Point of Sales System

Team presentation 1

T4

201211178 민경훈
201211187 배승현
201311283 송형선
201611299 정희승

INDEX

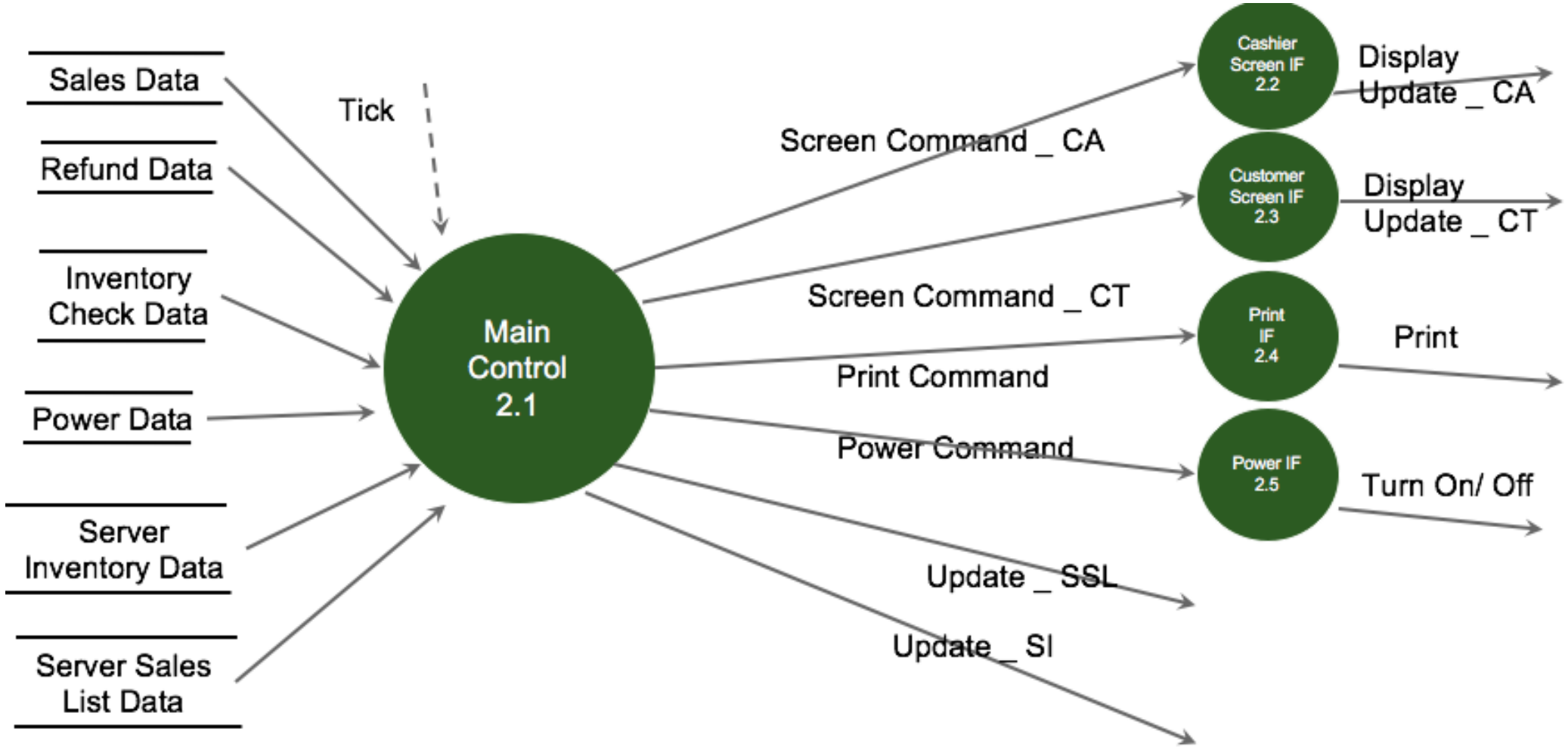
1. Revision

1.1 DFD Revision

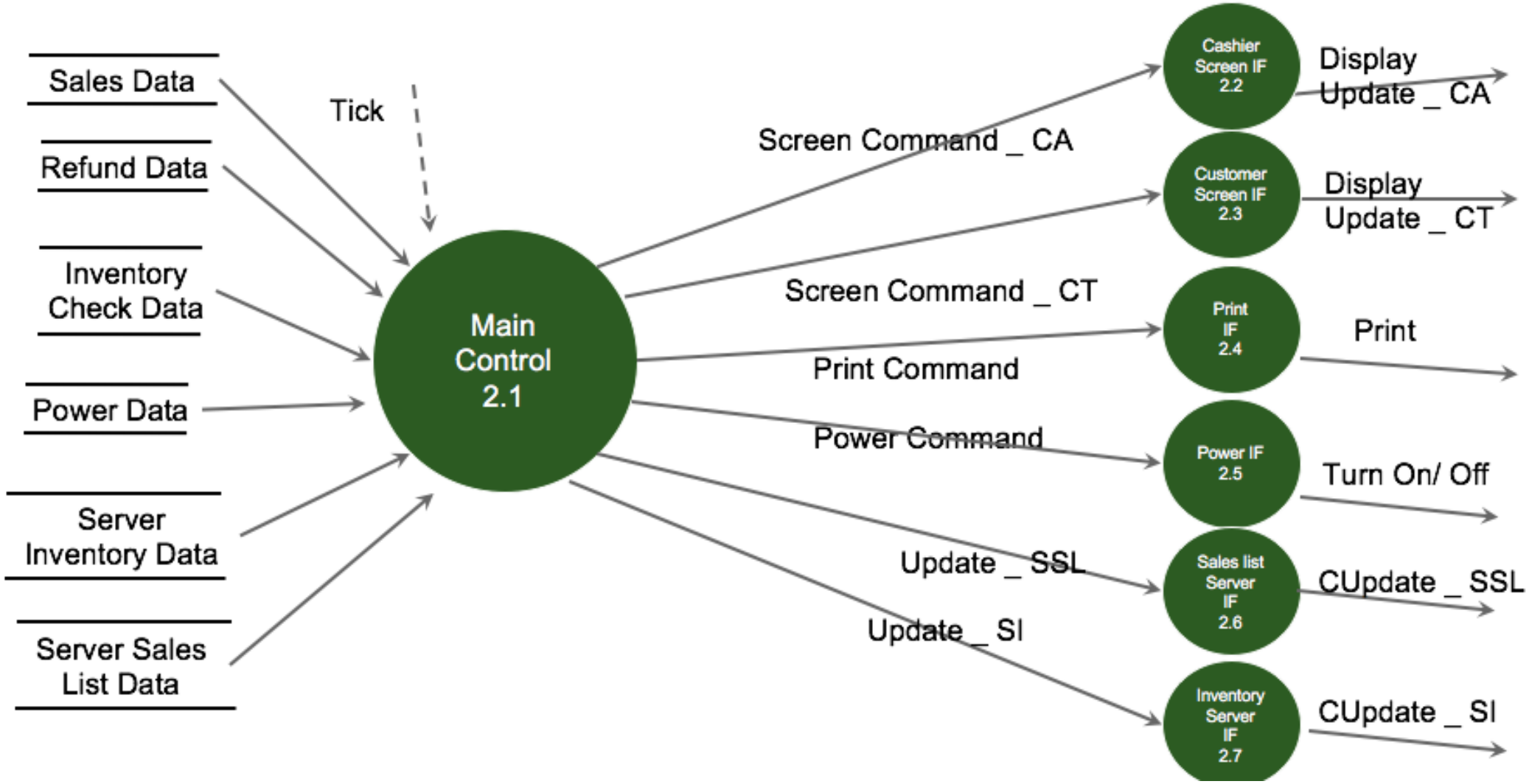
1.2 STD Revision

1.3 Structured Chart

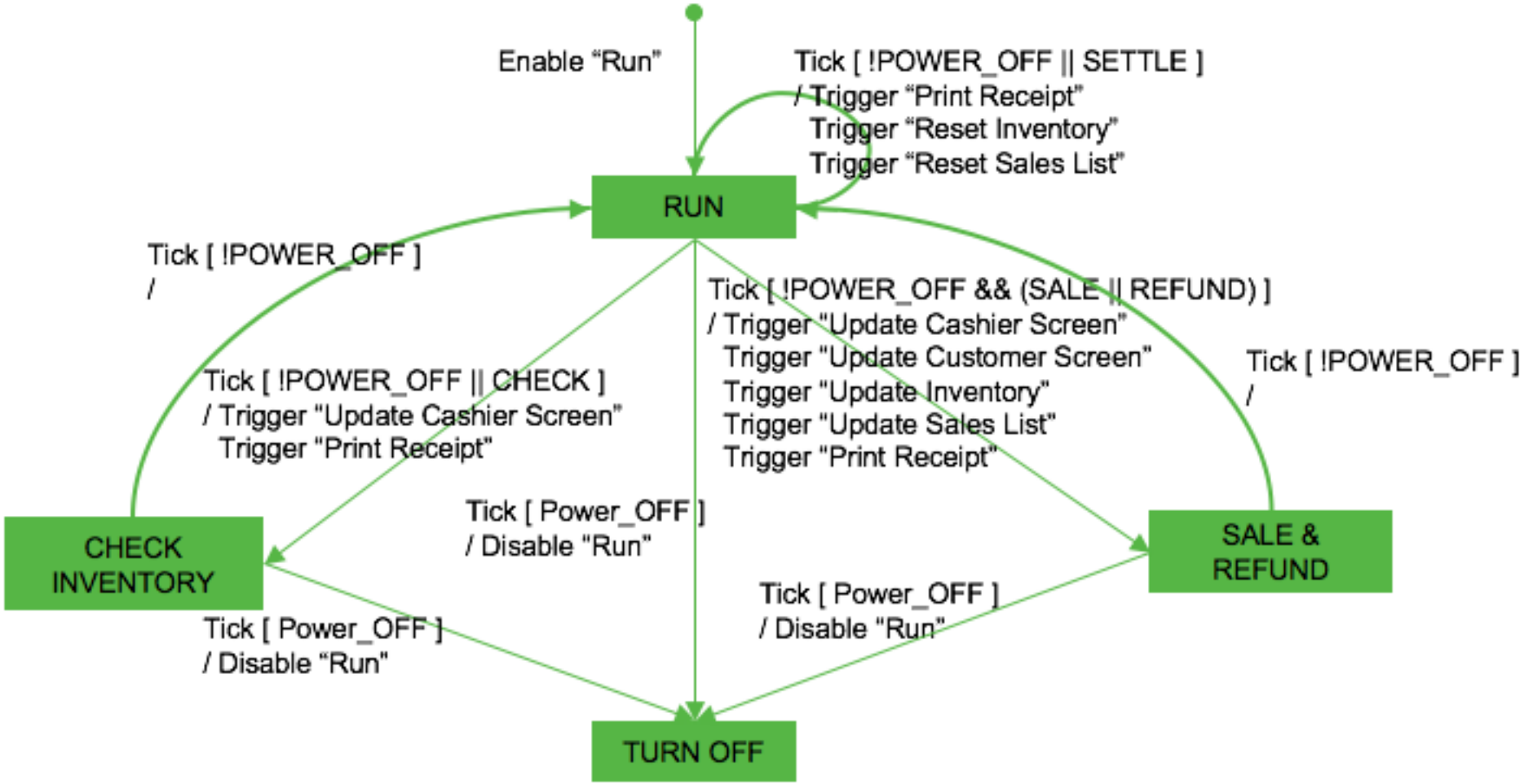
1.1 DFD Revision (Original)



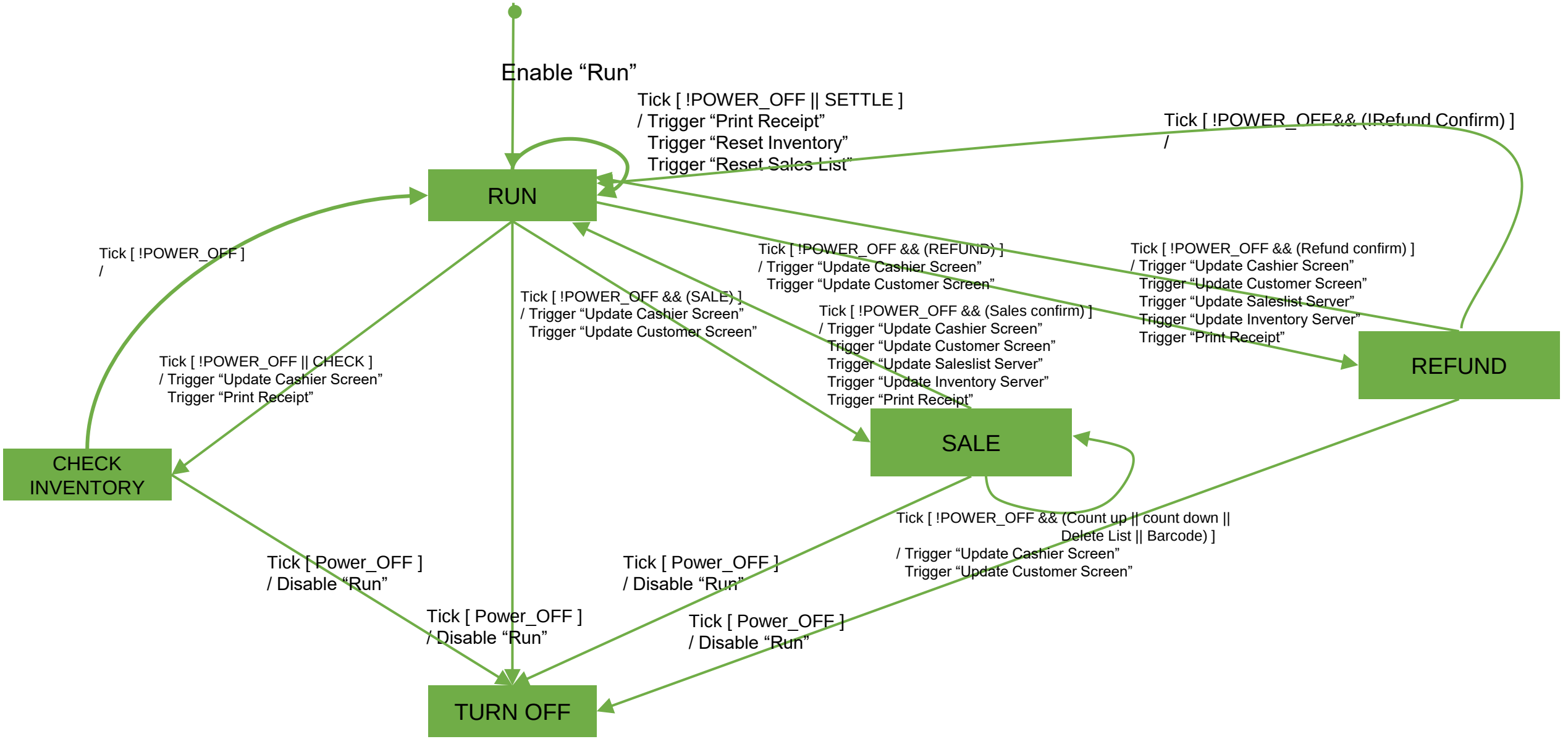
1.1 DFD Revision (Advanced)



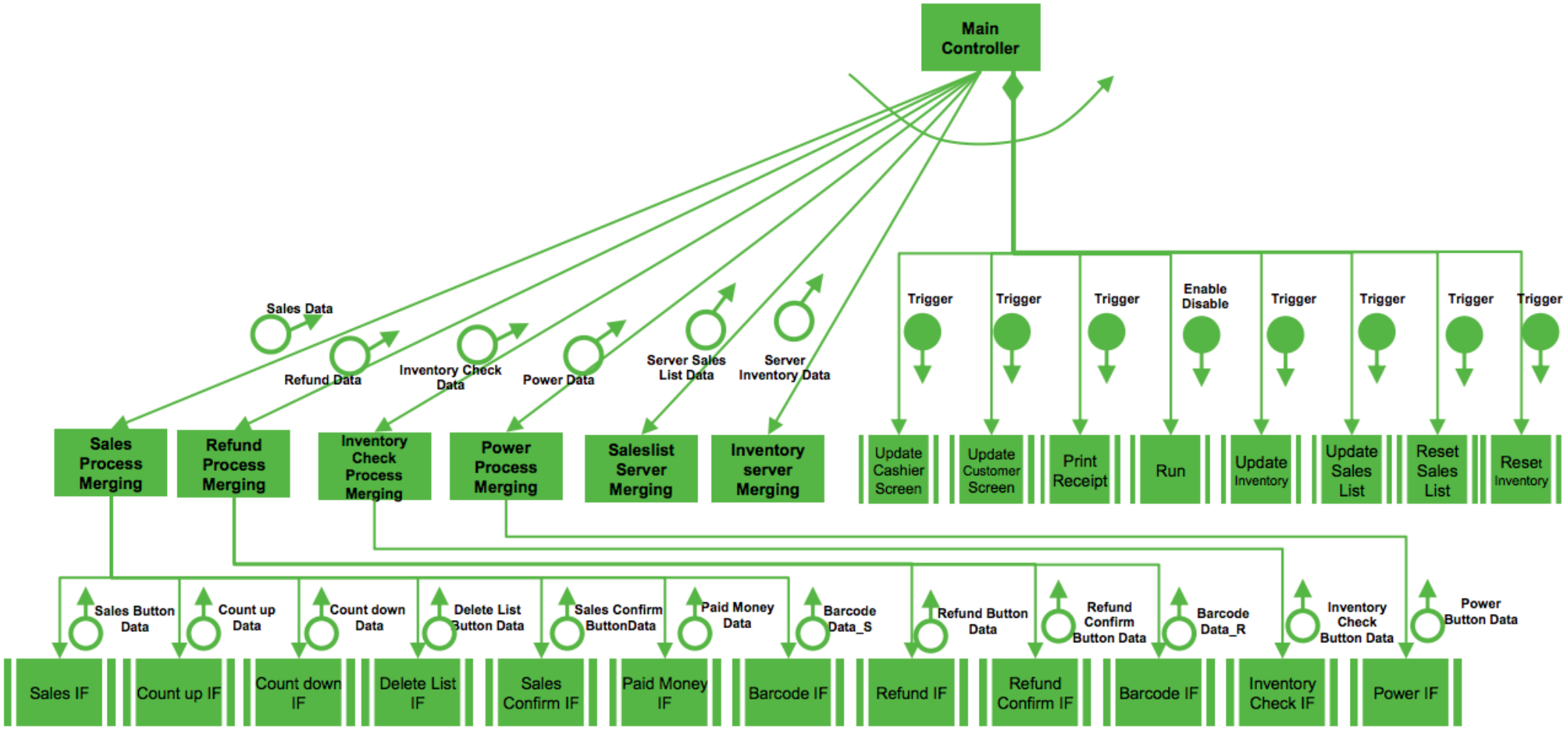
1.2 STD Revision (Original)



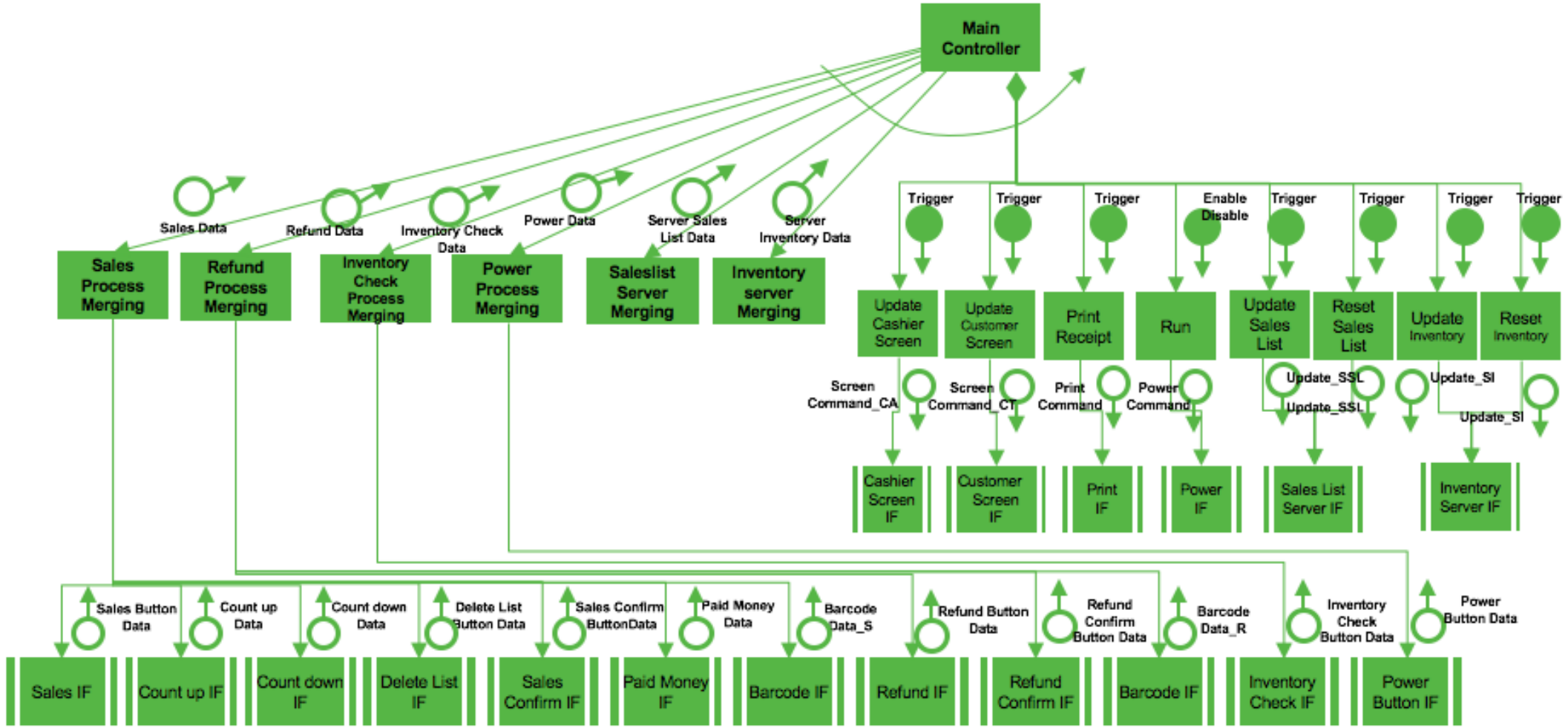
3.5. Data Flow Diagram _ Level 4



1.3 Structured Chart Revision (Original)

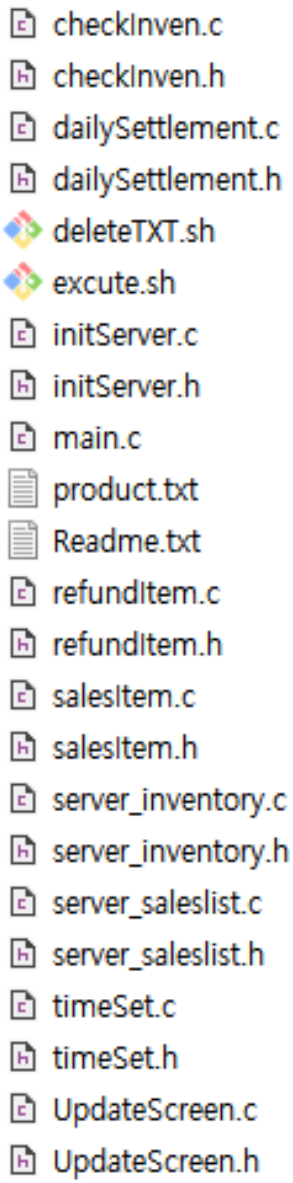


1.3 Structured Chart Revision (Advanced)



2. Implementation

Codes for POS

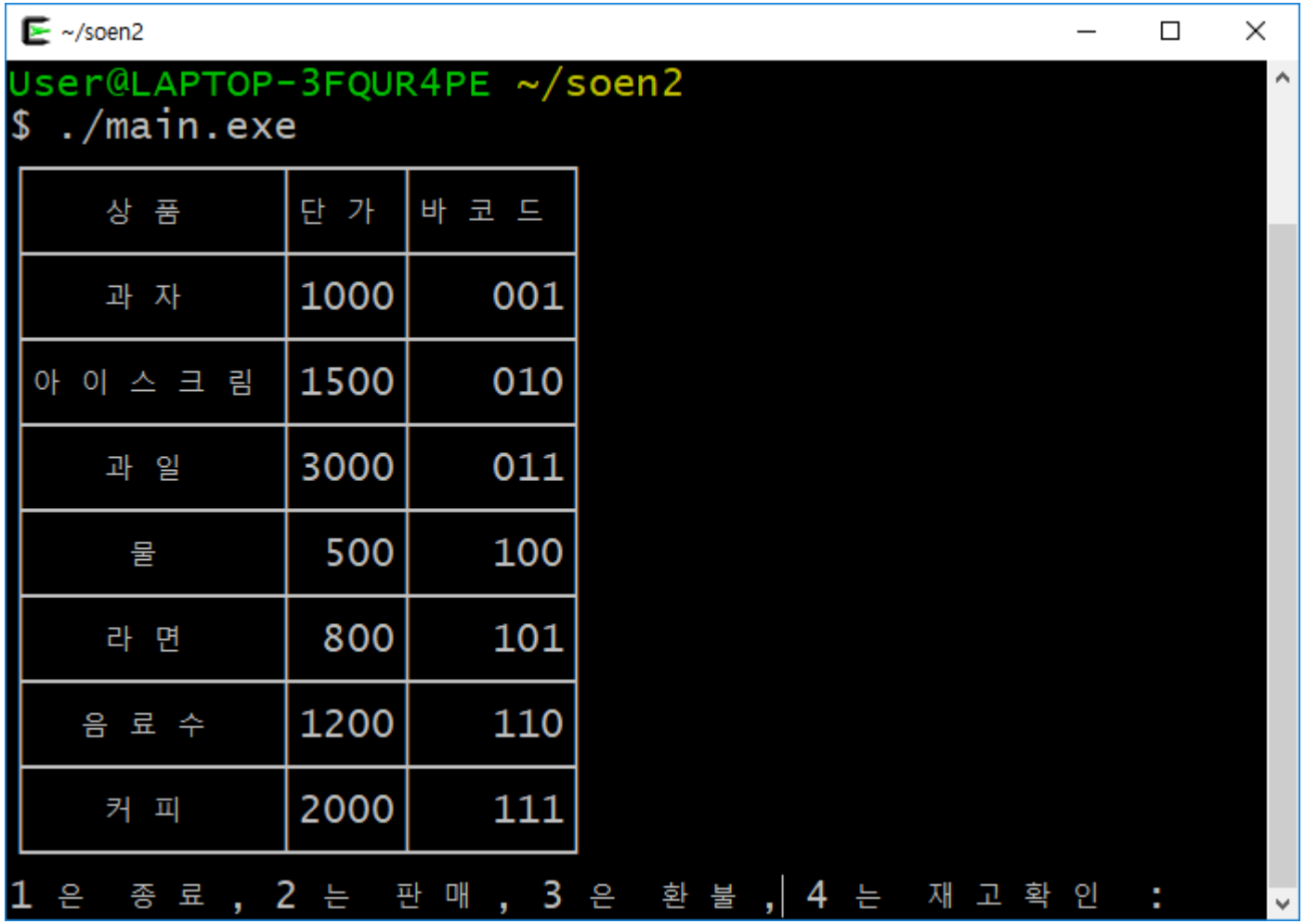


- **CheckInven :** 재고 확인을 위한 함수들
- **dailySettlement :** 정산을 위한 함수들
- **initServer :** 서버의 초기화 혹은 **Reset**
- **Main :** **Main Controller**
- **refundItem :** 환불을 위한 함수들
- **salesItem :** 구매를 위한 함수들
- **Server_Inventory :** 재고 서버의 구조체와 함수들
- **Server_saleslist :** 판매목록 서버의 구조체와 함수들
- **timeSet :** 시간의 흐름을 위한 함수들**(Interface)**
- **UpdateScreen :** **Screen** 출력을 위한 함수들

Main

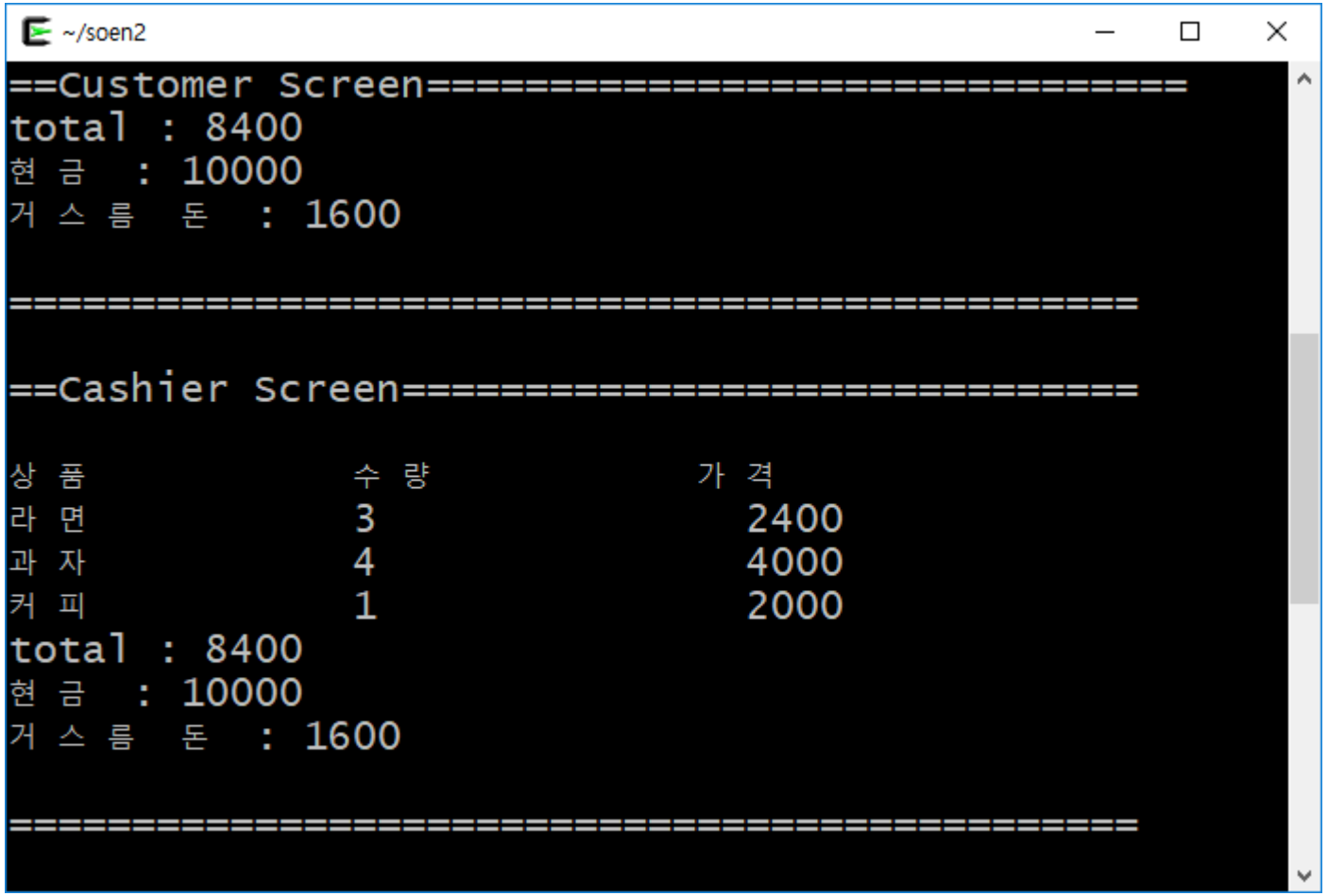
```
#include "server_inventory.h"
#include "server_saleslist.h"
#include "UpdateScreen.h"
#include "dailySettlement.h"
#include "initServer.h"
#include "salesItem.h"
#include "timeSet.h"
#include "checkInven.h"
#include "refundItem.h"

void Sale_Process_merging(TimeName *tn);
void settlement();
void Refund_Process_merging(TimeName *tn);
void Inventory_Check_Process_merging(TimeName
int Run();
void Power_Process_merging(TimeName *tn);
```



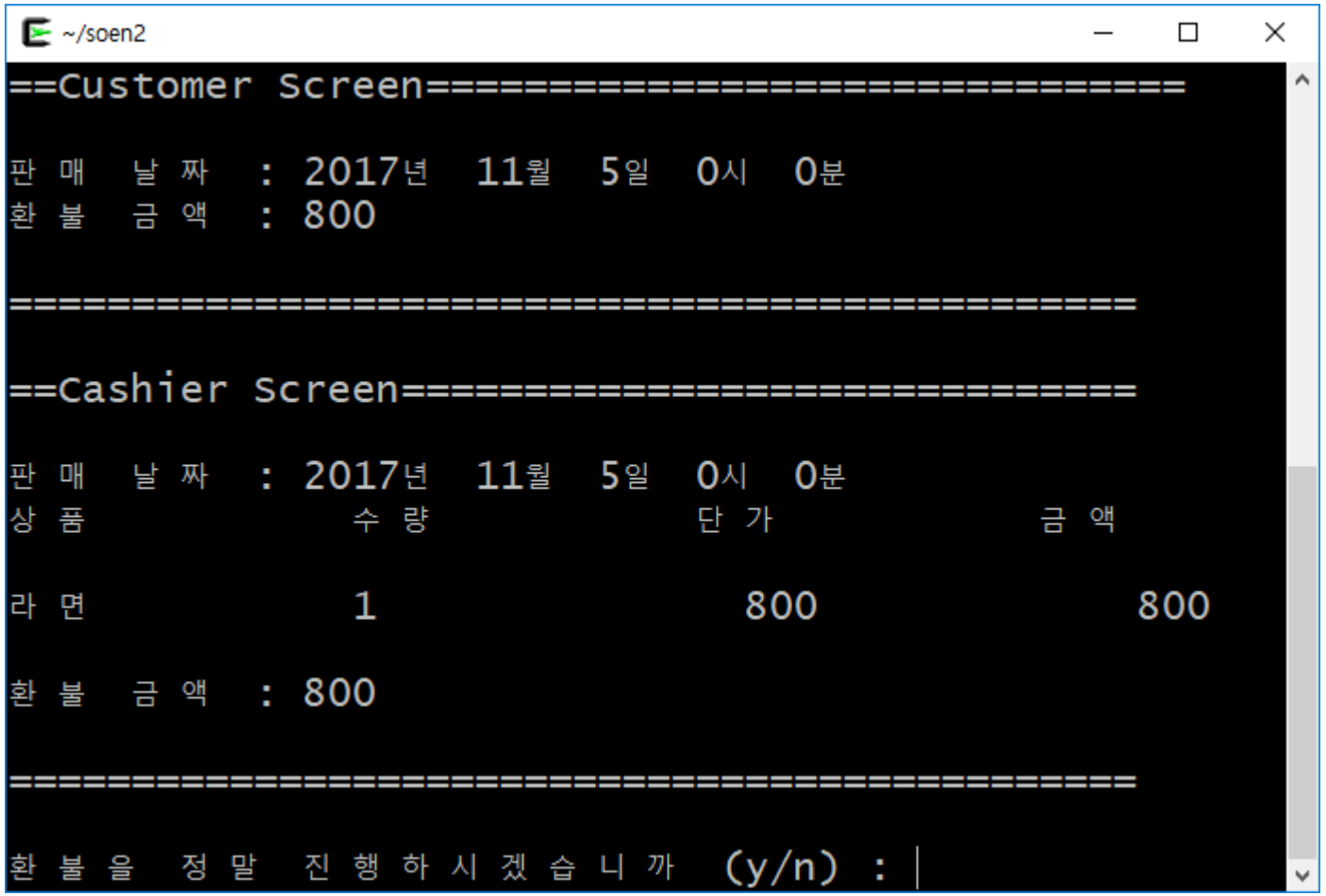
Sale

```
int decodeBarcode(ItemData* itd, char* bar_code);
int adjustQuantity(ItemData* itd, int Inum, int choiceNum);
int checkRecipt(int list[][2], int len, int Inum);
void Sale_Receipt_process(ItemData* itd,int list[][2], int len, char * reciptFileName, char* reciptID);
void salesWriter(char * saleFileName, ItemData* itd,int list[][2], int len, char* receipt_code);
```



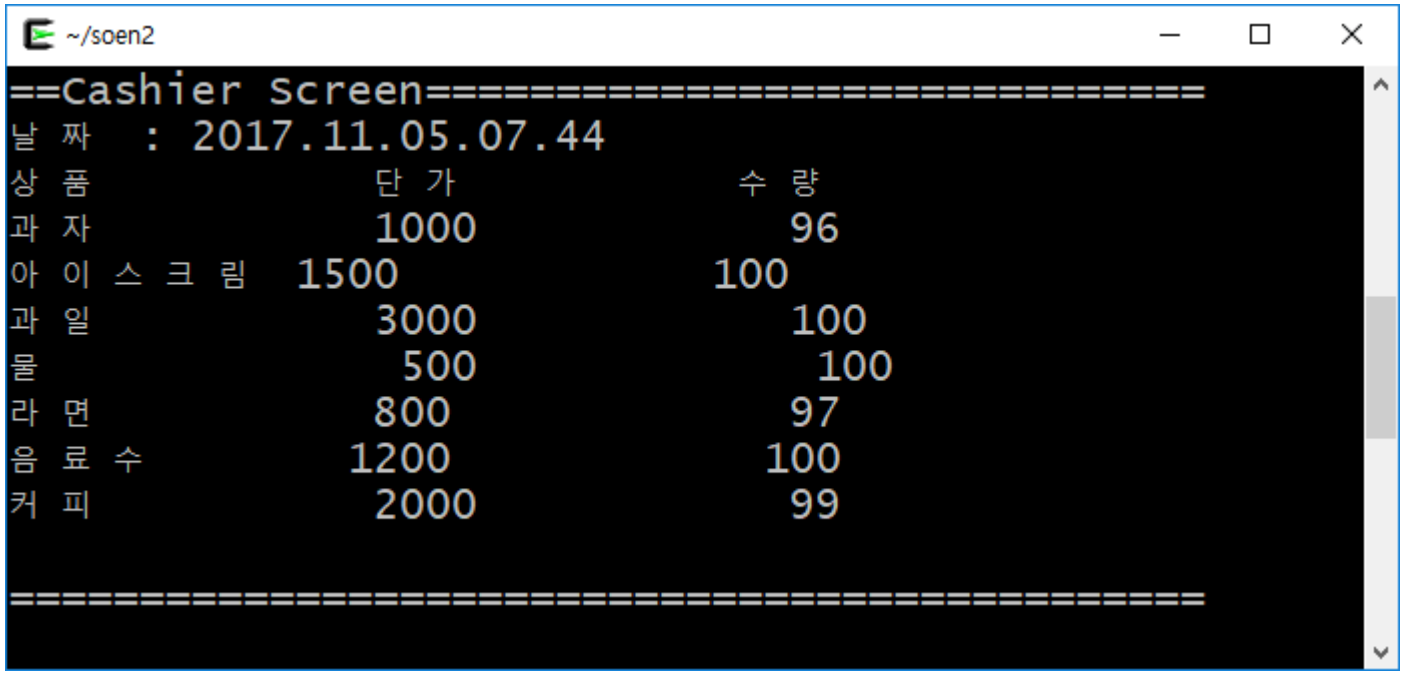
Refund

```
void setRefundDate(TimeName *tn, char* code);
void printRefundDisplay(TimeName* tn, SoldData* rd, int len);
void Refund_Receipt_process(SoldData* rd, char* FileName, int len, char* receiptNum);
void refillProduct(ItemData * itd, SoldData * rd, int itd_len, int rd_len);
void refundWriter(char* FileName, SoldData* rd, int len, char* receiptNum);
int LookupRefundList(char* code);
void SaveRefundList(char* code);
```



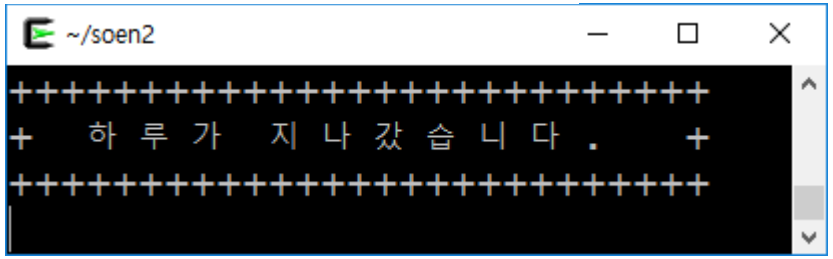
Check

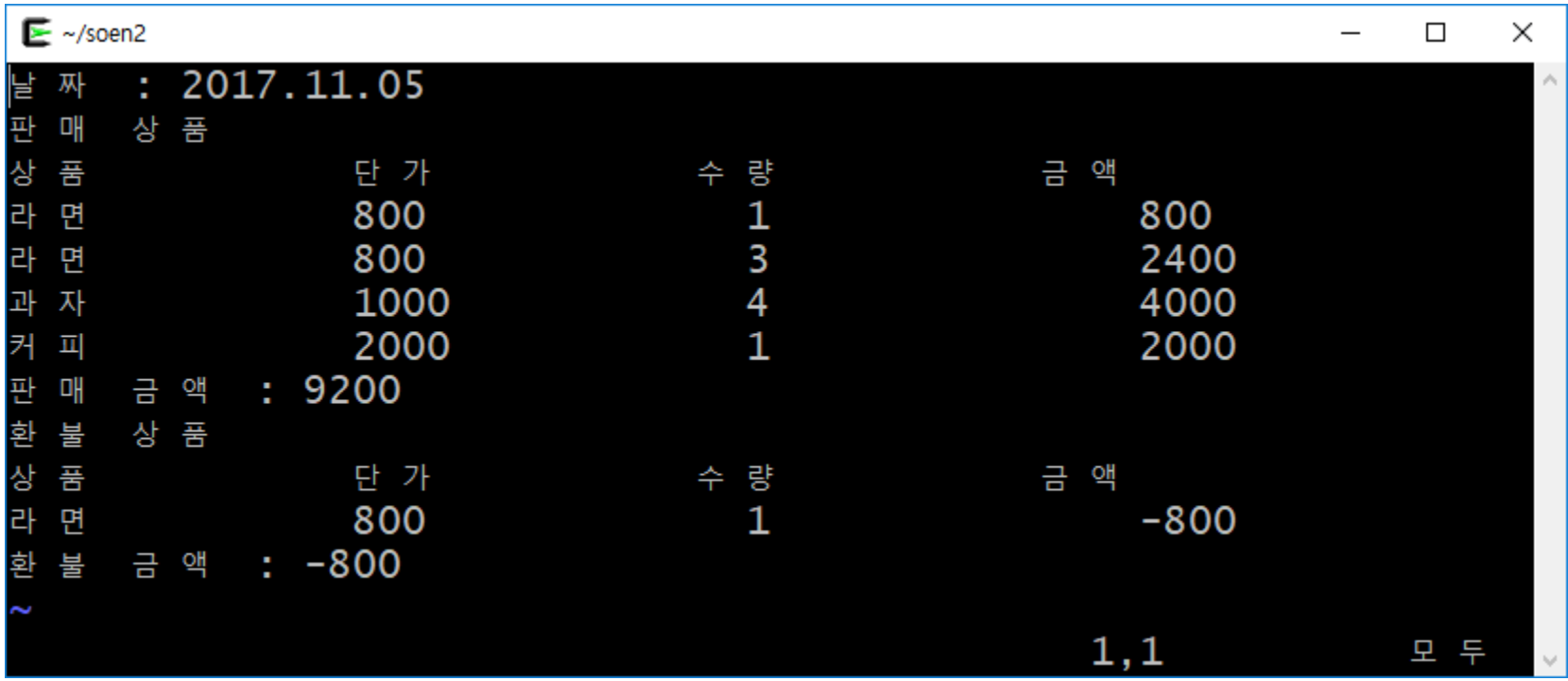
```
void Check_Receipt_process(char* receiptName, ItemData* itd, char* DateData);
```



Settlement

```
void printSettlementReport(TimeName* tn, int s_index);
void analyzeSaleManagement(char* serverName, SoldData ** sales, SoldData  ** refunds, int* saleLen, int* refundLen);
int findIndex(SoldData* sd, char * name, int len);
```





Inventory / Saleslist Server

initServer

```
#define PRODUCT_FILE "product.txt"
#define PRODUCT_COUNT 100

void initServer(TimeName* tn);
int Reset_Inventory(char * productFileName);
int Reset_Saleslist(char * FileName);
```

Server_Saleslist

```
typedef struct _SoldData
{
        char item[50];
        int quantity;
        int money;
        int pay;
}SoldData;

int Saleslist_server_Merging(char* saleProductFile, SoldData* rd, char* code);
int Update_Sales_List(char* FileName, SoldData* rd, int len, char* receiptNum);
```

Server_Inventory

```
typedef struct _ItemData
{
        char item[50];
        int money;
        char bar_code[4];
        int stock;
}ItemData;

int Update_Inventory(char * productFileName, ItemData* itd, int len);
int Print_Receipt(char * receiptFileName, char * str);
int Inventory_server_Merging(char* productFileName, ItemData* itd);
```

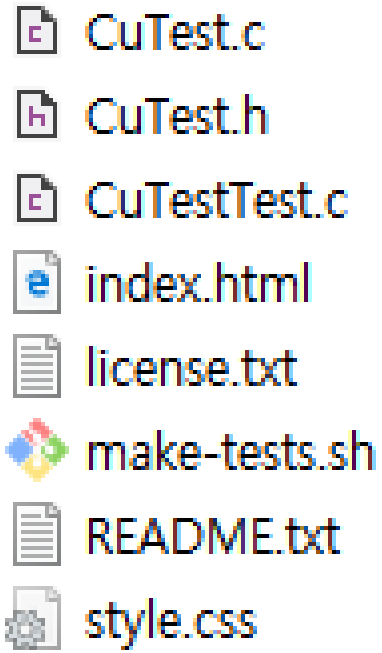
2. Unit Test

Unit Test Plan

Use
CuTest
Ver 1.5

For
Unit Test

In Linux (Cygwin)



Unit Test Plan

단순 **Data** 전달 **Process**나 **Contol** 전달 **Process**는 **Unit Test**에서 제외

Identifier	Feature	Feature
POS.UTC.111	**Sale Process** 가 흐르는지 확인	**SALE = TRUE/False**
POS.UTC.121	**Refund Process** 가 흐르는지 확인	**REFUND = TRUE/False**
POS.UTC.131	**Inventory Check Process**가 흐르는지 확인	**CHECK = TRUE/False**
POS.UTC.141	**Power Process**가 흐르는지 확인	**POWER_OFF = True/False**
POS.UTC.151	**SalesList Merging** 되는지 확인	**Saleslist_server_Merging = 0 or True**
POS.UTC.152	**Inventory Merging** 되는지 확인	**Inventory_server_Merging = 0 or 1**
POS.UTC.211	**Main Controller**가 제대로 수행되는지 확인	통합적인 부분(전달 프로세스)이므로 **Test** 제외
POS.UTC.212	**Update_Cashier_Screen**가 수행 되어지는지 확인	**Update_Cashier_Screen=0 or 1**
POS.UTC.213	**Update_Customer_Screen**가 수행 되어지는지 확인	**Update_Cashier_Screen=0 or 1**
POS.UTC.214	**Print_Receipt**가 수행 되어지는지 확인	**Print_Receipt=0 or 1**
POS.UTC.215	**Run**이 작동하는지 확인. 잘못된 데이터시 **Run =0**	**Run = 0 or 1**
POS.UTC.216	**Reset_Saleslist**가 수행 되어지는지 확인	**Reset_Saleslist = 0 or 1**
POS.UTC.217	**Update_Sales_List**가 수행 되어지는지 확인	**Update_Sales_List = 0 or 1**
POS.UTC.218	**Update_Inventory**가 수행 되어지는지 확인	**Update_Inventory = 0 or 1**
POS.UTC.219	**Reset_Inventory**가 수행 되어지는지 확인	**Reset_Inventory = 0 or 1**

Unit Test Result

Total Cases: 19
Fail: 0
Pass: 19