RVC 과제 리뷰

Junik Son

Statement of Purpose

Robot Vacuum Cleaner (RVC)

- An RVC automatically cleans and mops household surface.
- It goes straight forward while cleaning.
- If its sensors found an obstacle, it stops cleaning, turns aside, and goes forward with cleaning.
- If it detects dust, power up the cleaning for a while
- We do not consider the detail design and implementation on HW controls.
- We only focus on the automatic cleaning function.

- Do not consider "Dust"

Dust에 관한 것은 생각할 필요가 없다. 어차피 청소기이므로 청소기는 켜고 다니면 된다.(청소기 돌릴 때 먼지 있는 곳만 청소하려고 껐다 켰다 하지는 않기 때문에) 따라서 로봇 청소기가 Move하기만 하면 청소 기능은 켜고 다닌다.

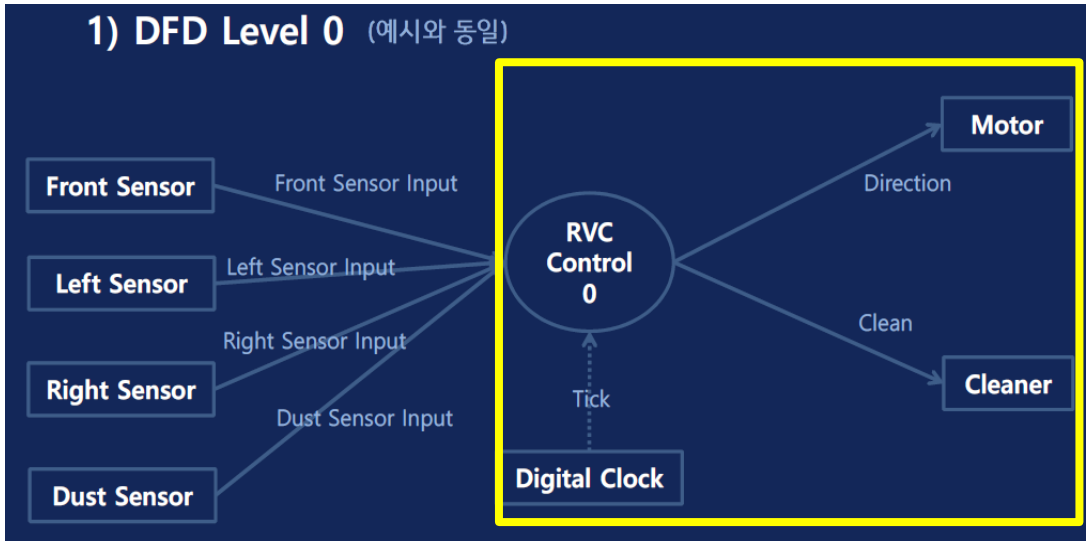
- Statement of Purpose는 개발할 시스템의 목적에 대한 설명임
  그 안에 dust에 대한 설명이 존재하므로, 무시해서는 안됨

Data Flow Diagram

3 Event list

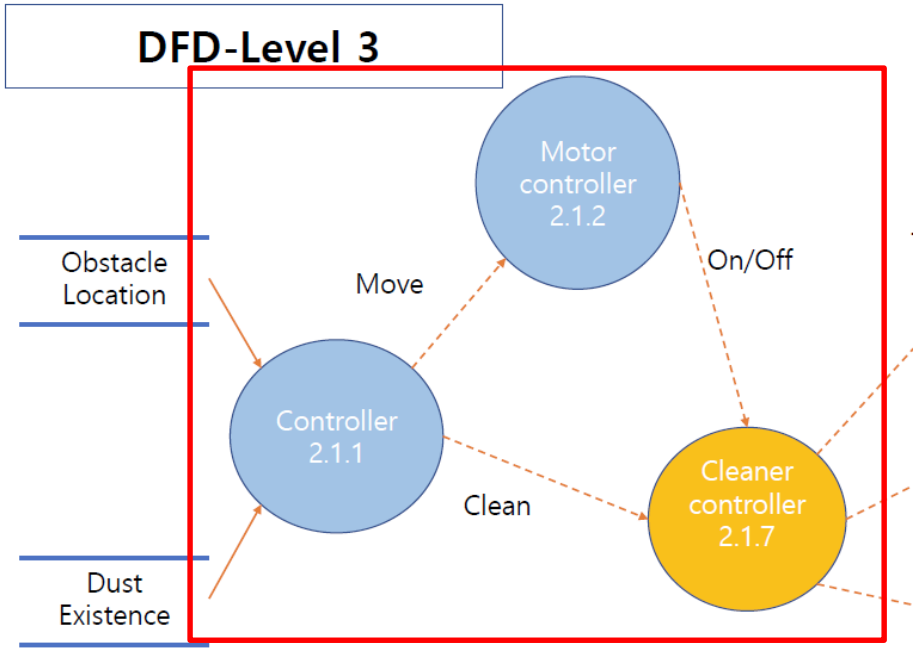
(새로 추가된 부분 바탕색 추가)

Input/Output Event	Description
Front Sensor Input	RVC 앞에 장애물을 탐지한다.
Left Sensor Input	RVC 왼쪽에 있는 장애물을 주기적으로 탐지한다.
Right Sensor Input	RVC 오른쪽에 있는 장애물을 주기적으로 탐지한다.
Dust Sensor Input	마루에 있는 먼지를 탐지한다.
Direction	모터에게 방향을 알린다.
Clean	활성화/비활성화/파워모드를 알린다.
Stop	Motor가 회전할 수 있는 모든 방향에 장애물이 있는 경우에 발생한다. (예외적인 상황이므로 control event에 해당)



- Event List에 Output Event를 추가하였을 경우 DFD Level 0, System Context Diagram에 추가된 Output이 나타나야 됨.

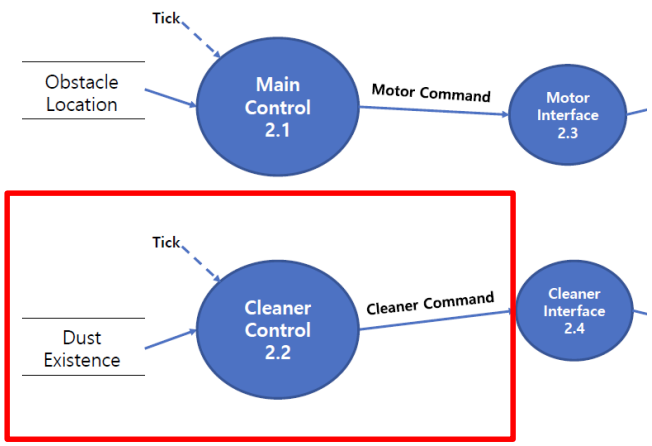
Data Flow Diagram



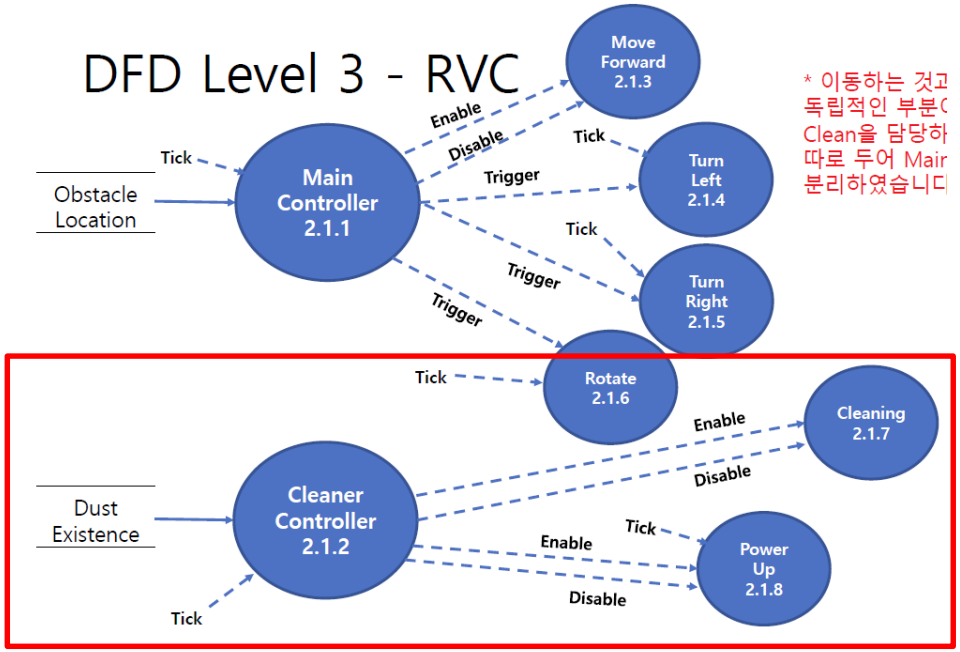
- Control Flow는 Control Process에서만 나갈 수 있음, Data Process에서는 나가지 못함
  - Control Flow는 Trigger, Enable, Disable만 가능
  - Data Flow는 Control Process, Data Process에서 둘 다 출력으로 사용 가능

Data Flow Diagram

DFD Level 2 - RVC



DFD Level 3 - RVC

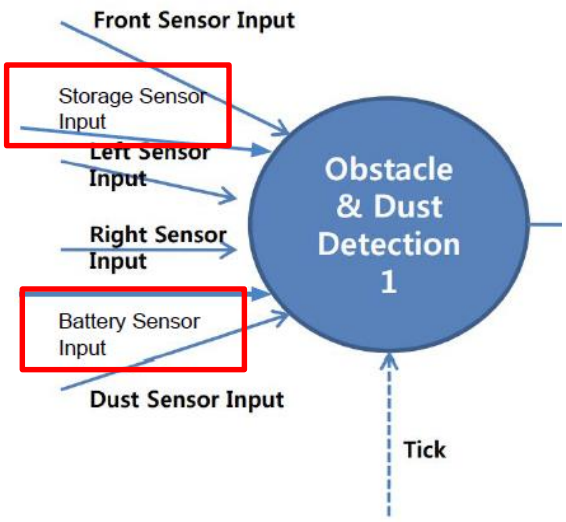


* 이동하는 것고
독립적인 부분(
Clean을 담당하
따로 두어 Mair
분리하였습니다

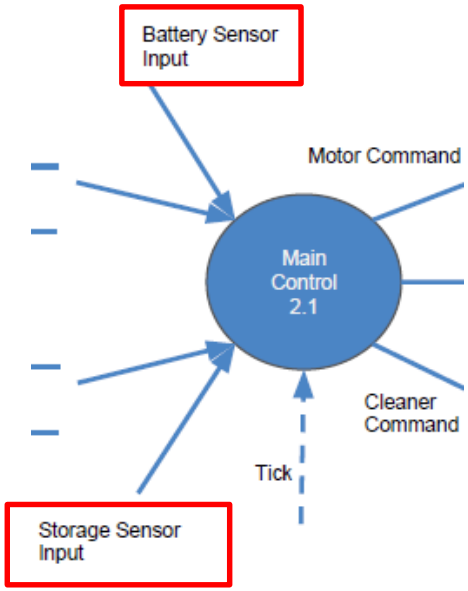
- Cleaner Control이 Cleaner Controller, Cleaning, Power Up으로 분해된다고 분석하였는데 Numbering이 잘못됨
  - Cleaner Controller 2.1.2 (X) -> Cleaner Controller 2.2.1(O)
  - Cleaning 2.1.7 (X) -> Cleaning 2.2.2 (O)
  - Power Up 2.1.8 (X) -> Power Up 2.2.3 (O)

Data Flow Diagram

RVC DFD Level 1



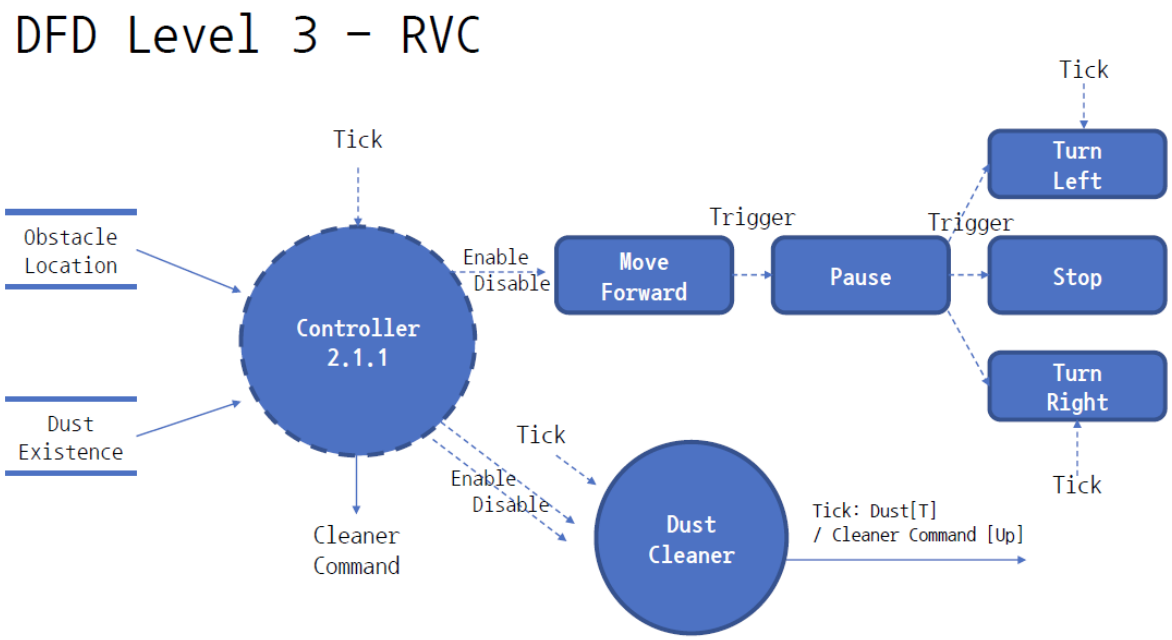
RVC DFD Level 2



- 상위 레벨의 DFD의 데이터의 입출력은 하위 레벨의 DFD의 데이터의 입출력과 같아야 함

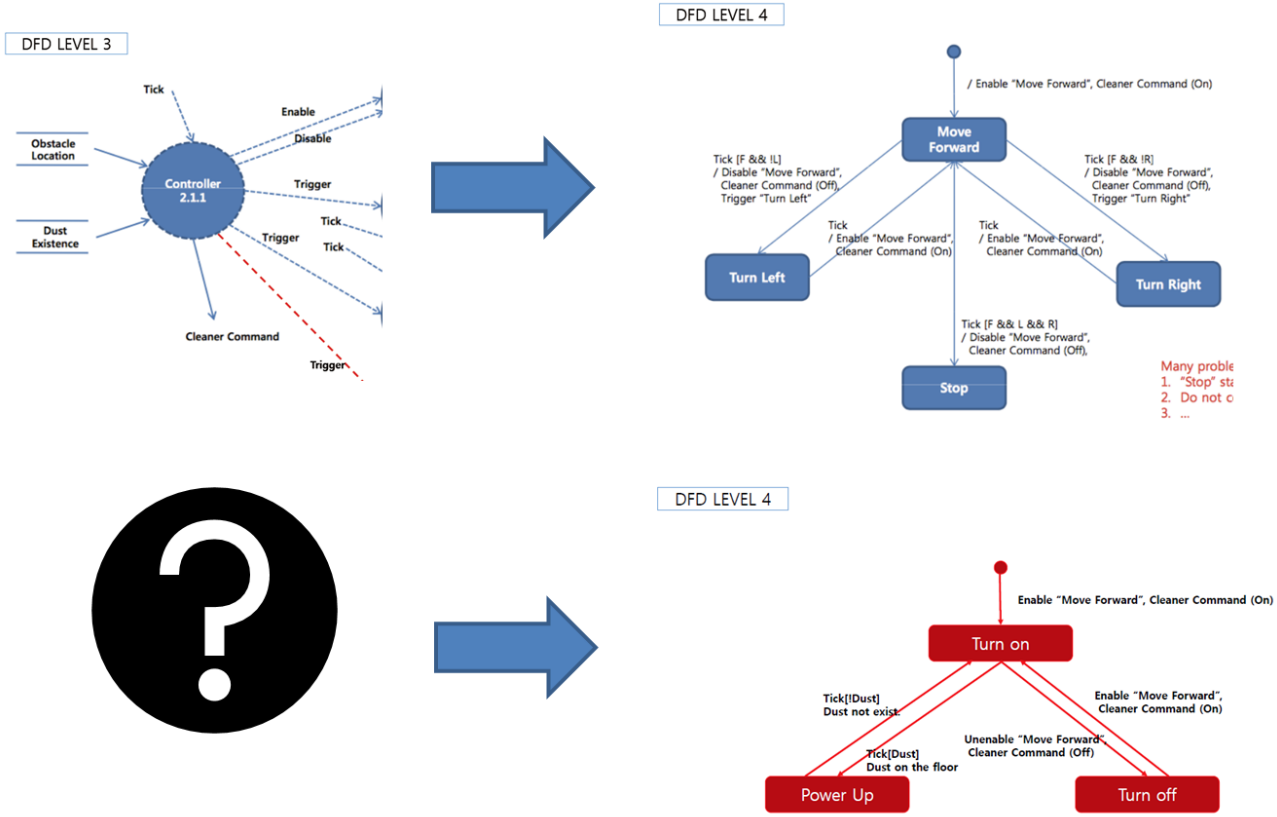
- DFD Level 1에서는 Obstacle & Dust Detection 1으로 Storage Sensor Input, Batter Sensor Input 데이터가 들어온다고 표현이 되어 있기 때문에 Level 2로 바뀌었을 때 두 개의 데이터는 Main Control 2.1의 입력이면 안되고 1.x 의 입력이여야 함

Data Flow Diagram



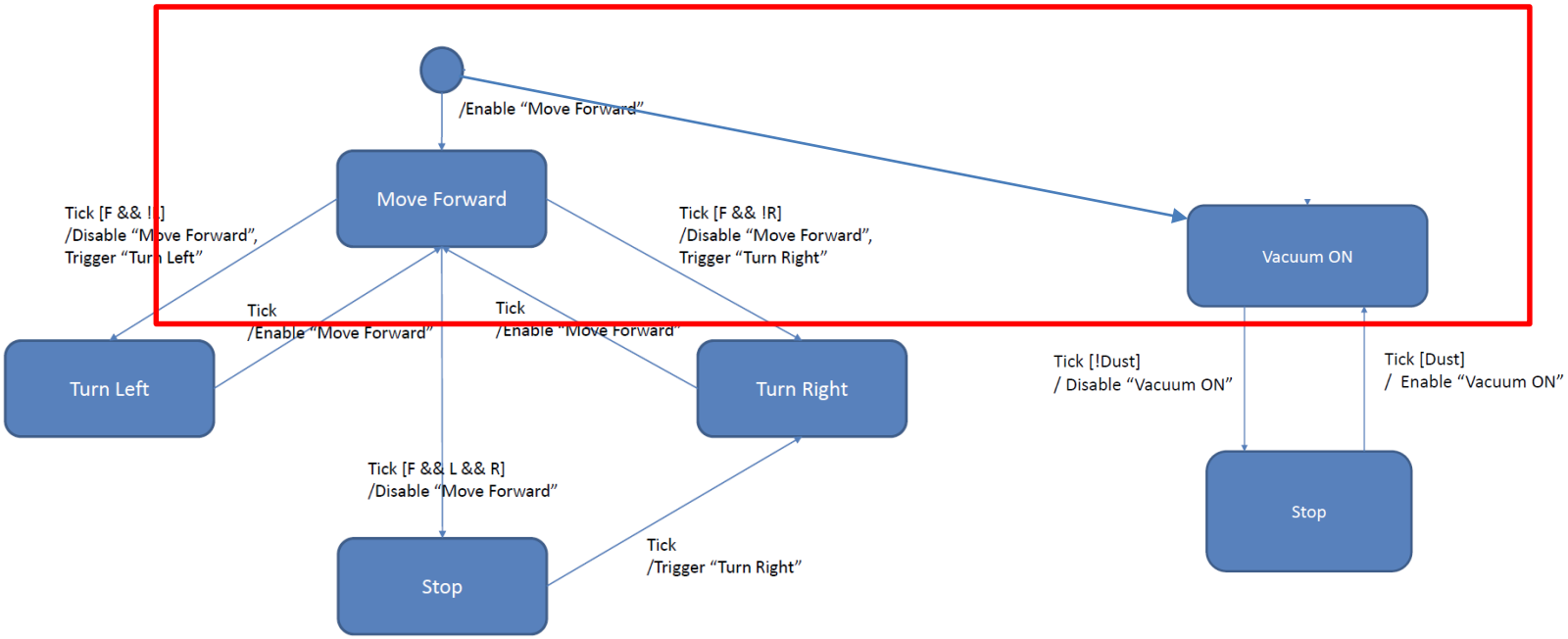
- Control Process와 Control Process의 State Transition Diagram의 상태는 같은 레벨에 있을 수 없음

State Transition Diagram



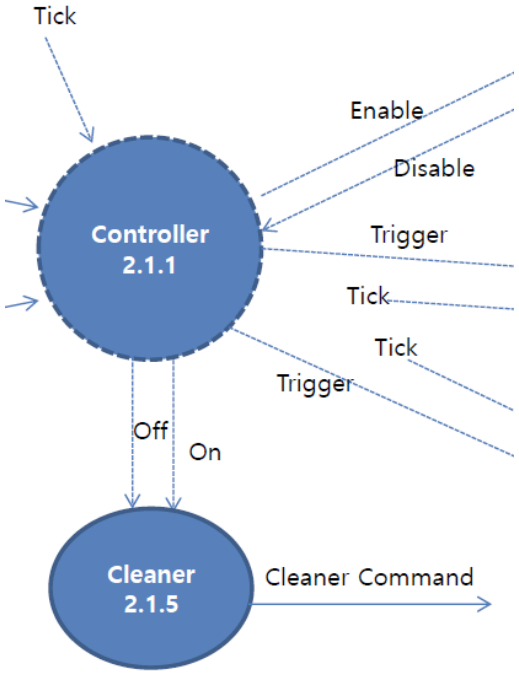
- State Transition Diagram은 Controller 와 1:1 관계임
  Ex) Controller를 2.1.1, 2.1.2 2개로 분석을 했으면 state transition diagram도 2개 필요

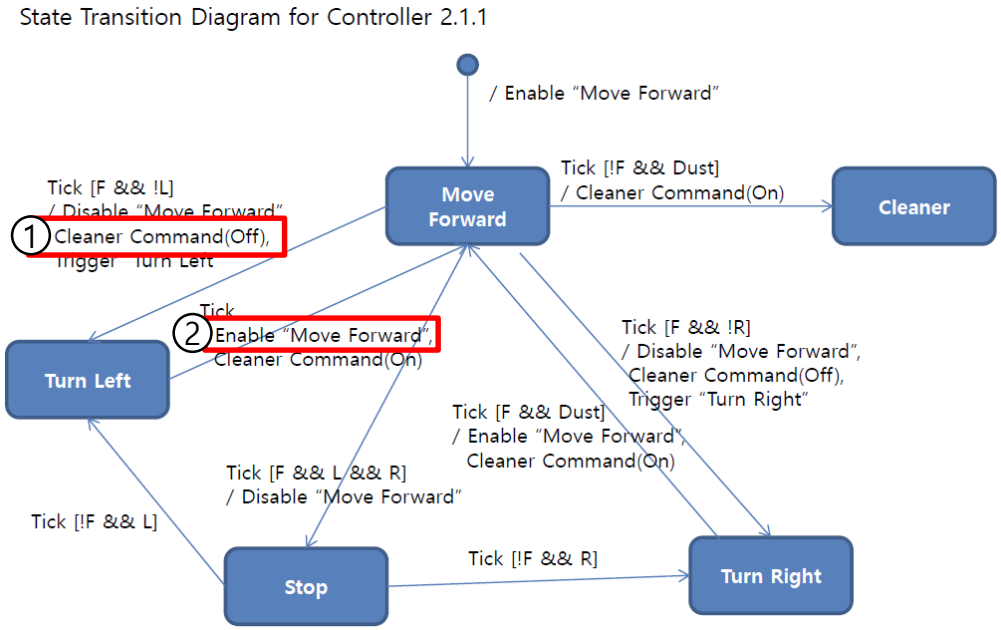
State Transition Diagram



- Initial State에서 조건이 없는 transition이 2개 있는 경우
  -> transition이 둘 다 발생해서 Controller의 상태가 2개가 되는 것이 아니라,
  둘 중 랜덤하게 하나의 transition만 발생함

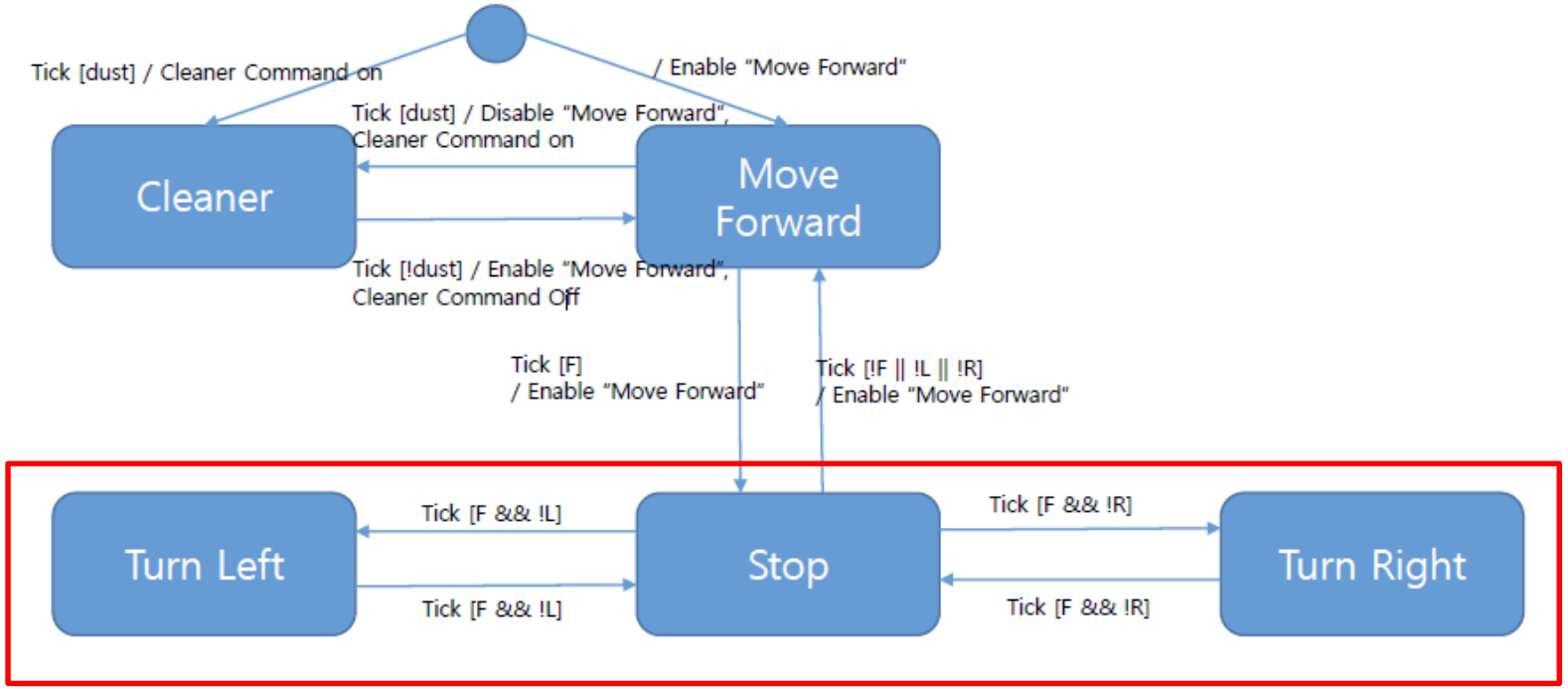
State Transition Diagram





- 1은 data flow의 data를 나타내는 것, 2는 control flow의 control을 나타내는 것
- DFD level3에서는 control process가 data flow를 출력 하지 않기 때문에 우측의 State Transition Diagram에서도 data flow에 해당하는 1은 삭제 되야 함.

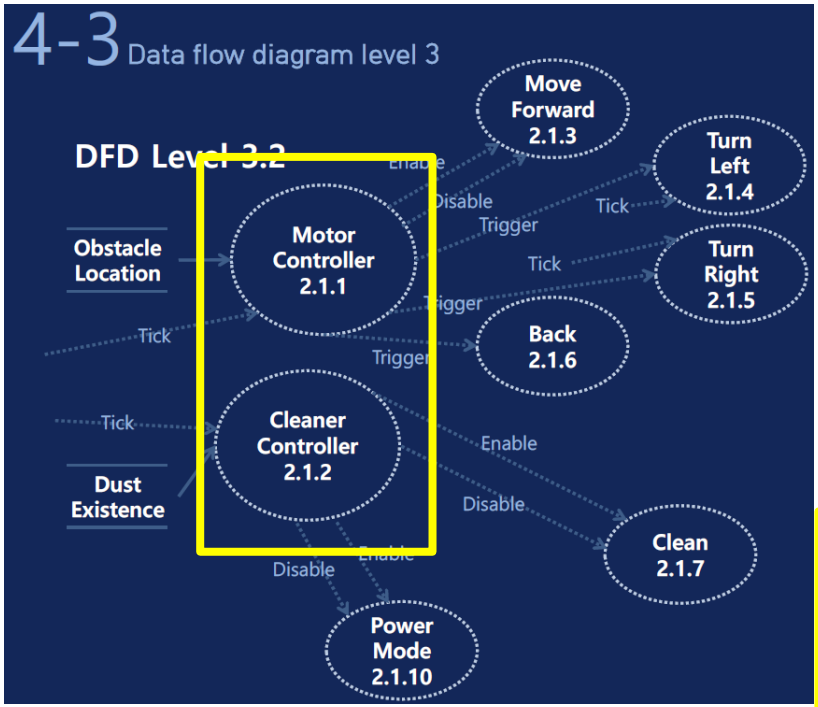
State Transition Diagram

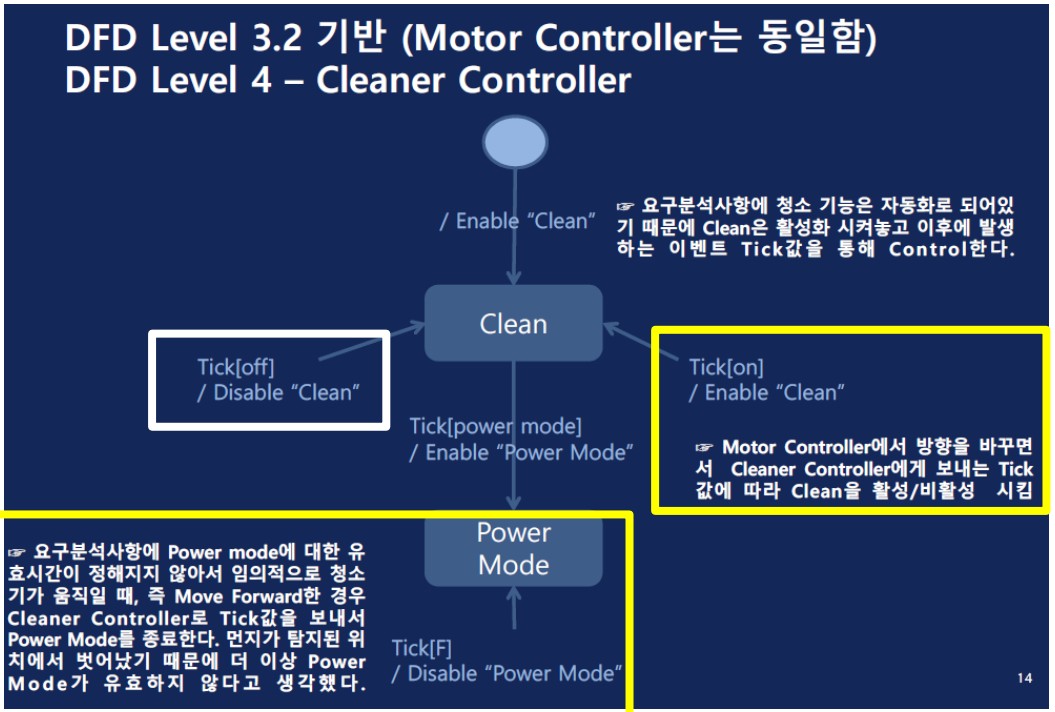


- Turn Left, Turn Right, Stop 은 컨트롤러의 상태를 의미하지 컨트롤러의 액션을 의미 하지 않음

- 따라서 Stop 상태에서 Turn Left로 상태를 전이하면서 액션(Turn Left)을 취하려면
- Tick [F && !L] / Trigger "Turn Left" <- 이런식으로 작성해야 함

State Transition Diagram





- Controller를 두 개 만들어서 두개의 Controller 사이의 동기화를 하기 위해서 on, off, F 라는 데이터를 사용하였는데, 이렇게 하기 위해서는 왼쪽 DFD level 3에 data flow가 표현 되어 있어야 함.