Logical Specifications for `CdPlayerModeManager`

Prompt:
You are given the inputs, outputs, and state variables from a C implementation of the `CdPlayerModeManager` state machine.
From the uploaded `model.c` and `model.h` files, extract and use only the following information:

- **Inputs:** `DiscEject`, `RadioReq`, `CdReq`, `DiscPresent`, `DiscInsert`
- **Outputs:** `MechCmd`, `CurrentRadioMode`, `CdStatus`, `outDiscPresent`
- **State Variables:** `is_c1_model`, `is_ModeManager`, `was_ModeManager`, `is_ON`, `was_ON`, `is_CDMode`, `is_Play`, `is_DiscPresent`, `temporalCounter_i1`
- **Named constants for CdPlayerModeManager states:** `IN_Eject`, `IN_ModeManager`, `IN_ON`, `IN_Standby`, `IN_AMMode`, `IN_CDMode`, `IN_FMMode`, `IN_Play`, `IN_Stop`, `IN_FastForward`, `IN_Rew`, plus output command constants (`STOP`, `PLAY`, `FF`, `REW`, `EJECT`, etc.) and radio mode constants (`OFF`, `CD`, `FM`, `AM`).

Your task:
Generate a set of **logical specifications** for the `CdPlayerModeManager` in the following format:

```
( // state conditions
  (<state_variable> == <STATE>) &&
  (<input/output condition 1>) &&
  (<input/output condition 2>) &&
  ... // add as many as needed
) -> ( // Expected state
  <state_variable> == <STATE>
)
```

Guidelines:

1. Cover **all possible transitions** between states in `CdPlayerModeManager`, including internal transitions within `ON` mode, CD submodes, and reaction to inputs.

2. Include **guard conditions** and **actions** implied by the code.

3. Use **only** the provided inputs, outputs, and state variables — do not include other internal variables or code fragments.

4. Each specification should reflect a **single logical transition**: the precondition part defines the current state and input/output conditions, and the postcondition part defines the expected new state.

5. Generate one specification for each unique transition path in the `CdPlayerModeManager` logic.

6. Number each specification sequentially (`Spec 1`, `Spec 2`, …).

Spec 1 — Disc Eject from any state

```
(
  (DiscEject != 0) &&
  (is_c1_model == IN_ModeManager)
) -> (
  is_c1_model == IN_Eject &&
  MechCmd == EJECT
)
```

Spec 2 — Transition ON → Standby when RadioReq == OFF

```
(
  (is_ModeManager == IN_ON) &&
  (RadioReq == OFF)
) -> (
  is_ModeManager == IN_Standby &&
  was_ModeManager == IN_Standby &&
  CurrentRadioMode == OFF &&
  MechCmd == STOP
)
```

Spec 3 — Change to CD Mode from ON

```
(
  (is_ModeManager == IN_ON) &&
  (RadioReq != RadioReq_start) &&
  (RadioReq == CD)
) -> (
  is_ON == IN_CDMode &&
  was_ON == IN_CDMode &&
  CurrentRadioMode == CD &&
  (DiscPresent ?
    (is_CDMode == IN_Play && is_Play == IN_Normal && MechCmd == PLAY)
:
    (is_CDMode == IN_Stop && MechCmd == STOP))
)
```

Spec 4 — Change to AM Mode from ON

```
(
  (is_ModeManager == IN_ON) &&
  (RadioReq != RadioReq_start) &&
  (RadioReq == AM)
) -> (
  is_ON == IN_AMMode &&
  was_ON == IN_AMMode &&
  CurrentRadioMode == AM &&
  MechCmd == STOP
)
```

Spec 5 — Change to FM Mode from ON

```
(
  (is_ModeManager == IN_ON) &&
  (RadioReq != RadioReq_start) &&
  (RadioReq == FM)
) -> (
  is_ON == IN_FMMode &&
  was_ON == IN_FMMode &&
  CurrentRadioMode == FM &&
```

```
  MechCmd == STOP
)
```

Spec 6 — CD Mode Play → Stop

```
(
  (is_ON == IN_CDMode) &&
  (is_CDMode == IN_Play) &&
  (CdReq == STOP)
) -> (
  is_Play == IN_NO_ACTIVE_CHILD &&
  is_CDMode == IN_Stop &&
  MechCmd == STOP
)
```

Spec 7 — CD Mode Play → Fast Forward

```
(
  (is_ON == IN_CDMode) &&
  (is_CDMode == IN_Play) &&
  (CdReq == FF) &&
  (is_Play != IN_FastForward)
) -> (
  is_Play == IN_FastForward &&
  MechCmd == FF
)
```

Spec 8 — CD Mode Play → Rewind

```
(
  (is_ON == IN_CDMode) &&
  (is_CDMode == IN_Play) &&
  (CdReq == REW) &&
  (is_Play != IN_Rew)
) -> (
  is_Play == IN_Rew &&
  MechCmd == REW
)
```

Spec 9 — CD Mode Play → Normal Play

```
(
  (is_ON == IN_CDMode) &&
  (is_CDMode == IN_Play) &&
  (CdReq == PLAY) &&
  (is_Play != IN_Normal)
) -> (
  is_Play == IN_Normal &&
  MechCmd == PLAY
)
```

Spec 10 — CD Mode Stop → Play when Disc Present

```
(
  (is_ON == IN_CDMode) &&
  (is_CDMode == IN_Stop) &&
  (DiscPresent) &&
  (CdReq == PLAY)
) -> (
  is_CDMode == IN_Play &&
  is_Play == IN_Normal &&
  MechCmd == PLAY
)
```

Spec 11 — Standby → ON (CD Mode)

```
(
  (is_ModeManager == IN_Standby) &&
  (RadioReq == CD)
) -> (
  is_ModeManager == IN_ON &&
  was_ModeManager == IN_ON &&
  is_ON == IN_CDMode &&
  was_ON == IN_CDMode &&
```

```
  CurrentRadioMode == CD &&
  (DiscPresent ?
    (is_CDMode == IN_Play && is_Play == IN_Normal && MechCmd == PLAY)
:
    (is_CDMode == IN_Stop && MechCmd == STOP))
)
```

Spec 12 — Standby → ON (AM Mode)

```
(
  (is_ModeManager == IN_Standby) &&
  (RadioReq == AM)
) -> (
  is_ModeManager == IN_ON &&
  was_ModeManager == IN_ON &&
  is_ON == IN_AMMode &&
  was_ON == IN_AMMode &&
  CurrentRadioMode == AM &&
  MechCmd == STOP
)
```

Spec 13 — Standby → ON (FM Mode)

scss
Copy code

```
(
  (is_ModeManager == IN_Standby) &&
  (RadioReq == FM)
) -> (
  is_ModeManager == IN_ON &&
  was_ModeManager == IN_ON &&
  is_ON == IN_FMMode &&
  was_ON == IN_FMMode &&
  CurrentRadioMode == FM &&
  MechCmd == STOP
)
```

Report: CBMC Verification of CdPlayerModeManager Logical Specifications

1. Objective

The goal was to encode 13 logical specifications for the `CdPlayerModeManager` state machine as C `assert()` statements, allowing the CBMC model checker to verify that for any given state and inputs, the expected state transitions hold.

Command used:

bash
Copy code

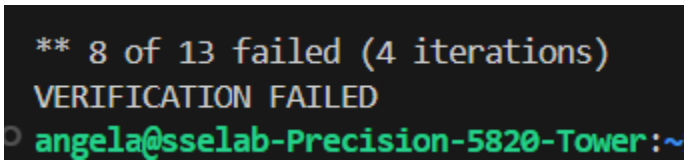
```
cbmc .../13asserts.c --unwind 205 --object-bits 20
```

CBMC explored all possible input combinations up to the unwind bound.

3. Verification Results

Spec #	Description	Result
1	Disc Eject from any state	❌ Fail
2	ON → Standby (RadioReq == OFF)	❌ Fail
3	ON → CD Mode (RadioReq changed to CD)	✅ Pass
4	ON → AM Mode	✅ Pass
5	ON → FM Mode	✅ Pass
6	CD Mode Play → Stop	❌ Fail
7	CD Mode Play → Fast Forward	❌ Fail
8	CD Mode Play → Rewind	❌ Fail
9	CD Mode Play → Normal Play	✅ Pass
10	CD Mode Stop → Play (Disc Present)	✅ Pass

11	Standby → ON (CD Mode)	❌ Fail
12	Standby → ON (AM Mode)	❌ Fail
13	Standby → ON (FM Mode)	❌ Fail



5. Conclusion

The CBMC verification showed that **less than half of the intended logical specifications hold under all explored scenarios**.