Defrost Board

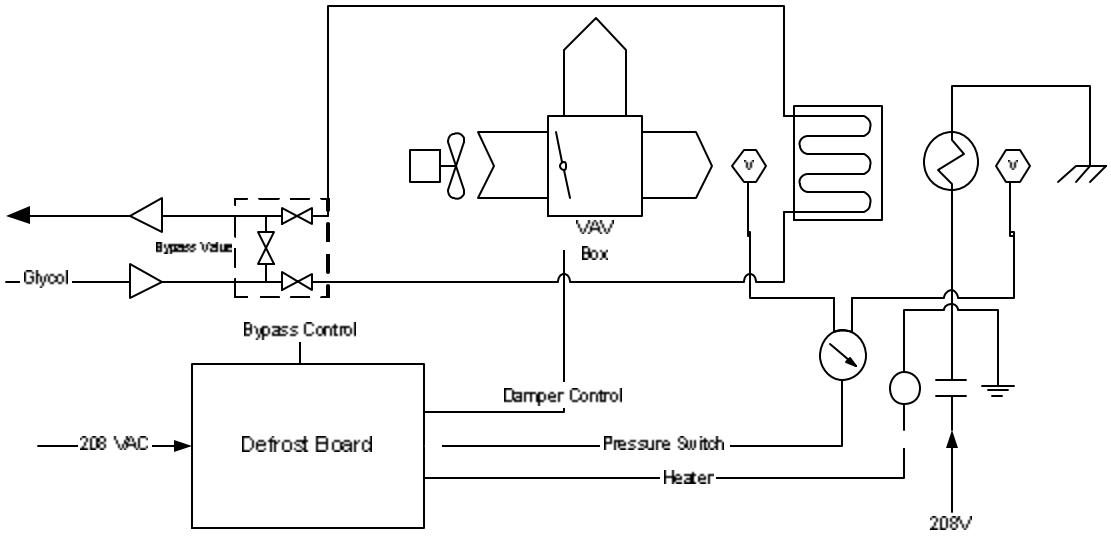
Hardware Design and Test Document

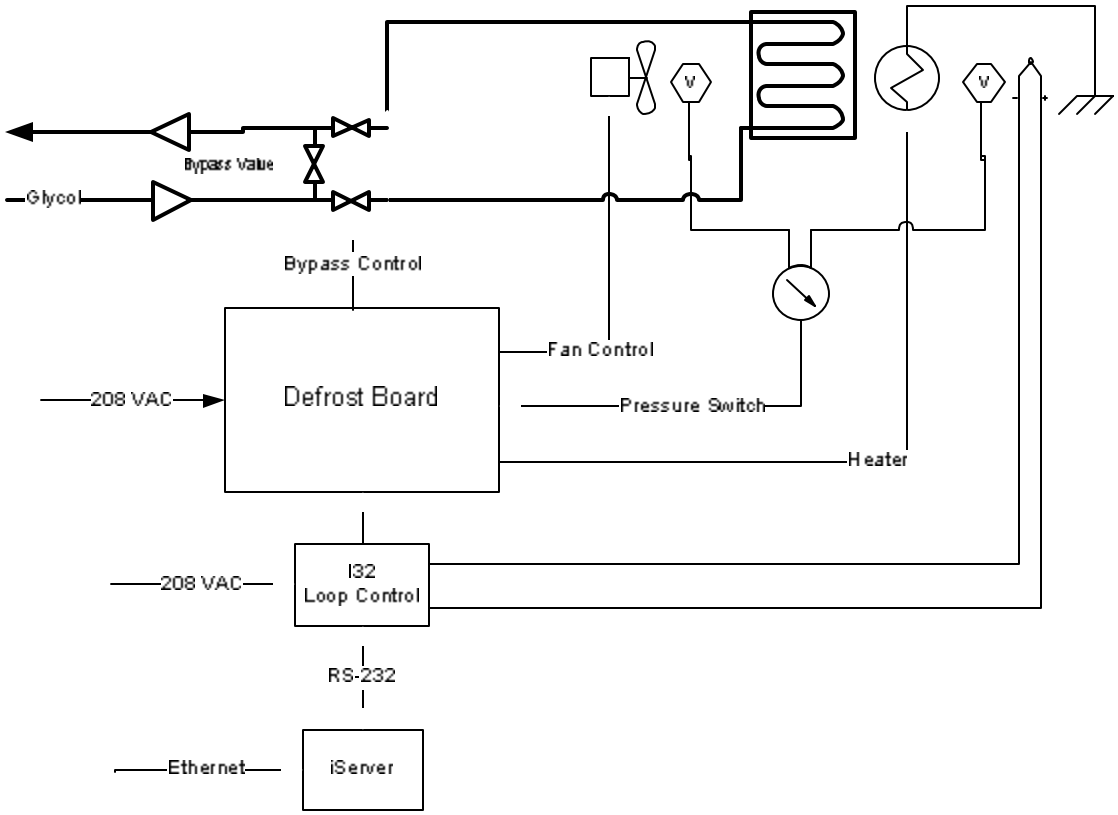
Revision History

Date	Revision	Author	Comment
December 4, 2003	New	Fred Keske	Initial Release
May 10, 2004	A	Fred Keske	Modified Figure 2
June 8, 2004	B	Fred Keske	Modified Figure 2

1 Defrost Overview

The defrost board is designed to control the defrosting of both the dome air handling unit and the mirror cooling unit. A defrost cycle occurs when the cooling coils ice up due to a high humidity condition. When the differential pressure transducer detects difference between the front of the cooling coil versus the back of the cooling coils microcontroller will take appropriate action depending on which cooling unit it is controlling. The two cooling diagrams are shown below in figures 1 and 2. Figure 1 is the Dome Air Handling Circuit and figure 2 is the Mirror Cooling Circuit.




Bypass Value
Glycol
Bypass Control
Defrost Board
208 VAC
Fan Control
Pressure Switch
Heater
208 VAC
I32
Loop Control
RS-232
Ethernet
iServer

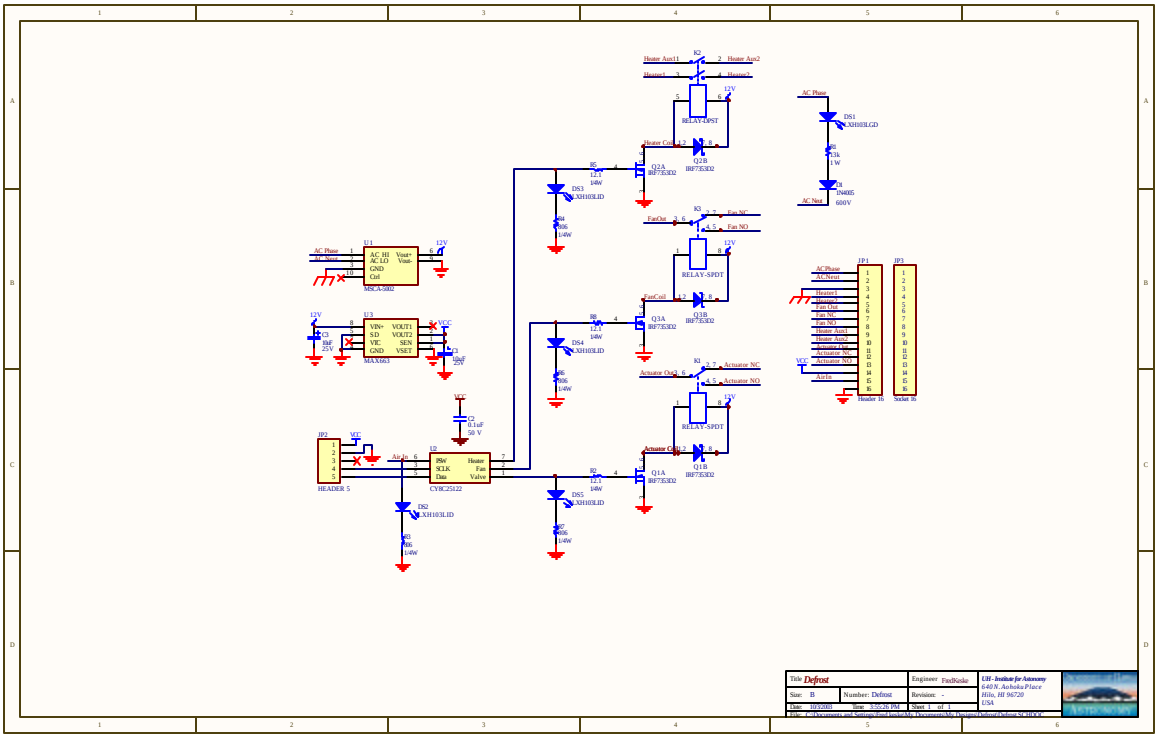


Figure 3 Defrost Circuit Schematic

1.2.1 Input Power

The Defrost Board is designed to operate from an input voltage of 85VAC to 265VAC with a maximum of 228mA operating current. An LED is illuminated will AC voltage is present.

1.2.2 12VDC Power Supply

The on-board AC to DC power supply is a off the self unit from Astrodyne that supplies 0.42A@12VDC.

1.2.3 5VDC Power Supply

The on-board DC to DC power supply is a linear regulator the supplies +5VDC to the microcontroller and to the LEDs. This function is performed with a MAX663.The part requires a maximum supply current of 15uA and will provide a minimum output current of 40mA@ 5 volts.

1.2.4 Discrete Outputs

There are three relay outputs driven with a FET (IRF7353D2). An LED is turned on when the respective relay coil is energized.

1.2.4.1 Heater

The heater relay is a DPST rated at 15A. This relay will drive the heater coil directly for the mirror cooling system and will drive an interposing relay for the dome cooling system. The second contact is reserved for status.

1.2.4.2 Fan

The fan relay is a SPDT rated at 16A. This relay will drive the fan for the mirror cooling system with the normally closed contacts and will drive the dampers for the dome cooling system with the normally open contacts.

1.2.4.3 Actuator

The actuator relay is a SPDT rated at 16A. This relay will drive the bypass valve for the mirror cooling system for the dome cooling system with the normally closed contacts.

1.2.5 Discrete Input

When the discrete input goes high, the defrost cycle is initiated. This signal originates from a pressure differential switch that detect when the air pressure difference exceeds the switch set point.

1.2.6 LEDs

The figure below shows the physical alignment of the LEDs on the Defrost circuit board. The functions of the LEDs are as follows:

- **Pwr** – This LED will illuminate when AC power is supplied to the unit. It does **not** indicate the health of the 12 volt or 5 volt power supplies.
- **Air** – This LED indicates that the discrete input to the PSoC is high. A defrost cycle will begin if the input held low (LED off) longer than 10 seconds.
- **Htr** – This LED will illuminate when the heater relay coil is energized which also indicates that the defrost cycle **is** active.
- **Fan** – This LED will illuminate when the fan relay coil is energized which also indicates that the defrost cycle is **not** active. The operation of this LED is identical to the **Act** LED.
- **Act** – This LED will illuminate when the actuator relay coil is energized which also indicates that the defrost cycle is **not** active. The operation of this LED is identical to the **Fan** LED.

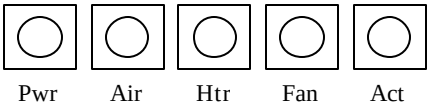


Figure 4 LED Alignment

1.2.7 Control

The control is for the defrost cycle is performed by the PSoC Microcontroller (CY8C25122). The cycle is initiated when the discrete input is triggered.

1.3 Board Connections

1.3.1 Edge Connector

JP1 and JP3 are a header and socket combination for easy board removal without removing the wire connections. The socket will accept a wire size from 22 to 12 AWG. Shown in figure below are the connections to JP1/JP3.

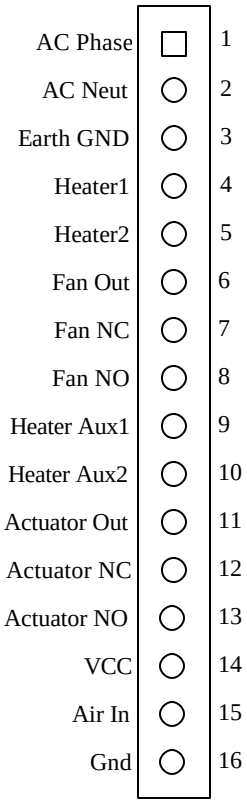


Figure 5 JP1/JP3 Connections

1.3.2 Programming Header

The Programming header is used to program the PSoC microcontroller which is shown in the figure below. It is to connect to the Y-programmer board via a ribbon cable (see section 2.1).

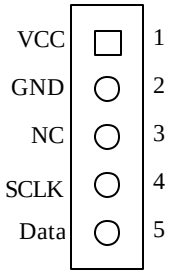
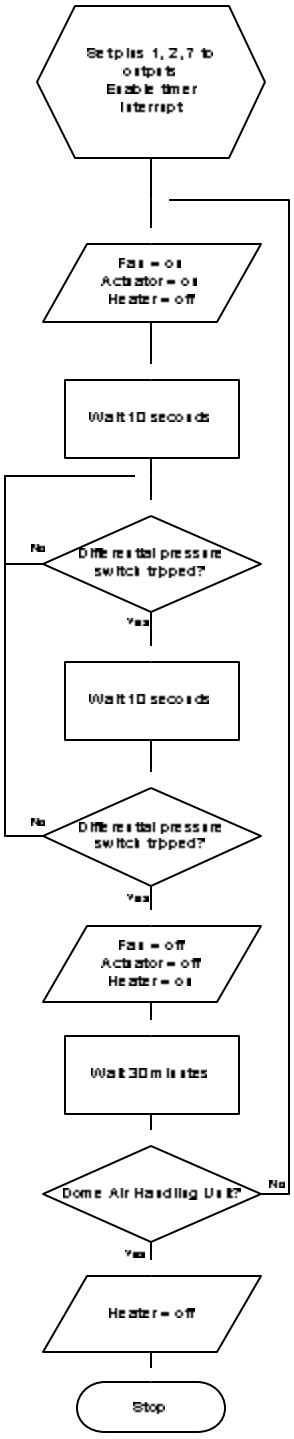


Figure 6 JP2 Connections

1.4 Firmware Characteristics

The operation of the defrost control is detailed in the figure 7 flowchart. Presently, the same source file is used for both the mirror cooling and air handling units. A complier switch is used to determine if the code will loop or stop after a defrost cycle is completed. Source compilation along with configuration is performed in the PSoC development environment.



1.4.1 Timer Module

A Timer32 module is placed within the microcontroller which provides a 32-bit timer function driven with a 93.75k Hz clock. This will give the required minimum 30 minutes defrost time. The two parameters that may have to be adjusted within main.c are the switch Hysteresis (set to 10 seconds) and the total defrost time (set to 30 minutes).

1.4.2 Code listing

The code listings (main.c and Timer32_1int.asm) are located in Appendix A. These are two files that were written from scratch (main.c) or were modified (Timer32_1int.asm). Specifically, the function Timer32_1INT was modified in Timer32_1int.asm. The rest of the files were generated automatically in PSoC Designer.

2 Program and Test

2.1 Programming the PSoC

To program the PSoC, the ICE unit must be plugged into the parallel port with the CAT 5 cable connected to the Y-programmer board. The 5-pin cable is then connected from the 5-pin header on the Y-programmer board to the 5-pin connector on the Defrost board. Next, PSoC Designer is started with the file "PSoC_Defrost.SOC" being opened. The "Program Part" icon is then pushed and the psoc_defrost.hex file is then selected. After positive acknowledgement, the part is now programmed.

2.2 Defrost Test Procedure

The following test procedure assumes that the unit is programmed for mirror cooling. If the unit is programmed for the dome air handler, eliminate the last step.

1. Connect a switch to pins 14 and 15 with the switch in the closed position
2. Connect an 85VAC to 265VAC power source to pins 1 and 2.
3. Verify that the Power, Air, Fan, and Actuator LEDs **are** illuminated.
4. Verify with a continuity meter that the following relay contacts are open:
   - Heater – Pins 4 and 5
   - Fan NC – Pins 6 and 7
   - Heater Aux – Pins 9 and 10
   - Actuator NC – Pins 11 and 12
5. Verify with a continuity meter that the following relay contacts are closed:
   - Fan NO – Pins 6 and 8
   - Actuator NO – Pins 11 and 13
6. Open the switch for at least 1 second, verify that the Air LED is **not** illuminated, and then close the switch within ten seconds with the Air LED once again becoming illuminated.
7. Verify that there is no change in Heater, Fan, and Actuator LED status.
8. Again, open the switch. Verify that the Air LED is **not** illuminated and leave the switch in the open position for at least 10 seconds.
9. After 10 seconds, verify that the Fan and Actuator LEDs are **not** illuminated and that the Heater LED **is** illuminated.
10. Verify with a continuity meter that the following relay contacts are closed:

- Heater – Pins 4 and 5
- Fan NC – Pins 6 and 7
- Heater Aux – Pins 9 and 10
- Actuator NC – Pins 11 and 12

11. Verify with a continuity meter that the following relay contacts are open:
    - Fan NO – Pins 6 and 8
    - Actuator NO – Pins 11 and 13
12. Close the switch with the Air LED once again becoming illuminated.
13. Wait 30 minutes and then repeat steps 3 through 12.

Appendix A: Code Listings

```
//-------------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------------

#include "psoc_defrostapi.h"

#define MINUTE          60
#define TIMER_CLK 93750L                                    // Needs to have the
'L' or the next two lines won't work
#define HYSTERESIS      10 * TIMER_CLK                      // 10 Seconds
#define DE_ICE          30 * MINUTE * TIMER_CLK             // 30 Minutes
#define HEATER          0x10
#define FAN             0x20
#define ACT             0x80
#define AIR_IN          4

extern unsigned char done;

void delay(DWORD timer_value);

void main(void)
{
        // Set pins 1, 2, and 7 to outputs
        PRT0DM0 = HEATER + FAN + ACT;
        Timer32_1_EnableInt();
        M8C_EnableGInt;
        while (1)
        {
                // Turn on fan and actuator
                PRT0DR = FAN + ACT;
                delay(HYSTERESIS);
                while (PRT0DR & AIR_IN)
                        ;

                delay(HYSTERESIS);
                if (!(PRT0DR & AIR_IN))
                {
                        // Turn off fan and actuator; turn on heater
                        PRT0DR = HEATER;
                        delay(DE_ICE);
                        #if (!LOOP)
                                PRT0DR = 0;
                                return;
                        #endif
                }
```

```
        }
}

void delay(DWORD timer_value)
{
        Timer32_1_WritePeriod(timer_value);
        Timer32_1_Start();
        while (!done)
                ;

        done = 0;
}
```

```
;-------------------------------------------------------------------------------
;  FILENAME:   Timer32_1int.asm
;   VERSION:   Rev B, 2002 Mar 30
;-------------------------------------------------------------------------------
;  DESCRIPTION:
;     Interrupt handler routine for Timer32 user module instance:
;       Timer32_1.
;-------------------------------------------------------------------------------
;          Copyright (c) Cypress MicroSystems 2000-2002.  All Rights Reserved.
;-------------------------------------------------------------------------------

include  "Timer32_1.inc"
AREA bss(RAM)
          _done: BLK 1

AREA text(ROM)

;-------------------------------------------------
;  Export interrupt handler
;     NOTE that interrupt handler is NOT exported
;     for access by C function.  Interrupt handlers
;     are not callable by C functions.
;-------------------------------------------------
export  _done
export   Timer32_1INT

;-------------------------------------------------------------------------------
;  FUNCTION NAME: Timer32_1Int
;
;  DESCRIPTION:
;     Timer32 interrupt handler for instance Timer32_1.
;
;     This is a place holder function.  If the user requires use of an interrupt
;     handler for this function, then place code where specified.
;
;  ARGUMENTS:
;     none.
;
;  RETURNS:
;     none.
;
;  SIDE EFFECTS:
;     none.
;
;  THEORY of OPERATION:
;     none.
```

```
;
;-------------------------------------------------------------------------
Timer32_1INT:
        mov [_done], 01h
   call Timer32_1_Stop
   reti


; end of file
```