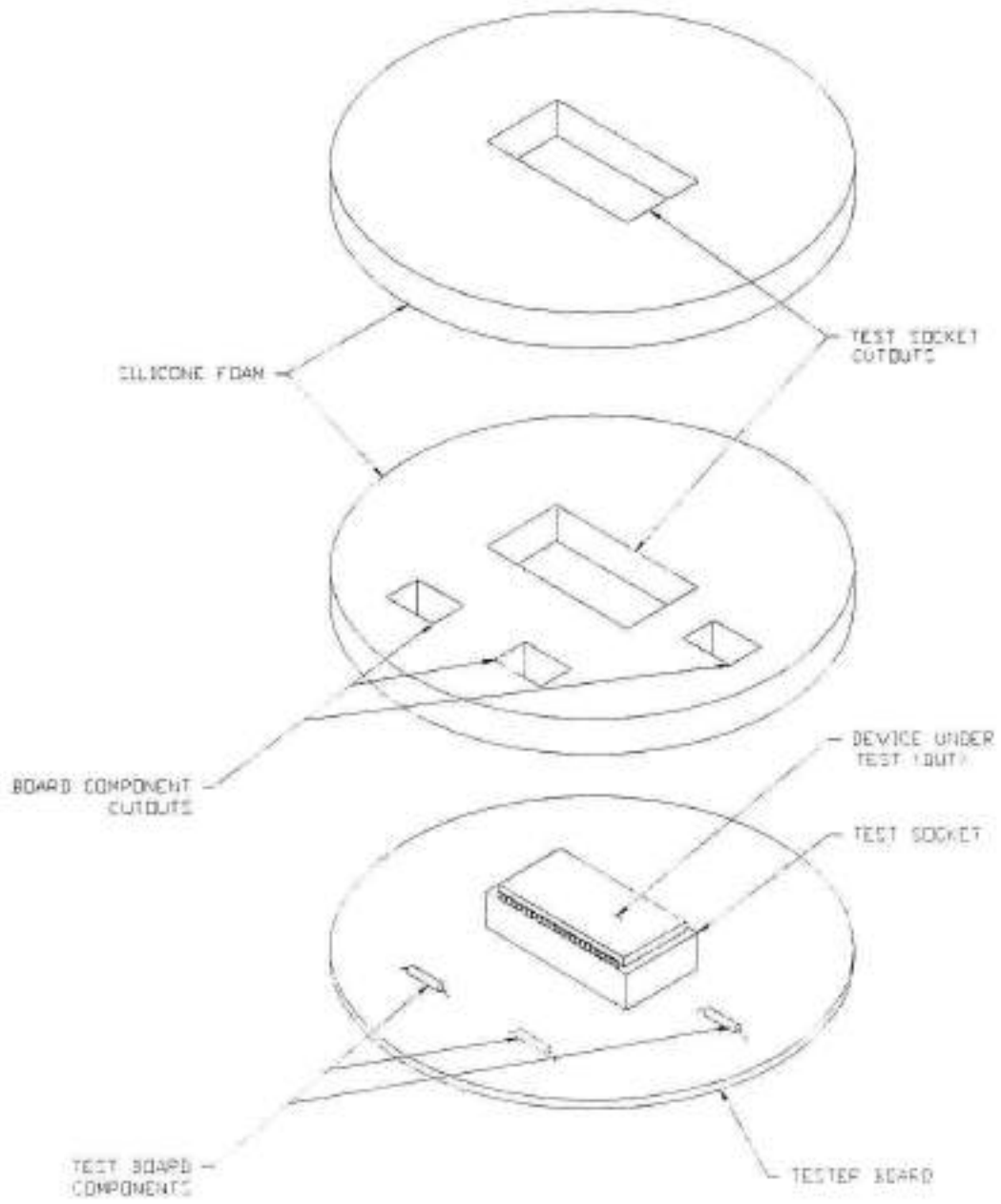


Figure 2-6. Typical Test Site



4221A

Figure 2-7. Typical Insulation for DUT

2-7.3 Air Purging

Low temperature testing can cause condensation and/or icing at the DUT site. Because of the low dew point (less than $-80\text{ }^{\circ}\text{C}$) of the ThermoStream outlet air, condensation within the thermal cap is not a problem. However, thermal conduction and forced convection may cause condensation to appear on the tester cables and/or interconnections, which are normally located in a room environment having dew points on the order of $15\text{ }^{\circ}\text{C}$. The TP04010A provides a test-site with a dry room-temperature air purge at the thermal cap. This air may be directed where condensation is not to accumulate.

NOTE: An optional DUT site purge kit (P/N ZAK105310) simplifies tying into the purge air by providing the user with all necessary fittings and tubing to direct air to the DUT site.

2-7.4 Direct DUT Temperature Sensing

For the direct DUT (dual loop) type of temperature control, install the T-type disk style DUT sensor supplied with the system.

1. Install the disk sensor on the DUT test socket to contact the DUT when inserted in the socket. See the detailed installation instructions provided with the sensor.
2. Plug the other end of the thermocouple wiring into the **SENSOR 2** connector on the system I/O panel (see Figure 2-8).

NOTE: If the optional T-type spring style DUT sensor is used instead of the T-type disk style, install per the supplied instructions and connect to the **SENSOR 2** connector on the I/O panel. The **Sensor 3** connector is supplied for use with K-type thermocouples.

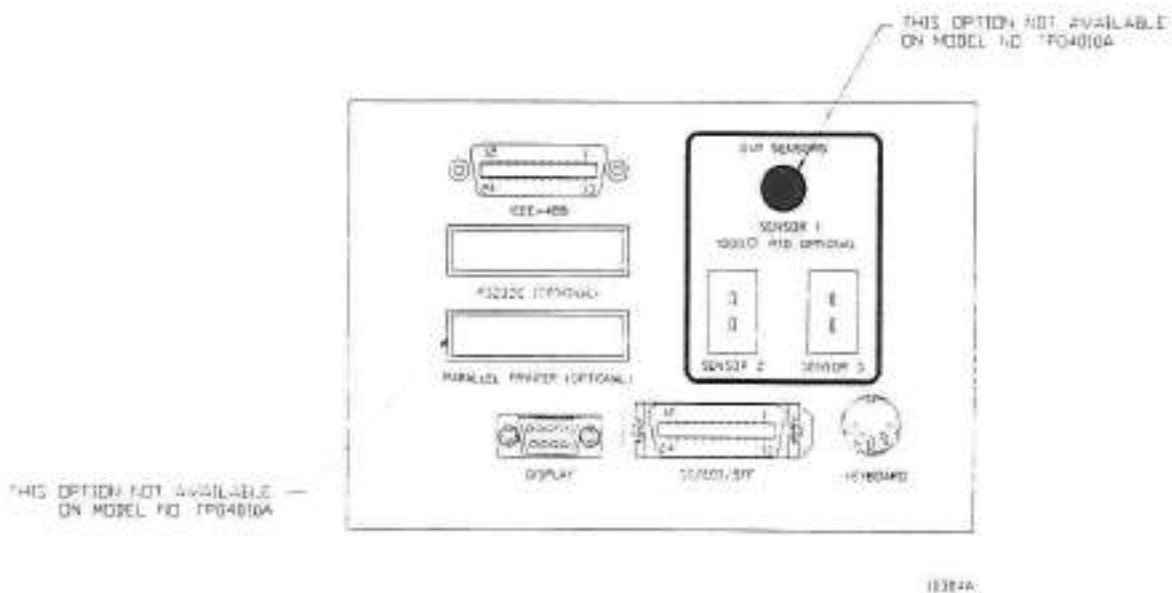


Figure 2-8. System I/O Panel

2-7.5 Remote Control Communications

IEEE-488 Interface

For remote control of the TP04010A, an IEEE-488 interface is standard for parallel communications with a tester or host computer. The parallel communications cable plugs into the IEEE-488 connector on the TP04010A I/O panel (refer to Figure 2-7). See Section 4, Remote Interfaces, for IEEE-488 interconnections information and special programming instructions.

RS232C Interface

An optional RS232C serial communications interface may be installed in the TP04010A at the time of purchase in addition to the standard IEEE-488 interface. In this case, the series communications cable plugs into the RS232C connector on the TP04010A I/O panel (refer to Figure 2-7). See Section 4, Remote Interfaces, for RS232C interconnections information and special programming instructions.

MCT Interface

A limited MCT Standard interface between the TP04010A and a tester is a standard feature. This interface cable plugs into the MCT STD INTERFACE connector on the TP04010A I/O panel. Interface connections include the Start-Test output from the TP04010A, the End-of-Test input from the tester, and the Stop-on-First-Fail input from the tester. See Section 4, Remote Interfaces, for MCT interconnections information and special programming instructions.

2-7.6 Peripheral I/O Panel Ports

The following I/O ports are available for connection to peripheral items:

KEYBOARD: A type DB-25 male connector for input of a programmer's keyboard (keyboard must be a PC AT compatible).

DISPLAY: A type DB-9 female connector for output to a remote monitor capable of displaying EGA (640 x 350) compatible graphics mode.

2-8 Initial Start-Up

The following procedure outlines how to start up the TP04010A initially. Once started and you are satisfied that the system works properly, it can be turned over for normal use as outlined in Section 3, System Operation.

1. Place the front panel **CONTROL BREAKER** at its **ON** position (refer to Figure 3-1).
2. Place the front panel **MAIN BREAKER** at its **ON** position.
3. Toggle the front panel **OFF/ON** switch to its **ON** position and then release.
4. Observe that the *System Startup Screen* (refer to Figure 3-7) appears on the CRT display of the operator control module (OCM).

5. At the *System Startup Screen*, push the *OCM Configuration* button to select the *Configuration* screen.
6. At the *Configuration* screen, select the parameters to give the desired system operation (refer to Par. 3-6.1). After making the proper selections, press the [ESC] key and observe that the *System Startup Screen* is displayed again.
7. Check the overall operation of the head module with its **HEAD** and **STAND** membrane switches (refer to Par. 3-2.2).
8. Listen to check that the air dryer is operating properly. You should hear a sharp retort from its muffler exhaust every 60 seconds.
9. Check the flow of air from nozzle output of the head module. With the front panel **FLOW** control turned fully clockwise, the air flow should be at its minimum. With the **FLOW** control turned fully counterclockwise, the air flow should be at its maximum.
10. When satisfied that the system is ready for use by the operator, toggle the front panel **OFF/ON** switch to its **OFF** position and then release for normal operation (refer to Section 3).

2-9 Repackaging

If the TP04010A System is to be shipped to another location, repackage the system in the original shipping carton (refer to Figure 2-3).

Note that prior approval is required before shipping the system to a Temptronic Sales/Service Office, or to the factory (refer to Par. 5-1). It is recommended that a tag be attached to the system giving the owner's name, address, telephone number, system model and serial numbers, and the reason for return.

Section 3

System Operation

3-1 Introduction

This section contains instructions for front panel (local) operation of the Model TP04010A Thermo-Stream Systems. Remote system operation is discussed in Section 4, Remote Interfaces. Familiarity with front panel operation is recommended before attempting remote operation.

3-2 Local Operation Controls

3-2.1 Electrical and Air Controls

Electrical Controls

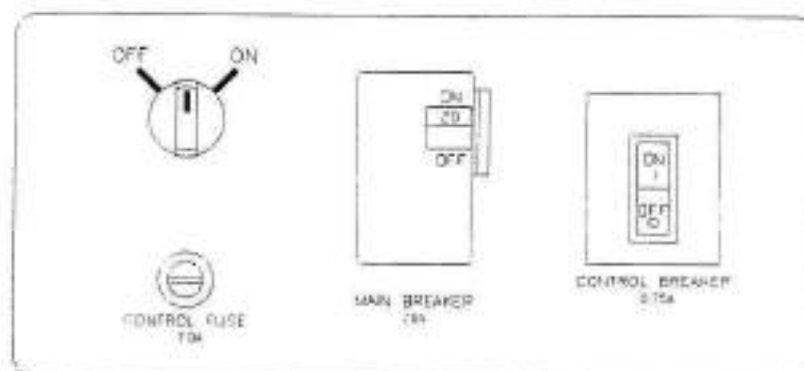
Figure 3-1 shows the power panel, which is located on the top right front of the frame module. This panel contains four components for controlling and protecting the ac power to the TP04010A.

OFF/ON momentary switch: Turn left to OFF to switch off ac power and the operation of the TP04010A. Turn right to ON to apply power and operate the TP04010A.

MAIN BREAKER: A 20-ampere circuit breaker in the primary ac power circuit to the TP04010A. This breaker is normally left in the ON (up) position.

CONTROL BREAKER: A 0.75-ampere circuit breaker in the secondary (control) relay circuit of the TP04010A. This breaker is normal left in the ON (up) position.

CONTROL FUSE: A 7.0-ampere fuse that protects the 24-volt ac power to the control relays in the TP04010A and the main ac power to the air dryer.



4033A

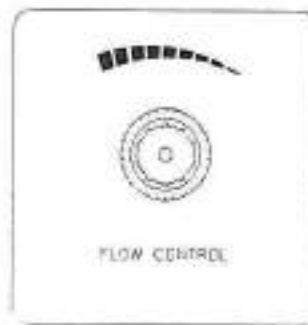
Figure 3-1. Power Panel

Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

Air Control

Figure 3-2 shows the air flow control panel, which is located near the top left front of the frame module. This panel contains the FLOW CONTROL valve for manual adjustment of the ThermoStream output at the head nozzle to the DUT. Turn this control counterclockwise to increase the flow or clockwise to decrease the flow. The amount of air flow is displayed on the operator screens, such as in Figure 3-9.



(32)

Figure 3-2. Air Flow Panel



3-2.2 Head and Manipulator

Figure 3-3 depicts the horizontal sweep limits of the thermal head manipulator (hinged arm configuration). This manipulator allows the thermal head to be positioned near the DUT site. Figure 3-4 shows the pivoting, turning, and tilting limits of the thermal head with respect to its thermal head manipulator. These three types of motion allow the thermal head to be aligned with the DUT site. Movements for setup of the manipulator and head are manually controlled; operational movements to speed up production testing are electrically controlled.

NOTE: These instructions are applicable to TP04010A Systems with an arm configuration. Skip to Paragraph 3.2.3 if your system has no arm configuration.

Manually Controlled Positioning

Manual travel and positioning of the head and manipulator can be set and held in the desired position by several mechanical locking devices. These locking devices are identified below (refer to Figures 3-3 and 3-4).

VERTICAL ARM LOCK (at top of vertical column exiting the frame module) can be tightened to clamp any horizontal sweep movement of the first arm of the manipulator. When this lock is loosened, the manipulator can be turned up to 165 degrees in either horizontal direction from the front of the system. When tightened, this lock disables the **STAND**  and  buttons (refer to Page 3-6).

HORIZONTAL ARM LOCK (at hinge between first and second arms) can be tightened to clamp any horizontal angular movement of the second arm from the first arm of the manipulator. When

this lock is loosened, the second arm can be turned up to 150 degrees in either horizontal direction from the centerline of the first arm.

END PIVOT LOCK (at rear of thermal head) can be tightened to clamp any horizontal pivoting movement of the thermal head with respect to the manipulator. When this lock is loosened, the thermal head can be pivoted up to 90 degrees in either horizontal direction from the centerline of its mounting (second) arm.

ANGLE PIVOT LOCK (at rear of thermal head) can be tightened to clamp any vertical pivoting movement of the thermal head with respect to the manipulator. When this lock is loosened, the thermal head can be pivoted on the manipulator up to 180 degrees in either direction from vertical (full pivot limited only by air hose at top of head). In addition, when the lock is loosened, the thermal head can be tilted forward or backward on the manipulator up to 7 degrees from vertical.

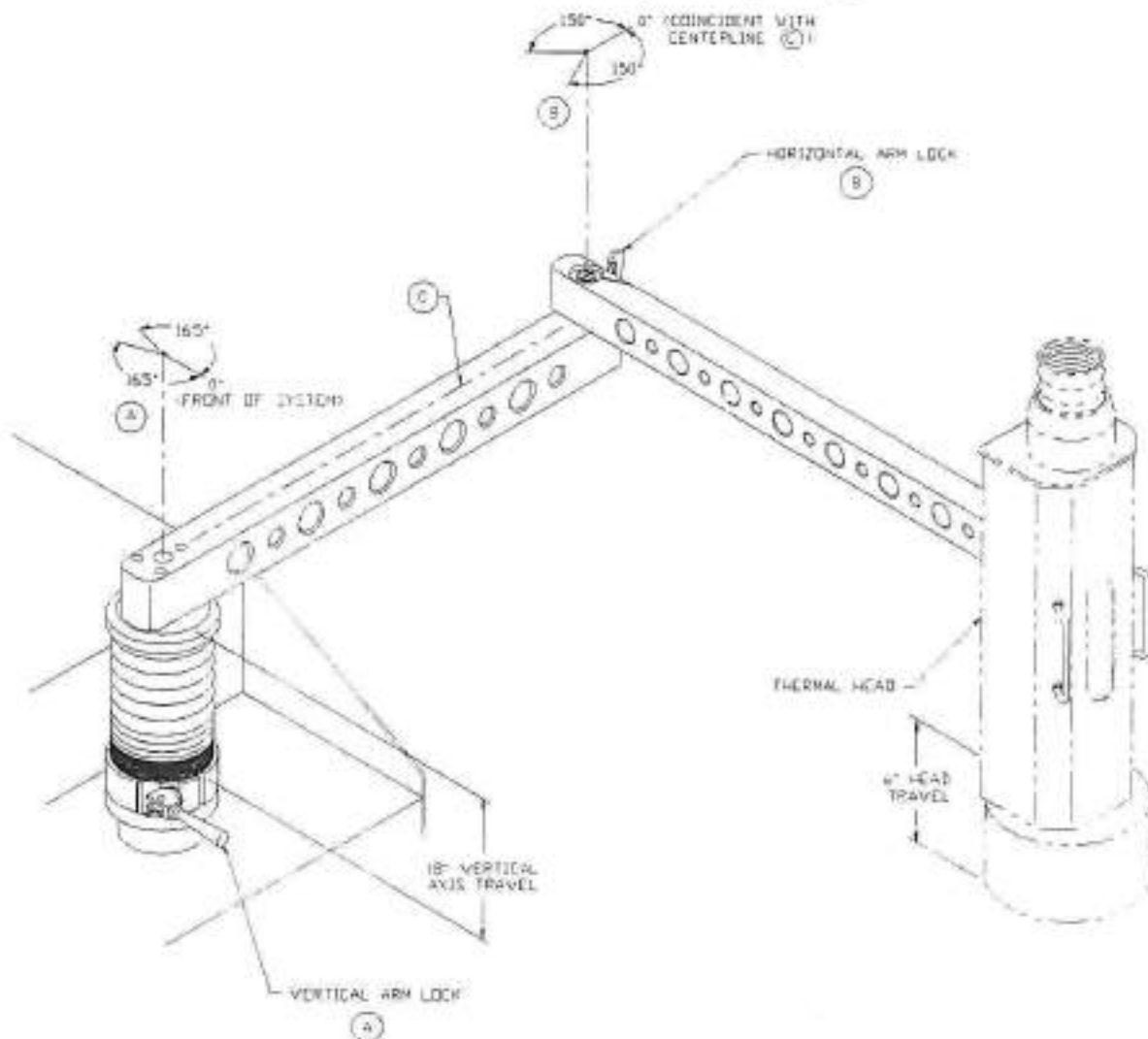


Figure 3-3. Thermal Head Manipulator Positioning

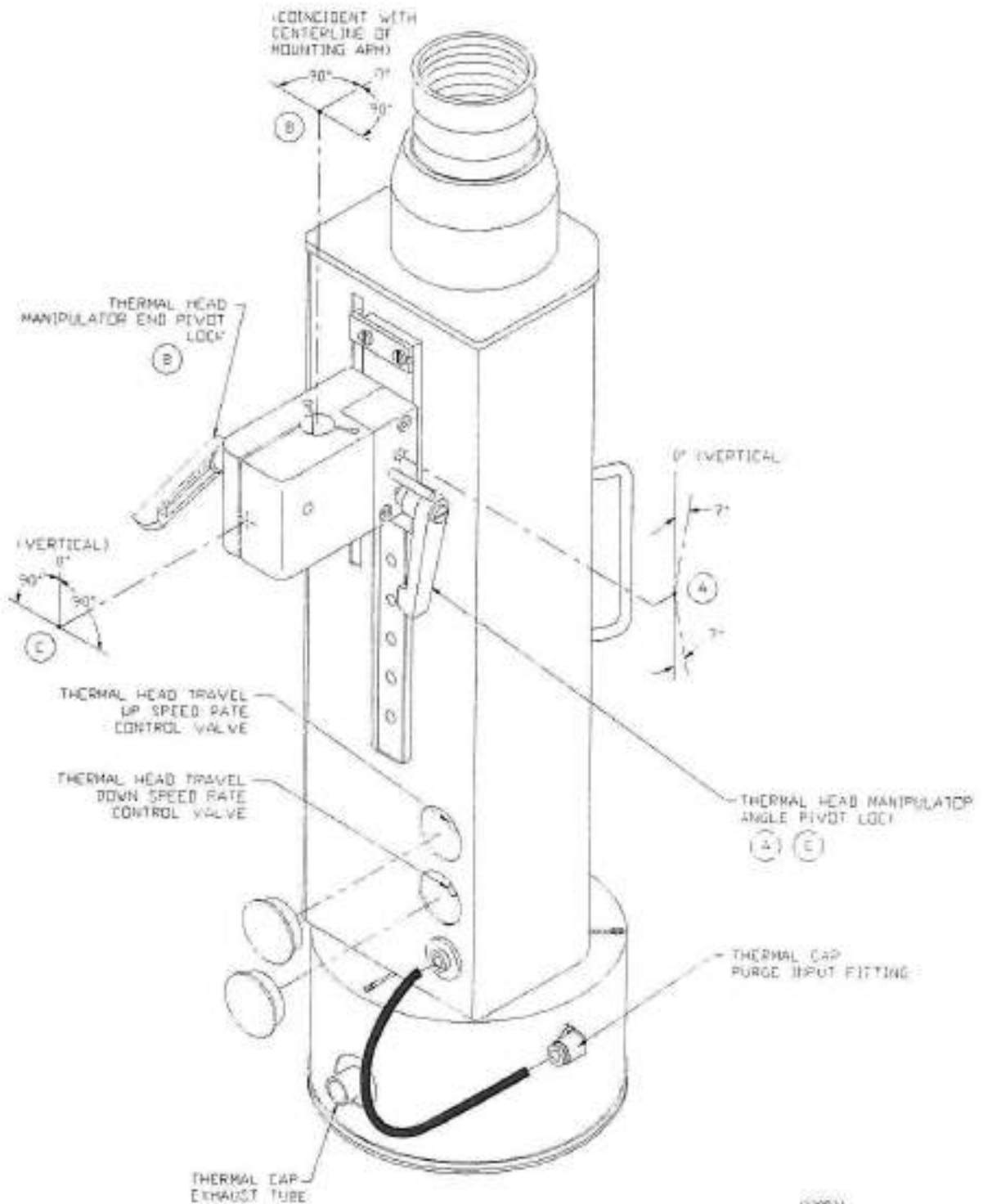










Figure 3-4. Thermal Head Positioning

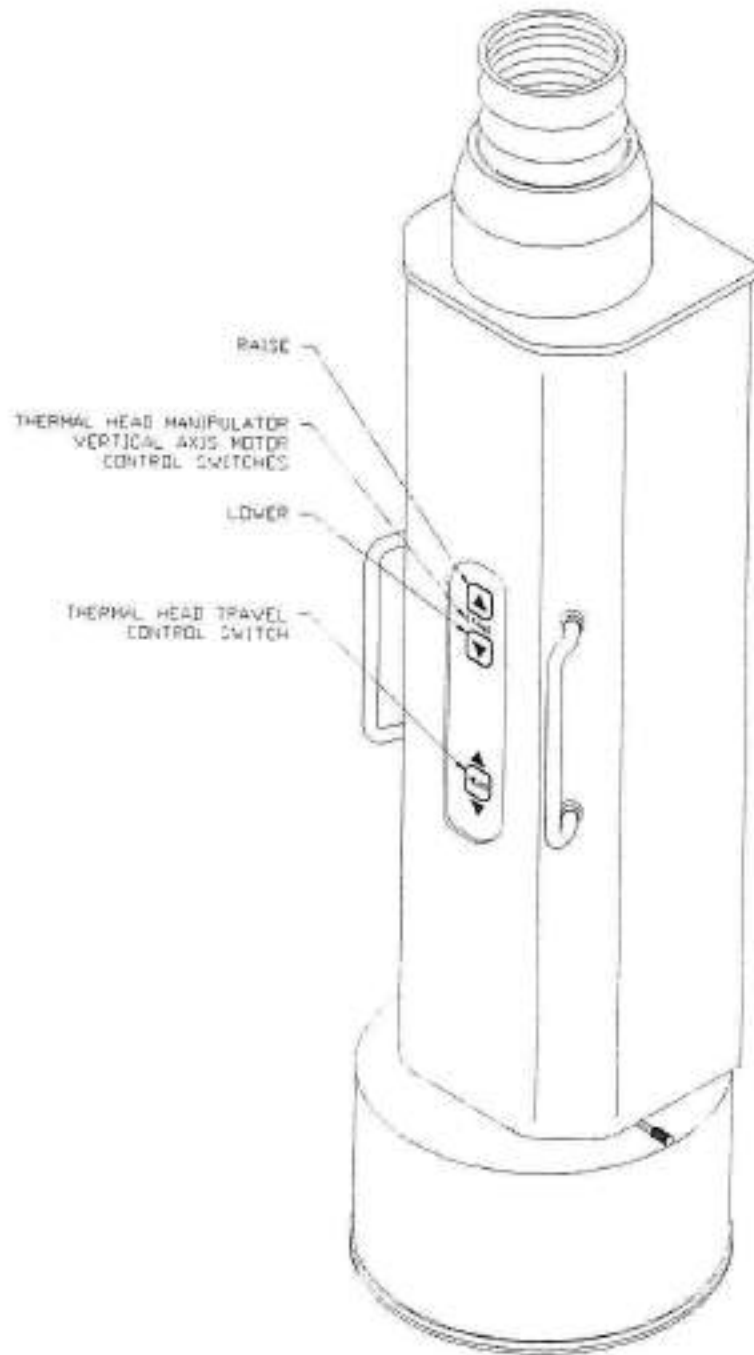
Electrically Controlled Positioning

Figure 3-5 shows the controls on the thermal head for vertical positioning of the thermal head and the manipulator (stand). These controls are two sets of membrane touch buttons, either  or  for up or down motion, respectively.

CAUTION. Always hold the handle grips on the head when positioning either the head or the stand.

HEAD  and  button raises or lowers the thermal head for a 6-inch travel with respect to the DUT site. Normally (refer to "Head motion" on Page 3-24), the head returns to its up position at the end of a test cycle. The speed of the head movement depends upon the attitude position of the head. Two adjustment controls (underneath snap buttons at rear of head, refer to Figure 3-4) can be set for the desired speed of head travel.

STAND  and  buttons raise and lower the manipulator (with thermal head) for a 16-inch vertical travel. Generally, the manipulator (with its thermal head lowered all the way) will be lowered until the head's thermal cap covers the DUT site. **NOTE:** the **STAND  and ** buttons are disabled when the **VERTICAL ARM LOCK** is tightened.



1032547

Figure 3-5. Head and Manipulator Electrical Controls

3-2.3 Operator Control Module

The operator control module (OCM) is a sloping panel assembly (see Figure 3-6) located in the top, left-corner of the TP04010A frame module. On its front panel, the OCM contains a 7-inch (diagonal size) cathode-ray tube display for the user. Below the display, seven function keys are located in a single row. Below this row of keys, a numeric keypad is grouped to the right of four cursor keys.

Display: The display shows the operational status of the TP04010A System. Typical operating menus show the programmed values (setpoints and soaks) and the resultant current values (DUT and ThermoStream temperatures, soak times, and cycle numbers). Also, other menus are available for test setups, system configuration, and operator maintenance.

Generally, the major portion of the screen display shows the menu information. At the bottom of the display, a row of up to seven screen selections identifies the possible functions (or screens) that can be selected next by one of the function keys. Immediately above the screen selections, a line of text gives a help tip for the highlighted selection.

Function Keys: Press one of these keys to select (highlight) the corresponding screen selection displayed along the bottom of the display.

Keypad: Contains 10 numeric keys, decimal key, and minus key for entering test parameter values (temperature setpoints, window tolerances, soaks, and cycles) as well as calibration values and an IEEE-488 address. Press the escape [ESC] key to exit from a routine and return to the previous selection without implementing a change. Press the [ENTER] key to implement (save) a selection.

If an external display and keyboard are used, they must be as specified in Paragraph 2-7.6. When using an external keyboard, the cursor keys are used to highlight the desired item followed by the [ENTER] key to select the choice.

Cursor (arrow) Keys: Used to highlight and select already entered (programmed) values or fields to be changed.

[HELP] Key: Used to access the appropriate help screen for more information on a displayed screen (or an error message).

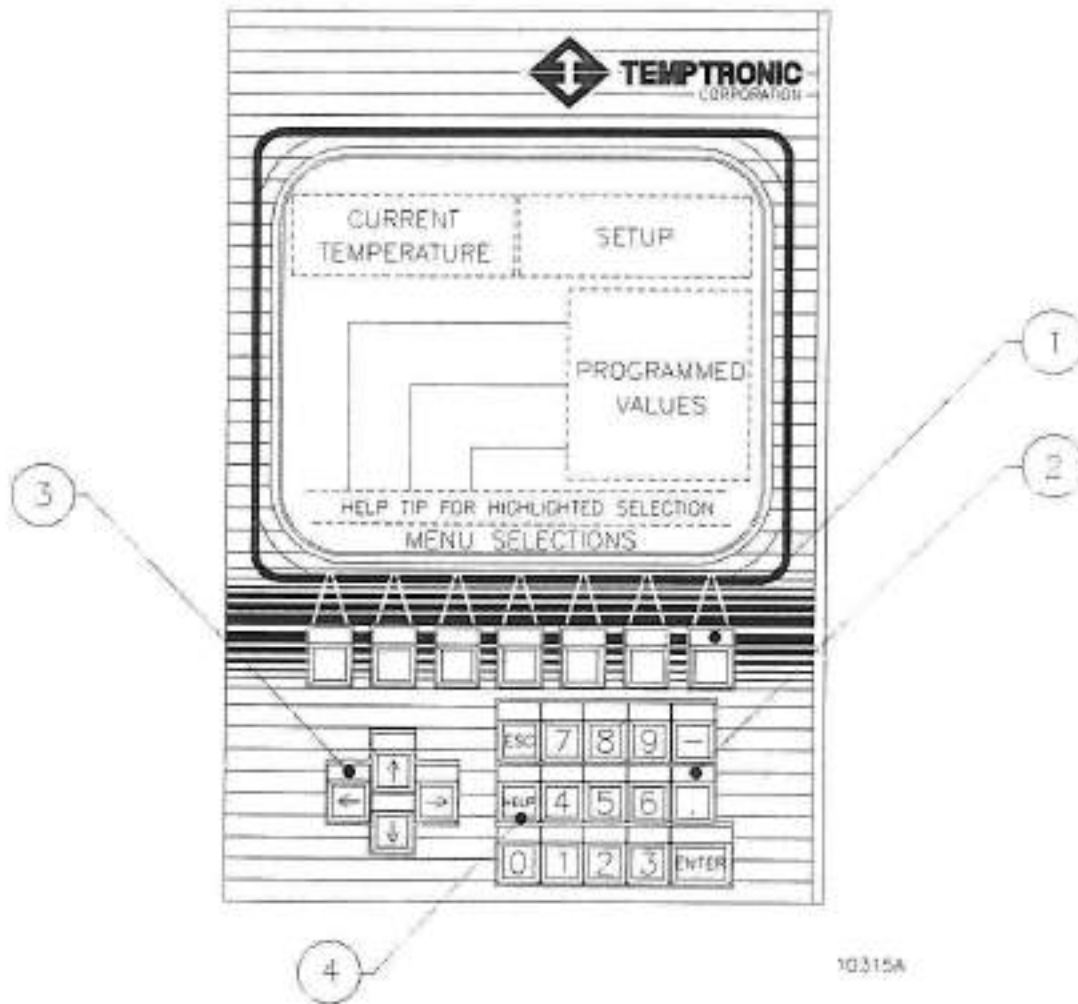


Figure 3-6. Operator Control Module (OCM)

3-3 Startup/Shutdown

3-3.1 Startup from Power On

To start up the TP04010A System initially or after it has been shutdown.

1. Check that the system is connected to the specified air source and the air is turned on.
2. Check that the system is connected to the specified ac power source and the front panel **CONTROL BREAKER** and **MAIN BREAKER** are turned on.
3. Toggle the front panel **OFF/ON** switch to the **ON** position and then release.
4. Next observe that the TP04010A System first displays a program status message that lasts about 30 seconds on the OCM.

After the program status message, the operating program will start up in either the Operator Mode (default mode) or the Variable Setpoint Mode (if selected on *System Configuration Screen*). The Operator Mode (Subsection 3-4) has a programmed selection of 12 different test setups (each setup includes a hot setpoint and a cold setpoint). The Variable Setpoint Mode (Subsection 3-5) offers an expanded programmed selection of 12 different test setups, each with 12 different setpoints as desired (cold to hot). For application, the Operator Mode facilitates production run testing, while the Variable Setpoint Mode accommodates the engineering evaluation and characterization of development devices.

Accessible from either the Operator Mode or Variable Setpoint Mode, the *Top Menu Screen* (Subsection 3-6) presents nine different selections. These selections allow you to switch quickly the operating mode, to access auxiliary function screens, and to shutdown the system.

3-3.2 System Shutdown

To shutdown the TP04010A System.

1. Place the system in a shutdown state if possible (*Top Menu Screen* displayed).
2. Toggle the **OFF/ON** switch to the **OFF** position and then release. Note that the OCM display turns off.

NOTE: When moving the TP04010A to a different test location, or when shutting down for an extended period, turn off and disconnect from its ac power source and air supply.

3-4 Operator Mode

3-4.1 Startup Sequence

At system start-up as soon as the power is turned on, observe that

- The electronics module fan turns on.
- The operator control module (OCM) will show a program status message and then several seconds later the *Startup Sequence Screen* (see Figure 3-7) will appear. (If this screen does not appear but is replaced by a *CMOS error* or *Setup error* message, see Subsection Par. 5-8.)

The *Startup Sequence Screen* displays a 60-second countdown delay of system startup. The delay purges the system's cold air line of condensed moisture each time the system is turned off for more than 2 hours. For shutdown intervals less than 2 hours, the system goes directly into the Operator Mode.

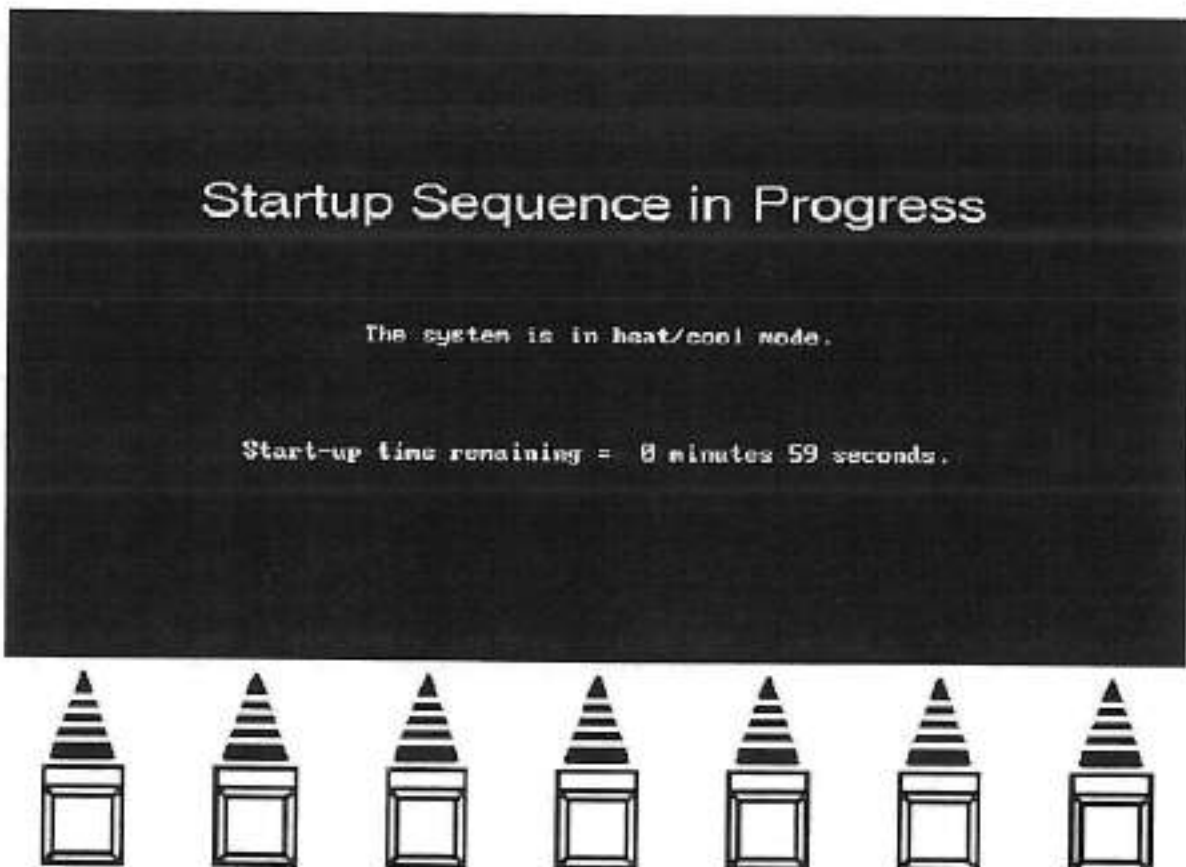


Figure 3-7. Startup Sequence Screen

3-4.2 Test Procedure

The next paragraphs describe the Operator Mode for testing at a manually selected setpoint (*Operator Main Screen*) or for testing by cycling between two hot and cold setpoints (*Operator Test Cycle Screen*). Other functions needed to turn the compressor (Air Chiller Module) on and off, to change the test setup parameters, and to access the *Top Menu Screen* are available on the *Operator Extended Screen*.

Operator Main Screen

The first operational screen in the Operator Mode is the *Operator Main Screen* (see Figure 3-8). Note that this screen displays the **Setup:** number in the upper right box, showing the selected setup number from 1 to 12. Also, the **Setpoint:** value in this box shows the programmed temperature of a selected setpoint (HOT, AMB, or COLD).

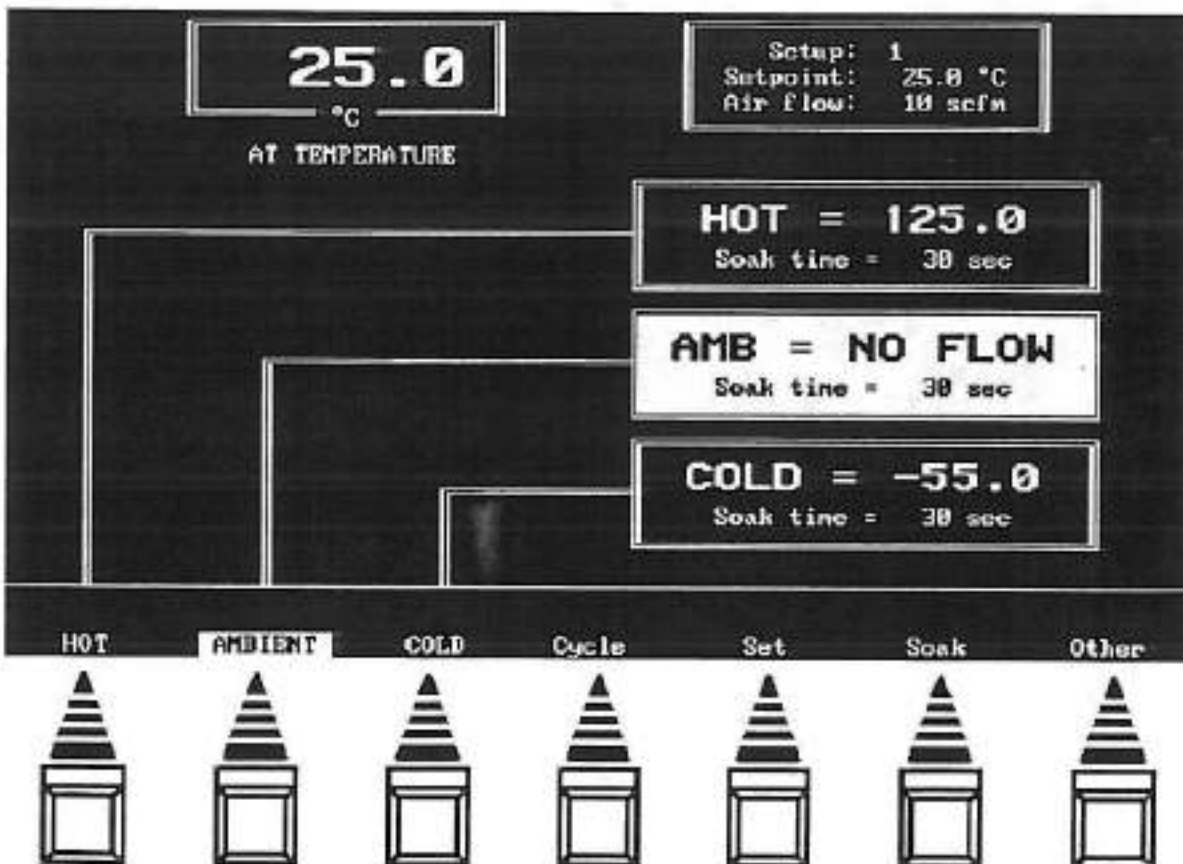


Figure 3-8. Operator Main Screen

The selections at the bottom of the screen on Figure 3-8 are summarized below. Press the OCM function key directly below a desired selection to activate it.

HOT: This selection displays the programmed temperature and soak time for the HOT setpoint. (Factory set values are 125 °C and 30 sec.)

AMBIENT: This selection displays **AMB = NO FLOW** and the soak time by default. Pressing the function key displays **AMB =** programmed temperature and the soak time (with air flow initiated). Alternate pressing of the function key toggles between the two **AMB** states. (Factory set values are 25 °C and 30 sec.)

COLD: This selection displays the programmed temperature and soak time for the COLD setpoint. (Factory set values are -55 °C and 30 sec.)

Cycle: Accesses the *Operator Test Cycle Screen* (Figure 3-9) and starts temperature cycling. (Select **Stop** on that screen to return to the *Operator Main Screen*.)

Set: Help tip displays "Choose hot, ambient or cold to change setpoint, or soak to change soak time." **NOTE:** Use this **Set** selection to make temporary changes in the test setup.

Setpoint Changes. Use the arrow left/right keys to make one of the setpoint selections (HOT, COLD, or AMBIENT). Observe that the selection is highlighted. Type in the new setpoint value desired and press [ENTER] key to save.

Soak Changes. After making the **Set** selection, then select **Soak**. Help tip displays "Choose hot, ambient or cold to change that soak time." With the selection highlighted, type in the new soak value desired and press [ENTER] key to save.

Soak: Restarts a soak time in process.

Other: Accesses the *Operator Extended Screen* (Figure 3-10) with other operator screen selections.

Operator Test Cycle Screen

The selections at the bottom of the *Operator Test Cycle Screen* (see Figure 3-9) are summarized below. Press the OCM function key directly below a desired selection to activate it.

Cycling: The system starts performing temperature cycles as selected at the *Operator Main Screen* (Figure 3-8). Each cycle starts going to the COLD setpoint. After the soak at that setpoint, the cycle continues to the HOT setpoint. After the soak at that setpoint, the cycle continues to the AMB setpoint. After the soak at that setpoint, the cycle repeats for the number of times programmed.

Stop: Stops the temperature cycling, and returns display to the *Operator Main Screen*. If a cycle is interrupted, the system continues to the next programmed setpoint and maintains temperature at that setpoint.

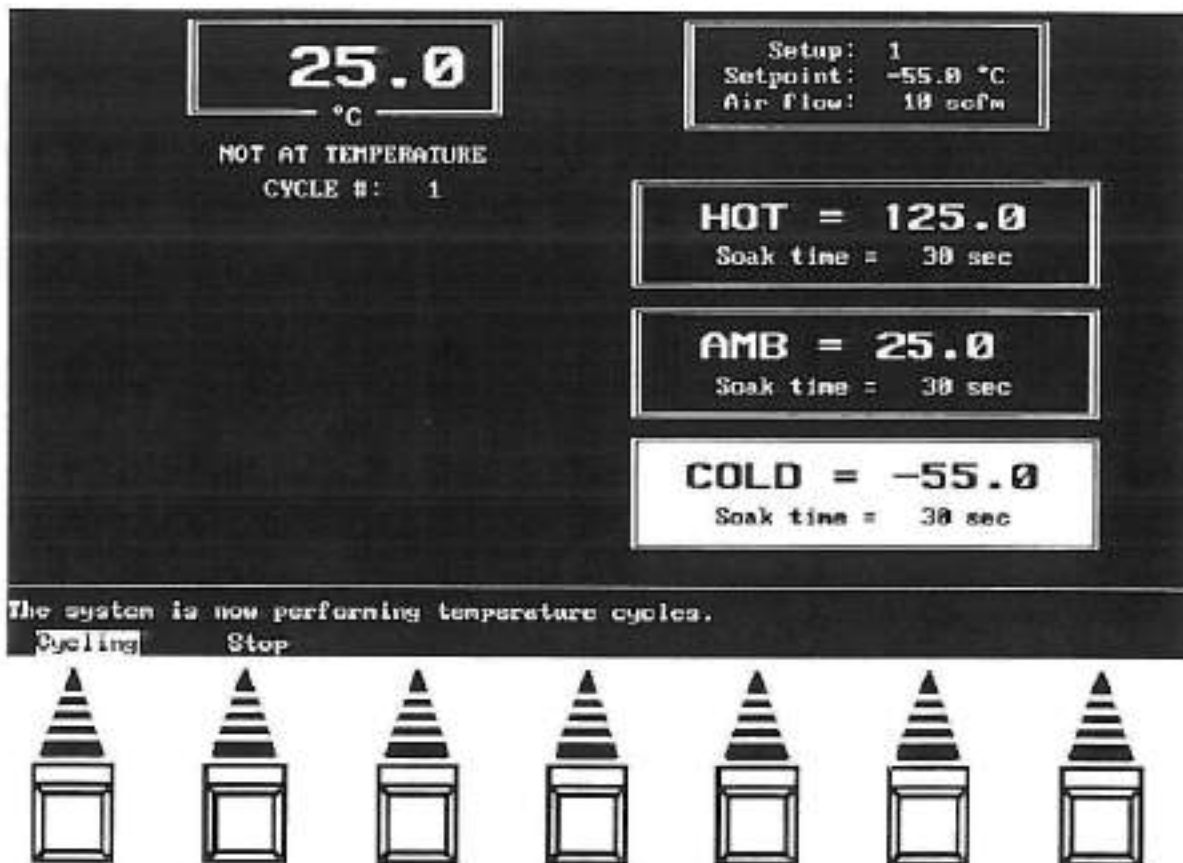


Figure 3-9. Operator Test Cycle Screen

Operator Extended Screen

The selections at the bottom of the *Operator Extended Screen* (see Figure 3-10) are summarized below. Press the OCM function key directly below a desired selection to activate it.

HOT: This selection displays the programmed temperature and soak time for the HOT setpoint.

AMBIENT: This selection displays **AMB = NO FLOW** and the soak time by default. Pressing the function key displays **AMB =** programmed temperature and the soak time (with air flow initiated). Alternate pressing of the function key toggles between the two **AMB** states.

COLD: This selection displays the programmed temperature and soak time for the COLD setpoint.

Compressor: Turns the compressor on or turns the compressor off. Toggles to the displayed state by pressing the function key.

Setup: Accesses the *Test Change Setup Screen* (Figure 3-11). (Press **Return** function key on that screen to return to the *Operator Extended Screen*.)

Top menu: Accesses the *Top Menu Screen* (Figure 3-21).

Return: Returns the display to the *Operator Main Screen* (Figure 3-8).

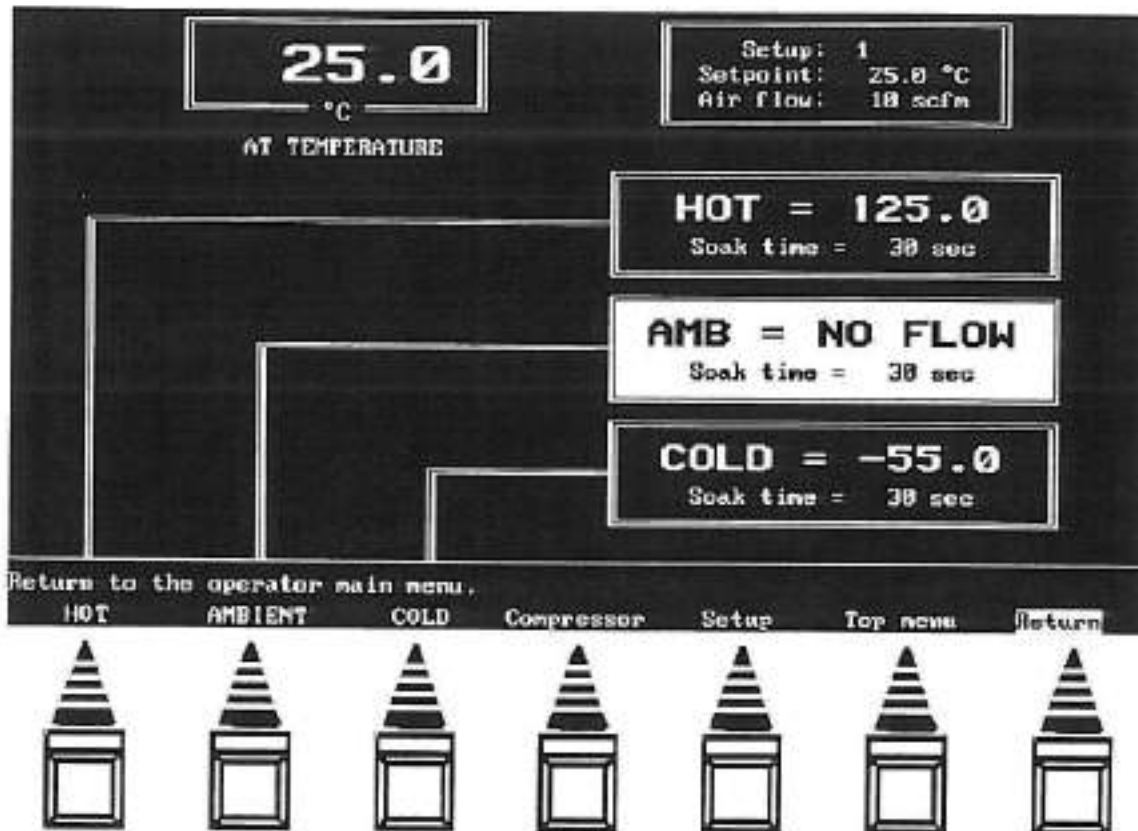


Figure 3-10. Operator Extended Screen

3-4.3 Test Setup

In the Operator Mode, up to 12 programmed test setup files can be stored in the TP04010A. Each test setup file contains the system setup parameters and the three temperature setpoint parameters to allow a DUT to be tested under the desired HOT, AMB, and COLD temperature conditions. Different test setup files are programmed through the *Operator Change Setup Screen* (see Figure 3-11). This figure shows the default parameters as programmed by the factory. From this base, or later test setup files, a new file can be programmed by changing the system setup parameters and the temperature setpoint parameters for a particular test. This combination of parameters is then saved (stored in memory) with a unique setup number from 1 through 12 on the *Operator Save Setup Screen* (Figure 3-12). Any one of the up to 12 setup files can be selected and loaded from the *Operator Load Setup Screen* (Figure 3-13) for controlling the TP04010A.

Operator Change Setup Screen

The selections at the bottom of the *Operator Change Setup Screen* (see Figure 3-11) are summarized below. Press the OCM function key directly below a desired selection to activate it.

View: Displays the test setup.

Change setup: See paragraph entitled "Changing System Setup Parameters" for details.

Change setpoints: See paragraph entitled "Changing Temperature Setpoint Parameters" for details.

NOTE: The last part in changing a test setup includes going to the **Save** selection.

Load: Accesses the *Operator Load Setup Screen* (Figure 3-13). See paragraph entitled "Loading a New Test Setup" for details.

Save: Accesses the *Operator Save Setup Screen* (see Figure 3-12). See paragraph entitled "Changing Temperature Setpoint Parameters" for details.

Return: Returns the display to the *Operator Extended Screen* (Figure 3-10).

Changing System Setup Parameters

To change the system setup parameters (test conditions) for a new setup file or to modify a parameter for an existing setup file.

1. Select the **Change setup** function on the *Operator Change Setup Screen* (refer to Figure 3-11) with the OCM function key to activate.
2. Change the setup parameters as needed in the left window of the screen. Use the up/down arrow keys to access (highlight) a data field. Type the change and then press the [ENTER] key to save that entry.
3. Check that the **DUT control** is set for the desired type, air (0) or direct DUT control (1). If direct DUT control is chosen, make sure the proper **DUT sensor** is selected, (1) or (2). A new

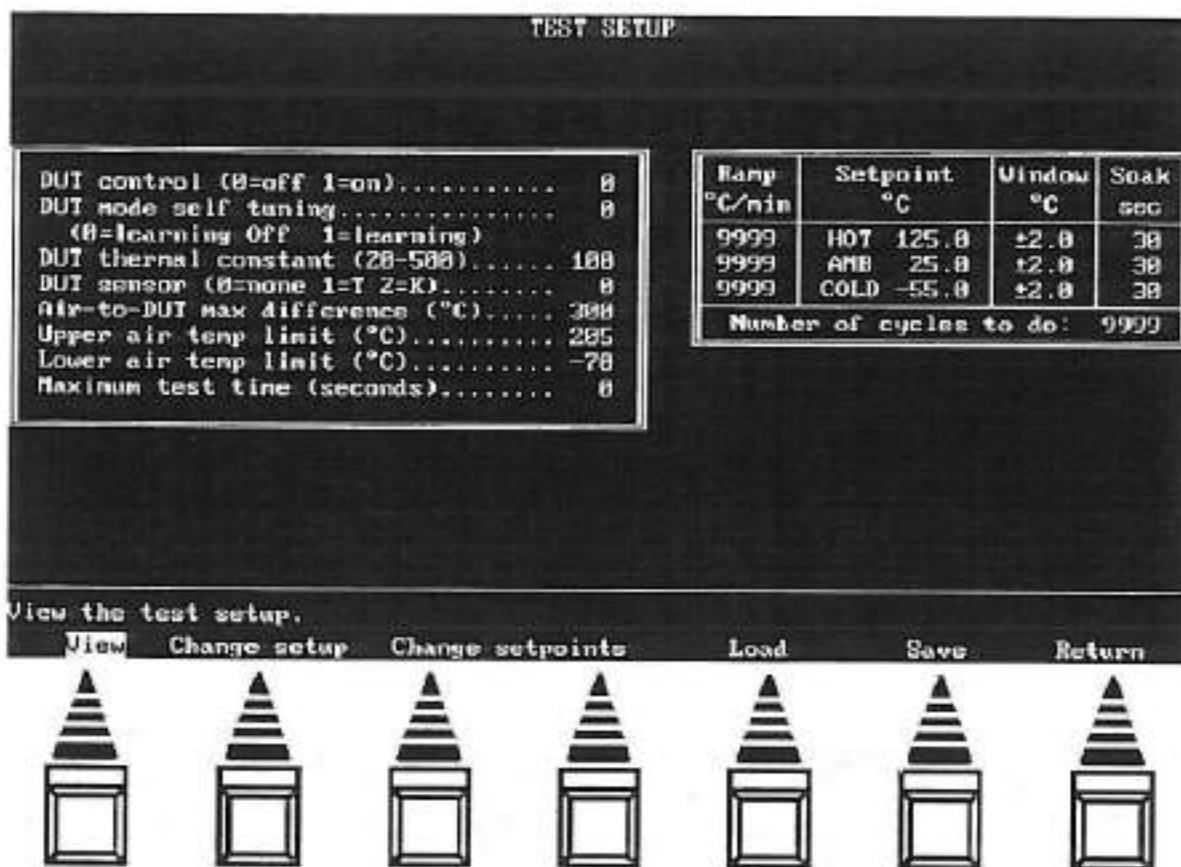


Figure 3-11. Operator Change Setup Screen

DUT sensor should be calibrated (refer to Sec. 6) before use. Also, for direct DUT control, check that the proper values are given for **Air-to-DUT max. temperature**, **Upper air temp limit**, and **Lower air temp limit**.

4. Check that the **DUT mode self-tuning** is set for the desired state. The learning state (1) allows the TP04010A to learn and adapt itself successively to match the DUT mass for the best compromise between minimal overshoot and fastest temperature transition time. If the learning state is off (0), the **DUT thermal constant** should be specified as best known.
5. Check that the **Maximum test time** is a valid one. It should be either 0 seconds to be ignored or a higher value that prevents the system from "hanging up" because an "end test" signal is not received in time from the host tester.
6. After making the needed changes, press the [ESC] key to leave the **Change setup** mode.

Changing Temperature Setpoint Parameters

To change the temperature setpoint parameters for a new setup file or to modify a parameter for an existing setup file,

1. Select the **Change setpoints** function on the *Operator Change Setup Screen* (refer to Figure 3-11) with the OCM function key to activate.
2. Change the setpoint parameters as needed in the right window of the screen. Use the arrow keys to access (highlight) a data field. Type the change and then press the [ENTER] key to save that entry.
3. After making the needed changes, press the [ESC] key to leave the **Change setpoints** mode.
4. Select the **Save** function on the *Operator Change Setup Screen* with the OCM function key to access the *Operator Save Setup Screen* (see Figure 3-12).
5. Use the up/down arrow keys to select the numbered line (1 to 12) on which to enter (or overwrite) the new test setup file (system setup parameters and temperature setpoint parameters).
6. Press the [ENTER] key to save the new test setup file at the selected line and return to the *Operator Change Setup Screen*.

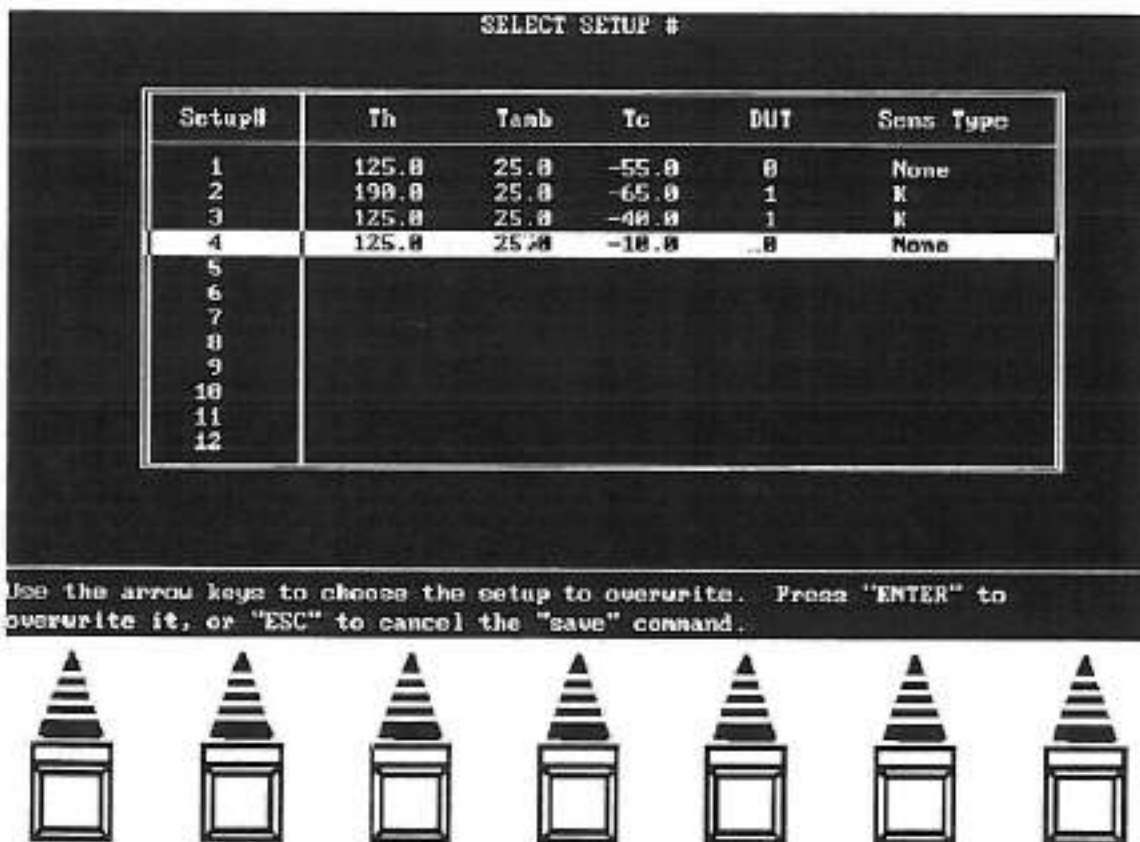


Figure 3-12. Operator Save Setup Screen

Loading a New Test Setup

To load a new test setup file for temperature control of the TP04010A,

1. Select the **Load** function on the *Operator Change Setup Screen* (refer to Figure 3-12) with the OCM function key to access the *Operator Load Setup Screen* (see Figure 3-13).
2. Use the up/down arrow keys to select the numbered line (1 to 12) with the desired test setup file to be used next. (Figure 3-13 shows an example of the *Operator Load Setup Screen* with four saved test setups and Setup #2 highlighted.)
3. Press the [ENTER] key to load the test setup file at the selected (highlighted) line and return to the *Operator Change Setup Screen*. Note that the selected test setup will be identified after **Setup:** at the upper right of the *Operator Main Screen*, *Operator Test Cycle Screen*, and *Operator Extended Screen*.

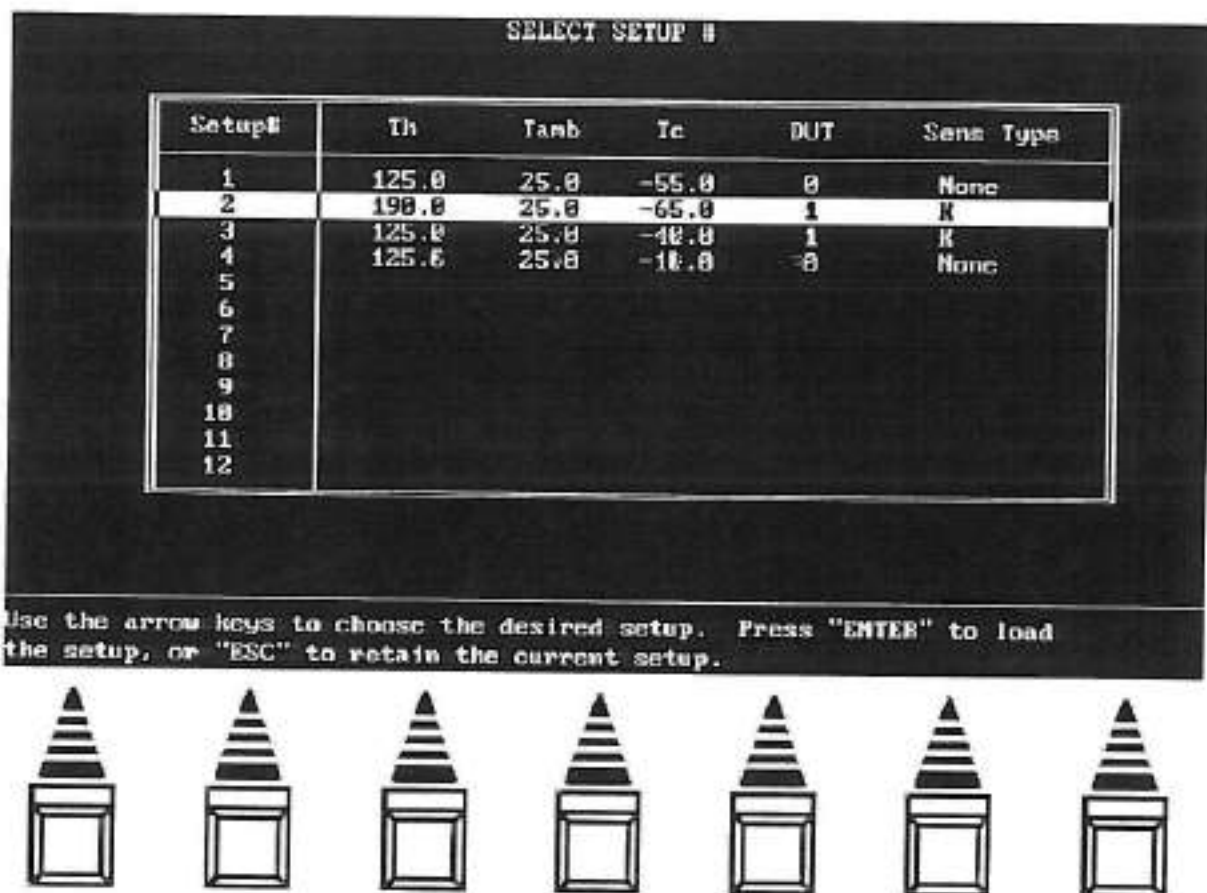


Figure 3-13. Operator Load Setup Screen



Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

3-5 Variable Setup Mode

3-5.1 Startup Sequence

At system start-up as soon as the power is turned on, observe that

- The electronics module fan turns on.
- The operator control module (OCM) will show a program status message and then several seconds later the *Temptronic Logo Screen* (see Figure 3-14) will appear. (If this screen does not appear but is replaced by a *CMOS error* or *Setup error* message, see Subsection. 5-8.)

To continue the normal startup sequence, press any key on the OCM keypad. Next the display switches to the *Startup Sequence Screen* (Figure 3-7). This screen displays a 60-second countdown delay of system startup. The delay purges the system's cold air line of condensed moisture each time the system is turned off for more than 2 hours. For shutdown intervals less than 2 hours, the system goes directly into the Variable Setup (V.S.) Mode.

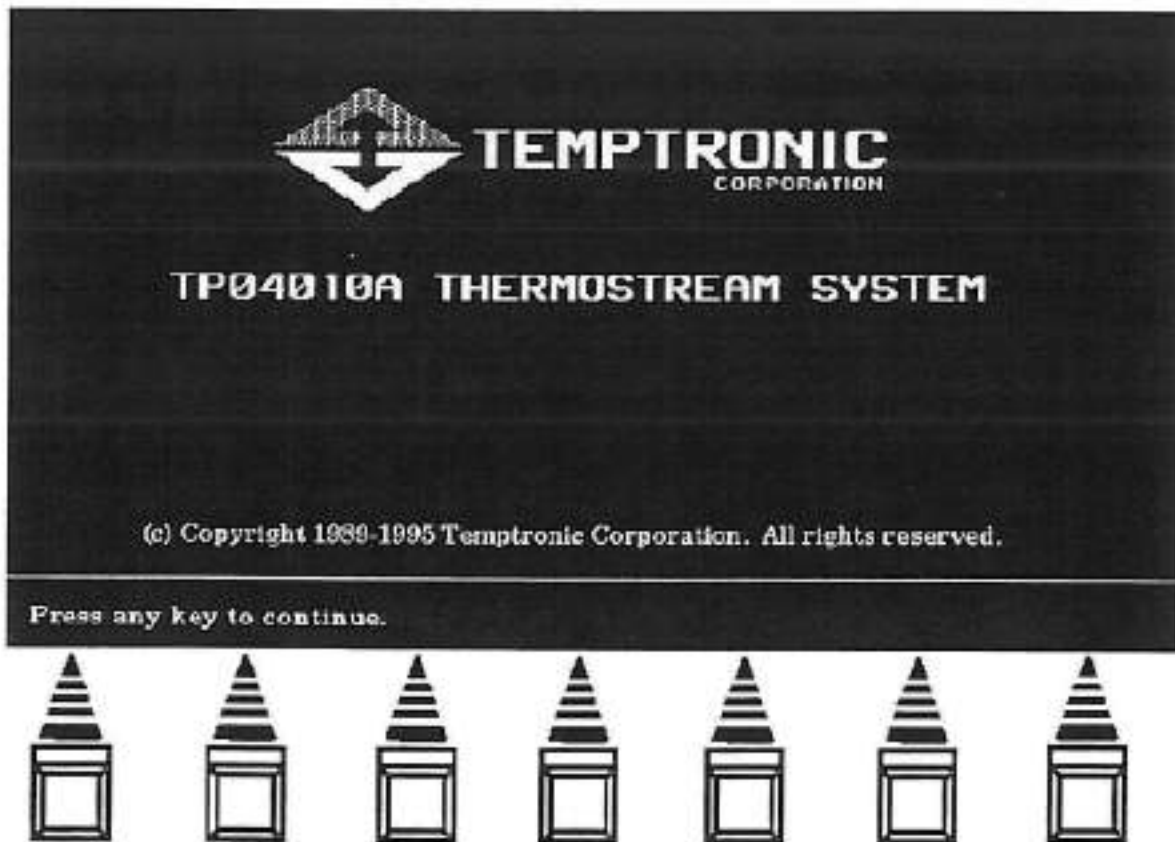


Figure 3-14. Temptronic Logo Screen

3-5.2 Test Procedure

The next paragraphs describe the Variable Setup Mode for testing at a manually selected setpoint (*V.S. Main Screen*) or for testing of cycling sequences from two up to 12 setpoints (*V.S. Test Cycle Screen*). Ramp and soak parameter for the cycling sequences are selected for the test setup (loaded from 1 of 12) on the *V.S. Ramp/Cycle Screen*.

V.S. Main Screen

The first operational screen in the Variable Setup Mode is the *V.S. Main Screen* (see Figure 3-15). Note that this screen displays the **Setup:** number in the upper right box, showing the selected setup number from 1 to 12. Also, the **Setpoint:** value in this box shows the programmed temperature of a selected setpoint or the next setpoint in a cycle underway.

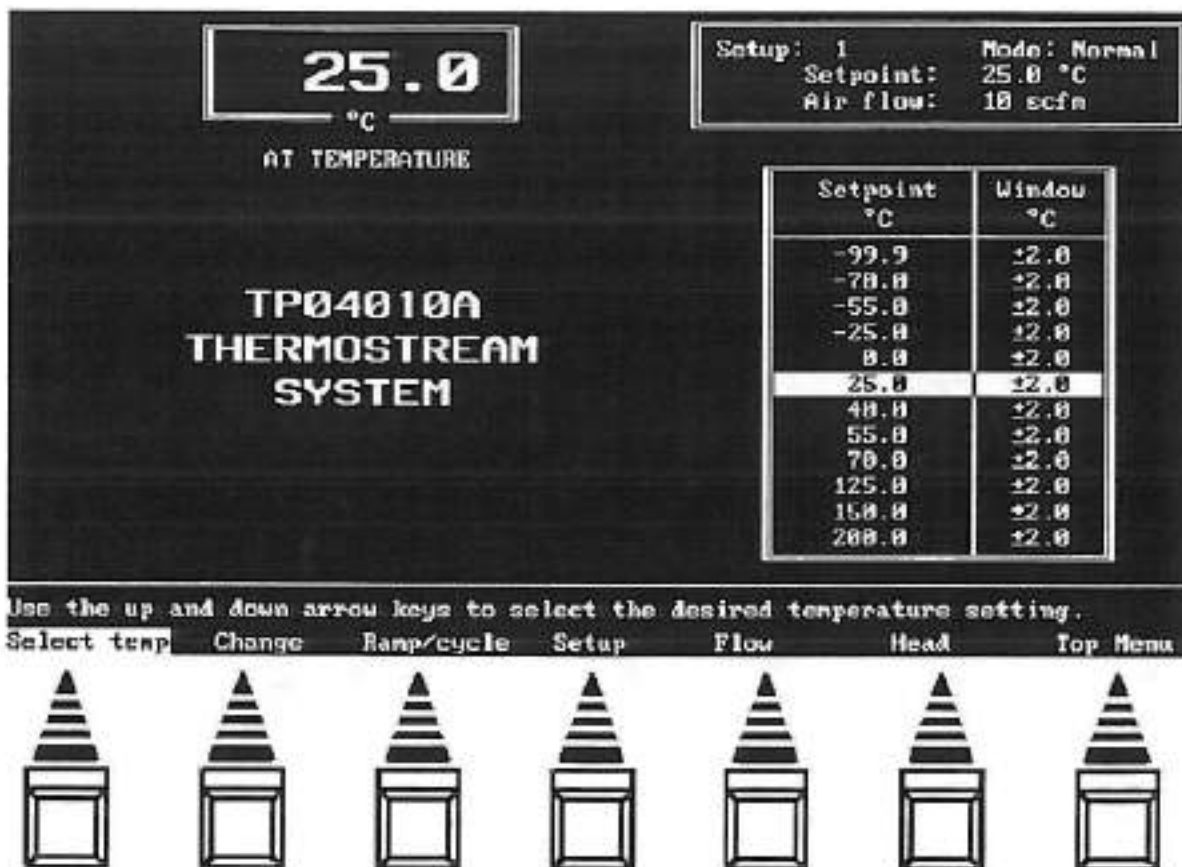


Figure 3-15. V.S. Main Screen

The selections at the bottom of the screen on Figure 3-15 are summarized below. Press the OCM function key directly below a desired selection to activate it.

Select temp: The help tip displays “Use the up and down arrow keys to select the desired temperature setting”. Press the [ENTER] key to select setpoint choice or the [ESC] key to abandon it. This selection of one setpoint starts temperature control at that setpoint. To stop control at that temperature, select a new setpoint or the ambient (25 °C) setpoint.

Change: The help tip displays “Use the arrow keys to select the field to change. Type the changes, and press ‘ENTER’ to save. Press ‘ESC’ to leave ‘change’ mode.” Use this selection to enter a temporary change to the test setup displayed.

Ramp/Cycle: Accesses the *V.S. Ramp/Cycle Screen* (Figure 3-16). (Use the **Return** selection on that screen to get back to the *V.S. Main Screen*.)

Setup: Accesses the *V.S. Test Setup Screen* (Figure 3-17). (Use the **Return** selection on that screen to get back to the *V.S. Main Screen*.)

Flow: Allows you to turn the main air flow on or off. The highlighted **Off** or **On** state indicates the next state to be changed to by pressing the [ENTER] key. Pressing the [ESC] key leaves the current state unchanged. Help screen has more details.

Head: Allows you to raise or lower the head module. The highlighted **Up** or **Down** state indicates the next state to be changed to by pressing the [ENTER] key. Pressing the [ESC] key leaves the current state unchanged. Help screen has more details.

Top Menu: Accesses the *Top Menu Screen* (Figure 3-21).

V.S. Ramp/Cycle Screen

The selections at the bottom of the *V.S. Ramp/Cycle Screen* (see Figure 3-16) are summarized below. Press the OCM function key directly below a desired selection to activate it.

Manual: Help tip displays "Use the up and down arrow keys to select a temperature to ramp to." Press [ENTER] to start temperature control at a selected setpoint. To stop control at that temperature, select a new setpoint or the ambient (25 °C) setpoint.

Change: Help tip displays "Use the arrow keys to select the field to change. Type the changes, and press 'ENTER' to save. Press 'ESC' to leave 'change' mode." Note that when making changes, two or more setpoints must be selected to form a cycle. A ramp value greater than 0 is needed to enable a setpoint for cycling. Don't forget to specify the number of cycles you want.

Cycle: Accesses the *V.S. Test Cycle Screen* (Figure 3-17). (Use the **Stop** selection on that screen to return to the *V.S. Ramp/Cycle Screen*.)

Flow: Allows you to turn the main air flow on or off. The highlighted **Off** or **On** state indicates the next state to be changed to by pressing the [ENTER] key. Pressing the [ESC] key leaves the current state unchanged. Help screen has more details.

Head: Allows you to raise or lower the head module. The highlighted **Up** or **Down** state indicates the next state to be changed to by pressing the [ENTER] key. Pressing the [ESC] key leaves the current state unchanged. Help screen has more details.

Return: Accesses the *V.S. Main Screen* (Figure 3-15).

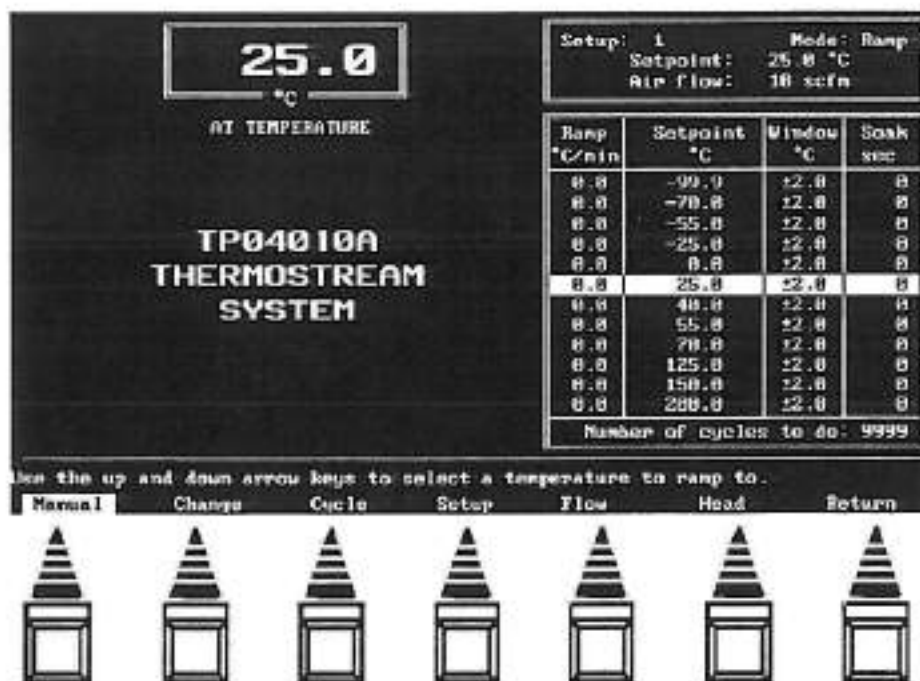


Figure 3-16. V.S. Ramp/Cycle Screen

V.S. Test Cycle Screen

The selections at the bottom of the *V.S. Test Cycle Screen* (see Figure 3-17) are summarized below. Press the OCM function key directly below a desired selection to activate it.

Cycling: The system starts performing temperature cycles as selected at the *V.S. Ramp/Cycle Screen* (Figure 3-16). Each cycle starts at the coldest setpoint and progresses through to the hottest setpoint. After the soak at the hottest setpoint, the cycle repeats for the number of times programmed.

Pause: The system interrupts the temperature cycle in process. Help tip on interim row of selections displays "Temperature cycling has been suspended by the operator."

NOTE: To resume temperature cycling, select **Continue** on the interim row of selections. Or, to stop temperature cycling at this point, select **Stop** on the interim screen and return to the *V.S. Ramp/Cycle Screen*.

Stop: Stops the temperature cycling, and returns display to the *V.S. Ramp/Cycle Screen*. If a cycle is interrupted or stopped, the system continues to the next programmed setpoint and maintains temperature at that setpoint (shown at **Cycle #**).

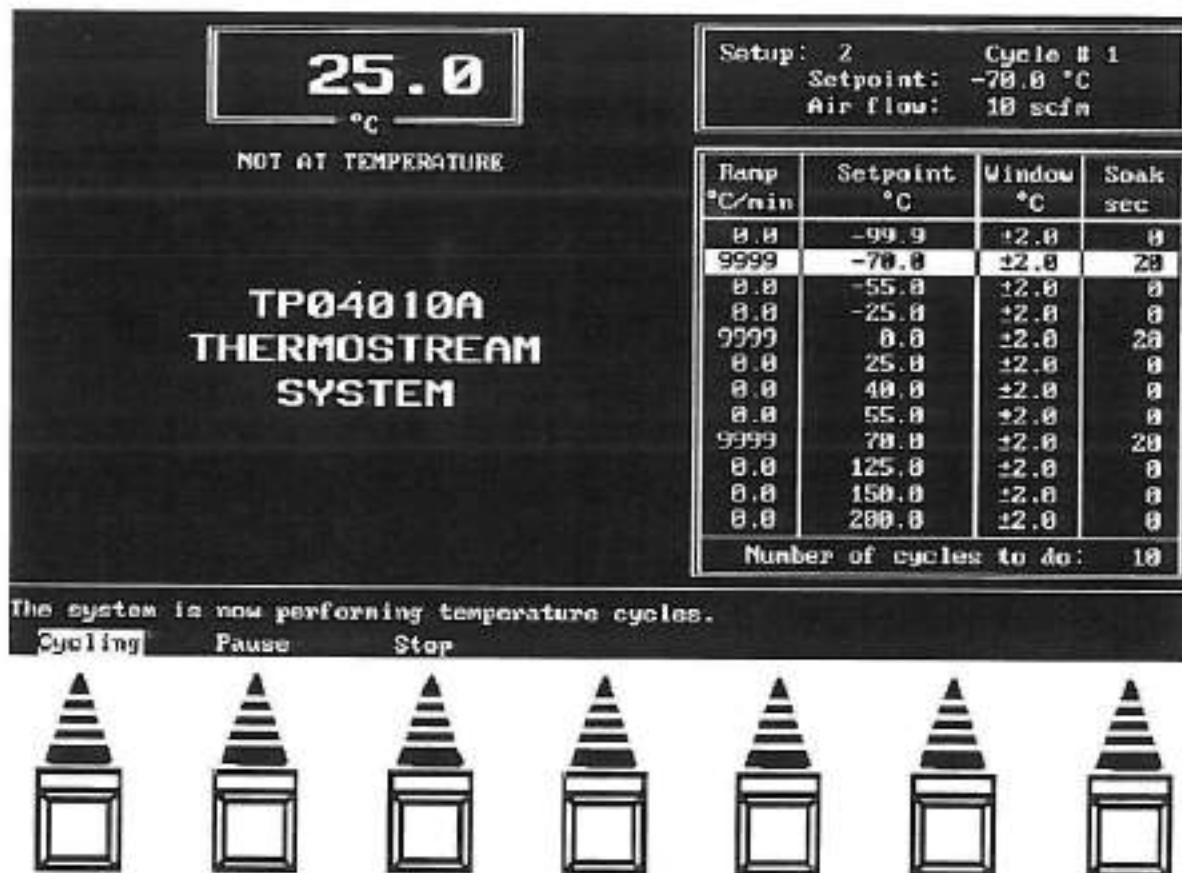


Figure 3-17. V.S. Test Cycle Screen

3-5.3 Test Setup

In the Variable Setup Mode, up to 12 programmed test setup files can be stored in the TP04010A. Each test setup file contains the system setup parameters and temperature setpoint parameters to allow a DUT to be tested over a ramp/cycle sequence of up to 12 setpoints. Different test setup files are programmed through the *V.S. Change Setup Screen* (see Figure 3-18). This figure shows the default parameters as programmed by the factory. From this base, or later test setup files, a new file can be programmed by changing the system setup parameters and the temperature setpoint parameters for a particular test. This combination of parameters is then saved (stored in memory) with a unique setup number from 1 through 12 on the *V.S. Save Setup Screen* (Figure 3-19). Any one of the up to 12 setup files can be selected and loaded from the *V.S. Load Setup Screen* (Figure 3-20) for controlling the TP04010A.

V.S. Change Setup Screen

The selections at the bottom of the *V.S. Change Setup Screen* (see Figure 3-18) are summarized below. Press the OCM function key directly below a desired selection to activate it.

View: Displays the test setup.

Change setup: Use this selection to make permanent (stored) changes in one or more test setups. See paragraph entitled "Changing System Setup Parameters" for details.

Change setpoints: See paragraph entitled "Changing Temperature Setpoint Parameters" for details.
NOTE: The last part in changing a test setup includes going to the **Save** selection.

Load: Accesses the *V.S. Load Setup Screen* (see Figure 3-20). See paragraph entitled "Loading a New Setup" for details.

Save: Accesses the *Operator Save Setup Screen* (see Figure 3-19). See paragraph entitled "Changing System Setup Parameters" for details.

Return: Returns the display to the *V.S. Main Screen* (Figure 3-15).

Changing System Setup Parameters

To change the system setup parameters (test conditions) for a new setup file or to modify a parameter for an existing setup file.

1. Select the **Change setup** function on the *V.S. Change Setup Screen* (refer to Figure 3-18) with the OCM function key to activate.
2. Change the setup parameters as needed in the left window of the screen. Use the up/down arrow keys to access (highlight) a data field. Type the change and then press the [ENTER] key to save that entry.
3. Check that the **DUT control** is set for the desired type, air (0) or direct DUT control (1). If direct DUT control is chosen, make sure the proper **DUT sensor** is selected, (1) or (2). A new DUT sensor should be calibrated (refer to Sec. 6) before use. Also, for direct DUT control.

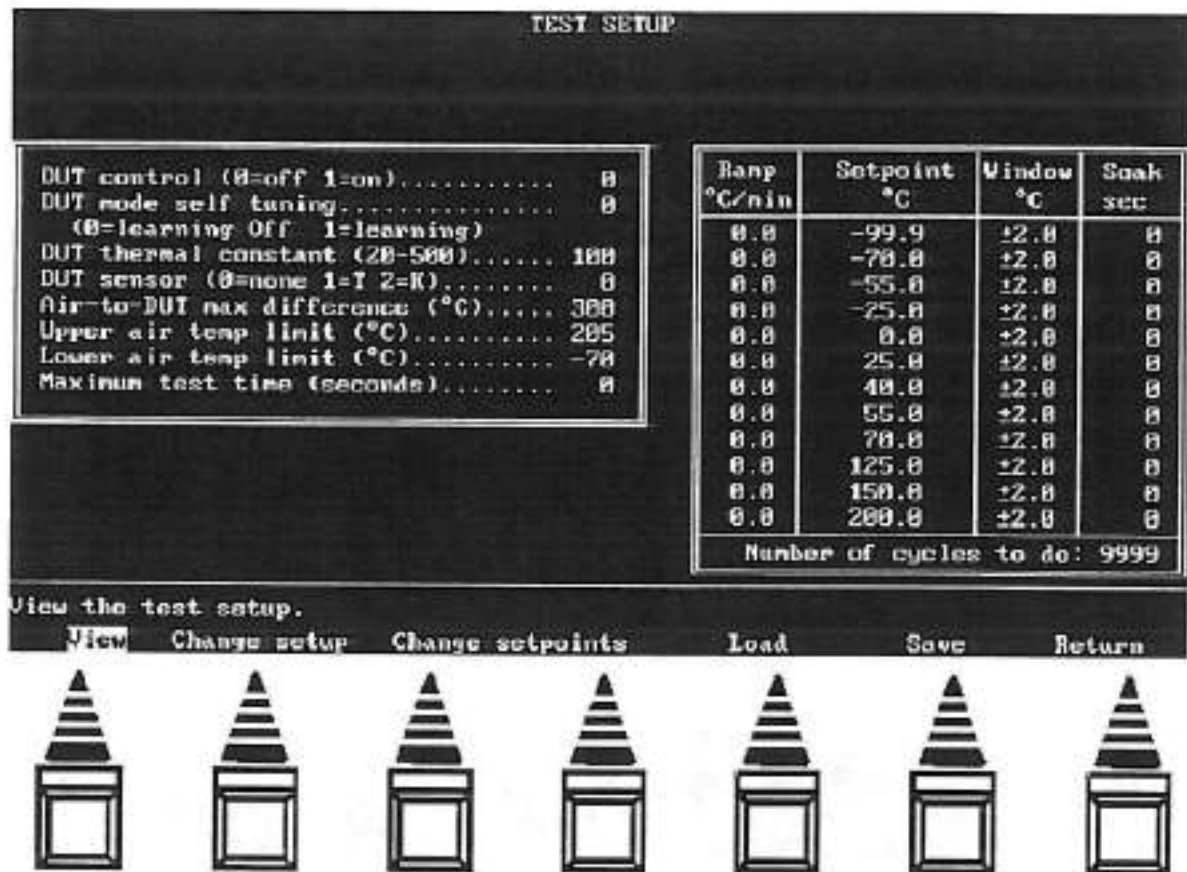


Figure 3-18. V.S. Change Setup Screen

check that the proper values are given for Air-to-DUT max. temperature, Upper air temp limit, and Lower air temp limit.

4. Check that the **DUT mode self-tuning** is set for the desired state. The learning state (1) allows the TP04010A to learn and adapt itself successively to match the DUT mass for the best compromise between minimal overshoot and fastest temperature transition time. If the learning state is off (0), the **DUT thermal constant** should be specified as best known.
5. Check that the **Maximum test time** is a valid one. It should be either 0 seconds to be ignored or a higher value that prevents the system from "hanging up" because an "end test" signal is not received in time from the host tester.
6. After making the needed changes, press the [ESC] key to leave the **Change setup** mode.

Changing Temperature Setpoint Parameters

To change the temperature setpoint parameters for a new setup file or to modify a parameter for an existing setup file,

1. Select the **Change setpoints** function on the *V.S. Change Setup Screen* (refer to Figure 3-18) with the OCM function key to activate.
2. Change the setpoint parameters as needed in the right window of the screen. Use the arrow keys to access (highlight) a data field. Type the change and then press the [ENTER] key to save that entry.
3. After making the needed changes, press the [ESC] key to leave the **Change setpoints** mode.
4. Select the **Save** function on the *V.S. Change Setup Screen* with the OCM function key to access the *V.S. Save Setup Screen* (see Figure 3-19).
5. Use the up/down arrow keys to select the numbered line (1 to 12) on which to enter (or overwrite) the new test setup file (system setup parameters and temperature setpoint parameters).
6. Press the [ENTER] key to save the new test setup file at the selected line and return to the *V.S. Change Setup Screen*.

SAVE TEST SETUP

| DUT control (0=off 1=on)..... 1 DUT mode self tuning..... 1 (0=learning Off 1=learning) DUT thermal constant (20-500)..... 175 DUT sensor (0=none 1=T 2=R)..... 2 Air-to-DUT max difference (°C)..... 150 Upper air temp limit (°C)..... 205 Lower air temp limit (°C)..... -78 Maximum test time (seconds)..... 100 | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Rang °C/min</th> <th style="text-align: left;">Setpoint °C</th> <th style="text-align: left;">Window °C</th> <th style="text-align: left;">Soak sec</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>-99.9</td><td>±2.0</td><td>0</td></tr> <tr><td>0.0</td><td>-70.0</td><td>±2.0</td><td>0</td></tr> <tr><td>150</td><td>-60.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>-30.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>0.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>30.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>60.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>90.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>120.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>150.0</td><td>±1.0</td><td>30</td></tr> <tr><td>150</td><td>180.0</td><td>±1.0</td><td>30</td></tr> <tr><td>0.0</td><td>200.0</td><td>±2.0</td><td>0</td></tr> </tbody> </table> <p style="text-align: right; margin-top: 5px;">Number of cycles to do: 50</p> | Rang °C/min | Setpoint °C | Window °C | Soak sec | 0.0 | -99.9 | ±2.0 | 0 | 0.0 | -70.0 | ±2.0 | 0 | 150 | -60.0 | ±1.0 | 30 | 150 | -30.0 | ±1.0 | 30 | 150 | 0.0 | ±1.0 | 30 | 150 | 30.0 | ±1.0 | 30 | 150 | 60.0 | ±1.0 | 30 | 150 | 90.0 | ±1.0 | 30 | 150 | 120.0 | ±1.0 | 30 | 150 | 150.0 | ±1.0 | 30 | 150 | 180.0 | ±1.0 | 30 | 0.0 | 200.0 | ±2.0 | 0 |
|--|---|----------------|----------------|--------------|-------------|-----|-------|------|---|-----|-------|------|---|-----|-------|------|----|-----|-------|------|----|-----|-----|------|----|-----|------|------|----|-----|------|------|----|-----|------|------|----|-----|-------|------|----|-----|-------|------|----|-----|-------|------|----|-----|-------|------|---|
| Rang °C/min | Setpoint °C | Window °C | Soak sec | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | -99.9 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | -70.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | -60.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | -30.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 0.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 30.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 60.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 90.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 120.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 150.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 150 | 180.0 | ±1.0 | 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 200.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

SETUP # TO OVERWRITE

| | | | | | |
|---|---|---|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |

Use the arrow keys to choose the setup to overwrite. Press "ENTER" to overwrite it, or "ESC" to cancel the "save" command.



Figure 3-19. V.S. Save Setup Screen

Loading a New Test Setup

To load a new test setup file for temperature control of the TP04010A,

1. Select the **Load** function on the *V.S. Change Setup Screen* (refer to Figure 3-18) with the OCM function key to access the *V.S. Load Setup Screen* (see Figure 3-20).
2. Use the up/down arrow keys to select the numbered line (1 to 12) with the desired test setup file to be used next. (Figure 3-20 shows an example of the *V.S. Load Setup Screen* with Setup #2 highlighted and its test parameters shown on the screen; test parameters shown on the screen correspond to the selected Setup number.)
3. Press the [ENTER] key to load the test setup file at the selected (highlighted) line and return to the *V.S. Change Setup Screen*. Note that the selected test setup will be identified after **Setup:** at the upper right of the *V.S. Main Screen*, *V.S. Test Cycle Screen*, and *V.S. Ramp/Cycle Screen*.

LOAD TEST SETUP

| DUT control (0=off 1=on)..... 0 DUT mode self tuning..... 0 (0=learning Off 1=learning) DUT thermal constant (20-500)..... 100 DUT sensor (0=none 1=T 2=K)..... 0 Air-to-DUT max difference (°C)..... 300 Upper air temp limit (°C)..... 205 Lower air temp limit (°C)..... -70 Maximum test time (seconds)..... 0 | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Ramp °C/min</th> <th>Setpoint °C</th> <th>Window °C</th> <th>Soak sec</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>-99.0</td><td>±2.0</td><td>0</td></tr> <tr><td>9999</td><td>-70.0</td><td>±2.0</td><td>20</td></tr> <tr><td>0.0</td><td>-55.0</td><td>±2.0</td><td>0</td></tr> <tr><td>0.0</td><td>-25.0</td><td>±2.0</td><td>0</td></tr> <tr><td>9999</td><td>0.0</td><td>±2.0</td><td>20</td></tr> <tr><td>0.0</td><td>25.0</td><td>±2.0</td><td>0</td></tr> <tr><td>0.0</td><td>40.0</td><td>±2.0</td><td>0</td></tr> <tr><td>0.0</td><td>55.0</td><td>±2.0</td><td>0</td></tr> <tr><td>9999</td><td>70.0</td><td>±2.0</td><td>20</td></tr> <tr><td>0.0</td><td>125.0</td><td>±2.0</td><td>0</td></tr> <tr><td>0.0</td><td>150.0</td><td>±2.0</td><td>0</td></tr> <tr><td>0.0</td><td>200.0</td><td>±2.0</td><td>0</td></tr> </tbody> </table> | Ramp °C/min | Setpoint °C | Window °C | Soak sec | 0.0 | -99.0 | ±2.0 | 0 | 9999 | -70.0 | ±2.0 | 20 | 0.0 | -55.0 | ±2.0 | 0 | 0.0 | -25.0 | ±2.0 | 0 | 9999 | 0.0 | ±2.0 | 20 | 0.0 | 25.0 | ±2.0 | 0 | 0.0 | 40.0 | ±2.0 | 0 | 0.0 | 55.0 | ±2.0 | 0 | 9999 | 70.0 | ±2.0 | 20 | 0.0 | 125.0 | ±2.0 | 0 | 0.0 | 150.0 | ±2.0 | 0 | 0.0 | 200.0 | ±2.0 | 0 |
|--|---|----------------|----------------|--------------|-------------|-----|-------|------|---|------|-------|------|----|-----|-------|------|---|-----|-------|------|---|------|-----|------|----|-----|------|------|---|-----|------|------|---|-----|------|------|---|------|------|------|----|-----|-------|------|---|-----|-------|------|---|-----|-------|------|---|
| Ramp °C/min | Setpoint °C | Window °C | Soak sec | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | -99.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9999 | -70.0 | ±2.0 | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | -55.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | -25.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9999 | 0.0 | ±2.0 | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 25.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 40.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 55.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9999 | 70.0 | ±2.0 | 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 125.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 150.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0.0 | 200.0 | ±2.0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| SETUP # TO LOAD | | | | | |
|-----------------|---|---|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |

| |
|----------------------------|
| Number of cycles to do: 10 |
|----------------------------|

Use the arrow keys to choose the desired setup. Press "ENTER" to load the setup, or "ESC" to retain the current setup.



Figure 3-20. V.S. Load Setup Screen

3-6 Top Menu Selections

3-6.1 Top Menu Screen

The *Top Menu Screen* (see Figure 3-21) can be accessed either from the Operator Mode or the Variable Setpoint Mode. Selections on this screen allow you to switch quickly the operating mode, to access auxiliary function screens, and to shutdown the system.



Figure 3-21. Top Menu Screen

3-6.2 Explanation of Screen Selections

The selections on the *Top Menu Screen* (Figure 3-21) are summarized below. Press the up/down arrow keys to highlight a selection and then press the [ENTER] key to activate that selection.

Run in Variable Setpoint mode (accesses *V.S. Main Screen*)

- > Run twelve-setpoint temperature control, ramp/cycle.

Run in Operator mode (accesses the *Operator Main Screen*)

- > Run temperature control, hot, ambient, cold, set soak time.

Change to heat-only mode /Change to heat/cool mode (pressing [ENTER] key toggles between the states)

In the "Change to heat-only mode":

- > Disable the compressor to run hot only.

In the "Change to heat/cool mode":

- > Enable the compressor to permit running both cold and hot.

Setup temperature parameters (accesses *V.S. Change Setup Screen*)

- > View, change, load, and save temperature & cycling parameters.

Configuration (accesses *System Configuration Screen*)

- > Change interface parameters, password, screen saver, time & date.

Calibration (accesses *Calibration Main Screen* – refer to Subsection 5-6 for procedure)

- > Calibrate one or more of the system temperature sensors.

Defrost (accesses *Defrost Cycle Screen* – refer to Subsection 5-7 for procedure)

- > Defrost the refrigeration system.

Shutdown

- > Shut the system down. (ThermoStream operating program)

Return to DOS

- > Shut the system down and return to DOS (TP04010A processor's root prompt)

3-7 System Configuration

3-7.1 System Configuration Screen

The *System Configuration Screen*, Figure 3-22, lists the programmable options that affect the overall system operation. These options will rarely need to be changed once the system is initially set up. Note that the system software version (User I/O version) is identified at the top of the screen.

Note that the *System Configuration Screen* is only accessible from the *Top Menu Screen* (refer to Figure 3-21).

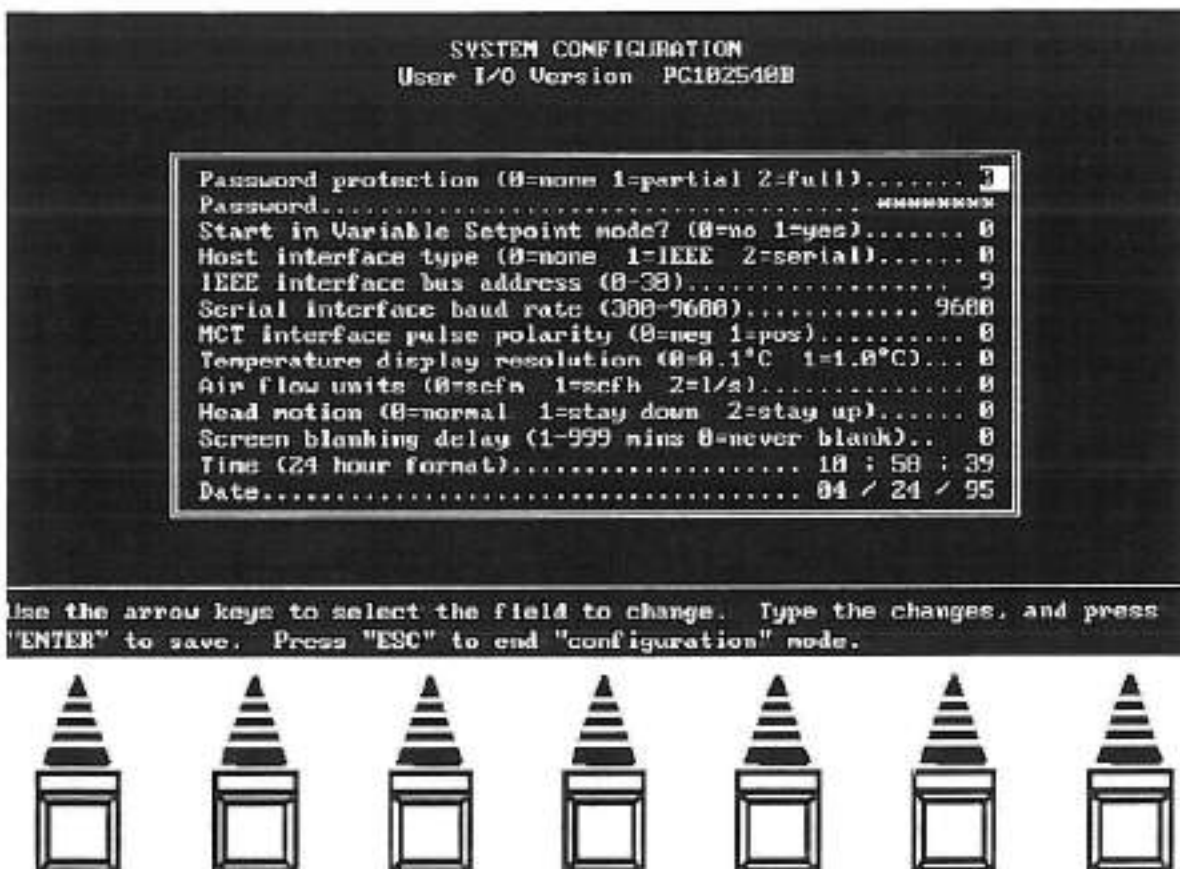


Figure 3-22. System Configuration Screen

3-7.2 Explanation of Configuration Options

Password protection

The TP04010A System provides three levels of security by using a password system. The operator will be prompted for a password when attempting to access a secured function.

- 0 = none:** No security present, no password required. The operator has full access to all system functions. (Default setting when system is shipped.)
- 1 = partial:** Limited security, allows the operator to startup and shutdown the system (Par. 3-3) and to perform Operator Mode testing (Par. 3-4). The operator can change setup parameters temporarily through the **Set** function of the *Operator Main Screen*, but cannot load or save test setup files. The operator cannot change from Operator Mode to Variable Setpoint Mode, or vice versa.
- 2 = full:** Highest security, allows the operator to only operate the system on the *Operator Main Screen*. The operator cannot change any parameters nor load or save test setup files. However, any temperature setpoint on that screen can still be selected.

Password:

The password can be from 1 to 8 characters, alphanumeric string in any combination or order. Any password containing alpha characters requires the use of an external keyboard. Since the numeric keypad of the TP04010A does not have the ability to generate alpha characters, using them in a password can enhance the security level of the system. Simply remove the external keyboard once the password has been entered and return it when password use is required. (The system is shipped with the factory entered password "12345678".)

Start in Variable Setpoint mode?

- 0 = no:** System starts up in Operator Mode (default mode when shipped).
- 1 = yes:** System starts up in Variable Setpoint Mode.

Host interface type

Allows you to select the type of system communication for interfacing to an external host controller.

- 0 = none:** Used for no interface. (Default setting when system is shipped.)
- 1 = IEEE:** Select for IEEE-488 interfacing, see Section 4.
- 2 = serial:** Select for the optional RS-232C interfacing, see Section 4.

IEEE interface bus address

Allows the system address to be set from 0 to 30 as specified by the IEEE-488 standard. (The system is shipped with the factory entered address "9".)

Serial interface baud rate

Allows the system baud rate to be set from 300 to 9600. Accepted values are: 300, 600, 1200, 2400, 4800, or 9600. (The system is shipped with the factory set rate at "9600".) The number of data bits is fixed at 8 with no parity bit and one stop bit.

MCT interface pulse polarity

Allows you to change the polarity of the Start-Test, End-of-Test, and Stop-on-First-Fail signals given and received by the TP04010A.

0 = neg: Sets all signals to negative polarity, see Section 4. (Default setting when shipped.)

1 = pos: Sets all signals to positive polarity, see Section 4.

Temperature display resolution

Allows you to change the displayed temperature window in the upper left-hand area of the menu screens. This does not affect the temperature control accuracy of the system in any way.

0 = 0.1C: Sets the display to show a typical temperature as 25.0. (Default setting when shipped.)

1 = 1.0C: Sets the display to show a typical temperature as 25.

Air flow units

Allows you to change the displayed flow units that appear in the system status window in the upper right-hand area of the menu and test setup screens.

0 = scfm: Sets the display to show flow units in standard cubic feet per minute. (Default setting when shipped.)



1 = scfh: Sets the display to show flow units in standard cubic feet per hour.



2 = l/s: Sets the display to show flow units in liters per second.

Head motion

Allows you to select the type of thermal head vertical travel during the test cycle.

0 = normal: Causes the thermal head to automatically travel down at start of test and travel up at end of test (default setting when shipped). At end of test, the ThermoStream air flow stops (head up).

1 = stay down: Causes the thermal head to stay down automatically at all times, and disables the **HEAD**  and  button. At end of test, the ThermoStream air flow stops.

2 = stay up: Causes the thermal head to stay up automatically at all times, and disables the **HEAD**  and  button. At end of test, the ThermoStream air flow stops.

Screen blanking delay

Allows you to select the time delay before the current screen display will be replaced with a Temptronic Corporation logo if there is no operator activity. Once the screen is blanked, pressing any key will restore the current display. The screen will not blank when the system is displaying a help screen, the System Configuration Screen, the Defrost Screen, or a calibration screen.

1-999 mins: The screen will be blanked after the specified delay within this time range.

0=never blank: Screen blanking is disabled. (Default setting when shipped.)

Time (24 hour format)

Allows you to set the system clock to the current time. There are three fields: hours, minutes, and seconds. Each field is changed separately.

Date

Allows you to set the system calendar to the current date. There are three fields: month, day, and year. Each field is changed separately.

3-8 Error Messages

Major system faults are identified to the operator on the Operator Control Module display as error messages. For each error displayed, you may access a help screen which provides the corrective action to be performed to correct the error.

The following is a list of the error messages and the recommended corrective action. Errors are listed in their order of software priority. Each error successively higher on the list takes precedence and is displayed over those below it in the event of multiple errors.

Internal Error

The TP04010A hardware is not operating properly. Turn the system off, wait one minute, and turn it on again. If the internal error persists, turn the system off and refer to the system service manual or contact the Temptronic Service Department. This error message is shown in Figure 3-23.

Overheat

The airstream thermocouple sensor has detected an over temperature condition and the heaters have been automatically shut down. Press the [ESC] key to try to clear the error condition. If the overheat error persists, turn the TP04010A off, wait one minute, and turn it on again. If the error still reoccurs, turn the system off and refer to the system service manual or contact the Temptronic Service Department.

Low Input Air Pressure

The air pressure at the air inlet to the TP04010A is less than the minimum required for proper system operation. Check the air input pressure at its source and bring to within specified range.

This error will clear automatically when the problem is corrected. If the error persists after attempting the corrective measures described above, turn the system off and refer to the system service manual or contact the Temptronic Service Department.

Air Open Loop

The airstream temperature is not properly being read and/or controlled by the TP04010A. Press the [ESC] key to try to clear the error condition. If the air open loop error persists, turn the TP04010A off, wait one minute, and turn it on again. If the error still reoccurs, turn the system off and refer to the system service manual or contact the Temptronic Service Department.

Low Flow

The main air flow rate is less than 2 scfm. Try to adjust the front-panel FLOW control for a higher reading on the *Operator Main Screen* or *V/S Main Screen*. If a flow rate above 4 scfm cannot be obtained, this error may indicate the need to defrost the system. Please refer to Section 5 for further information and the defrosting procedure.

This error will clear automatically when the problem is corrected.

NOTE: When alternate air flow units have been selected and set on the *System Configuration Screen* those units (in equivalent flow) should be applied to the above statement. Simply convert the values above into the alternate air flow units.

No DUT Sensor Selected

DUT control mode has been selected, but no DUT sensor type has been chosen on the *Operator Change Setup Screen* or *V.S. Change Setup Screen*. Return to the setup screen and either select a DUT sensor type, or turn off DUT control mode. This error message is shown in Figure 3-24

DUT Open Loop

Temperature is not properly being read by the DUT sensor. Check to see that the sensor selected on the *Operator Change Setup Screen* or *V.S. Change Setup Screen* is plugged into the TP04010A, and is not shorted or open. This error will clear automatically when the problem is corrected. If the error persists after attempting the corrective measures described above, turn the system off and refer to the system service manual or contact the Temptronic Service Department.

Flow Sensor Hardware Error

The air flow sensor in the TP04010A is defective. Turn the system off and refer to the system service manual or contact the Temptronic Service Department.

System Error

An unspecified problem exists with the TP04010A. Press the [ESC] key to try to clear the error condition. If the system error persists, turn the TP04010A off, wait one minute, and turn it on again. If the error reoccurs, turn the system off and refer to the system service manual or contact the Temptronic Service Department.

Out of Range

The current temperature setpoint is above the upper air temperature limit setting, or below the lower air temperature limit setting. Either change the value of the current setpoint to a temperature within those limits, or go to the *Test Setup Screen* and change the upper or lower air temperature limit values. This error will clear automatically when the problem is corrected.

Improper Software Version

The version of software currently in the system is not compatible with the version of control board read-only memory (ROM). Restart the system, and if the problem persists, contact the Temptronic Service Department.

Test Failed

The tester has signaled the TP04010A that the device under test has failed to pass a test, and so temperature cycling has stopped before completion. Press the [ESC] or [ENTER] key to permit the TP04010A to resume normal operation.

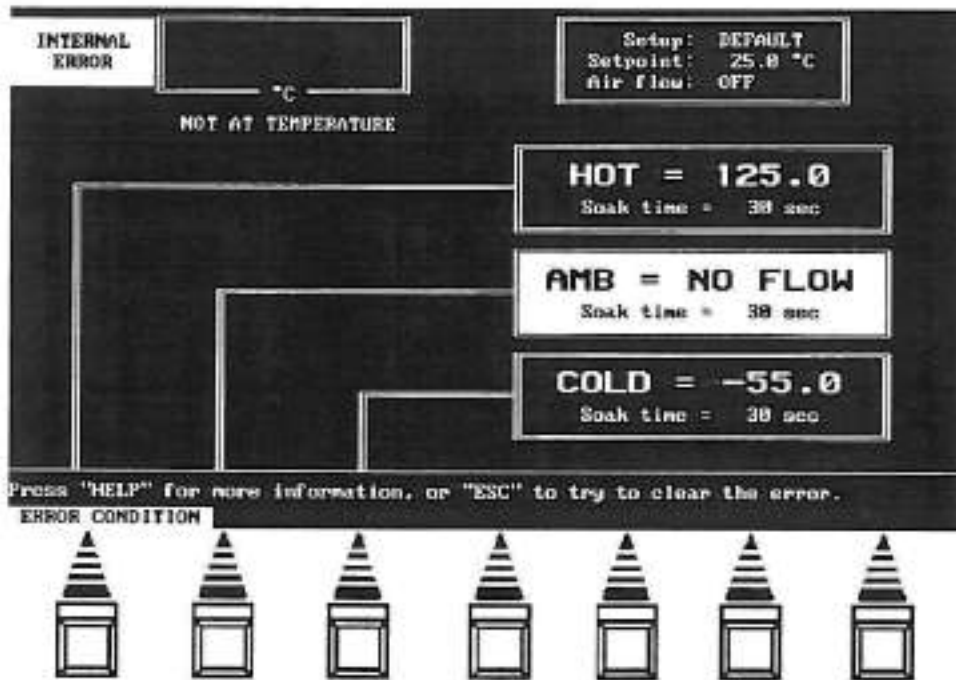


Figure 3-23. Internal Error

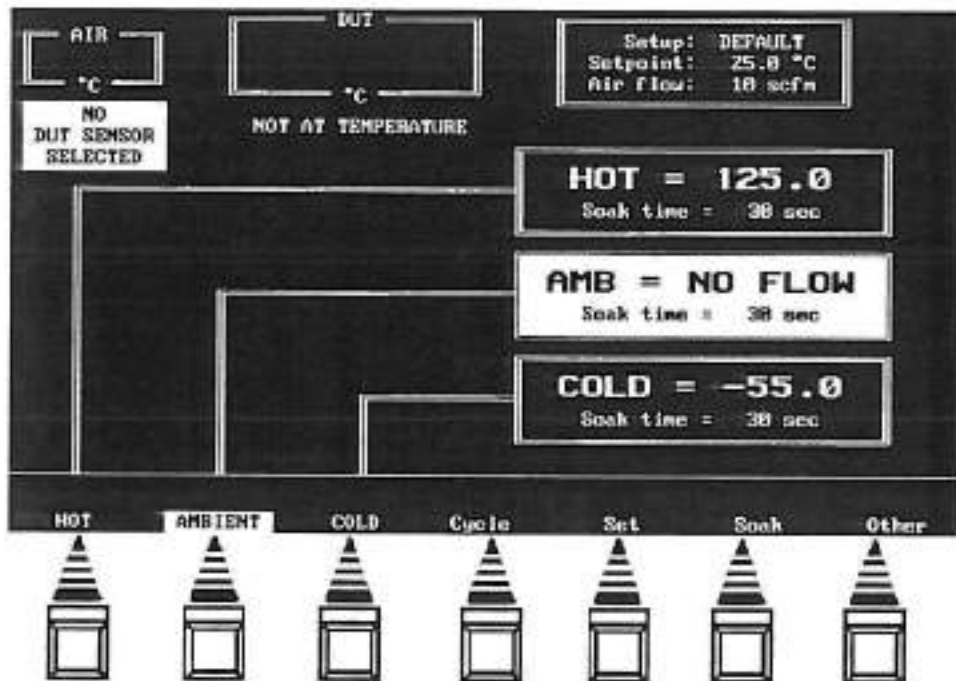


Figure 3-24. No DUT Sensor Selected Error

Section 4

Remote Interfaces

4-1 Introduction

The TP04010A System provides three I/Os for three different communication interfaces: the MCT Standard Interface, the IEEE-488 (Parallel) Interface, and the optional RS232C (Serial) Interface. The system must be configured by the operator for the specific interface to be used. This is done using the *System Configuration Screen* (see Figure 4-1), which is accessed from the *System Startup Screen* (refer to Section 3). The system warns the operator when the selected interface hardware is not present.

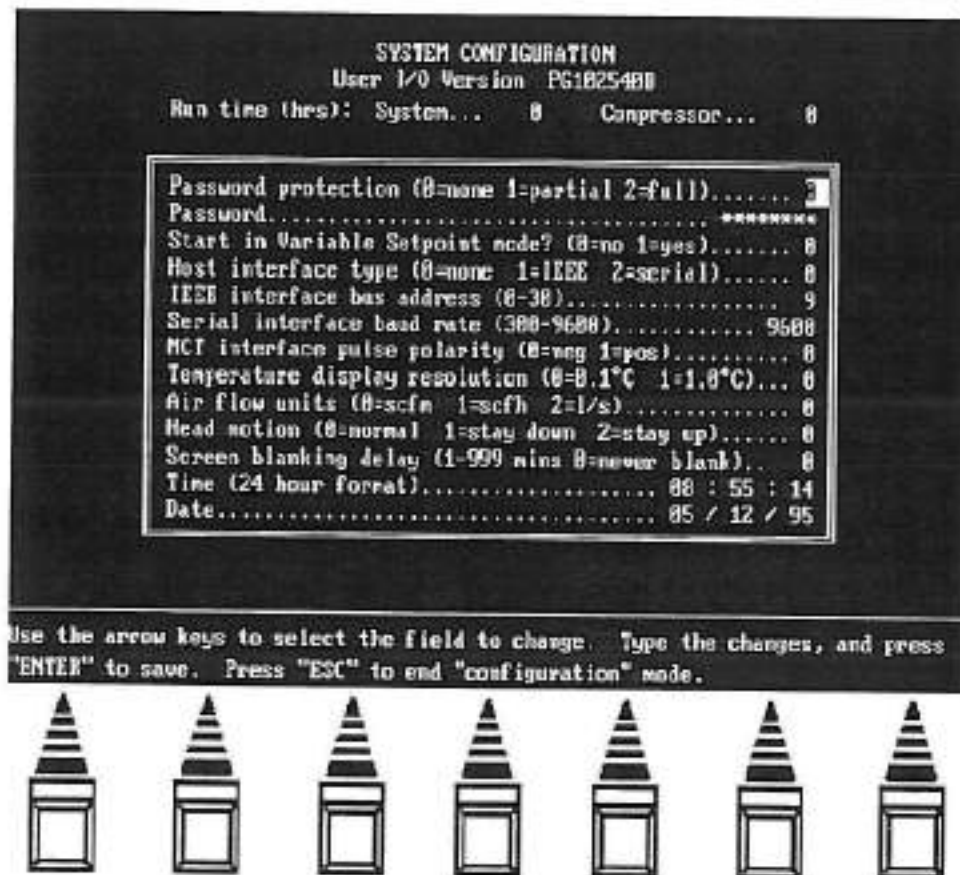


Figure 4-1. System Configuration Screen

Familiarization with the specifications and standards of the chosen interface is required for a successful interface. The following sections provide only that information specific to the TP04010A System and identifies which commands are supported.

NOTE: The system must be in an operating mode capable of temperature control before remote control can be exercised.

4-2 MCT Interface

Normally, the MCT Interface (Start Test [ST]/ End of Test [EOT]) is two solid-state relay (SSR) closures: one closure to start test and one closure to end test. see Figure 4-2. The input and output pulse polarity is settable on the *System Configuration Screen*, refer to Figure 4-1. The connector pinout is provided in Table 4-1.

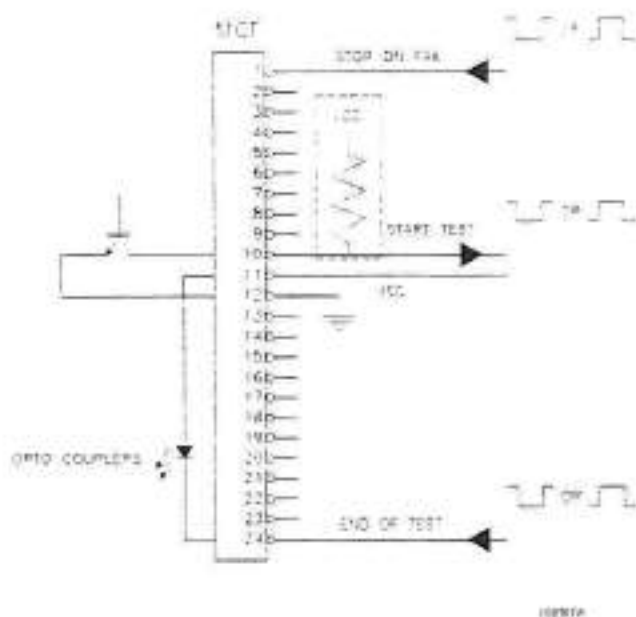


Figure 4-2. MCT Interface

Table 4-1. MCT Interface Connector Pin Designations.

| Pin No. | Signals | Functions |
|---------|-------------------------------|--|
| 1 | STOP ON FAIL (test failed) | An input pulse as short as 11 msec received by the TP04010A to signal the system (when in cycle mode) to abort temperature cycling and display "TEST FAILED" |
| 10 | START TEST (ready to test) | An output (120 to 150 msec pulse) from the TP04010A to signal AT TEMP condition |
| 11 | TESTER VCC | +5 TO +12 Vdc |
| 12 | TESTER GROUND | Ground |
| 24 | END OF TEST | An input (as short as 11 msec pulse) to the TP04010A which when in cycle mode will shorten the maximum set test time |

The part number for the connector supplied on the system I/O panel is TRW57-40240. the user must obtain the mating connector.

4-3 IEEE-488/RS232C Host Interface

4-3.1 Parallel/Serial Programming

When a TP04010A System is to be integrated into an automated test station the following system configuration selections must be properly set to permit control by the host.

- > Host interface type must be set to either "1" or "2" depending on the interface to be used.
- > Depending on the interface selected, either the IEEE interface bus address or the serial interface baud rate must be properly set.

Use the system's Operator Control Module (OCM) to access the *System Configuration Screen* from the *System Startup Screen*.

The Temptronic TP04010A ThermoStream System host interface instruction set was designed to be in substantial compliance with IEEE Standard 488.2-1987 ("IEEE Standard Codes, Formats, Protocols, and Common Commands For Use with ANSI/IEEE Std.488.1-1987 IEEE Standard Digital Interface for Programmable Instrumentation"). Please refer to that standard for general programming information.

With a few exceptions the IEEE-488 and RS32C versions of the TP04010A host interface use the same instructions. Several instructions (see Para 4-5.3) were added to the RS232C version to compensate for the lack of dedicated hardware lines. The RS232C version does not transmit until it has a complete response message (a string terminated by a line feed) to send to the host. (The interface output buffer is 250 bytes in size.) The "message available" bit in the status byte is always sent as 0 when using the serial interface.

Remote mode operation blocks out any screen selections on the *Main Menu Screen* (refer to Section 3) and substitutes the word "REMOTE". The help tip line on that screen changes to "The system is currently in remote mode. Press "ESC" to return to local". If the host locks out local control, the help tip line changes to "The system is currently in remote mode. Local controls are locked out"

Copies of IEEE-488 and RS232C demo programs (idemo.c and sdemo.c) are provided in Appendixes A and B of this manual. A 3.5-inch, 1.44 MB, standard MSDOS format disk containing these demo programs is provided in the back cover of this manual.

4-3.2 Device Specific Commands and Queries

| | |
|-------|--|
| ADMD | Set air-to-DUT maximum difference ADMD nnn -- where nnn is 10 - 300 degrees C in 1 degree increments. |
| ADMD? | Read air-to-DUT maximum difference |
| AUXC? | Read auxiliary conditions register bit 9 -- ramp mode 7 -- reserved 6 -- ready for operation = 1, startup sequence = 0 5 -- flow on 4 -- DUT mode 3 -- heat only mode = 1, compressor on = 0 2 -- head up 1 -- reserved 0 -- reserved |
| CLER | Clear device-specific (reported by EROR?) errors |
| COOL | Turn the compressor on or off COOL 1 -- turns the compressor on COOL 0 -- turns the compressor off |
| CYCC | Set ramp/soak cycle count CYCC nnnn -- where nnnn is the # (1 - 9999) of cycles to do |
| CYCC? | Read number of ramp/soak cycles to do |
| CYCL | Start/stop ramp/soak cycling CYCL 1 -- starts CYCL 0 -- stops NOTE: When all cycles have been completed or when cycling was stopped on failure, it is necessary to send a CYCL 0 command to reset the system. |
| CYCL? | Ramp/soak cycle number (current value if cycling, last value if not) |
| DSNS | Set DUT sensor type DSNS n -- where n is 0-2 0 -- no DUT sensor connected 1 -- type T thermocouple 2 -- type K thermocouple |
| DSNS? | Read DUT sensor type |
| DUTC | Set device thermal constant DUTC nnn -- where nnn is nominally 100 but can range from 20-500. A higher number corresponds to a higher mass device. |

| | |
|-------|---|
| DUTC? | Read device thermal constant |
| DUTM | Turn DUT mode on or off DUTM 0 -- off (air control) DUTM 1 -- on (actual DUT control) |
| EROR? | Read device-specific error register (16 bits) bit 14 -- no DUT sensor selected 13 -- improper software version 9 -- flow sensor hardware error 8 -- DUT open loop 7 -- internal error 4 -- low input air pressure 3 -- low flow 2 -- setpoint out of range 1 -- air open loop 0 -- overheat |
| FLOW | Turn flow on or off FLOW 1 -- on FLOW 0 -- off |
| FLRL? | Read air flow in liters/second. |
| FLWR? | Read air flow rate in scfm |
| HEAD | Raise or lower the test head (same as STND) HEAD 1 -- put head down HEAD 0 -- put head up |
| LLIM | Set lower air temperature limit LLIM nnn -- where nnn is -20.5 to +25 degrees C. |
| LLIM? | Read lower air temperature limit |
| LRNM | Turn DUT learning on or off LRNM 0 -- off (control DUT with current DUT control parameters) LRNM 1 -- on (adaptive DUT control) |
| NEXT | Step to the next setpoint during temperature cycling NOTE: Stepping will occur whether or not the device is at temperature. NEXT will cause an error if the system is not in cycling mode. |
| RAMP | Set ramp rate for the setpoint currently in use RAMP is a 4 character entry -- where the rate is nn.n from 0 to 99.9 in 0.1 steps and nnn to nnnn from 100 to 9999 in whole increment steps. |
| RAMP? | Read ramp rate |

| | |
|-------|--|
| RMPC | Enter/leave ramp/cycle mode RMPC 1 -- enter ramp/cycle mode RMPC 0 -- leave ramp/cycle mode |
| RMPS | Same as RMPC |
| RSTO | Reset device-specific functions. The TP04010A is forced to the <i>Main Menu Screen</i> , with setpoint number 1 (ambient) active. Any device specific errors are reset. The current configuration, setup, and setpoint information are left unchanged. RSTO will not reset an improper software version error, and will not force the system out of the startup sequence. NOTE: The fourth character is the letter O, not a zero. This command may take up to 5 seconds to complete. |
| SETD? | Read dynamic temperature setpoint (changes during a ramp) |
| SETN | Select a setpoint to be the current setpoint SETN n -- where n is 0 - 2. Setpoint #0 is HOT, #1 is AMBIENT, #2 is COLD |
| SETN? | Read current setpoint # |
| SETP | Set current setpoint's temperature SETP nnn.n - - where nnn.n is the temperature (negative sign required for negative temperatures) |
| SETP? | Read current temperature setpoint |
| SFIL | Load setup SFIL n, where n is 1 - 12. NOTE: This command may take up to 5 seconds to complete. |
| SOAK | Set soak time for the setpoint currently in use SOAK nnnn -- where nnnn is 0 - 9999 seconds |
| SOAK? | Read soak time |
| STND | Raise or lower the test head (same as HEAD) STND 1 -- put head down STND 0 -- put head up |
| TECR? | Read temperature condition register (at temp/not at temp) bit 5 -- stopped cycling ("stop on fail" signal was received) 4 -- end of all cycles 2 -- end of test when in ramp/cycle mode 1 -- not at temperature 0 -- at temperature (soak time has elapsed) |

- TEMP?** Read main temperature
Returns air temperature when in air control mode or DUT temperature when in DUT mode
NOTE: A returned value greater than 400 degrees C indicates an invalid temperature reading.
- TESE** Set temperature event status enable (mask) register
TESE nnn -- where nnn is 0-255
- TESE?** Read temperature event status enable (mask) register
- TESR?** Read temperature event status register
bit 7 -- unexpected shutdown (disk full, etc)
5 -- stopped cycling ("stop on fail" signal was received)
4 -- end of all cycles
3 -- end of one cycle
2 -- end of test when in ramp/cycle mode
1 -- not at temperature
0 -- at temperature (soak time has elapsed)
NOTE: The above bits are latched, and are automatically cleared when the register is read.
- TMPA?** Read air temperature
Always returns air temperature, whether in the air control mode or DUT mode.
NOTE: A returned value greater than 400 degrees C indicates an invalid temperature reading.
- TMPD?** Read actual DUT temperature
Always returns actual DUT temperature, whether in the air control mode or DUT mode.
NOTE: A returned value greater than 400 degrees C indicates an invalid temperature reading.
- TTIM** Set maximum test time (if no end of test input or NEXT command is received)
TTIM nnnn -- where nnnn is 0-9999 seconds
- TTIM?** Read maximum test time
- ULIM** Set upper air temperature limit
ULIM nnn -- where nnn is +25 to +225 degrees C
- ULIM?** Read upper air temperature limit

- WHAT?** Read current system menu (what the system is doing)
- bit 5 = Mode change setpoints
 - 9 = all cycles completed
 - 10 = Main Menu Screen
 - 11 = no cycles performed
 - 14 = test setup
 - 16 = stopped cycling on failure (DUT failed)
 - 25 = Menu Other Screen
 - 26 = soak time change
 - 28 = cycle
 - 60 = improper software version
 - 63 = startup sequence
- WNDW** Set current setpoint's temperature window command
WNDW n -- where n is the window 0 1-9.9 degrees C
- WNDW?** Read current setpoint's temperature window

4-4 IEEE-488 Interface

4-4.1 Parallel Bus Interface Parameters

The IEEE-488 interface requires that an address be assigned for each device on the bus. The address is settable on the *System Configuration Screen*, refer to Figure 4-1.

4-4.2 Mandatory IEEE-488.2 Common Commands and Queries

- *CLS Clear status (SESR, TESR) registers
- *ESE Set standard event status enable (mask) register
 *ESE nnn -- where nnn is 0 - 255
- *ESE? Read standard event status enable (mask) register
- *ESR? Read standard event status register
 bit 7 -- power on -- NOT USED
 6 -- user request -- NOT USED
 5 -- command error (cme)
 4 -- execution error (exe)
 3 -- device dependent error (dde)
 2 -- query error (qye)
 1 -- request control -- NOT USED
 0 -- operation complete -- NOT IMPLEMENTED
 NOTE: The above bits are latched and are automatically cleared when the register TP04010A, 0, PG102540A is read.
- *IDN? Read product identification information (Manufacturer, model, serial number, software version). Returns TEMPTRONIC, TP04010A, 0, PG102540A. The last of the fields in the return string will vary with the particular software version.
- *RST Reset device-specific functions. Any device-specific errors are reset. The software configuration and the currently chosen test setup (including setpoints) are loaded. *RST will not reset an improper software version error, and will not force the system out of the startup sequence. This command may take up to 5 seconds to complete.
- *SRE Set service request enable (mask) register
 *SRE nnn -- where nnn is 0 - 255.
- *SRE? Read service request enable (mask) register

- *STB? Read status byte
 bit 7 -- ready
 6 -- master status summary bit
 5 -- standard event (esr) summary bit
 4 -- message available (IEEE only, serial always 0)
 3 -- temperature event (tesr) summary bit
 2 -- device - specific error (error) summary bit
 1 -- not used (always 0)
 0 -- "
- *TST? Self test (dummy, always returns "passed")

NOT IMPLEMENTED -- *OPC, *OPC?, *WAI

See IEEE Standard 488.2-1987 for more information about the commands listed above.

4-5 RS232C Interface (optional)

4-5.1 Serial Interface Connector

The system I/O panel provides a standard DB25P (25-pin male) connector, see Table 4-2 for pinout.

Table 4-2. Interface Connector Pin Designations.

| Pin Number | Function/Signal Level |
|------------|---|
| 1 | Chassis ground |
| 2 | Serial data out |
| 3 | Serial data in |
| 4 | RTS -- always high |
| 5 | CTS -- must be high (tie to pin 4) |
| 6 | DSR -- handshaking input. The ThermoStream will stop sending output when DSR goes low, and resume when it returns high. |
| 7 | Signal ground |
| 20 | DTR -- handshaking output. Goes low when the ThermoStream input buffer has filled and it CANNOT accept new commands |

If the host computer is incapable of hardware handshaking (using lines 6 and 20), tie pin 6 of the host to pin 4.

NOTE: Ensure not to overflow the 250 byte input buffer when using this configuration.

4-5.2 Serial Interface Parameters

A number of parameters are associated with the serial interface. see Table 4-3. The data bits, parity, and stop bits are fixed and the baud rate is settable on the *System Configuration Screen*, refer to Figure 4-1

Table 4-3. Communication Parameters.

| Parameters | Definitions |
|------------|--|
| Baud Rate | 300, 600, 1200, 2400, 4800, or 9600 baud |
| Data Bits | Fixed at 8 |
| Parity | Fixed at no parity |
| Stop Bits | Fixed at one (1) |

4-5.3 Serial Interface Special Commands

- %GL Go into local mode
- %LL Local lockout
- %RM Go into remote mode
- %S? Read serial poll
- ! Device clear. The TP04010A will echo back the !. The system will automatically perform a device clear (and send a !) if it detects a hardware communications error.
- ^ The TP04010A sends a ^ (caret mark) as a service request (SRQ) indicator. The ^ is sent as soon as the output buffer is empty, and is not followed by a semicolon, carriage return, line feed, or other delimiter. A ^ will never otherwise be found in a response string.



Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

Section 5

Operator Maintenance

5-1 Introduction

The TP04010A ThermoStream System has been designed and manufactured to withstand the rigors of constant use in most manufacturing environments. To ensure many hours of trouble free service from your system, a minimum of routine maintenance consistent with that for similar equipment should be performed on a regular basis.

This section contains only those procedures considered appropriate for routine maintenance.

Refer to the Warranty, a copy is supplied near the end of this manual, concerning system service support and direct any questions to:

Temptronic Service Department

Phone: 1-800-558-5080

Telex: 211-938; answer back SERVSUR

Fax: 1-617-969-2475

5-2 Inspection and Cleaning

5-2.1 *Inspection*

Weekly inspection is recommended for frequently used systems to ensure normal operation with no deterioration in performance.

- > Inspect exposed hoses and cables for cuts and or abrasions; reroute and repair as required.
- > Inspect for free air flow at all ventilated panel areas; remove any restrictions.
- > Inspect for open liquid containers resting on the system; remove when found. The system is not waterproof.

5-2.2 *Cleaning*

- > Keep the system clean for reliable operation.
- > Clean the display using any commercially available CRT cleaner and a soft lint-free cloth. Do not use any abrasive cleaner or paper towel.

5-3 Maintenance Log

The system records its total run time and compressor run time. Total system run time is the accrued time the system has been on. This includes the time when the system is in either the operational or standby modes. Compressor run time is the accrued time that the compressor has been on. Each record will log up to 65,355 hours before resetting to 0.

To access this record:

1. Return the system to the *System Startup Screen*.
2. Select **Configuration** with the function key to access the *System Configuration Screen* (see Figure 5-1).
3. Press the [ENTER] key and the right most function key so the run time line will show.

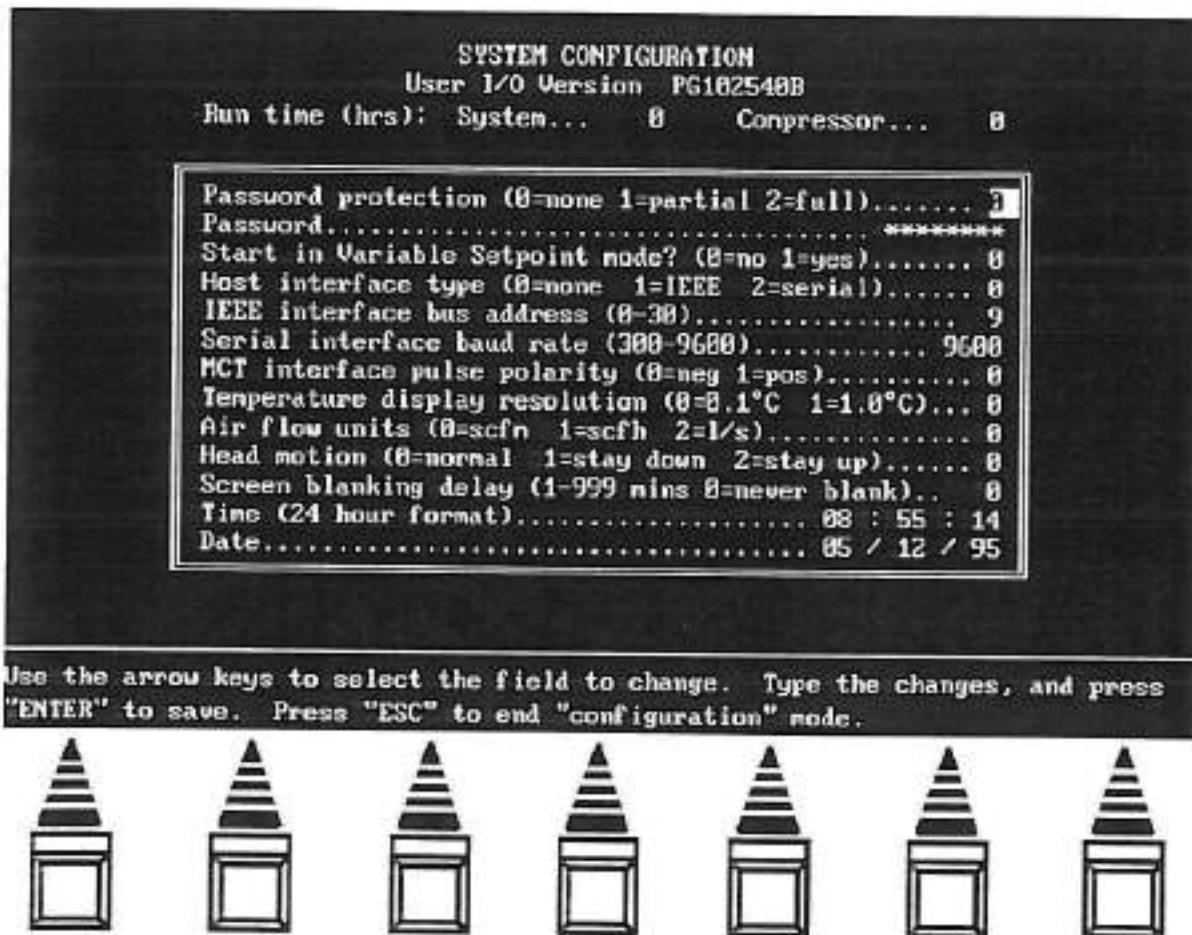


Figure 5-1. System Configuration Screen Showing Run Time

The following log (Table 5-1) is provided as a suggested preventive maintenance schedule. Space has been provided for additional items found useful from actual system performance and experience for each installation.

Table 5-1. Maintenance Log

| Maintenance Item | Run Time (thousands of hours) | | | | | | | |
|--------------------------|-------------------------------|-----|-----|-----|-----|-----|-----|-----|
| | 0.5 | 1.0 | 1.5 | 2.0 | 2.5 | 3.0 | 3.5 | 4.0 |
| Air calibration | | [] | | [] | | [] | | [] |
| Back-up batteries | | | | # | | | | # |
| Pneumatic module filters | [] | # | [] | # | [] | # | [] | # |
| Air dryer filter(s) | [] | # | [] | # | [] | # | [] | # |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

[] Check and perform maintenance as required.

Check and replace as indicated.



Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

5-4 Back-Up Battery Replacement

The electronics module has two back-up batteries for its non volatile memory (see Figure 5-2). These batteries need to be replaced yearly. We recommend that the batteries be replaced prior to performing the temperature sensor calibration if the previous calibration was done 1 year earlier or before. Replacement batteries should be 1.5-volt, size AA alkaline batteries.

To replace the back-up batteries

1. Turn off the ac power to the TP04010A System.
2. Gain access to the electronics module by removing the frame module front-panel (see Front Panel Removal in Par. 5-5).
3. Turn on the ac power to the TP04010A System.
4. Remove both old batteries from their holder on the front of the electronics module.
5. Insert two new batteries, being careful to observe the directions of polarity as marked in the holder.

Note: If batteries are replaced with the power off, calibration values will be lost and the system will have to be re-calibrated.

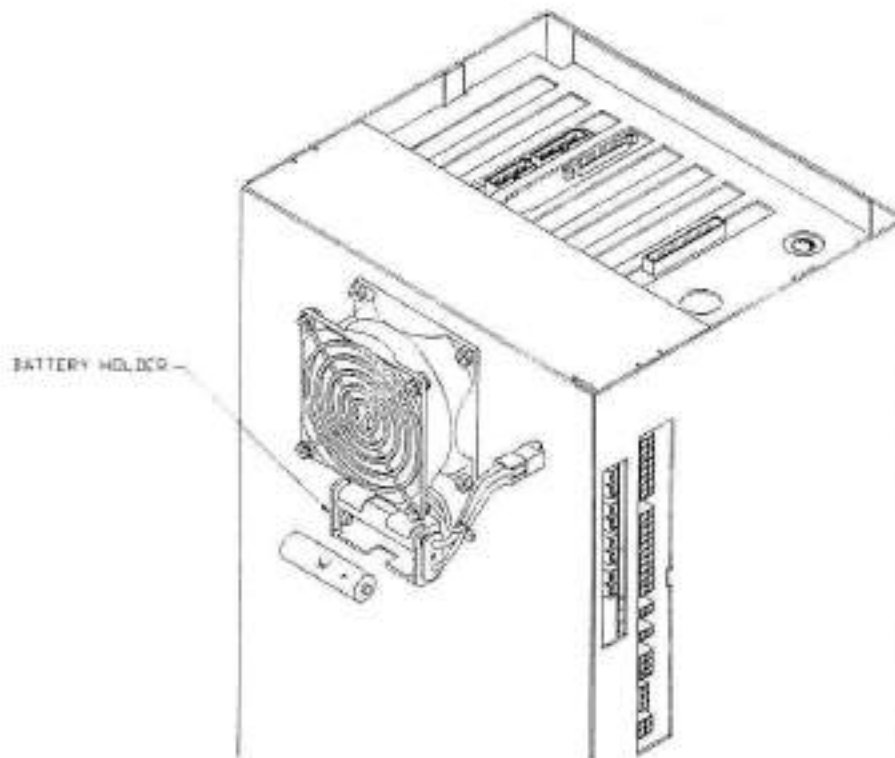


Figure 5-2. Back-Up Batteries Replacement

5-5 Air Path Maintenance

Two air filters on the front of the pneumatics module and the post filter(s) on the air dryer module should be checked monthly to see if they are dirty or filled with liquids and need to be replaced. They should be replaced at least once every 6 months. Also, replace the exhaust muffler on the air dryer module if the muffler becomes cracked or ineffective.

WARNING: Make sure you turn off the ac power and the compressed air to the TP04010A System before attempting to perform the following procedure. Do not perform this procedure unless you are qualified to do so.

5-5.1 Front Panel Removal

To remove the front panel from off the frame module,

1. Remove the four button-head screws securing the front panel, one at each corner.
2. Swing out the bottom of the front panel and lift free of the frame.
3. To reinstall the front panel, reverse the order of disassembly.

5-5.2 Air Filter Servicing

To check or replace the air filters,

1. Turn off the ac power to the TP04010A System.
2. Turn off the system's air pressure supply and disconnect the supply line from the AIR INPUT fitting on the power and air input panel at the rear of the frame module. Bleed all air from the system (turn on ac power to the TP04010A just long enough to exhaust air in the system).
3. Remove the front panel if servicing the pneumatics module air filters (see previous paragraph). The post filter(s) on the air dryer module is accessible on the output of the dryer.
4. Press the tab button at the top front of the perforated regulator bowl guard (see Figure 5-3), turn one quarter turn counterclockwise, pull down and away from its cap mount, and remove the guard.
5. Unscrew the polycarbonate bowl and remove it and its seal from the cap mount.
6. Unscrew the filter element and remove it and its seal from the cap mount.
7. Inspect the filter element for dust particles, oil contamination, and any broken structure. Inspect the polycarbonate bowl for any cracks or deformations.
8. Replace the filter element or polycarbonate bowl as needed.
9. Reassemble the air filter in the reverse order of disassembly (make sure seals are in place)

10. Install the system front panel if removed.
11. Reinstall the air pressure line at the rear of the frame module, and turn on the air source.
12. Turn on the ac power to the TP04010A System.

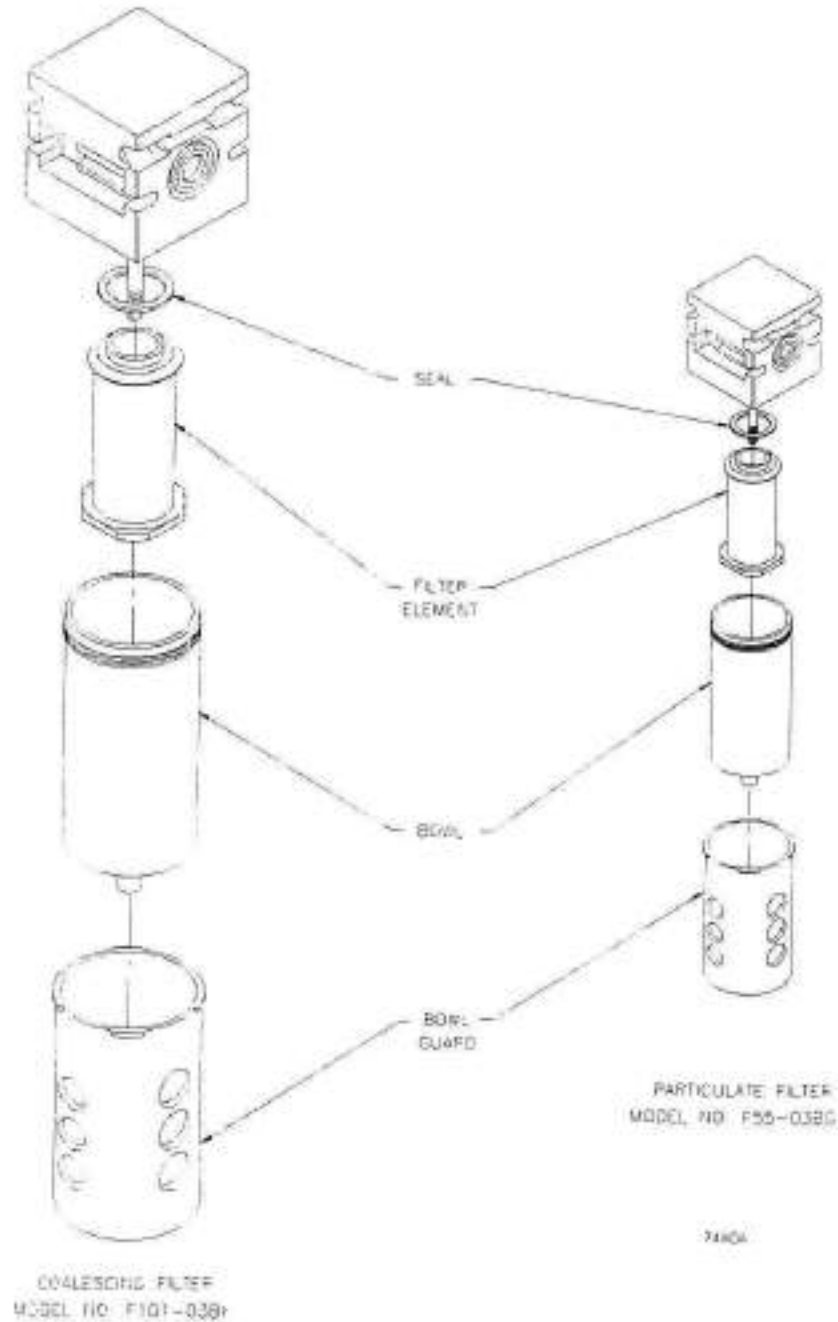


Figure 5-3. Air Filter Element Replacement

5-5.3 Exhaust Muffler Replacement

To replace a faulty exhaust muffler in the air dryer module,

1. Turn off the ac power to the TP04010A System.
2. Turn off the system's air pressure supply and disconnect the supply line from the AIR INPUT fitting on the power and air input panel at the rear of the frame module. Bleed all air from the system (turn on ac power to the TP04010A just long enough to exhaust air in the system).
3. Using a screwdriver, remove four corner screws in the air dryer cover and remove the cover.
4. Locate the faulty exhaust muffler, unscrew by hand, and discard the muffler (see Figure 5-4).
5. Screw in a new exhaust muffler and just hand tighten. Do not over tighten or the muffler will break.
6. Replace the cover on the air dryer module.
7. Reinstall the air pressure line at the rear of the frame module, and turn on the air source.
8. Turn on the ac power to the TP04010A System.

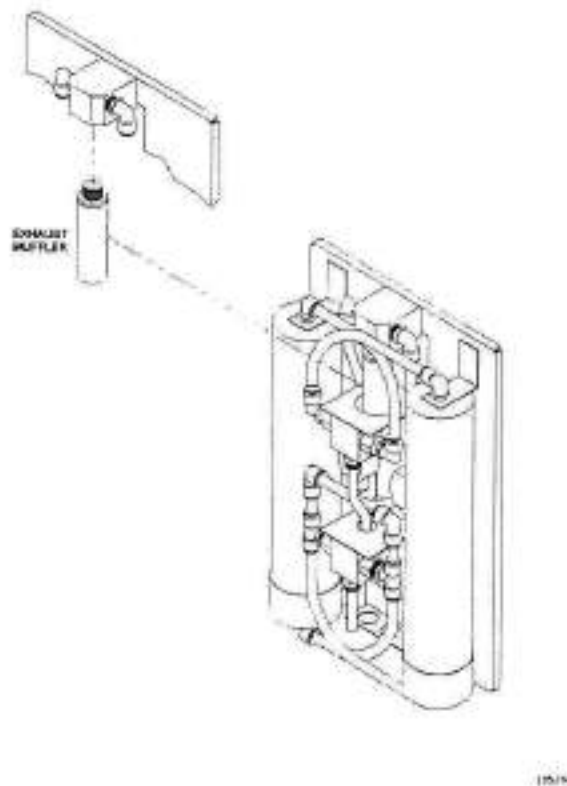


Figure 5-4. Standard Air Dryer Module (cover removed)

5-6 Calibration

5-6.1 Verification Procedure

Before the TP04010A System is calibrated, or at any time the system calibration is questioned, perform the verification procedure as outlined below. If periodic verifications show that the calibration performance has stabilized and is well within specifications, then the follow-up calibration procedure is not required. Readings of the verification performance must be recorded in Table 5-1 to evaluate the deviations for determining how often calibrations are needed.

To ensure an accurate system calibration, follow the proper temperature sensing techniques (contact the Temptronic Service Office if you have any questions). The verification procedure must be performed by a qualified technician. Verify that the instrumentation (external precision temperature monitor and thermocouple sensor) used to sense the working surface temperature is in calibration. This instrumentation must be calibrated against a primary or transfer standard. Verification is performed at two temperatures extremes. It can also be done at other temperatures (typical test setpoints) if desired.

Low Temperature Verification

1. Program the TP04010A System for air temperature sensing with an Air-to-DUT max temperature of 10 °C (refer to Par. 3-4.3 or 3-5.3) and an air flow rate of 8 SCFM.
2. Mount the thermocouple sensor of the precision temperature monitor in the air path near the main flow thermocouple in the output nozzle of the head module.
3. Start system operation at a cold setpoint and set the system temperature until the temperature monitor reads -60.00 °C for a 5-minute period.
4. Read the system setpoint temperature and record the value in Table 5-1. The system setpoint value must be within -60 ± 0.5 °C for acceptable performance.
5. Stop system operation for 15 minutes.
6. Program the TP04010A System for DUT sensing (type T or K used for testing) and an air flow rate of 8 SCFM.
7. Mount the thermocouple sensor of the temperature monitor on a DUT near the DUT temperature sensor.
8. Start system operation at a hot setpoint and set the system temperature until the temperature monitor reads -60.00 °C for a 5-minute period.
9. Read the system setpoint temperature and record the value in Table 5-1. The system setpoint value must be within -60 ± 0.5 °C for acceptable performance.
10. Stop system operation for 15 minutes.

High Temperature Verification

1. Program the TP04010A System for air temperature sensing with an Air-to-DUT max temperature of 10 °C (refer to Par. 3-4.3 or 3-5.3) and an air flow rate of 8 SCFM.
2. Mount the thermocouple sensor of the precision temperature monitor in the air path near the main flow thermocouple in the output nozzle of the head module.
3. Start system operation at a cold setpoint and set the system temperature until the temperature monitor reads +200.00 °C for a 5-minute period.
4. Read the system setpoint temperature and record the value in Table 5-1. The system setpoint value must be within $+200 \pm 0.5^\circ \text{C}$ for acceptable performance.
5. Stop system operation for 15 minutes.
6. Program the TP04010A System for DUT sensing (type T or K, whichever is used for testing) and an air flow rate of 8 SCFM.
7. Mount the thermocouple sensor of the temperature monitor on a DUT near the DUT temperature sensor.
8. Start system operation at a hot setpoint and set the system temperature until the temperature monitor reads +200.00 °C for a 5-minute period.
9. Read the system setpoint temperature and record the value in Table 5-1. The system setpoint value must be within $+200 \pm 0.5^\circ \text{C}$ for acceptable performance.
10. Stop system operation for 15 minutes.
11. If desired, repeat the above Low Temperature Verification and High Temperature Verification steps for other temperatures (typical test setpoints) and record their system setpoint temperatures in Table 5-1 for evaluation.

Table 5-1. Calibration Verification Table

| | Factory Verification | 1st Verification | 2nd Verification | 3rd Verification | 4th Verification |
|--------------------------------------|-------------------------|---------------------|---------------------|---------------------|---------------------|
| Low Temperature Verification | | | | | |
| Air Temp Sensor | °C | °C | °C | °C | °C |
| DUT Temp Sensor () | °C | °C | °C | °C | °C |
| High Temperature Verification | | | | | |
| Air Temp Sensor | °C | °C | °C | °C | °C |
| DUT Temp Sensor () | °C | °C | °C | °C | °C |
| Test Setpoint °C | | | | | |
| Air Temp Sensor | °C | °C | °C | °C | °C |
| DUT Temp Sensor () | °C | °C | °C | °C | °C |
| Test Setpoint °C | | | | | |
| Air Temp Sensor | °C | °C | °C | °C | °C |
| DUT Temp Sensor () | °C | °C | °C | °C | °C |
| Test Setpoint °C | | | | | |
| Air Temp Sensor | °C | °C | °C | °C | °C |
| DUT Temp Sensor () | °C | °C | °C | °C | °C |
| Date and initials | | | | | |
| | | | | | |

5-6.2 Calibration Procedure

Calibrate the system temperature sensors when required as determined by the Calibration Verification procedure (refer to Par. 5-6.1). Calibration of the sensors must be performed by a qualified technician. A calibration instrument (external precision temperature monitor and temperature sensor) is required for the temperature sensor calibration. This instrumentation must be calibrated against a primary or transfer standard.

Thermocouple Panel Access

Remove the system front panel to access the sensor circuit input connections on the thermocouple panel (right side of electronics module)

1. Remove the four button-head screws securing the front panel, one at each corner.
2. Swing out the bottom of the front panel and lift free of the frame.
3. To reinstall the front panel, reverse the order of disassembly.

Procedure

The calibration procedure is made easier by system semi-automatic operations directed by selections from a series of OCM screens. When **Calibration** is selected at the *Top Menu Screen* (Figure 3-21), the display switches to the *Calibration Main Screen* (see Figure 5-5). This screen starts a menu aided procedure for calibration of the three different temperature sensors. The selections at the bottom of the *Calibration Main Screen* are summarized below. Press the OCM function key directly below a desired selection to activate it.

Air: Calibrate the air thermocouple (accesses the *Air Thermocouple Low Calibration Screen*).

DUT - T: Calibrate the DUT type T thermocouple (accesses the *DUT Type T Thermocouple Low Calibration Screen*).

DUT - K: Calibrate the DUT type K thermocouple (accesses the *DUT Type K Thermocouple Low Calibration Screen*).

Return: Return to the Top Menu screen (accesses the *Top Menu Screen*, Figure 3-21).

Each temperature sensor calibration requires a precision temperature calibrator instrument to be substituted for the sensor, at the thermocouple panel. The procedure for calibrating each of the air, type T, or type K temperature sensors is very similar. Each calibration goes through a two-step procedure at -60°C and then at $+200^{\circ}\text{C}$. Carefully follow the detailed setup instructions on each screen. After the calibration instrument has been set to the specified temperature, press the [ENTER] key. During the **PLEASE WAIT ...** flag on the screen, the system will automatically take a reading, store it, and then advance to the next step of the calibration procedure.

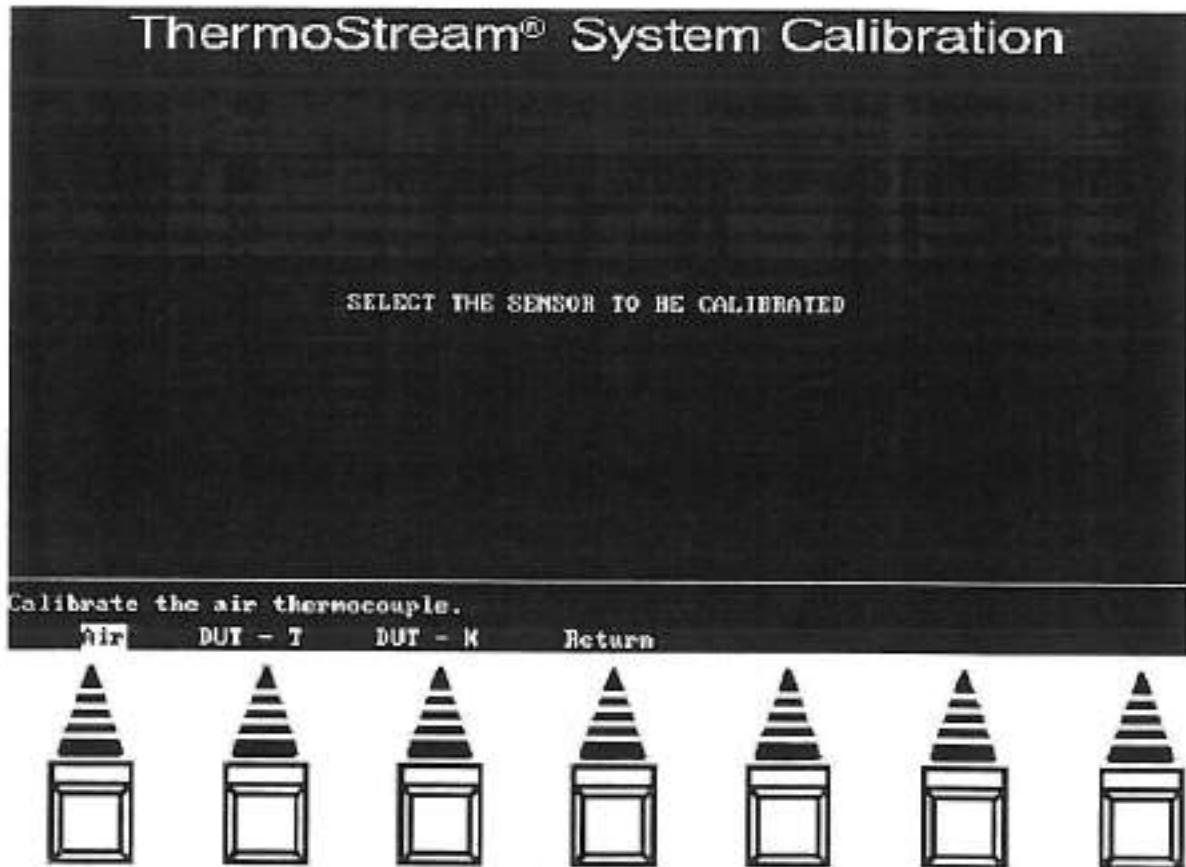


Figure 5-5. Main Calibration Screen

As an example of sensor calibration, the air temperature sensor screens are shown next.

Figure 5-6 Air Thermocouple Low Calibration Screen (-60 °C)

Figure 5-7 Air Thermocouple High Calibration Screen (+200 °C)

Figure 5-8 Air Thermocouple End Calibration Screen

ThermoStream® System Calibration

AIR THERMOCOUPLE

Remove the system front panel (see Operator's manual, Calibration section). Unplug P13 from J13, located on the right side top of the electronic enclosure (see diagram) and replace it with the output plug from the calibration instrument.

Set the calibrator for a temperature of -60°C.

Front of electronic enclosure →

| | |
|-----|---------------|
| ■ ■ | J38 OUT - T |
| ■ ■ | J35 OUT - K |
| ■ ■ | J13 Air ←---- |
| ■ ■ | J15 Purge |
| ■ ■ | J52 Spare |

Side View

Press "ENTER" when ready to proceed, or "ESC" to cancel the calibration.



Figure 5-6. Air Thermocouple Low Calibration Screen

ThermoStream® System Calibration

AIR THERMOCOUPLE

Remove the system front panel (see Operator's manual, Calibration section). Unplug P13 from J13, located on the right side top of the electronic enclosure (see diagram) and replace it with the output plug from the calibration instrument.

Set the calibrator for a temperature of 288°C.

Front of electronic enclosure →

| | |
|-----|---------------|
| ■ ■ | J38 OUT - T |
| ■ ■ | J35 OUT - K |
| ■ ■ | J13 Air ←---- |
| ■ ■ | J16 Purge |
| ■ ■ | J52 Spare |

Side View

Press "ENTER" when ready to proceed, or "ESC" to cancel the calibration.



Figure 5-7. Air Thermocouple High Calibration Screen

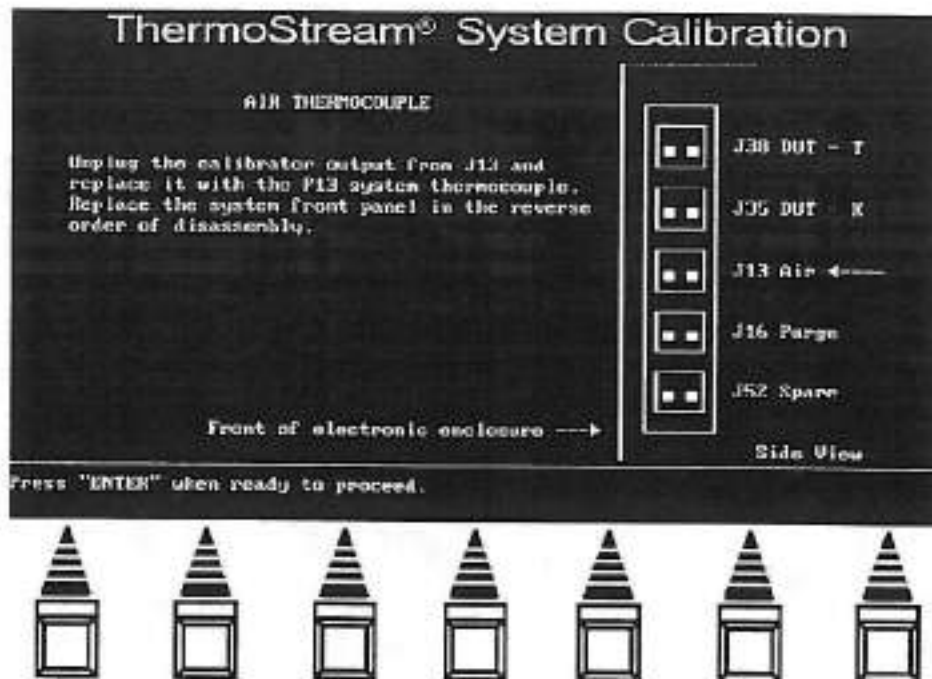


Figure 5-8. Air Thermocouple End Calibration Screen

5-7 Air Chiller Module Defrosting

When **Defrosting** is selected at the *Top Menu Screen* (Figure 3-21), the display switches to the *Defrost Cycle Screen* (see Figure 5-9). Accessing the *Defrost Cycle Screen* immediately starts the defrost cycle. This cycle continues until time-out (6 hours) or until terminated by pressing the [ESC] key.

Defrosting permits recovery from a "freeze-up" condition created when moisture in the air supply is allowed to enter the heat exchanger of the air chiller where it condenses and freezes. The indication that a "freeze-up" condition is occurring is a diminishing air flow at the head nozzle for a set value of air flow. When this condition occurs, you need to determine what has failed and is allowing moisture into the heat exchanger.

Check:

- > Air dryer coalescing filter -- empty if necessary.
- > Pneumatic module coalescing filter -- empty if necessary.
- > Air dryer operation.

When the defrost cycle is initiated:

- > The compressor of the air chiller module is turned off.

- > The system will not permit temperature control.
- > The air flow through the system will continue to force air through the heat exchanger to aid in the defrosting and removal of moisture from it.

The defrost cycle may be aborted at any time and the system returned to normal temperature control operation. However, if the moisture in the heat exchanger is not removed or is allowed to continue to enter the heat exchanger, the "freeze-up" will reoccur.

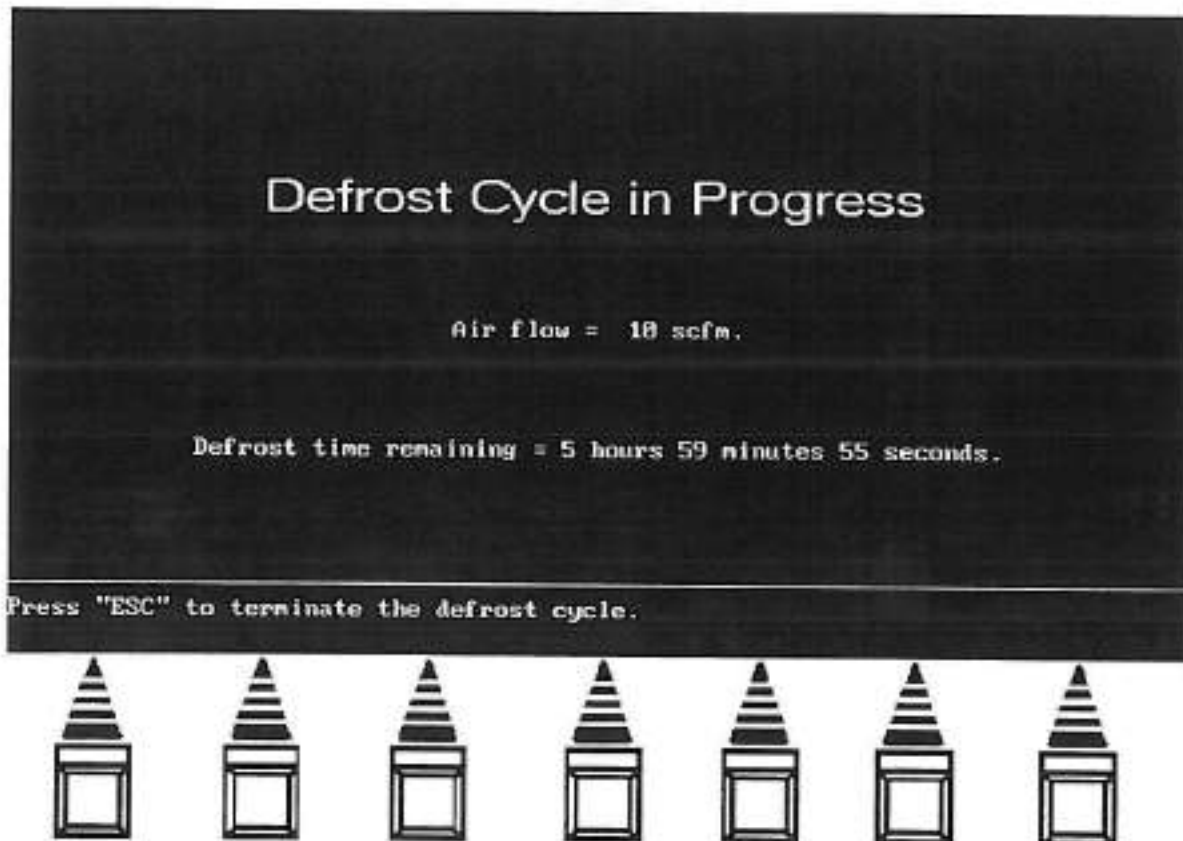


Figure 5-9. Defrost Cycle Screen

5-8 BIOS Setup

A discharged back-up battery or changes in the processor's motherboard may require that the CMOS setup utility be re-configured. If the stored CMOS information is incorrect, the system will not operate correctly. During bootup, a message such as *CMOS error or Setup error* will be displayed. The operating program will halt and no system screens will be displayed.

The following procedure outlines how to set up the CMOS information correctly for normal system operation. **NOTE:** An external keyboard must be connected at the system I/O panel to enter alphabetic responses. An external EGA display monitor is also recommended. See page 2-11 in this manual for requirements.

1. Turn on power to the TP04010A System and press the [ESC] key when prompted as the operating program boots. The setup screen should appear.
2. Observe that the STANDARD window becomes highlighted. Then press the [ENTER] key.
3. Follow the instructions on the STANDARD screen and.
 - a. After **Date**: enter the month, day, year
 - b. After **Time**: enter the hour, minute, second.
 - c. After **Drive A**: type in NOT INSTALLED.
4. Use the arrow keys to highlight the ADVANCED SETUP window. Then press the [ENTER] key.
5. Follow the instructions on the ADVANCED SETUP screen and.
 - a. After **Floppy Drive Seek at Boot**: type in DISABLED.
 - b. After **Video Shadow**: type in DISABLED.
 - c. After **Internal Cache**: type in DISABLED.
 - d. After **External Cache**: type in DISABLED.
 - e. After **Video ROM Cache**: type in DISABLED.
 - f. After **System ROM Cache**: type in DISABLED.
6. Use the arrow keys to highlight the CHIPSET window. Then press the [ENTER] key.
7. Follow the instructions on the CHIPSET screen and.
 - a. After **Auto Configuration**: type in DISABLED.



8. Press the [ESC] key to access the EXIT screen. Note that SAVE CHANGES AND EXIT becomes highlighted. Then press the [ENTER] key to exit the CMOS setup.
9. Turn off power to the TP04010A System and then restart. Observe that the operating program boots normally and stops with a Temptronic program screen.

Appendix A

IEEE-488 Demo Program

The following information is supplied to users of the Temptronic Corporation Model TP04010A Thermo-Stream System to assist them with using the IEEE-488 parallel interface.

Temptronic Corporation does not guarantee this material and is not responsible for any errors nor is liable for damages as the result of use of this material. The information in this appendix is subject to change without notice.

Contents

| | |
|---|------|
| demo.c -- TP04010A ThermoStream system IEEE bus demo program | A-2 |
| buserror () -- bus error and timeout handler | A-3 |
| send () -- send a string to the TP04010A | A-4 |
| receive () -- receive a string from the TP04010A | A-4 |
| initialize () -- initialize the TP04010A IEEE bus interface | A-5 |
| cleardeverror () -- clear TP04010A-specific errors | A-6 |
| poll_temp_test () -- test of temperature polling | A-6 |
| poll_statusbyte_test () -- test of status byte polling | A-7 |
| srq_test () -- test of service request when "at temperature" | A-9 |
| multitemp_test () -- test of multiple temperature setpoints | A-11 |
| cycle_test () -- test of cycle mode | A-13 |
| main () | A-15 |
| GPIB.COM CONFIGURATION INFORMATION | A-16 |

```

/* idemo.c -- TP04010A ThermoStream system IEEE bus demo program          */
/* 2.00 Modified from HEC's TP04000A version 7/25/94 by JM                */
/* This program, written in Microsoft Quick C ver 2.0,                    */
/* demonstrates control of the TP04010A by an IBM-compatible              */
/* pc via the remote IEEE-488 (GPIB) bus interface.                       */
/* This file must be linked with the *cib*.obj file supplied              */
/* by National Instruments Corp.                                          */
/* A National Instruments GPIB-PCII interface card must be                */
/* present in the pc, and the National GPIB.COM device driver             */
/* must be properly configured and loaded for the program to             */
/* See the end of this program for configuration information.             */
/* IMPORTANT: This program assumes that the TP04010A has been            */
/* manually started by the operator, and is displaying the               */
/* "Main Menu" screen. The IEEE interface is NOT active                  */
/* until the system reaches the "Main" menu                               */
/* screens.                                                                */
/* IEEE.c -- TP04010A ThermoStream system IEEE bus checkout - prod. use  */
/* ver 1.02A 1/16/91 JAS                                                  */
/* This program was modified for use in a production environment          */
/* to simulate control of the TP04010A by an IBM-compatible              */
/* PC via the remote IEEE-488 (GPIB) bus interface.                      */

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "time.h"

/* National Instruments requires that the following 3 variables be defined */
int ibsta; /* status word */
int iberr; /* error code */
int ibcnt; /* number of bytes sent */

int tp; /* TP04010A handle */
int bd; /* IEEE interface board handle */
char wrt[512]; /* string sent to TP04010A */
char rd[512]; /* string received from TP04010A */
char spr; /* serial poll response */
int stb; /* status byte */
int eror; /* device-specific error register */
int tesr; /* temperature event status register */
double temp; /* temperature */

int errcount; /* ERROR COUNT */
int attemperature; /* "at temperature" flag */
time_t prevtime, starttime; /* used for timing */

```

```

.....
/*          buserror() -- bus error and timeout handler          */
/*          */
/* All calls to National Instruments' routines should be followed by a */
/* check of the return value or ibsta (which are the same) to see if an */
/* error or bus timeout occurred.  If bit 15 is set, the problem is an */
/* error.  If bit 14 is set, the problem is a bus timeout.          */
/* The following procedure (as illustrated in this function) should be */
/* followed to reliably clear a bus error:                          */
/*   1. Send interface clear (IFC)                                  */
/*   2. Send selected device clear (SDC)                            */
/*   3. Send *ESR? followed by a read of the response from the TP04010A */
/*      to read and clear the standard event status register.      */
/*      Alternatively, a *CLS command (with no following read) may be */
/*      sent instead of *ESR? if there is no desire to determine the */
/*      reason for the error.                                        */
/* Erc is an error code which will be displayed on the screen to aid in */
/* finding the source of the error.                                */
.....

void buserror(int erc)
{
  char erd[16]; /* returned error info */

  if (ibsta & 0x4000) /* if the problem was a bus timeout */
  {
    printf("Bus timeout -- location %d",erc);
    ibsic(bd); /* send interface clear */
    ibclr(tp); /* send selected device clr */
    ibwrt(tp,"*ESR?\n",6); /* Try to read the standard event status */
    /* register. The response may give the */
    /* reason for the bus timeout to aid in */
    /* troubleshooting, and the read will */
    /* clear the register. */
    if (!(ibrd(tp,erd,15) & 0xC000)) /* if no further error, read valid */
    {
      strtok(erd,"\n"); /* replace the terminating line feed with null */
      printf(" Standard event status register = %s\n",erd);
    }
    else printf("\n"); /* unable to get status, just return */
  }
  else /* problem was some other error */
  {
    printf("IEEE bus error -- location %d .Program aborting\n",erc);
    exit(1);
  }
}

```

```

.....
/*          send() -- send a string to the TP04010A          */
/*          */
/* This function sends a string and checks for IEEE bus errors. */
/* An "end of string" (line feed) character is added.          */
/* If a bus error or timeout occurs, an error recovery routine (buserror) */
/* is called.          */
/* Erc is an error code so that the location of the error can be displayed */
/* to aid in troubleshooting.          */
/* This function returns the value of ibsta.          */
/*          */
int send(char *s, int ERC)
{
    int rv; /* return value = ibsta for this send */

    strcpy(wrt,s);
    strcat(wrt,"\n");
    rv = ibwrt(tp,wrt,strlen(wrt));

    /* Check for error. Bit 15 set = bus error, bit 14 set = bus timeout. */
    if (rv & 0xC000) buserror(ERC);
    return(rv);
}

.....
/*          receive() -- receive a string from the TP04010A          */
/*          */
/* This function receives a string and checks for IEEE bus errors. */
/* If a bus error or timeout occurs, an error recovery routine (buserror) */
/* is called.          */
/* Erc is an error code so that the location of the error can be displayed. */
/* This function returns the value of ibsta.          */
/* The returned string is in the global variable rd[].          */
/*          */
int receive(int ERC)
{
    int rv; /* return value = ibsta for this receive */

    rv = ibrd(tp,rd,256); /* read up to 256 characters from the TP04010A */

    /* Check for error. Bit 15 set = bus error, bit 14 set = bus timeout. */
    if (rv & 0xC000)
    {
        buserror(ERC);
        strcpy(rd,""); /* make the returned string a null string */
    }
    else /* if a valid string returned */
        strtok(rd,"\n"); /* replace the terminating line feed with a null */
    return(rv);
}

```

```
.....  
/*      initialize() -- initialize the TP04010A IEEE bus interface      */  
/*      */  
/* This function should be called at the beginning of every program to */  
/* ensure that the temperature controller will be able to receive and */  
/* process bus commands. */  
/* The following procedure (as illustrated in this function) should be */  
/* followed to reliably initialize the TP04010A IEEE bus interface: */  
/* 1. Send interface clear (IPC) */  
/* 2. Send selected device clear (SDC) */  
/* 3. Send *CLS */  
/* 4. Send *RST */  
/* 5. Send *STB? followed by a read of the response from the TP04010A */  
  
void initialize(void)  
{  
    /* Send IPC (interface clear) to terminate any possible bus handshake */  
    /* in progress. */  
    if (ibsic(bd) < 0) printf("Error during ibsic()\n");  
  
    /* Send SDC (selected device clear) command to reset the TP04010A IEEE */  
    /* interface. The IEEE command input and output buffers are cleared. */  
    if (ibclr(tp) < 0) printf("Error during ibclr()\n");  
  
    /* Send *CLS (clear status) to clear the TP04010A's status registers */  
    send("*CLS",3);  
  
    /* Send *RST (machine function reset) command to force the TP04010A to */  
    /* the "Main Menu" screen. This command can take several seconds. */  
    send("*RST",4);  
  
    /* Ask for the TP04010A's status byte. The response will not be sent */  
    /* immediately because of the time it takes for the previous *RST */  
    /* command to complete. */  
    send("*STB?",5);  
    receive(6); /* get the response back in rd[] */  
    if ((atoi(rd) & 0x80) != 0x80) /* if "ready" bit in status byte not set */  
    {  
        printf("TP04010A not ready\n, Program aborting.");  
        exit(1);  
    }  
}
```

```

.....
/*      cleardeverror() -- clear TP04010A-specific errors      */
/*      */
/* This function uses the CLER (clear error) bus command to try to clear */
/* certain errors that are related to the temperature control functions of */
/* the TP04010A. These errors are reported back in response to the ERROR? */
/* query. Three errors must be cleared by the CLER command -- overheat, */
/* main thermocouple open loop, and purge thermocouple open loop. The */
/* other errors will clear themselves automatically when the problem is */
/* corrected. Calling this function for those errors that are */
/* automatically clearable is not necessary, but may provide a convenient */
/* way of delaying a few seconds and then rereading the device error */
/* register so that transient errors can be ignored. */
.....

void cleardeverror(void)
{
    time_t st;

    errcount++; /* increment the error count */
    send("CLER",7);

    /* wait 3-4 seconds to make sure CLER has taken effect */
    st = time(NULL);
    while ((time(NULL)-st)<=3); /* wait here */
}

.....
/*      poll_temp_test() -- test of temperature polling      */
/*      */
/* One conceptually simple method of waiting until a temperature setpoint */
/* has been reached is to periodically ask for the current temperature */
/* with the TEMP? query. To ensure that the TP04010A is functioning */
/* properly, the device error register should also be read with the ERROR? */
/* query. */
.....

void poll_temp_test()
{
    char setpointstr[16];
    float setpoint;
    printf("Temperature polling test in progress. Target = 130\n");
    setpoint=130.0;
    /* Turn off all reasons for a service request (not used for this test), */
    /* use setpoint number 0 (Hot setpoint), and put the */
    /* head down (which also turns the airflow on). These commands need */
    /* only be sent once. */
    send("**SRE 0;SETN 0;HEAD 1;DUTM 0",10);

    /* Convert the desired setpoint to a string and send to the TP04010A */
    sprintf(setpointstr,"SETP %1.1f",setpoint); /* setpoint string */
    send(setpointstr,11);

    /* Wait up to 1 minute to get to temperature, sampling every second. */
}

```



```

/* When the temperature is at the setpoint +/- 3 degrees, display an */
/* "at temperature" message. */
errcount = 0; /* reset the error counter */
starttime = time(NULL); /* start of one minute period */
do
{
  prevtime = time(NULL);
  do
  {
    while ((time(NULL)-prevtime)); /* wait here until 1 sec elapsed */

    /* Check for TP04010A-specific errors (such as overheat) */
    /* If an error is detected, try clearing it. Give up after three */
    /* unsuccessful error-clearing attempts. */
    send("EROR?",11);
    receive(12); /* get the reponse in rd[] */
    error = atoi(rd); /* convert string to an integer */
    if (error) cleardeverror(); /* try to clear a TP04010A-specific error */
    if (errcount = 3) /* if error cannot be cleared in 3 tries */
    {
      printf("Unclearable TP04010A-specific error. EROR = %X. ",error);
      printf("Program aborting.\n");
      exit(1);
    }

    attemperature = 0; /* assume not at temperature */

    /* read the temperature */
    send("TEMP?",11);
    if (!(receive(12) & 0xC000)) /* get the response, and if valid */
    {
      temp = atof(rd); /* convert temperature to floating point */
      printf("Temperature = %s\n",rd);
      if ((temp = setpoint-3) && (temp<= setpoint+3)) attemperature = 1;
    }
  }
  while (((time(NULL)-starttime)<) && !attemperature);

  if (attemperature) printf("At temperature\n\n");
  else printf("Did not reach temperature in 1 minute.\n\n");
}

/*****
/* poll_statusbyte_test() -- test of status byte polling */
/* */
/* Both the "at temperature" and error conditions of the TP04010A can be */
/* simply and reliably monitored by periodically polling the status byte */
/* with the *STB? query. */
*****/

void poll_statusbyte_test()
{
  char setpointstr[16];
  float setpoint;

```

```

printf("Status byte polling test in progress. Target = 100\n");
setpoint=100.0;
/* Turn off all reasons for a service request (not used for this test). */
/* Turn on the "at temperature" bit of the temperature event status */
/* enable register. Use setpoint number 0 (Hot setpoint) */
/* Set the soak time to 10 seconds, the temperature tolerance */
/* window to +/- 2 degrees. Put the head down (which also turns the */
/* airflow on. These commands need only be sent once. */
send("*SRE 0;TESE 1;SETN 0;SOAK 10;WNDW 2;HEAD 1",15);

/* Convert the desired setpoint to a string and send to the TP04010A */
sprintf(setpointstr,"SETP %1.1f",setpoint); /* setpoint string */
send(setpointstr,16);

/* Resend the setpoint number. This forces the temperature condition */
/* register "at temperature" bit off, even if the value of the new */
/* setpoint is the same as the previous setpoint. This is needed to */
/* guarantee that there will be a temperature condition register */
/* transition from "not at temp" to "at temp." It is this transition */
/* that sets the "at temperature" bit in the temperature event status */
/* register. The *CLS (clear status registers) command is needed to */
/* get rid of a possible preexisting "at temp" condition in the */
/* temperature event status register. (Do not confuse the condition */
/* and status registers.) Both the SETN (with appropriate setpoint */
/* number) and *CLS commands should be sent after every new setpoint. */
send("SETN 0;*CLS",17);

/* Wait up to 1 minute to get to temperature, reading the status byte */
/* every second. */
attemperature = 0; /* reset "at temperature" flag */
errcount = 0; /* reset the error counter */
starttime = time(NULL); /* start of one minute period */
do
{
    prevtime = time(NULL);
    do
    {
        while ((time(NULL)-prevtime)); /* wait here until 1 sec elapsed */

        send("*STB?",19);
        receive(19); /* get the reponse in rd[] */
        stb = atoi(rd); /* convert string to an integer */

        /* Check for TP04010A-specific errors (such as overheat). */
        /* If an error is detected, try clearing it. Give up after three */
        /* unsuccessful error-clearing attempts. */
        if (stb & 4) /* if the device-specific error bit is set */
        {
            send("EROR?",20);
            receive(21);
            error = atoi(rd);
            cleardeverror();
        }
        if (errcount = 3) /* if error cannot be cleared in 3 tries */
        {

```

```

    printf("Unclearable TP04010A-specific error.  ERROR = %X.  ",error);
    printf("Program aborting.\n");
    exit(1);
  }

  /* If the temperature event summary bit is set, "at temperature" */
  /* must have occurred because only the "at temp" bit of the */
  /* temperature event status enable register is enabled. */
  if (stb & 8)
  {
    send ("*CLS",22); /* clear temperature event status register */
    attemperature = 1;
  }
}
while ((time(NULL)-starttime)<) && !attemperature);

if (attemperature) printf("At temperature\n\n");
else printf("Did not reach temperature in 1 minute.\n\n");
}

/*****
/*  srq_test() -- test of service request when "at temperature"      */
/*                                                                */
/* The service request line of the IEEE bus interface can be used to signal */
/* when the desired temperature is reached without continually polling the */
/* TP04010A.                                                         */
*****/

void srq_test()
{
  char setpointstr[16];
  float setpoint;
  printf("Service request test in progress.  Target temp = 125\n");
  setpoint=125.0;
  /* do a serial poll to clear any pending service request */
  ibrsp(tp, &spr);

  /* Turn on the "at temperature" bit of the temperature event status */
  /* enable register.  Use setpoint number 0 (Hot setpoint)          */
  /* Set the soak time to 10 seconds, the temperature tolerance     */
  /* window to +/- 3 degrees.  Put the head down (which also turns the */
  /* airflow on.  Clear the status registers to prevent an immediate SRQ */
  /* service request) from some old condition.  Set the "temperature */
  /* event" and "device error" bits of the service request enable */
  /* register on.  These commands need only be sent once. */
  send("TESE 1;SETN 0;SOAK 10;WNDW 3;HEAD 1;*CLS;*SRE 12",30);

  /* Convert the desired setpoint to a string and send to the TP04010A */
  sprintf(setpointstr,"SETP %1.1f",setpoint); /* setpoint string */
  send(setpointstr,31);

  /* Resend the setpoint number.  This forces the temperature condition */
  /* register "at temperature" bit off, even if the value of the new */
  /* setpoint is the same as the previous setpoint.  This is needed to */

```

```

/* guarantee that there will be a temperature condition register */
/* transition from "not at temp" to "at temp." It is this transition */
/* that sets the "at temperature" bit in the temperature event status */
/* register. The *CLS (clear status registers) command is needed to */
/* get rid of a possible preexisting "at temp" condition in the */
/* temperature event status register. (Do not confuse the condition */
/* and status registers.) Both the SETN (with appropriate setpoint */
/* number) and *CLS commands should be sent after every new setpoint. */
send("SETN 0;*CLS",32);

/* Wait up to 1 minute for an "at temp" service request to occur. */
attemperature = 0; /* reset flag */
errcount = 0; /* reset the error counter */
starttime = time(NULL);
do
{
/* call ibwait() with a 0 parameter to update service request state */
if (ibwait(bd,0) & 0x1000) /* if srq has occurred */
{
printf("SRQ received\n");
/* Do a serial poll. If an error occurs, force the poll */
/* response to 0. */
if (ibrsp(tp,&spr) spr = 0;
/* Check for TP04010A-specific errors (such as overheat). */
/* If an error is detected, try clearing it. Give up after three */
/* unsuccessful error-clearing attempts. */
if (spr & 4) /* if the device-specific error bit is set */
{
.. send("EROR?",33);
.. receive(34);
.. error = atoi(rd);
.. cleardeverror();
}
if (errcount = 3) /* if error cannot be cleared in 3 tries */
{
.. printf("Unclearable TP04010A-specific error. EROR = %X. ",
.. error);
.. printf("Program aborting.\n");
.. exit(1);
}

/* If the temperature event summary bit is set, read the "at */
/* temperature" bit of the temperature event status register */
/* to verify the cause of the interrupt. */
if (spr & 8)
{
.. send("TESR?",35); /* Read the temperature event status reg */
..... /* (reading the register also clears it) */
.. receive(36);
.. tesr = atoi(rd); /* make into an integer */
.. if (tesr & 1) attemperature = 1; /* if "at temp" bit is set */
}
}
}

```



```

while ((time(NULL)-prevtime) < 45 | attemperature);
    ..... /* wait for up to 1 minute */

if (attemperature) printf("At temperature\n\n");
else printf("Did not reach temperature in 1 minute.\n\n");
}

/*****
/*  multitemp_test() -- test of multiple temperature setpoints          */
/*  */
/*  This function illustrates how a tester might preload the temperature */
/*  setpoints with several different temperature values, and then step from */
/*  one temperature to another as desired. The TP04010A's status byte is */
/*  polled, and the temperature event summary bit is used to give a "begin */
/*  test" signal to the tester.                                          */
void multitemp_test(void)
{
    int sn;          /* setpoint number */
    char sendstr[32];

    printf("Multiple temperature setpoint test in progress\n");

    /* Put the head up (and turn the airflow off). */
    send("HEAD 0",60);

    /* Wait for "head up" command to take effect. A minimum of one second */
    /* must separate any of the four following commands to guarantee      */
    /* proper head and flow operation: head up, head down, flow on,      */
    /* flow off. */
    starttime = time(NULL);
    do { }
    while ((time(NULL)-starttime)); /* wait here until 2 sec elapsed */

    /* Load parameters into the setpoints that will be used */
    send("SETN 2;SETP 20;WNDW 3;SOAK 10",61);
    send("SETN 1;SETP 25;WNDW 3;SOAK 10",61);
    send("SETN 0;SETP 125;WNDW 3;SOAK 10",61);

    /* Disable all service requests. Turn on the "at temperature" bit of */
    /* the temperature event status enable register. Put the head down. */
    send("SRE 0;TESE 1;HEAD 1;COOL 1",63);

    /* for each setpoint */
    for (sn=0; sn<=2; sn++)
    {
        /* Convert the desired setpoint to a string and send to the TP04010A */
        sprintf(sendstr,"SETN %d;CLS",sn);
        send(sendstr,64);

        /* Wait up to 5 minutes to get to temperature, reading the status */
        /* byte every second. */
        attemperature = 0;          /* reset "at temperature" flag */
    }
}

```

```

errcount = 0;          /* reset the error counter */
starttime = time(NULL); /* start of 5 minute period */
do
{
    prevtime = time(NULL);
    do
    {
        while ((time(NULL)-prevtime)); /* wait here until 1 sec elapsed */

        send("**STB?",65);
        receive(65);          /* get the reponse in rd[] */
        stb = atoi(rd);      /* convert string to an integer */

        /* Check for TP04010A-specific errors (such as overheat). */
        /* If an error is detected, try clearing it. */
        if (stb & 4)          /* if the device-specific error bit is set */
        {
            .. send("EROR?",65);
            .. receive(65);
            .. error = atoi(rd);
            .. cleardeverror();
        }
        if (errcount = 3)    /* if error cannot be cleared in 3 tries */
        {
            .. printf("Unclearable TP04010A-specific error.  EROR = %X.  ",
                .. error);
            .. printf("Program aborting.\n");
            .. exit(1);
        }

        /* If the temperature event summary bit is set, "at temperature" */
        /* must have occurred because only the "at temp" bit of the */
        /* temperature event status enable register is enabled. */
        if (stb & 8)
        {
            .. send ("**CLS",65); /* clear temperature event status register */
            .. attemperature = 1;
        }
    }
    while ((time(NULL)-starttime) <&& !attemperature);

    if (attemperature)
    {
        printf("At temperature\n");
        /* PERFORM YOUR TEST HERE */
    }
    else printf("Did not reach temperature in 5 minutes.\n");
}

/* Put the head up */
send("HEAD 0",65);

printf("\n");
}

```

```

/*****
/*          cycle_test() -- test of cycle mode          */
/*          */
/* This function illustrates how a tester might preload the temperature */
/* setpoints with several different temperature values, and then use cycle */
/* mode to automatically step (in order) from one setpoint to the next. */
/* The TP04010A's status byte is polled, and the temperature event summary */
/* bit is used to give a "begin test" signal to the tester. */
/* Note that this function offers less flexibility to the tester than the */
/* multitemp_test() function. Cycle mode is most useful when the TP04010A */
/* is controlling a "dumb" tester. */
void cycle_test(void)
{
    int allcyclesdone;

    printf("Cycle mode test in progress\n");

    /* Put the head up (and turn the airflow off). Go into ramp mode. */
    /* Set the maximum test time to 1 minute, and set up to do 2 cycles. */
    send("HEAD 0;RMPC 1;TTIM 60;CYCC 2",70);

    /* Wait for "head up" command to take effect. A minimum of one second */
    /* must separate any of the four following commands to guarantee */
    /* proper head and flow operation: head up, head down, flow on, */
    /* flow off. */
    starttime = time(NULL);
    do {
    while ((time(NULL)-starttime)); /* wait here until 2 sec elapsed */

    /* Load parameters into the setpoints that will be used */
    /* A ramp rate of 9999 means transition the temperature as fast as */
    /* possible. */
    send("SETN 0;RAMP 9999;SETP 125;WNDW 3;SOAK 10",71);
    send("SETN 1;RAMP 9999;SETP 25;WNDW 3;SOAK 10",71);
    send("SETN 2;RAMP 9999;SETP 15;WNDW 3;SOAK 10",71);

    /* Disable all service requests. Turn on the "at temperature" bit of */
    /* the temperature event status enable register. Put the head down. */
    /* Turn cycling on. Start at the first setpoint. Clear status regs. */
    send("SRE 0;TESE 17;HEAD 1;CYCL 1;SETN 0;*CLS",73);

    allcyclesdone = 0; /* flag -- becomes 1 when all cycles are done */
    do
    {
        /* Wait up to 5 minutes to get to temperature, reading the status */
        /* byte every second. */
        atemperature = 0; /* reset "at temperature" flag */
        errcount = 0; /* reset the error counter */
        starttime = time(NULL); /* start of 5 minute period */
        do
        {
            prevtime = time(NULL);
            do
            {

```

```

while ((time(NULL)-prevtime)); /* wait here until 1 sec elapsed */

send("**STB?",75);
receive(75);          /* get the response in rd[] */
stb = atoi(rd);       /* convert string to an integer */

/* Check for TP04010A-specific errors (such as overheat). */
/* If an error is detected, try clearing it. */
if (stb & 4)          /* if the device-specific error bit is set */
{
..send("EROR?",75);
..receive(75);
..error = atoi(rd);
..cleardeverror();
}
if (errorcount = 3)  /* if error cannot be cleared in 3 tries */
{
..printf("Unclearable TP04010A-specific error.  EROR = %X.  ",
..      error);
..printf("Program aborting.\n");
..exit(1);
}

/* If the temperature event summary bit is set, "at temperature" */
/* must have occurred because only the "at temp" bit of the */
/* temperature event status enable register is enabled. */
if (stb & 8)
{
..send("TESR?",75); /* Read the temperature event status reg */
..... /* (reading the register also clears it) */
..receive(75);
..tesr = atoi(rd); /* make into an integer */
..if (tesr & 1) attemperature = 1; /* if "at temp" bit is set */
..if (tesr & 16) allcyclesdone = 1;
}
}
while ((time(NULL)-starttime,) && !attemperature
&& !allcyclesdone);

if (allcyclesdone)
printf("All temperature cycles done\n");
else
{
if (attemperature)
{
..printf("At temperature\n");
../* PERFORM YOUR TEST HERE */
}
else printf("Did not reach temperature in 5 minutes.\n");

/* Step to the next temperature */
send("NEXT",75);
}
}
while (!allcyclesdone);

```



```
    /* put the head up and end cycling mode */
    send("HEAD 0;CYCL 0;RMPC 0",76);
}

/*****
/*                               main()                               */
*****/

int main()
{
    printf("TP04010A IEEE BUS INTERFACE TEST PROGRAM ver 2.00\n\n");

    /* Find the IEEE board. It has been assigned the name "GPIB0" by */
    /* the GPIB.COM device driver. */
    bd = ibfind("GPIB0");

    /* Check for an error from ibfind() -- indicated by a negative return */
    if (bd < 0)
    {
        printf("Board ibfind() failed. Program aborting.\n");
        exit(1);
    }

    /* Find the Temptronic TP04010A. It has been assigned the name "DEV1" */
    /* by the GPIB.COM device driver. */
    tp = ibfind("DEV1");

    /* Check for an error from ibfind() */
    if (tp < 0)
    {
        printf("TP04010A ibfind() failed. Program aborting.\n");
        exit(1);
    }

    /* Set the TP04010A bus timeout interval to 10 seconds */
    ibtmo(tp,13);

    /* Initialize the TP04010A IEEE interface. This should be done at the */
    /* beginning of every program to ensure that the temperature */
    /* controller will be able to receive and process bus commands. */
    initialize();

    /* The following functions illustrate 3 different methods of waiting */
    /* until one desired airstream temperature has been reached. */

    poll_temp_test();          /* poll temperature and error registers */
    .....

    poll_statusbyte_test();   /* poll the status byte -- RECOMMENDED */
    .....

    srq_test();               /* wait for a service request */
    .....
}
```

```

/* The following functions illustrate 2 different methods of stepping */
/* from one preset temperature to another under control of the host */
/* system. */

multitemp_test();

cycle_test();

/* Place the TP04010A IEEE bus off line */
send("**STB?",80); /* Wait for any pending commands to complete by */
receive(80); /* waiting here until a response comes back from */
..... /* the TP04010A. */
ibsic(bd);
ibclr(tp);
ibloc(tp); /* return TP04010A to local control */
ibonl(tp,0);

printf("\nEND\n");
return(0);
)

/*****
/*
/* GPIB.COM CONFIGURATION INFORMATION */
/*
/* Use National Instruments' IBCONF.EXE configuration program to set up */
/* the GPIB.COM device driver. */
/* Board GPIB0 should be connected to DEV1, which will be the TP04010A. */
/*
/* GPIB0 Board settings: */
/* Primary GPIB Address.....21 */
/* Secondary GPIB Address.....NONE */
/* Timeout setting.....T10s */
/* EOS byte.....0AH */
/* Terminate Read on EOS.....yes */
/* Set EOI with EOS on Write.....yes */
/* Type of compare on EOS.....7-bit */
/* Set EOI w/last byte of write...yes */
/* GPIB-PC Model.....PC2 */
/* Board is System Controller....yes */
/* Local Lockout on all devices...no */
/* Disable Auto Serial Polling...yes */
/* High-speed timing.....no */
/* Interrupt jumper setting.....NONE */
/* Base I/O Address.....[match hardware setting of your card] */
/* DMA channel.....NONE */
/* Internal Clock Freq (in MHz)...8 */
*****/

```

```
/*                                                                    */
/* DEVI device settings:                                             */
/*   Primary GPIB Address.....[match address on TP04010A config screen] */
/*   Secondary GPIB Address.....NONE                                */
/*   Timeout setting.....710s                                       */
/*   EOS byte.....0AH                                               */
/*   Terminate Read on EOS.....yes                                   */
/*   Set EOI with EOS on Write.....yes                               */
/*   Type of compare on EOS.....7-bit                               */
/*   Set EOI w/last byte of write...yes                             */
/*                                                                    */
/*.....*/
```



Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

Appendix B

RS232C Demo Program

The following information is supplied to users of the Temptronic Corporation Model TP04010A ThermoStream System to assist them with using the RS232C serial interface.

Temptronic Corporation does not guarantee this material and is not responsible for any errors nor is liable for damages as the result of use of this material. The information in this appendix is subject to change without notice.

Contents

| | |
|--|------|
| <code>sdemo.c</code> -- TP04010A ThermoStream system serial interface demo program..... | B-2 |
| <code>delay ()</code> -- wait a certain number of seconds | B-3 |
| <code>pcserialinit ()</code> -- pc serial board initialization..... | B-4 |
| <code>swrt ()</code> -- write a string to the serial interface..... | B-5 |
| <code>srd ()</code> -- read a string from the serial interface | B-6 |
| <code>deviceclear ()</code> -- send device clear characters to the TP04010A | B-7 |
| <code>serialerror ()</code> -- serial interface error and timeout handler | B-8 |
| <code>send ()</code> -- send a string to the TP04010A..... | B-9 |
| <code>receive ()</code> -- receive a string from the TP04010A..... | B-9 |
| <code>serialpoll ()</code> -- do a serial poll of the TP04010A | B-10 |
| <code>checkforsrq ()</code> -- see if the TP04010A is requesting service | B-10 |
| <code>initialize ()</code> -- initialize the TP04010A serial interface..... | B-11 |
| <code>cleardeverror ()</code> -- clear TP04010A-specific errors | B-12 |
| <code>poll_temp_demo ()</code> -- demonstration of temperature polling..... | B-12 |
| <code>poll_statusbyte_demo ()</code> -- demonstration of status byte polling..... | B-14 |
| <code>srq_demo ()</code> -- demonstration of service request when "at temperature"..... | B-16 |
| <code>multitemp_demo ()</code> -- demonstration of multiple temperature setpoints..... | B-19 |
| <code>cycle_demo ()</code> -- demonstration of cycle mode | B-23 |
| <code>main ()</code> | B-27 |
| RS232 SERIAL INTERFACE CABLE INFORMATION | B-29 |

```

/* sdemo.c -- TP04010A ThermoStream system serial interface demo program */
/* ver 2.00 8/1/94 Modified from HEC's tp04000 version */
/* This program, written in Microsoft Quick C ver 2.0, */
/* demonstrates control of the TP04010A by an IBM-compatible */
/* PC via the remote serial interface. */
/* For universality, this program uses bios interrupt 14H to */
/* send and receive serial data. */
/* Pinout and cable information are shown at the end of this */
/* source file. */
/* IMPORTANT: Be sure that the baud rate used for this demo */
/* matches the setting on the TP04010A configuration screen. */
/* The default settings for this demo program are the COM1: */
/* serial port, and 9600 baud. To use a different pc serial */
/* port and/or baud rate, invoke this program followed by the */
/* desired port number (1-4) and baud rate. This demo uses */
/* the computer's default serial timeout interval (usually */
/* several seconds). The timeout interval is the maximum time */
/* that the host pc will wait to send or receive a character. */
/* If you experience occasional serial timeouts, especially */
/* during a read, you may wish to increase the timeout value */
/* by invoking SDEMO with a T1 to T8 parameter, where the */
/* number is the amount to multiply the default timeout. For */
/* example, type SDEMO 2 1200 T2 to run this program using */
/* COM2:, 1200 baud and double the default timeout interval. */
/* NOTE: The remote serial interface offers a great deal of */
/* programming flexibility, but also some complexity. There */
/* are usually several ways to accomplish the same task. Much */
/* of the complexity of the sample code in this program is a */
/* result of error checking and handling. You may want to */
/* increase or decrease the amount of error checking based on */
/* your particular application and reliability requirements. */
/* IMPORTANT: This program assumes that the TP04010A has been */
/* manually started by the operator, and is displaying the */
/* Main screen. The serial interface is NOT */
/* active until the system reaches the "Main" screens. */

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "time.h"
#include "dos.h"
#include "ctype.h"

```

```
char comport;      /* pc COM port to be used for demo (1-4) */
int baudrate;     /* baud rate for demo (300-9600) */
int timeout;      /* serial timeout multiplier */

char wrt[512];     /* string sent to TP04010A */
char rd[512];     /* string received from TP04010A */
int srqreceived;  /* Flag -- nonzero if a service request character (^) */
    ..          /* has been received. */
unsigned char spr; /* serial poll response */
int stb;         /* status byte */
int error;      /* device-specific error register */
int tesr;      /* temperature event status register */
double temp;   /* temperature */

int ssta;      /* host pc serial status word */
int errcount; /* error count */
int attemperature; /* "at temperature" flag */
time_t starttime; /* start of a timing interval */

union REGS inregs, outregs;

/*****
/*          delay() -- wait a certain number of seconds          */
/*          *****/
void delay(int delaytime) /* delaytime is delay in seconds */
{
    time_t start;
    start = time(NULL);
    while ((time(NULL)-start)/delaytime);
}

```

```

/*****
/*          pserialinit() -- pc serial board initialization          */
/*          */
/* This function checks that the desired serial (com) port is present in */
/* the host pc, and initializes it to the desired baud rate.          */
int pserialinit(char comport, int baudrate)
{
    long serialbaseaddr;
    unsigned char br; /* encoded form of baud rate */
    int x;
    switch (baudrate)
    {
        case 300: br = 0x40;
        .. break;
        case 600: br = 0x60;
        .. break;
        case 1200: br = 0x80;
        .. break;
        case 2400: br = 0xA0;
        .. break;
        case 4800: br = 0xC0;
        .. break;
        default: br = 0xE0; /* 9600 baud */
    }

    /* check for serial port present (is base address legal?) */
    serialbaseaddr = 0x00400000L + 2*(comport-1);
    x=(int far*)serialbaseaddr;
    if (!(xx200) || (x0x3ff))
    {
        return(1); /* pc has no serial port if base address is out of range */
    }
    else
    {
        /* use pc bios interrupt 14H to initialize the serial port */
        inregs.x.dx = comport-1; /* com1 - com4 */
        inregs.h.ah = 0; /* initialize the port */
        inregs.h.al = br | 3; /* no parity, 1 stop, 8 data bits */
        int86(0x14, &inregs, &outregs);

        if (outregs.h.ah & 1) /* is an old character in the uart? */
        {
            inregs.h.ah = 2; /* read it to flush it if so */
            int86(0x14, &inregs, &outregs);
        }

        return(0);
    }
}

```



```
.....  
/*          swrt() -- write a string to the serial interface          */  
/*          */  
/* This function uses pc bios interrupt 14H to write one or more characters */  
/* over the serial interface. It returns a 0 if all characters were      */  
/* written, or returns 0x80 if an interface timeout occurred.          */  
/*          */  
int swrt(char *wrt, int cnt)  
{  
    int i,j;  
    inregs.x.dx = comport-1;  
  
    for (i=0; i ... ; i++)  
    {  
        for (j=0; j<timeout; j++)  
        {  
            inregs.h.al = wrt[i];    /* character to send */  
            inregs.h.ah = 1;         /* "send" subfunction of bios int 14H */  
            int86(0x14, &inregs, &outregs);  
            ssta = outregs.h.ah & 0x80; /* returned status, 0x80 = timeout */  
            if (!ssta) break;        /* character successfully sent */  
        }  
        if (ssta) return(ssta);     /* return immediately if timeout */  
    }  
    return(0); /* zero means entire string sent without problem */  
}
```

```

/*****
/*          srd() -- read a string from the serial interface          */
/*          */
/* This function uses pc bios interrupt 14H to read one or more characters */
/* from the serial interface.  It returns a 0 if an end of string (line */
/* feed) character was read, or if the desired maximum number of */
/* characters were read.  It returns nonzero if an error occurred.  The */
/* service request flag is set if the srq character is detected.      */
*****/

int srd(char *srdstr, int cnt)
{
    int i,j;
    inregs.x.dx = comport-1;

    for (i=0; i ... i++)
    {
        for (j=0; j<timeout; j++)
        {
            inregs.h.ah = 2;          /* "receive" subfunction of bios int 14H */
            int86(0x14, &inregs, &outregs);
            srdstr[i] = outregs.h.al; /* put byte in string */
            ssta = outregs.h.ah;     /* returned status, 0x80 = timeout */
            if (!(ssta & 0x80)) break; /* character successfully received */
        }
        if (ssta) break;             /* return immediately if error */
        if (srdstr[i] == 10) break; /* done if line feed received */
        if (srdstr[i] == '^')       /* if service request received */
        {
            srqreceived = 1;        /* set flag */
            i--;                    /* don't put in received string */
        }
    }
    i++;
    srdstr[i] = 0; /* add terminating null to received string */
    return(ssta); /* zero means entire string received without problem */
}

```

```
/*.....*/
/*      deviceclear() -- send device clear characters to the TP04010A      */
/*      */
/* This function sends a series of 3 device clear characters (!) to the    */
/* TP04010A. After each is sent, an attempt is made to read the "!" which */
/* the temperature controller should echo back. A delay before checking    */
/* the first character's echo is needed to ensure that the character      */
/* received is not left over from an earlier transmission. A missing or   */
/* incorrect echo of the first character does not necessarily indicate a  */
/* problem that cannot be cleared, but the two succeeding clear commands  */
/* should echo back correctly. If only one of them echoes back, the      */
/* problem may be a mismatch in baud rate. If the device clear was      */
/* successful, this function returns a 0.                                  */
/*.....*/

int deviceclear(void)
{
    int i;

    inregs.x.dx = comport-1;
    inregs.h.ah = 1;      /* send a character */
    inregs.h.al = '!';   /* device clear character */
    int86(0x14, &inregs, &outregs);
    delay(2);           /* wait one second */
    inregs.h.ah = 2;     /* receive a character */
    int86(0x14, &inregs, &outregs);

    for (i=0; i<=1; i++)
    {
        inregs.h.ah = 1;      /* send a character */
        inregs.h.al = '!';   /* device clear character */
        int86(0x14, &inregs, &outregs);
        inregs.h.ah = 2;     /* receive a character */
        int86(0x14, &inregs, &outregs);
        if ((outregs.h.ah & 0x80) || (outregs.h.al != '!')) return(1);
    }
    return(0);
}
}
```

```

/*****
/*      serialerror() -- serial interface error and timeout handler      */
/*                                                                              */
/* All serial interface reads and writes should be followed by a          */
/* check of the return value or ssta (which are the same) to see if an    */
/* error or interface timeout occurred.  If Bit 7 is set, the problem is a  */
/* timeout.  If another bit is set, the problem is an error.              */
/* The following procedure (as illustrated in this function) should be     */
/* followed to reliably clear a serial interface error:                    */
/*   1. Send device clear                                                  */
/*   2. Send *ESR? followed by a read of the response from the TP04010A   */
/*      to read and clear the standard event status register.             */
/*      Alternatively, a *CLS command (with no following read) may be     */
/*      sent instead of *ESR? if there is no desire to determine the      */
/*      reason for the error.                                              */
/* Erc is an error code which will be displayed on the screen to aid in   */
/* finding the source of the error.                                        */
*****/

void serialerror(int erc)
{
    char erd[16]; /* returned error info */

    if (ssta & 0x80) /* if the problem was an interface timeout */
    {
        printf("Serial interface timeout -- location %d",erc);
        deviceclear(); /* send device clr */
        swrt("*ESR?\n",6); /* Try to read the standard event status */
        /* register. The response may give the */
        /* reason for the timeout to aid in */
        /* troubleshooting, and the read will */
        /* clear the register. */
        if (!srd(erd,15)) /* if no further error, read valid */
        {
            strtok(erd,"\r\n"); /* Replace the terminating carriage return */
            /* or line feed with a null (which is the */
            /* string terminator in C). */
            printf("    Standard event status register = %s\n",erd);
        }
        else printf("\n"); /* just end line if unable to get status */
    }
    else /* problem was some other error */
        printf("Serial interface error -- location %d\n",erc);
}

```

```

/*****
/*          send() -- send a string to the TP04010A          */
/*          */
/* This function sends a string and checks for serial interface errors. */
/* An "end of string" (line feed) character is added. */
/* If a serial interface error or timeout occurs, an error recovery routine */
/* (serialerror) is called. */
/* Erc is an error code so that the location of the error can be displayed */
/* to aid in troubleshooting. */
/* This function returns the value of ssta. */
*****/

int send(char *s, int erc)
{
    int rv; /* return value = ssta for this send */

    strcpy(wrt,s);
    strcat(wrt,"\n");
    rv = swrt(wrt,strlen(wrt));

    /* Check for error. */
    if (rv) serialerror(erc);
    return(rv);
}

/*****
/*          receive() -- receive a string from the TP04010A          */
/*          */
/* This function receives a string and checks for serial interface errors. */
/* If a serial interface error or timeout occurs, an error recovery routine */
/* (serialerror) is called. */
/* Erc is an error code so that the location of the error can be displayed. */
/* This function returns the value of ssta. */
/* The returned string is in the global variable rd[] */
*****/

int receive(int erc)
{
    int rv; /* return value = ssta for this receive */

    rv = srd(rd,256); /* read up to 256 characters from the TP04010A */

    /* Check for error. Bit 7 set = interface timeout, other bits = error. */
    if (rv)
    {
        serialerror(erc);
        strcpy(rd,""); /* make the returned string a null string */
    }
    else /* if a valid string returned */
        strtok(rd,"\r\n"); /* Replace the terminating carriage return or */
        /* line feed with a null. */
    return(rv);
}

```

```

/*****
/*      serialpoll() -- do a serial poll of the TP04010A      */
/*      */
/* This function sends a serial poll query to the TP04010A and returns the */
/* integer value of the response. It also clears the srq received flag. */
int serialpoll(void)
{
    char sprd[8];          /* serial poll response string */

    srqreceived = 0;      /* reset service request received flag */
    swrt("%S?\n",4);
    if (!(srd(sprd,7))) /* if no error */
        return(atoi(sprd));
    else return(0);
}

/*****
/*      checkforsrq() -- see if the TP04010A is requesting service */
/*      */
/* This function checks to see if any character has been sent by the */
/* TP04010A. If the character is not the service request indicator (^), */
/* then the character will be lost. Therefore this function should not be */
/* used when there is the possibility that the TP04010A will send any */
/* other character. */
/* This function returns nonzero if a service request character has been */
/* sent by the TP04010A. */
int checkforsrq(void)
{
    if (srqreceived) return(1); /* Done if the srq character has already */
    /* ..... /* been received by the srd() function. */

    inregs.x.dx = comport-1;
    inregs.h.ah = 3;          /* serial interface status function */
    int86(0x14, &inregs, &outregs);
    if (outregs.h.ah & 1) /* if a character has been received */
    {
        inregs.h.ah = 2; /* get the character */
        int86(0x14, &inregs, &outregs);
        if (outregs.h.al == '^') /* if srq detected */
        {
            srqreceived = 1; /* set global variable */
            return(1); /* done */
        }
    }
    return(0); /* if here, no srq */
}

```

```
/*-----*/
/*      initialize() -- initialize the TP04010A serial interface      */
/*      */
/* This function should be called at the beginning of every program to */
/* ensure that the temperature controller will be able to receive and */
/* process serial interface commands.                                  */
/* The following procedure (as illustrated in this function) should be */
/* followed to reliably initialize the TP04010A serial interface:    */
/* 1. Send a ! (device clear) and receive it correctly echoed back 2  */
/*    times in succession.                                           */
/* 2. Send !RM (remote mode)                                         */
/* 3. Send *CLS                                                       */
/* 4. Send *RST                                                       */
/* 5. Send *STB? followed by a read of the response from the TP04010A */
/*-----*/

void initialize(void)
{
    /* Send ! (device clear) command to reset the TP04010A serial host */
    /* interface. The interface input and output buffers are cleared. */
    if (deviceclear()) printf("Error during device clear\n");

    /* Send !RM to put into remote mode and *CLS (clear status) to clear */
    /* the TP04010A's status registers */
    send("!RM;*CLS",3);

    /* Send *RST (machine function reset) command to force the TP04010A to */
    /* the "Main" screen. This command will take several seconds. */
    send("*RST",4);

    /* Ask for the TP04010A's status byte. The response will not be sent */
    /* immediately because of the time it takes for the previous *RST */
    /* command to complete. */
    send("*STB?",5);
    receive(6); /* get the response back in rd[] */
    if ((atoi(rd) & 0x90) != 0x90) /* if "ready" bit in status byte not set */
        printf("TP04010A not ready\n");
}

```

```

/*****/
/*      cleardeverror() -- clear TP04010A-specific errors      */
/*      */
/* This function uses the CLER (clear error) command to try to clear */
/* certain errors that are related to the temperature control functions of */
/* the TP04010A. These errors are reported back in response to the EROR? */
/* query. Three errors must be cleared by the CLER command -- overheat, */
/* air open loop, and purge heat failure. The other errors will clear */
/* themselves automatically when the problem is corrected. Calling this */
/* function for those errors that are automatically cleared is not */
/* necessary, but may provide a convenient way of waiting a few seconds */
/* and then rereading the device error register so that transient errors */
/* can be ignored. */

```

```

void cleardeverror(void)
{
    time_t st;

    errcount++; /* increment the error count */
    send("CLER",7);

    /* wait 3-4 seconds to make sure CLER has taken effect */
    delay(4);
}

```

```

/*****/
/*      poll_temp_demo() -- demonstration of temperature polling      */
/*      */
/* One conceptually simple method of waiting until a temperature setpoint */
/* has been reached is to periodically ask for the current temperature */
/* with the TEMP? query. To ensure that the TP04010A is functioning */
/* properly, the device error register should also be read with the EROR? */
/* query. */

```

```

void poll_temp_demo()
{
    char setpointstr[16];
    float setpoint;
    printf("Temperature polling demonstration. Target = 70\n");
    setpoint=70.0;
    /* Turn off all reasons for a service request (not used for this demo), */
    /* use setpoint 0 (Hot setpoint), and put the */
    /* head down (which also turns the airflow on). These commands need */
    /* only be sent once. */
    send("SRE 0;SETN 0;HEAD 1",10);

    /* Convert the desired setpoint to a string and send to the TP04010A */
    sprintf(setpointstr,"SETP %1.1f",setpoint); /* setpoint string */
    send(setpointstr,11);
}

```



```
/* Wait up to 1 minute to get to temperature, sampling every second. */
/* When the temperature is at the setpoint +/- 3 degrees, display an */
/* "at temperature" message. */
errcount = 0;          /* reset the error counter */
starttime = time(NULL); /* start of one minute period */
do
{
    delay(2); /* wait here until 1 second has elapsed */

    /* Check for TP04010A-specific errors (such as overheat) */
    /* If an error is detected, try clearing it. Give up after three */
    /* unsuccessful error-clearing attempts. */
    send("EROR?",12);
    receive(13);          /* get the reponse in rd[] */
    error = atoi(rd);     /* convert string to an integer */
    if (error) clearerror(); /* try to clear a TP04010A-specific error*/
    if (errcount = 3)     /* if error cannot be cleared in 3 tries */
    {
        printf("Unclearable TP04010A-specific error. EROR = %u. ",error);
        send("HEAD 0",14); /* head up and flow off */
        printf("Program aborting.\n");
        exit(1);
    }

    attemperature = 0; /* assume not at temperature */

    /* read the temperature */
    send("TEMP?",15);
    if (!receive(16)) /* get the response, and if valid */
    {
        temp = atof(rd); /* convert temperature to floating point */
        printf("Temperature = %s\n",rd);
        if ((temp = setpoint-3) && (temp <= setpoint+3)) attemperature=1;
    }
}
while (((time(NULL)-starttime)<) && !attemperature);

if (attemperature) printf("At temperature\n\n");
else printf("Did not reach temperature in 1 minute.\n\n");
}
```

```

/*****
/*      poll_statusbyte_demo() -- demonstration of status byte polling      */
/*      */                                                                    */
/* Both the "at temperature" and error conditions of the TP04010A can be  */
/* simply and reliably monitored by periodically polling the status byte  */
/* with the *STB? query.                                                    */
*/
*/

void poll_statusbyte_demo()
{
    int seer;                /* standard event status register value */
    char setpointstr[16];
    float setpoint;
    printf("Status byte polling demonstration. Target = 100\n");
    setpoint=100.0;
    /* Clear the status registers so that subsequent errors can be detected.*/
    /* Turn off all reasons for a service request (not used for this demo).*/
    /* Turn on the error bits (command, execution, device dependent, query)*/
    /* of the standard event status enable register. Turn on the "at      */
    /* temperature" bit of the temperature event status enable register.  */
    /* Put the head down (which also turns the airflow on). Use setpoint  */
    /* #0 (Hot setpoint). These commands need only be sent once.          */
    send("*CLS:*SRE 0:*ESE 60:TESE 1:HEAD 1:SETN 0",17);

    /* Set the soak time to 10 seconds, the temperature tolerance window to */
    /* +/- 2 degrees. These commands need only be resent when they change. */
    send("SOAK 10;WNDW 2",18);

    /* Convert the desired setpoint to a string and send to the TP04010A */
    sprintf(setpointstr,"SETP %1.1f",setpoint); /* setpoint string */
    send(setpointstr,19);

    /* Resend the setpoint number. This forces a restart of the soak time */
    /* interval, even if the value of the new setpoint is the same as the */
    /* previous setpoint. As long as the soak time is set to a nonzero    */
    /* value, this in turn forces the temperature condition register      */
    /* "at temperature" bit off. This is needed to guarantee that there   */
    /* will be a temperature condition register transition from           */
    /* "not at temp" to "at temp." It is this transition that sets the    */
    /* "at temperature" bit in the temperature event status register.     */
    /* Even if the soak time is set to zero, a "not at temp" to "at temp" */
    /* transition of the temperature condition register will occur if the  */
    /* setpoint number is changed, or if the old temperature reading is   */
    /* out of the window (tolerance) of the new value, and the new       */
    /* setpoint number or value is not immediately changed back to the old */
    /* one.                                                                 */
    /*

    /* The TESR? (temperature event status register) query is used to get  */
    /* rid of a possible preexisting "at temp" condition in the temperature*/
    /* event status register. (Do not confuse the condition and status    */
    /* registers.) Reading the register also clears it. The value         */
    /* read is not used. Both SETN (with the appropriate setpoint number) */
    /* and TESR? should be sent after every new setpoint to ensure that an */
    /* "at temperature" status will never be missed or erroneously       */
    /* indicated under unusual circumstances.                             */
    */
}

```

```
send("SETN 0;TESR?",20);
receive(21); /* must read the result of the TESR? query */

/* Wait up to 1 minute to get to temperature, reading the status byte */
/* every second. */
attemperature = 0; /* reset "at temperature" flag */
errcount = 0; /* reset the error counter */
starttime = time(NULL); /* start of one minute period */
do
{
    delay(2);

    send("**STB?",22); /* ask for the status byte */
    receive(23); /* get the response in rd[] */
    stb = atoi(rd); /* convert string to an integer */

    /* Check for a command, execution, device dependent or query error. */
    /* The most likely cause is an error during initialization. */
    if (stb & 32) /* See if the standard event summary bit in */
    /* ..... /* the status byte is set. */
    {
        send("**ESR?",24); /* read the error (which also clears it) */
        receive(25);
        sesr = atoi(rd);
        printf("Standard event error = %d.\n",sesr);
    }

    /* Check for TP04010A-specific errors (such as overheat). If an */
    /* error is detected, try clearing it. Give up after three */
    /* unsuccessful error-clearing attempts. */
    if (stb & 4) /* if the device-specific error bit is set */
    {
        send("ERROR?",26); /* reading the condition reg does NOT clear it*/
        receive(27);
        error = atoi(rd);
        cleardeverror(); /* try to clear the error */
    }
    if (errcount = 3) /* if error cannot be cleared in 3 tries */
    {
        printf("Unclearable TP04010A-specific error. ERROR = %u. ",error);
        send("HEAD 0",28); /* head up and flow off */
        printf("Program aborting.\n");
        exit(1);
    }

    /* If the temperature event summary bit is set, "at temperature" */
    /* must have occurred because only the "at temp" bit of the */
    /* temperature event status enable register is enabled. */
    if (stb & 8) attemperature = 1;
}
while (((time(NULL)-starttime)<) && !attemperature);
```

```

    if (attemperature) printf("At temperature\n\n");
    else printf("Did not reach temperature in 1 minute.\n\n");
}

/*****
/*  srq_demo() -- demonstration of service request when "at temperature"  */
/*  */
/*  The service request character (^) the serial interface can be used to  */
/*  signal when the desired temperature is reached without continually  */
/*  polling the TP04010A.  */
void srq_demo()
{
    int sse;          /* standard event status register value */
    char setpointstr[16];
    float setpoint;
    printf("Service request demonstration. Target = 90\n");
    setpoint=90.0;
    /* Temporarily disable all reasons for a service request and clear */
    /* status registers. */
    send("SRE 0;CLS",29);

    /* do a serial poll to clear any leftover pending service request */
    spr = serialpoll();

    /* Turn on the error bits (command, execution, device dependent, query)*/
    /* of the standard event status enable register. Turn on the "at  */
    /* temperature" bit of the temperature event status enable register.  */
    /* Put the head down (which also turns the airflow on). Use setpoint  */
    /* #0 (Hot setpoint). These commands need only be sent once. */
    send("ESE 60;TESE 1;HEAD 1;SETN 0",30);

    /* Set the soak time to 10 seconds, the temperature tolerance window to */
    /* +/- 3 degrees. These commands need only be resent when they change. */
    send("SOAK 10;WNDW 3",31);

    /* Convert the desired setpoint to a string and send to the TP04010A */
    sprintf(setpointstr,"SETP %1.1f",setpoint); /* setpoint string */
    send(setpointstr,32);

    /* Resend the setpoint number. This forces a restart of the soak time  */
    /* interval, even if the value of the new setpoint is the same as the  */
    /* previous setpoint. As long as the soak time is set to a nonzero  */
    /* value, this in turn forces the temperature condition register  */
    /* "at temperature" bit off. This is needed to guarantee that there  */
    /* will be a temperature condition register transition from  */
    /* "not at temp" to "at temp." It is this transition that sets the  */
    /* "at temperature" bit in the temperature event status register.  */
    /* Even if the soak time is set to zero, a "not at temp" to "at temp"  */

```

```
/* transition of the temperature condition register will occur if the */
/* setpoint number is changed, or if the old temperature reading is */
/* out of the window (tolerance) of the new value, and the new */
/* setpoint number or value is not immediately changed back to the old */
/* one. */

/* The TESR? (temperature event status register) query is used to get */
/* rid of a possible preexisting "at temp" condition in the temperature*/
/* event status register. (Do not confuse the condition and status */
/* registers.) Reading the register also clears it. The value */
/* read is not used. Both SETN (with the appropriate setpoint number) */
/* and TESR? should be sent after every new setpoint to ensure that an */
/* "at temperature" status will never be missed or erroneously */
/* indicated under unusual circumstances. */
send("SETN 0;TESR?",33);
receive(34); /* must read the result of the TESR? query */

/* Enable the standard event, temperature event, and device error bits */
/* in the service request enable register. An SRQ (service request) */
/* can occur any time after this command is sent. */
send("SRE 44",35);

/* Wait up to 1 minute for an "at temp" service request to occur. */
attemperature = 0; /* reset flag */
errcount = 0; /* reset the error counter */
starttime = time(NULL);
do
{
/* Call checkforsrq(). The result returned is nonzero if an SRQ */
/* was received. If the SRQ input of your system is connected to */
/* a hardware interrupt (it is not in this demo), the interrupt */
/* service routine should only set a flag to show that a service */
/* request is pending. Commands should not be sent to the */
/* TP04010A from within the interrupt service routine because the */
/* SRQ might have occurred while the computer was in the middle of */
/* sending some other string. The resulting mixture would be an */
/* illegal string. */
if (checkforsrq()) /* if srq has occurred */
{
printf("SRQ received\n");
/* Do a serial poll. If an error occurs, the poll */
/* response will be 0. */
spr = serialpoll();
/* Check for TP04010A-specific errors (such as overheat). */
/* If an error is detected, try clearing it. Give up after 3 */
/* unsuccessful error-clearing attempts. */
if (spr & 4) /* if the device-specific error bit is set */
{
.. send("EROR?",36);
.. receive(37);
.. error = atoi(rd);
.. cleardeverror();
}
if (errcount = 3) /* if error cannot be cleared in 3 tries */
{
```

```

..printf("Unclearable TP04010A-specific error.  ERROR = %u.  ",
..      error);
..send("HEAD 0",38); /* head up and flow off */
..printf("Program aborting.\n");
..exit(1);
}

/* Check for a standard event status register error.  The most */
/* likely cause is an error during this demo's initialization. */
if (spr & 32)
{
..send("*ESR?",39); /* read the error (which also clears it) */
..receive(40);
..sesr = atoi(rd);
..printf("Standard event error = %d.\n",sesr);
}

/* If the temperature event summary bit is set, read the */
/* temperature event status register to verify the cause of */
/* the interrupt.  Actually, since only the "at temperature" */
/* bit is enabled in this demo, the read is not necessary. */
if (spr & 8)
{
..send("TESR?",41); /* Read the temperature event status reg */
..... /* (reading the register also clears it) */
..receive(42);
..tesr = atoi(rd); /* make into an integer */
..if (tesr & 1) attemperature = 1; /* if "at temp" bit is set */
}
}
}
while ((time(NULL)-starttime)< 60 && !attemperature);
..... /* wait for up to 1 minute */

if (attemperature) printf("At temperature\n\n");
else printf("Did not reach temperature in 1 minute.\n\n");
}

```

```
.....  
/* multitemp_demo() -- demonstration of multiple temperature setpoints */  
/* */  
/* This function illustrates how a tester might preload the temperature */  
/* setpoints with several different temperature values, and then step from */  
/* one temperature to another as desired. The TP04010A's status byte is */  
/* polled, and the temperature event summary bit is used to give a "begin */  
/* test" signal to the tester. */  
  
/* Initialization routine for the multitemp demo. It returns non-zero if */  
/* an error occurred. */  
  
int multitemp_init(void)  
{  
    int sesr;          /* standard event status register */  
  
    /* Clear the status registers (so that an error occurring during this */  
    /* routine may be detected), disable all service requests, turn on */  
    /* the "at temperature" bit of the temperature event status enable */  
    /* register, and put the head up (which also turns the air flow off). */  
    /* Turn the compressor ON. */  
    send("CLS;SRE 0;TESR 1;HEAD 0;COOL 1",43);  
  
    /* Wait for "head up" command to take effect. For TP04010A software */  
    /* versions 0.62 and earlier, a minimum of one second must separate */  
    /* any of the following four commands to guarantee proper head and */  
    /* flow operation: head up, head down, flow on, flow off. For */  
    /* versions FG70090F and later, no separation between commands is */  
    /* necessary, but because of pneumatic delays, it will still take a */  
    /* second or two for the head to actually move and the flow to start. */  
    delay(2);  
  
    /* Load parameters into the setpoints that will be used */  
    send("SETN 2;SETP 15;WNDW 3;SOAK 10",44);  
    send("SETN 1;SETP 25;WNDW 3;SOAK 10",45);  
    send("SETN 0;SETP 125;WNDW 3;SOAK 10",48);  
  
    /* Check the standard event status register to make sure that the */  
    /* previous commands were executed without error. */  
    send("ESR?",49);  
    receive(50);      /* get the response in rd[] */  
    sesr = atoi(rd);  /* convert string to an integer */  
  
    /* If receive() returned a null string to indicate an error, or if */  
    /* there was a command syntax error, or the command could not be */  
    /* executed, or there was a device dependent or query error, */  
    /* return nonzero. */  
    if ((rd[0]==0) || (sesr & 60)) return(1);  
    else return(0);  
}
```

```

void multitemp_demo(void)
{
    int sn;           /* setpoint number */
    int tryagain;    /* nonzero if want to retry a setpoint after error */
    char sendstr[32];

    printf("Multiple temperature setpoint demonstration\n");

    if (multitemp_init() /* initialize TP04010A setpoints, etc. */
        /* if unsuccessful */
        send("**RST",51); /* reset the system */
        if (multitemp_init() /* try again */
            {
                printf("Could not initialize. Demonstration skipped.\n\n");
                return;
            }
        )
    {
        send("HEAD 1",52); /* put the head down (which also turns the flow on) */
        tryagain=1; /* want to retry each setpoint after an error */

        /* for each setpoint */
        for (sn=0; sn<=2; sn++)
        {
            /* Convert the desired setpoint to a string & send to the TP04010A. */
            /* Clear the status registers so that a new change in status (such */
            /* as getting to temperature) will show up in the status byte and */
            /* thus be detected. Since this part of the demo does not check */
            /* for standard event errors (see explanation below), the *CLS */
            /* command which clears both the standard event and temperature */
            /* event registers can be used instead of TESR? followed by a read.*/
            sprintf(sendstr,"SETN %d;*CLS",sn);
            send(sendstr,53);

            /* Wait up to 5 minutes to get to temperature, reading the status */
            /* byte every second. */
            atemperature = 0; /* reset "at temperature" flag */
            errcount = 0; /* reset the error counter */
            starttime = time(NULL); /* start of 5 minute period */
            do
            {
                delay(2); /* wait here until 1 second has elapsed */

                send("**STB?",54);
                receive(55); /* get the response in rd[] */
                stb = atoi(rd); /* convert string to an integer */

                /* Check for TP04010A-specific errors (such as overheat). */
                /* IF an error is detected, try clearing it. */
                if (stb & 4) /* if the device-specific error bit is set */
                {
                    ..send("ERROR?",56);
                    ..receive(57);
                    ..error = atoi(rd);
                    ..cleardeverror();
                }
            }
        }
    }
}

```



```
        if (errcount = 3) /* if error cannot be cleared in 3 tries */
        {
.. printf("Unclearable TP04010A-specific error. EROR = %u. ",
.. error);
.. send("HEAD 0",58); /* head up and flow off */
.. printf("Program aborting.\n");
.. exit(1);
        }

        /* Note that the standard event summary bit of the status byte */
        /* is ignored in this routine. If an error occurs, it is */
        /* either a transient error that can be ignored, or the system */
        /* will not reach temperature and the setpoint will be retried.*/

        /* If the temperature event summary bit is set, "at temperature"*/
        /* must have occurred because only the "at temp" bit of the */
        /* temperature event status enable register is enabled. */
        if (stb & 8) attemperature = 1;
    }
while (((time(NULL)-starttime),) && !attemperature);

if (attemperature)
{
    tryagain=1; /* allow the next setpoint to be retried */
    send("TEMP?",59); /* get the exact temperature */
    receive(60);
    printf("At temperature. Temperature = %s\n",rd);
    /* PERFORM YOUR TEST HERE */
}
else
{
    printf("Did not reach temperature in 5 minutes.\n");
    if (tryagain) /* retry the same setpoint one more time */
    {
.. printf("Retrying...\n");
.. tryagain=0; /* only retry each setpoint once */

.. /* It may be that the device needs more than the allotted */
.. /* time to reach the desired temperature. If so, just */
.. /* retrying the same setpoint is sufficient. But if the */
.. /* problem was that operator interference or erroneous */
.. /* remote mode commands or noise changed some of the */
.. /* TP04010A settings, a reset command will restore the */
.. /* system to a known state. */
.. send("**RST",61); /* reset the system */
    }
}
```

```

.. /* Reinitialize for this routine. The *RST command does */
.. /* not change any of the setpoint values, but it does */
.. /* force the system to the "Main" screen. */
.. multitemp_init();

.. /* *RST raises the head, so it must be lowered again */
.. send("HEAD 1",62);

.. sn--; /* retry the same setpoint number */
   |
   | else tryagain=1; /* allow a retry at the next setpoint */
   |
}

/* Put the head up */
send("HEAD 0",63);

printf("\n");

/* As an alternative to preloading several setpoints during initial- */
/* ization, the same setpoint number can be reused by sending new */
/* values for the desired temperature, soak time, etc. That way, */
/* there is no limit to the number of test temperatures. For */
/* example: */
/* */
/*     send("SETN 0;SETP 75;WINDW 2;SOAK 10;SETN 0",99); */
/* */
/* The setpoint number (if changed) must be sent at the beginning of */
/* the string so that the setpoint, window and soak values will apply */
/* to the desired setpoint number. The setpoint number is sent at */
/* the end so that the soak time will be reset to the value just sent. */
/* See the poll_statusbyte_demo() for more detail. */

```

```
.....  
/*          cycle_demo() -- demonstration of cycle mode          */  
/*          */  
/* This function illustrates how a tester might preload the temperature */  
/* setpoints with several different temperature values, and then use cycle */  
/* mode to automatically step (in order) from one setpoint to the next. */  
/* The TP04010A's status byte is polled, and the temperature event summary */  
/* bit is used to give a "begin test" signal to the tester. */  
/* Note that this function offers less flexibility to the tester than the */  
/* multitemp_demo() function. Cycle mode is most useful when the TP04010A */  
/* is controlling a "dumb" tester. */  
  
/* Initialization routine for the cycle demo. It returns non-zero if */  
/* an error occurred. */  
  
int cycle_demo_init(void)  
{  
    int sesr;          /* standard event status register */  
  
    /* Check the standard event status register to make sure that the */  
    /* previous commands were executed without error. */  
  
    /* Clear the status registers (so that an error during this routine may */  
    /* be detected), disable all service requests. Turn on the "at temp" */  
    /* and "end of all cycles" bits of the temperature event status enable */  
    /* register, so that either of these occurrences will set the */  
    /* temperature event summary bit in the status byte. Put the head up */  
    /* (and turn the airflow off), and go into ramp mode. Set the maximum */  
    /* test time to 1 minute, and set up to do 2 cycles. */  
    send("HEAD 0",164);  
    delay(2);  
    send("*SRE 0;TESE 17;RMPC 1;TTIM 60;CYCC 2;*CLS",64);  
  
    /* Because of pneumatic delays, it takes a */  
    /* second or two for the head to actually move and the flow to start. */  
    delay(2); /* wait two seconds */  
  
    /* Load parameters into the setpoints that will be used */  
    /* A ramp rate of 9999 means transition the temperature as fast as */  
    /* possible. */  
    send("SETN 2;RAMP 9999;SETP 10;WNDW 3;SOAK 10",65);  
    delay(2); /* wait two seconds */  
    send("SETN 1;RAMP 9999;SETP 25;WNDW 3;SOAK 10",67);  
    delay(2); /* wait two seconds */  
    send("SETN 0;RAMP 9999;SETP 125;WNDW 3;SOAK 10",69);  
    delay(2); /* wait two seconds */  
  
    /* Set the ramp rates of the unused setpoints to 0 so that they will be */  
    /* skipped. */  
  
    /* Check the standard event status register to make sure that the */  
    /* previous commands were executed without error. */  
    send("*ESR?",77);  
    receive(78); /* get the response in rd[] */
```

```

sesr = atoi(rd);      /* convert string to an integer */

/* If receive() returned a null string to indicate an error, or if */
/* there was a command syntax error, or the command could not be */
/* executed, or there was a device dependent or query error,      */
/* return nonzero. */
if ((rd[0]==0) || (sesr & 60)) return(1);
else return(0);
}

void cycle_demo(void)
{
    int allcyclesdone;

    printf("Cycle mode demonstration\n");

    if (cycle_demo_init())      /* Initialize TP04010A setpoints, etc. */
    {                          /* if unsuccessful */
        send("**RST",79);      /* reset the system */
        if (cycle_demo_init()) /* try again */
        {
            printf("Could not initialize. Demonstration skipped.\n\n");
            return;
        }
    }

    /* Put the head down, start cycling mode, and clear status registers */
    /* (so that the new temperature events will be detected). Since this */
    /* part of the demo does not check for standard event errors (see the */
    /* explanation below), the *CLS command which clears both the standard */
    /* event and temperature event registers can be used instead of TESR? */
    /* followed by a read. */
    send("HEAD 1;CYCL 1;*CLS",80);

    allcyclesdone = 0; /* flag -- becomes 1 when all cycles are done */

    do
    {
        /* Wait up to 5 minutes to get to temperature, reading the status */
        /* byte every second. */
        atemperature = 0; /* reset "at temperature" flag */
        errcount = 0; /* reset the error counter */
        starttime = time(NULL); /* start of 5 minute period */
        do
        {
            delay(2); /* wait here until 1 second has elapsed */

            send("**STB?",81);
            receive(82); /* get the response in rd() */
            stb = atoi(rd); /* convert string to an integer */

            /* Check for TP04010A-specific errors (such as overheat). */
            /* IF an error is detected, try clearing it. */
            if (stb & 4) /* if the device-specific error bit is set */
            {

```

```

.. send("EROR?",83);
.. receive(84);
.. error = atoi(rd);
.. cleardeverror();
  }
  if (errcount = 3) /* if error cannot be cleared in 3 tries */
  {
.. printf("Unclearable TP04010A-specific error. EROR = %u. ",
.. error);
.. send("HEAD 0",85); /* head up and flow off */
.. printf("Program aborting.\n");
.. exit(1);
  }

/* Note that the standard event summary bit of the status byte */
/* is ignored in this routine. If an error occurs, it is */
/* either a transient error that can be ignored, or the system */
/* will not reach temperature. */

/* If the temperature event summary bit is set, see if the */
/* "at temperature" and/or "end of all cycles" bits are set. */
if (stb & 8)
{
.. send("TESR?",86); /* Read the temperature event status reg */
..... /* (reading the register also clears it) */
.. receive(87);
.. tesr = atoi(rd); /* make into an integer */
.. if (tesr & 1) attemperature = 1;
.. if (tesr & 16) allcyclesdone = 1;
}
}
while (((time(NULL)-starttime.) && !attemperature
&& !allcyclesdone);

if (allcyclesdone)
  printf("All temperature cycles done\n");
else
  {
    if (attemperature)
    {
.. send("TEMP?",88); /* get the exact temperature */
.. receive(89);
.. printf("At temperature. Temperature = %s\n",rd);
.. /* PERFORM YOUR TEST HERE */
    }
    else
    {
.. printf("Did not reach temperature in 5 minutes.\n");

.. /* Not much can be done except to restart the whole cycling */
.. /* process from the beginning. When cycling, a particular */
.. /* setpoint cannot be retried. A reset (*RST) will take */
.. /* the system out of cycling mode, so you will need to */
.. /* resend at least RMPC 1;HEAD 1;CYCL 1 to restart cycling.*/
    }
  }

```

```
        /* Step to the next temperature */  
        send("NEXT",90);  
    }  
    }  
    while (!allcyclesdone);  
  
    /* put the head up and end cycling mode */  
    send("HEAD 0;CYCL 0;RMPC 0",91);  
}
```

```
.....  
/*                               main()                               */  
  
int main(int argc, char *argv[])  
{  
    int i;  
  
    comport = 1;      /* default com port = COM1: */  
    baudrate = 9600; /* default baud rate = 9600 */  
    timeout = 1;     /* default timeout multiplier = 1: */  
  
    printf("TP04010A RS232 SERIAL INTERFACE DEMONSTRATION PROGRAM ver 2.00");  
    printf("\n\n");  
  
    /* if command line arguments were given, set COM port and/or baud rate */  
    for (i=1; i<argc; i++)  
    {  
        if (toupper(argv[i][0]) == 'T') /* if a timeout multiplier */  
        {  
            timeout = atoi(argv[i][1]); /* Make rest of string a number */  
            if (timeout < 1 || timeout > 8) timeout=1; /* only 1-8x ok */  
        }  
        else switch (atoi(argv[i])) /* convert argument string to integer */  
        {  
            case 1: /* if argument is 1-4, assume it specifies a COM: port */  
            case 2:  
            case 3:  
            case 4: comport = atoi(argv[i]);  
            ..      break;  
            case 300: /* if one of these, assume it specifies a baud rate */  
            case 600:  
            case 1200:  
            case 2400:  
            case 4800:  
            case 9600: baudrate = atoi(argv[i]);  
        }  
    }  
    /* Check that the RS232 serial interface card is present in the pc, */  
    /* and initialize it. A nonzero return means there was an error. */  
    if (pcserialinit(comport, baudrate))  
    {  
        printf("\nPC serial interface not present or could not be initialized.");  
        printf(" Program aborting.\n");  
        exit(1);  
    }  
  
    /* Initialize the TP04010A's serial interface. This should be done at */  
    /* the beginning of every program to ensure that the temperature */  
    /* controller will be able to receive and process interface commands. */  
    initialize();  
}
```

```
/* The following functions illustrate 3 different methods of waiting */
/* until one desired airstream temperature has been reached. */

poll_temp_demo();          /* poll temperature and error registers */
.....

poll_statusbyte_demo();   /* poll the status byte -- RECOMMENDED */
.....

srq_demo();               /* wait for a service request */
.....

/* The following functions illustrate 2 different methods of stepping */
/* from one preset temperature to another under control of the host */
/* system. */

multitemp_demo();

cycle_demo();

/* Return the TP04010A to local control */
send("**STB?",92);        /* Wait for any pending commands to complete by */
receive(93);             /* waiting here until a response comes back from */
..... /* the TP04010A. */
send("%GL",94);          /* return TP04010A to local control */

printf("\nEND\n");
return(0);
)
```



```

/*****
/*
/*          RS232 SERIAL INTERFACE CABLE INFORMATION          */
/*
/*
/* The RS232 serial interface of the TP04010A is configured as "data
/* terminal equipment" (DTE). The connector on the system I/O panel is a
/* 25 pin male D connector (DB25P). The pinout is as follows:
/*
/*
/* PIN #          FUNCTION
/* 1      Chassis ground
/* 2      Serial data out
/* 3      Serial data in
/* 4      RTS (request to send) output -- always high
/* 5      CTS (clear to send) input -- must be high
/* 6      DSR (data set ready) handshaking input -- The TP04010A will
/* stop sending output when this input goes low, and will resume
/* when it returns to high.
/* 7      Signal ground
/* 20     DTR (data terminal ready) handshaking output -- This output
/* will go low when the TP04010A input buffer has filled and it
/* cannot accept new commands.
/*
/*
/*
/* Because the serial port on an IBM-compatible pc is also configured as
/* DTE, a "null modem" cable (with female connectors on both ends) is
/* needed to connect a pc to the TP04010A. A pc/xt requires a 25 pin
/* female D connector at the pc end, while a pc/at requires a 9 pin
/* female D connector. The wiring for the cable is as follows:
/*
/*
/* TP04010A END PIN #    25 PIN PC END #    9 PIN PC END #
/* 1                      1                NO CONNECT
/* 2                      3                2
/* 3                      2                3
/* 4                      5                8
/* 5                      4                7
/* 6                      20               4
/* 7                      7                5
/* 20                     6                6
/*
/* As an alternative to wiring pins 4 and 5 of the TP04010A through the
/* cable of the pc, they may be connected to each other. This will have
/* no adverse effect on system operation. If for some reason the host
/* computer (the pc) is incapable of hardware handshaking using the DTR
/* and DSR signal lines, pins 4 and 6 of the TP04010A can be connected
/* together. In this case, care must be taken not to overflow the
/* TP04010A's 250-byte input and output buffers.
/*
/*****

```

Operator Notes

Use this page to record your notes. Refer to the "Readers Comment" card at the back of this manual to send any comments back to Temptronic.

Index

A

Air Chiller Module Defrosting, 5-15
Air Control, 3-3
Air Dryer Mounting, 2-6
Air Path Maintenance, 5-6
Air Requirements, 1-2
Assembly Instructions, 2-5

B

Back-Up Battery Replacement, 5-5
BIOS Setup, 5-17
Buck/Boost Transformer Mounting, 2-6

C

Calibration
 Calibration procedure, 5-12
 Calibration verification table, 5-11
 Verification procedure, 5-9
Compressed Air Interconnection, 2-3
Configuration Options, 3-33

D

Defrost Cycle, 5-15
Displays
 Air Thermocouple End Calibration Screen, 5-15
 Air Thermocouple High Calibration Screen, 5-14
 Air Thermocouple Low Calibration Screen, 5-14
 Defrost Cycle Screen, 5-16
 Main Calibration Screen, 5-13
 No DUT sensor selected error, 3-38
 Operator Change Setup Screen, 3-17
 Operator Extended Screen, 3-15
 Operator Load Setup Screen, 3-19
 Operator Main Screen, 3-12
 Operator Save Setup Screen, 3-18
 Operator Test Cycle Screen, 3-14
 Startup Sequence Screen, 3-11
 System Configuration Screen, 3-22, 4-1, 5-2
 Temptronic Logo Screen, 3-21
 Top Menu Screen, 3-30

V.S. Change Setup Screen, 3-27
V.S. Load Setup Screen, 3-29
V.S. Main Screen, 3-22
V.S. Ramp/Cycle Screen, 3-24
V.S. Save Setup Screen
V.S. Test Cycle Screen, 3-25

E

Electrical Controls, 3-1
Error Messages, 3-36
Exhaust Muffler Replacement, 5-8

F

FLOW control, 3-3

G

General Data, 1-1

H

Head and Manipulator
 Electrically controlled positioning, 3-6
 Manually controlled positioning, 3-3

I

IEEE-488 Demo Program, A-1
IEEE-488 Interface
 Mandatory IEEE-488.2 common commands and queries, 4-9
 Parallel bus interface parameters, 4-9
IEEE-488/RS232C Host Interface
 Device specific commands and queries, 4-4
 Parallel/serial programming, 4-3
Initial Start-Up, 2-11
Inspection and Cleaning, 5-1
Insulation Kit, 2-7

L

Local Operation Controls, 3-1

M

Maintenance Log, 5-2
MCT Interface, 4-2

O

Operator Control Module (OCM), 3-8
Operator Maintenance, 5-1

Operator Mode

Startup sequence, 3-11
Test procedure, 3-12

P

Packing Carton, 2-1
Parallel/Serial Programming, 4-3
Performance Report Forms, 1-3
Placement of TP04010A, 2-2
Power Interconnections, 2-2
Power Panel, 3-1
Power Requirements, 1-2
Preparation for Use, 2-1

R

Reader Comments Card, 1-3
Receipt of Shipment, 2-1
Registration Cards, 1-3
Remote Interfaces, 4-1
Repackaging, 2-12
RS232C Demo Program, B-1
RS232C Interface
 Serial interface connector, 4-10
 Serial interface parameters, 4-11
 Serial interface special commands, 4-11

S

Shipping Size, 2-1
Shipping Weight, 2-1

Shutdown Procedure, 3-10
Specifications, 1-2
Startup Procedure, 3-10
System Configuration
 Explanation of configuration options, 3-33
 System Configuration Screen, 3-32
System Interconnections, 2-6
System Interfacing
 Air purging, 2-10
 Direct DUT temperature sensing, 2-10
 ESD protection, 2-7
 IEEE-488 interface, 2-11
 MCT interface, 2-11
 Peripheral I/O panel ports, 2-11
 RS232C interface, 2-11
 ThermoStream interfacing, 2-7
System Operation, 3-1

T

Temptronic Service Department, 5-1
Test Setup
 Changing system setup parameters, 3-17
 Changing Temperature Setpoint Parameters, 3-19
 Loading a new test setup, 3-19
Thermal Cap Attachment, 2-5
Top Menu Selections
 Explanation of screen selections, 3-31
 Top Menu Screen, 3-30

U

Unpacking Instructions, 2-3
User Interface Performance Report, 1-3
User/Owner Registration Cards, 1-3

V

Variable Setup Mode
 Startup sequence, 3-21
 Test procedure, 3-22
 Test setup, 3-26

Warranty

Temptronic Corporation warrants all equipment of its manufacture to be free from defects in materials and workmanship for a period of one year from the date of shipment to the original buyer. The liability under this warranty is limited to replacement parts and labor on equipment when the equipment is returned prepaid to the factory or its authorized service center with prior authorization from Temptronic Corporation, and upon examination by Temptronic Corporation, is determined to be defective. At Temptronic Corporation's option, a service representative may be dispatched to the equipment location.

As an additional protection, Temptronic Corporation warrants that for a period of 90 days from the date of shipment to the original buyer, there will be no charge for service related shipping of parts and/or equipment or for authorized travel of a service representative to the equipment location. After 90 days, all costs incurred for shipping the equipment or parts thereof or for travel are the responsibility of the buyer. Our warranty for this equipment is rendered void if the unit has been repaired, taken apart or modified, or attempted to be, unless such actions have been taken in accordance with written instructions received from Temptronic Corporation. The warranty is also void if the equipment has been subjected to abuse, accident or other abnormal conditions.

IF ANY FAULT DEVELOPS, THE FOLLOWING STEPS SHOULD BE TAKEN:

1. Notify Temptronic Corporation by calling 1-800-558-5080. Overseas customers should contact the local Temptronic authorized service center. Please be prepared with the model number, serial number and full details of the difficulty. Upon receipt of this information, service data or shipping instructions will be provided by Temptronic Corporation. Do not return the unit for repair without first contacting the factory or its representative for instructions.
2. After the initial 90 day period, on receipt of shipping instructions, forward the equipment prepaid to the factory or its authorized service center as instructed. If requested, an estimate of the charges will be made before work begins, especially with those cases where the Temptronic Corporation product is not covered by the warranty.
3. If the original carton and packing are not available, the product should be packed in a container with a strong exterior and surrounded by a protective layer of shock-absorbing material. Temptronic Corporation advises returning the equipment at full value to the carrier.

Temptronic Corporation reserves the right to make changes in design at any time without incurring any obligation to install the same changes on units previously purchased.

This warranty states the essence of the obligations or liabilities on the part of Temptronic Corporation. **THE FORMAL, COMPLETE AND EXCLUSIVE STATEMENT OF TEMPTRONIC CORPORATION'S WARRANTY IS CONTAINED IN ITS QUOTATIONS, ACKNOWLEDGEMENTS AND INVOICES.** Temptronic Corporation neither assumes, nor authorizes any person to assume for it, any liability in connection with the sale of its equipment other than those set forth herein.

