EECS4312 eHealth Project

Siraj Rauff (cse23188@cse.yorku.ca)
Skyler Layne (cse23170@cse.yorku.ca)

December 3, 2015

You may work on your own or in a team of no more than two students. **Submit only one document under one Prism account.**

Prism account used for submission: cse23188

Keep track of your revisions in the table below.

Revisions

Date	Revision	Description
date please	1.0	Initial requirements document

Requirements Document:
for Patient care eHealth System

Contents

List of Figures

List of Tables

1. System Overview

The System Under Development (SUD) is a computer system to create and manage health prescription records for Ontario.

This requirements document is specifically for prescription management. The purpose of the eHealth Patient care System is to maintain physicians, medications, patients, and patient prescriptions. The system will also control the undesirable interactions between medications, that is when two medications conflict in some way with one another. Only specialist physicians should be allowed to prescribe undesirable interactions while general physicians should be allowed to prescribe medications, as long as they do not create undesirable interactions.

2. Context Diagram

The System Under Description (SUD) is a computer *controller* to keep track of the physicians, patients, patient prescriptions, medications, and medication interactions. The monitored variables and controlled variables for this computer system can be found in Table 1 and Table 2 respectively.

The system must keep track of abstract state which isn't available to the user. For a list of the abstract states within the controller see Table **??**.

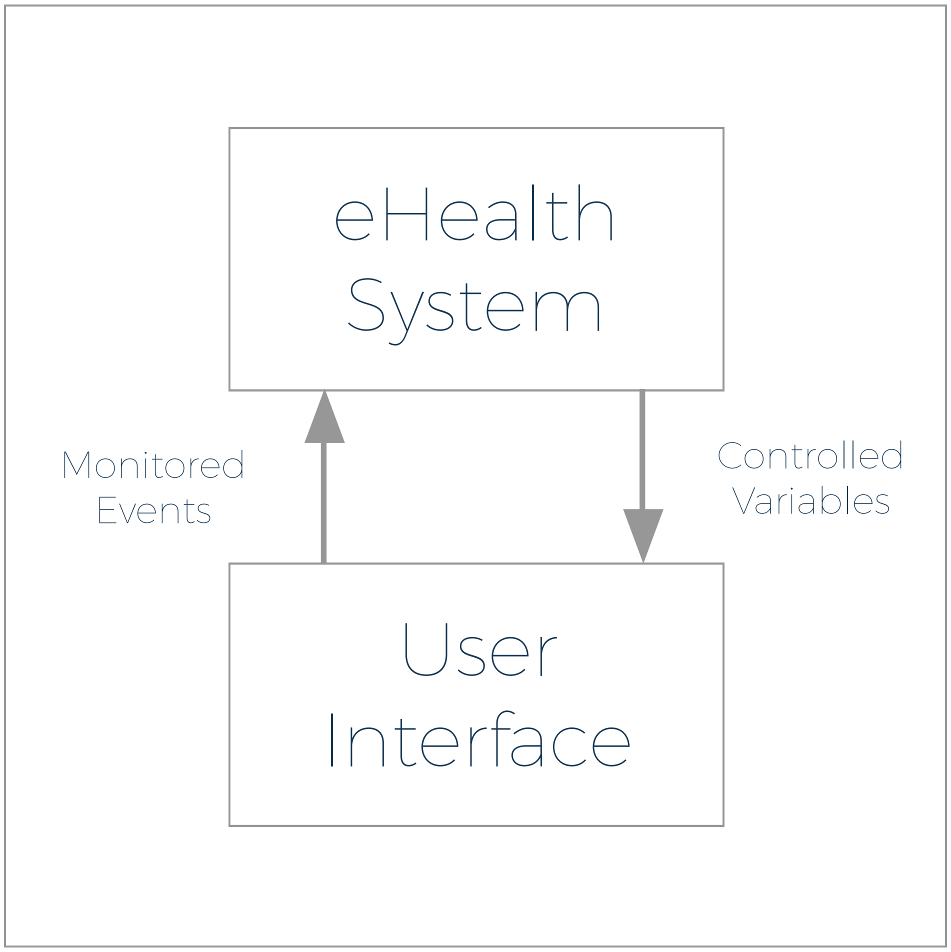

Figure 1: Context Diagram

3. Goals

The high-level goals (G) of the system are:

- G1— The user should be able to add doctors (generalists, and specialists)
- G2— The user should be able to add patients
- G3— The user should be able to add medications
- G4— The user should be able to add perscriptions
- G5— The user should be able to add interactions between medications
- G6— The user should only be able to add a prescription with a dangerous interaction if the doctor is a specialist

todo

4. Monitored Variables

The monitored variables are a subset of those described in [?].[1] There is a single status variable m_st that is *invalid* whenever any one of the operator inputs or temperature sensor are in a failed state. Otherwise types and ranges are as in [?].

Name	Type	Range	Units	Physical Interpretation
m_tm	$\mathbb{R}$	68.0 .. 105.0	°F	actual temperature of Isolette air temperature from sensor
m_dl	$\mathbb{Z}$	97 .. 99	°F	desired lower temperature set by operator
m_dh	$\mathbb{Z}$	98 .. 100	°F	desired higher temperature set by operator
m_al	$\mathbb{Z}$	93 .. 98	°F	lower alarm temperature set by operator
m_ah	$\mathbb{Z}$	99 .. 103	°F	higher alarm temperature set by operator
m_st	Enumerated	{valid, invalid}		status of sensor and operator settings
m_sw	Enumerated	{on, off}		switch set by operator

Table 1: Monitored Variables

[1]With some change of nomenclature. Monitored variables have an “m” prefix.

todo

5. Controlled Variables

The controlled variables are a subset of those described in [?].[2] In addition, there is a mode display c_md and a message display c_ms.[3]

Name	Type	Range	Units	Physical Interpretation
c_hc	Enumerated	{on, off}		heat control: command to turn heat source on or off
c_td	$\mathbb{Z}$	$\{0\} \cup \{68 \,..\, 105\}$	°F	displayed temperature of Isolette (zero when Isolette is off)
c_al	Enumerated	{off, on}		sound alarm to call nurse
c_md	Enumerated	{off, init, normal, failed}		mode of Isolette operation (failed if $m\_st = invalid$)
c_ms	Enumerated	{ok, invalid, config, low, high}		messages to display to nurse

Table 2: Controlled Variables

[2]With some change of nomenclature. Controlled variables have a "c" prefix.
[3]The mode "off" is added to that of Fig. A-4 in [?], and the mode transitions have been changed.

6. Mode Diagram

REQ1 states The *controller* shall operate in one of four modes: *off*, *init*, *normal* and *fail*, shown in Fig. 2. As shown in the figure, the Isolette will begin in the *off* mode, and will enter *init* mode when the nurse flips *m_sw* into the *on* position. The Isolette will only be able to move from the *init* mode ot the *normal* mode if it is properly configured such that the desired range is valid and not overlapping with the alarm levels, and both the sensors and operator controls are working (see REQ6). Once inside the *normal* mode, the controller will move to the *fail* mode if either the controls or sensor fails, as specified in REQ5, and will only return to the *normal* mode once they are both working correctly (REQ11). In any of these states, if the nurse switches *m_sw* to *off* the controller will switch to the off mode (REQ12).

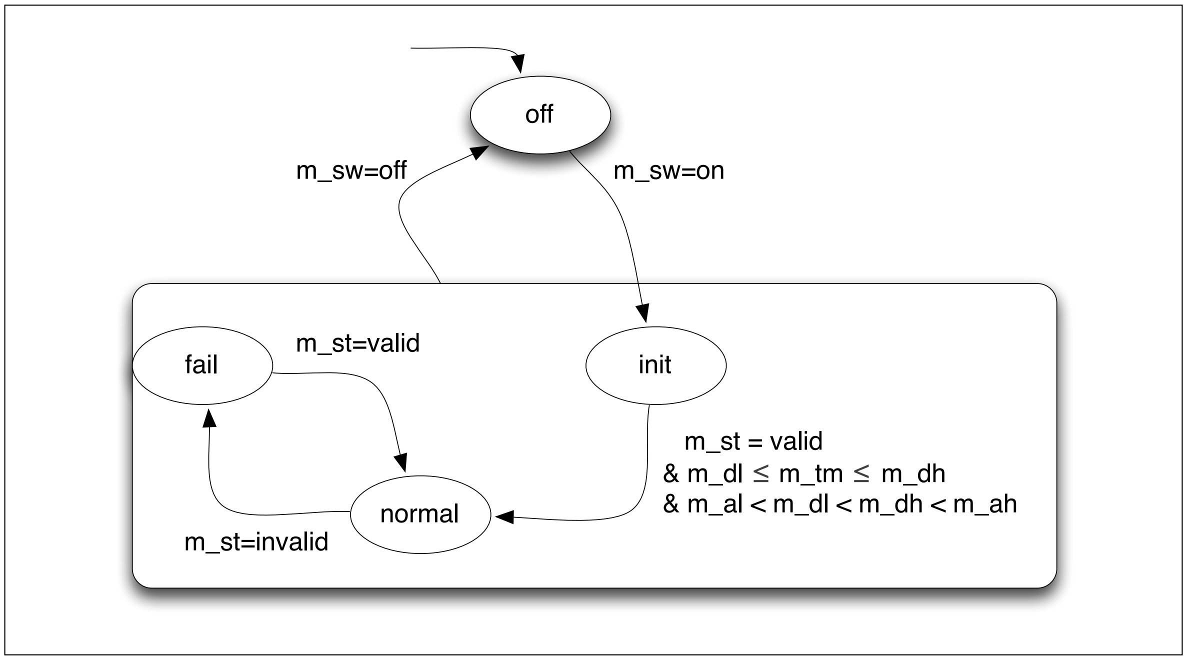

Figure 2: Statechart for the modes variable *c_md*

7. E/R-descriptions

7.1. Requirements Descriptions

REQ1	The *controller* shall operate in one of four modes: *off*, *init*, *normal* and *fail*.	See statechart in Fig. 1.

Rationale: The Isolette should consist of separate states to indicate to the nurse whether it is safe or not for the child to be kept inside. States should exist to the nurse as messages, either of normal status or of an error. Errors such as failed sensors or display interfaces, or an invalid configuration of the *desired temperature range* and *alarm values* should be displayed to the nurse.

REQ2	In the *normal* mode, the temperature controller shall maintain current temperature inside the Isolette within a set temperature range (the *desired* range).	The *desired* temperature range is *m_dl .. m_dh*. If the current temperature *m_tm* is outside this range, the controller shall turn the heater on or off via the controlled variable *m_hc* to maintain the desired state.

Rationale: The *desired temperature range* will be set by the nurse to the desired range based on the infant's weight and health. The controller shall maintain the current temperature within this range under normal operation.

REQ3	In *normal* mode, the controller shall activate an alarm whenever • the current temperature falls outside the *alarm* temperature range (either through temperature fluctuation or a change in the alarm range by an operator), or • a failure is signalled in any of the input devices (temperature sensor and operator settings).	The alarm temperature range is *m_al* .. *m_ah*. Monitored variable *m_st* shows "invalid" when any of the input signals fail.

Rationale: During normal operation, if any conditions occur that could affect the wellbeing of the infant, the nurse should be notified by an alarm. This could include sensor or interface failure, or current temperature exceeding alarm values - even if this is caused by the nurse adjusting the values.

REQ4	Once the alarm is activated, it becomes deactivated in one of two ways: • The nurse turns off the Isolette; • The alarm has lasted for 10 seconds, and after 10 seconds or more the alarm conditions are removed.	The alarm *c_al*, will remain on until *c_md* goes to state off, or has been held for 10 seconds.

Rationale: The alarm should stay on as long as the alarm condition remains. Once the conditions have been cleared, the alarm should only turn off after it has been on for at least 10 seconds to ensure that the nurse was in fact notified of the temperature fluctuation into dangerous areas or the possible failure of a component. This is because quick fluctuations in temperature or a component that could be shorting out momentarily might not occur long enough for the alarm to sound and notify the nurse.

REQ5	In *normal* mode,the controller will enter the *fail* mode if the sensors or operator controls stop working.	If monitored variable *m_st* shows "invalid", the controller will enter the fail state. See statechart in Fig. 1.

Rationale: During normal operation, if the sensor or controls malfunction, the controller should enter the *fail* mode display a message through the interface indicating so. No assumptions should be made in this state as the accuracy of the monitored data can no longer be trusted to be accurate.

REQ6	In *init* the controller will only be able to transition to *normal* if • The current temperature is in the desired range and • The desired range is valid and • The alarm levels do not overlap the desired range and • The operator controls and sensors are working	The controller must only transition to *normal* if $m\_al < m\_dl < m\_dh < m\_ah$ and *m_st* does not show "invalid"

Rationale: After the nurse has configured the Isolette, it should not proceed to *normal* mode before first validating the consistency of the configuration and ensuring that the environment is adjusted until it is consistent with the configuration. This ensures the proper functioning of the Isolette as well as ensuring that the desired temperature range is reached before the infant is placed inside which is critical to the infant's wellbeing.

7.2. Environmental Descriptions

ENV7	The current temperature received from the sensor is a a real number in the range 68.0 to 105.0°F.	The temperature of the system *m_tm* will only operate within the given range

Rationale: The system will never encounter temperatures outside the range of 68.0 to 105.0°F in the environments they are deployed in, and does not have to deal with any values outside this range.

ENV8	The desired and alarm temperatures received from the operator are all in increments of 1°F.	*m_ah*, *m_al*, *m_dh*, and *m_dl* in Table 1: Monitored Variables.

Rationale: The operator interface for the nurse is designed in such a way that the increments of the input temperatures from the control interface must change by whole numbers.

ENV9	Failure of the sensors or operator settings will cause the status to become invalid.	If the sensor or operator settings fail, *m_st* becomes invalid.

Rationale: The environment has ways of detecting failures of the operator interface or the temperature sensor and indicating their status to the controller. The Isolette must constantly be monitoring the status of the sensors and controls to ensure that safe assumptions can be made on their values so as to not endanger the infant.

ENV10	The alarm levels may be set such that they are overlapping with the desired temperature range.	The range of *m_al* overlaps slightly with *m_dl*, and *m_ah* overlaps with *m_dh*, the system should account for this and ensure the Isolette is properly configured, and display a message to the nurse if it is not.

Rationale: The operator controls allow the nurse to set *m_al* anywhere from $97 \mathrel{..} 98$, and *m_dl* to any value in $97 \mathrel{..} 99$. It is then possible for the nurse to set *m_al* such that it will cause an alarm while *m_tm* is in the desired range of *m_dl* .. *m_dh*. The same is true for *m_ah* and *m_dh*. The Isolette must be aware of this possibility, and display an appropriate error message to the nurse if the configuration of the Isolette is not valid.

8. Abstract variables needed for the Function Table

Name	Conditional
$c1(i)$	$m\_st(i) = valid$
$c2(i)$	$m\_dl(i) \leq m\_tm(i) \leq m\_dh(i)$
$c3(i)$	$m\_al(i) < m\_dl(i) < m\_dh(i) < m\_ah(i)$
$c4(i)$	$c1(i) \land c2(i) \land c3(i)$
$c5(i)$	$m\_tm(i) \leq m\_al(i) + 0.5$
$c6(i)$	$m\_tm(i) \geq m\_ah(i) - 0.5$
$c7(i)$	$c1(i) \land c3(i) \land m\_al(i) < m\_tm(i) < m\_ah(i)$
$c8(i)$	$\neg c1(i) \lor \neg c3(i) \lor c5(i) \lor c6(i)$
$held\_for(i)$	$(\forall (j : int) : i - 10 \leq j \land 0 \leq j \implies c\_al(j) = on)$

Figure 3: Abstract Variables used in Function Tables

9. Function Tables

9.1. Function Table for Heat Control: c_hc

Monitored Inputs *c_md(i)*			c_hc(i)
$i = 0$			off
$i > 0$	$c\_md(i) = off \lor \neg c1 \lor \neg c3$		off
	$\neg c\_md(i) = off \land c1 \land c3$	c2	NC
		$m\_tm(i) < m\_dl(i)$	on
		$m\_tm(i) > m\_dh(i)$	off

Figure 4: Function Table for heat control: *c_hc*

9.2. Function Table for Temperature Display: c_td

Monitored Inputs $m\_tm(i)$, $c\_md(i)$	$c\_td(i)$
$c\_md(i) = normal$	$m\_tm(i)$
$\neg c\_md(i) = normal$	0

Figure 5: Function Table for Temperature Display: *c_td*

9.3. Function Table for Mode: c_md

Monitored Inputs $m\_sw(i)$, $c\_md(i\text{-}1)$				$c\_md(i)$
$i = 0$				off
$i > 0$	$m\_sw(i)$ = off			off
	$m\_sw(i)$ = on	$c\_md(i\text{-}1) = off$		init
		$c\_md(i\text{-}1) = normal \vee c\_md(i\text{-}1) = failed$	$c1(i)$	normal
			$\neg\, c1(i)$	failed
		$c\_md(i\text{-}1) = init$	$c4(i)$	init
			$\neg\, c4(i)$	normal

Figure 6: Function Table for Mode: *c_md*

9.4. Function Table for Messages: c_ms

Monitored Inputs $m\_al(i)$, $m\_ah(i)$, $m\_tm(i)$	$c\_ms(i)$
$\neg\, c1(i)$	invalid
$\neg\, c3(i)$	config
$m\_tm(i) < m\_al(i)$	low
$m\_tm(i) > m\_ah(i)$	high
ELSE	ok

Figure 7: Function Table for Messages: *c_ms*

9.5. Function Table for Alarm: c_al

Monitored Inputs *m_al(i)*, *m_ah(i)*, *m_tm(i)*				*c_al(i)*
$i = 0$				off
$i > 0$	*c_al(i-1) = off*	$c7(i)$		NC
		$\neg c7(i)$		on
	c_al(i-1) = on	$c8(i)$		NC
		$\neg c8(i)$	$held\_for(i)$	off
			$\neg held\_for(i)$	on

Figure 8: Function Table for Alarm: *c_al*

10. Validation

todo

You must also provide and prove in PVS one important safety invariant for the heat control *c_hc* and one important safety invariant for the alarm control *c_al*.

Include the PVS sources in the appendix to this document but summarize the proofs here (top.summary).

```
***
*** top (23:34:28 11/15/2015)
*** Generated by proveit - ProofLite-6.0.9 (3/14/14)
*** Trusted Oracles
***   MetiTarski: MetiTarski Theorem Prover via PVS proof rule metit
***
 Proof summary for theory top
    Theory totals: 0 formulas, 0 attempted, 0 succeeded (0.00 s)

 Proof summary for theory Time
    r2d_TCC1.............................proved - complete   [shostak](0.23 s)
    d2r_TCC1.............................proved - complete   [shostak](0.03 s)
    held_for_TCC1........................proved - complete   [shostak](0.08 s)
    Theory totals: 3 formulas, 3 attempted, 3 succeeded (0.33 s)

 Proof summary for theory isolette
    c_md_ft_TCC1.........................proved - complete   [shostak](0.03 s)
    c_md_ft_TCC2.........................proved - complete   [shostak](0.03 s)
    c_md_ft_TCC3.........................proved - complete   [shostak](0.05 s)
    c_md_ft_TCC4.........................proved - complete   [shostak](0.10 s)
    c_md_ft_TCC5.........................proved - complete   [shostak](0.06 s)
    c_md_ft_TCC6.........................proved - complete   [shostak](0.03 s)
    c_md_ft_TCC7.........................proved - complete   [shostak](0.02 s)
    c_md_ft_TCC8.........................proved - complete   [shostak](0.02 s)
    c_md_ft_TCC9.........................proved - complete   [shostak](0.02 s)
    c_td_ft_TCC1.........................proved - complete   [shostak](0.01 s)
    c_hc_ft_TCC1.........................proved - complete   [shostak](0.07 s)
    c_hc_ft_TCC2.........................proved - complete   [shostak](0.11 s)
    c_hc_ft_TCC3.........................proved - complete   [shostak](0.07 s)
    c_hc_ft_TCC4.........................proved - complete   [shostak](0.05 s)
    c_hc_ft_TCC5.........................proved - complete   [shostak](0.03 s)
    c_al_ft_TCC1.........................proved - complete   [shostak](0.03 s)
    c_al_ft_TCC2.........................proved - complete   [shostak](0.04 s)
    c_al_ft_TCC3.........................proved - complete   [shostak](0.00 s)
    c_al_ft_TCC4.........................proved - complete   [shostak](0.07 s)
    c_al_ft_TCC5.........................proved - complete   [shostak](0.04 s)
    c_al_ft_TCC6.........................proved - complete   [shostak](0.03 s)
    inv_hc_holds.........................proved - complete   [shostak](0.34 s)
    inv_al_holds.........................proved - complete   [shostak](2.71 s)
    Theory totals: 23 formulas, 23 attempted, 23 succeeded (3.98 s)

Grand Totals: 26 proofs, 26 attempted, 26 succeeded (4.32 s)
```

Figure 9: Validated Isolette

11. Use Cases

See Section A2 of [?] for some use cases. The use cases need to be adapted to the revised descriptions of the previous sections of this document.

12. Acceptance Tests

In this section, the use cases have to be converted into precise acceptance tests (using the function table to describe pre/post conditions) to be run when the design and implementation are complete.

13. Traceability

Matrix to show which acceptance tests passed, and which R-descriptions they checked.

14. Glossary

The definition of important terms is placed in this section. You are not required to complete this.

A. Additional Requirements

REQ11	In *fail* mode, the controller shall only return to *normal* mode if • The sensor is working and • The operator controls are working	In *fail* mode the controller shall return to *normal* mode when *m_st* returns "valid"

REQ12	In any mode, the controller will transition to the *off* mode if the nurse turns the switch off.	The controller will transition to *off* mode from any mode if *m_sw* becomes *off*

B. Isolette PVS

```
isolette[delta:posreal]: THEORY
BEGIN

  %% Import timing resolution
  importing Time[delta]
  i: VAR DTIME

  %% TYPE declarations
  SWITCH: TYPE = {on, off}
  MSTATE: TYPE =  {valid, invalid}
  CONTROL: TYPE = {on, off}
  STATE: TYPE = {off, init, normal, failed}
  ERROR: TYPE = {ok, invalid, config, low, high}
  DISPLAY: TYPE = {i: nat | i = 0 OR (68 <= i AND i <= 105)}
        CONTAINING 0
  ALARM: TYPE = {on, off}
  TM: TYPE+ = {r: real | r >= 68.0 AND r <= 105.0} CONTAINING 68.0
  DL: TYPE+ = {i: nat | i >= 97 AND i <= 99} CONTAINING 97
  DH: TYPE+ =  {i: nat | i >= 98 AND i <= 100} CONTAINING 98
  AL: TYPE+ = {i: nat | i >= 93 AND i <= 98} CONTAINING 93
  AH: TYPE+ = {i: nat | i >= 99 AND i <= 103} CONTAINING 99

  %% Monitored Variables
  m_tm: [DTIME -> TM]
  m_dl: [DTIME -> DL]
  m_dh: [DTIME -> DH]
  m_al: [DTIME -> AL]
  m_ah: [DTIME -> AH]
  m_st: [DTIME -> MSTATE]
  m_sw: [DTIME -> SWITCH]

  %% Controlled Variables
  c_hc: [DTIME -> CONTROL]
  c_td: [DTIME -> DISPLAY]
  c_al: [DTIME -> ALARM]
  c_md: [DTIME -> STATE]
  c_ms: [DTIME -> ERROR]
```

```
al_on(i): bool = c_al(i) = on %% Alarm is on

% General Function table conditions
c1(i): bool = m_st(i) = valid
c2(i): bool = m_dl(i) <= m_tm(i) <= m_dh(i)
c3(i): bool = m_al(i) < m_dl(i) < m_dh(i) < m_ah(i)
c4(i): bool = c1(i) AND c2(i) AND c3(i)
c5(i): bool = m_tm(i) <= m_al(i) + 0.5
c6(i): bool = m_tm(i) <= m_al(i) - 0.5
c7(i): bool = c1(i) AND c3(i) AND m_al(i) < m_tm(i) < m_ah(i)
c8(i): bool = NOT c1(i) OR NOT c3(i) OR c5(i) OR c6(i)
held_for(i): bool = held_for(al_on, 10)(i)

% Mode Function Table
c_md_ft(i): bool =
COND
      i = 0 ->  c_md(i) = off,
      i > 0 ->
      COND
              m_sw(i) = off -> c_md(i) = off,
              m_sw(i) = on ->
              COND
                      c_md(i-1) = off -> c_md(i) = init,
                      c_md(i-1) = normal OR c_md(i-1) = failed ->
                      COND
                              NOT c1(i) -> c_md(i) = failed,
                              c1(i) -> c_md(i) = normal
                      ENDCOND,
                      c_md(i-1) = init ->
                      COND
                              NOT c4(i) ->  c_md(i) = normal,
                              c4(i) ->  c_md(i) = init
                       ENDCOND
              ENDCOND
      ENDCOND
ENDCOND
```

```
% Temperature Display Function Table
c_td_ft(i): bool =
COND
      c_md(i) = normal -> c_td(i) = m_tm(i),
      NOT c_md(i) = normal -> c_td(i)= 0
ENDCOND

% Heat Control Function Table
c_hc_ft(i): bool =
          COND
              i = 0 -> c_hc(i) = off,
              i > 0 ->
                COND
                  c_md(i) = off OR (NOT c1(i))
                    OR (NOT c3(i)) -> c_hc(i) = off,
                 (NOT c_md(i) = off) AND c1(i) AND c3(i) ->
                      COND
                            c2(i) ->  c_hc(i) =  c_hc(i-1),
                                  m_tm(i) < m_dl(i)
                                    -> c_hc(i) = on,
                                  m_tm(i) > m_dh(i)
                                    -> c_hc(i) = off
                      ENDCOND
                  ENDCOND
          ENDCOND
```

```
% Alarm Function Table
c_al_ft(i): bool =
COND
      i = 0 ->  c_al(i) = off,
      i > 0 ->
      COND
              c_al(i-1) = off ->
              COND
                      c7(i) ->  c_al(i) =  c_al(i-1),
                      NOT c7(i) -> c_al(i) = on
              ENDCOND,
              c_al(i-1) = on ->
              COND
                      c8(i) ->  c_al(i) =  c_al(i-1),
                      NOT c8(i) ->
                      COND
                              held_for(i) -> c_al(i) = off,
                              NOT held_for(i) -> c_al(i) = on
                      ENDCOND
               ENDCOND
      ENDCOND
ENDCOND

% Message Display Function Table
c_ms_ft(i): bool =
          IF m_st(i) = invalid THEN c_ms(i) = invalid
          ELSIF NOT c3(i) THEN c_ms(i) = config
          ELSIF m_tm(i) < m_al(i) THEN c_ms(i) = low
          ELSIF m_tm(i) > m_ah(i) THEN c_ms(i) = high
          ELSE c_ms(i) = ok
          ENDIF


% Isolette Specification
isolette(i): bool = c_hc_ft(i) AND c_td_ft(i) AND c_al_ft(i)
      AND c_md_ft(i) AND c_ms_ft(i)
```

```
  % Checks
  inv_hc(i): bool = NOT (c1(i) AND c3(i)) IMPLIES c_hc(i) = off
  inv_hc_holds: CONJECTURE (FORALL i: isolette(i)) =>
                                (FORALL i: inv_hc(i))

  inv_al(i): bool = i > 0 AND (
    (NOT al_on(i-1) AND NOT c7(i))
    OR (al_on(i-1) AND c8(i))
    OR (al_on(i-1) AND NOT c8(i) AND NOT held_for(i))
  ) IMPLIES c_al(i) = on
  inv_al_holds: CONJECTURE (FORALL i: i>0 IMPLIES isolette(i))
                                => (FORALL i: i>0 IMPLIES inv_al(i))

END isolette
```