

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

2. Implementation

3. Unit Test

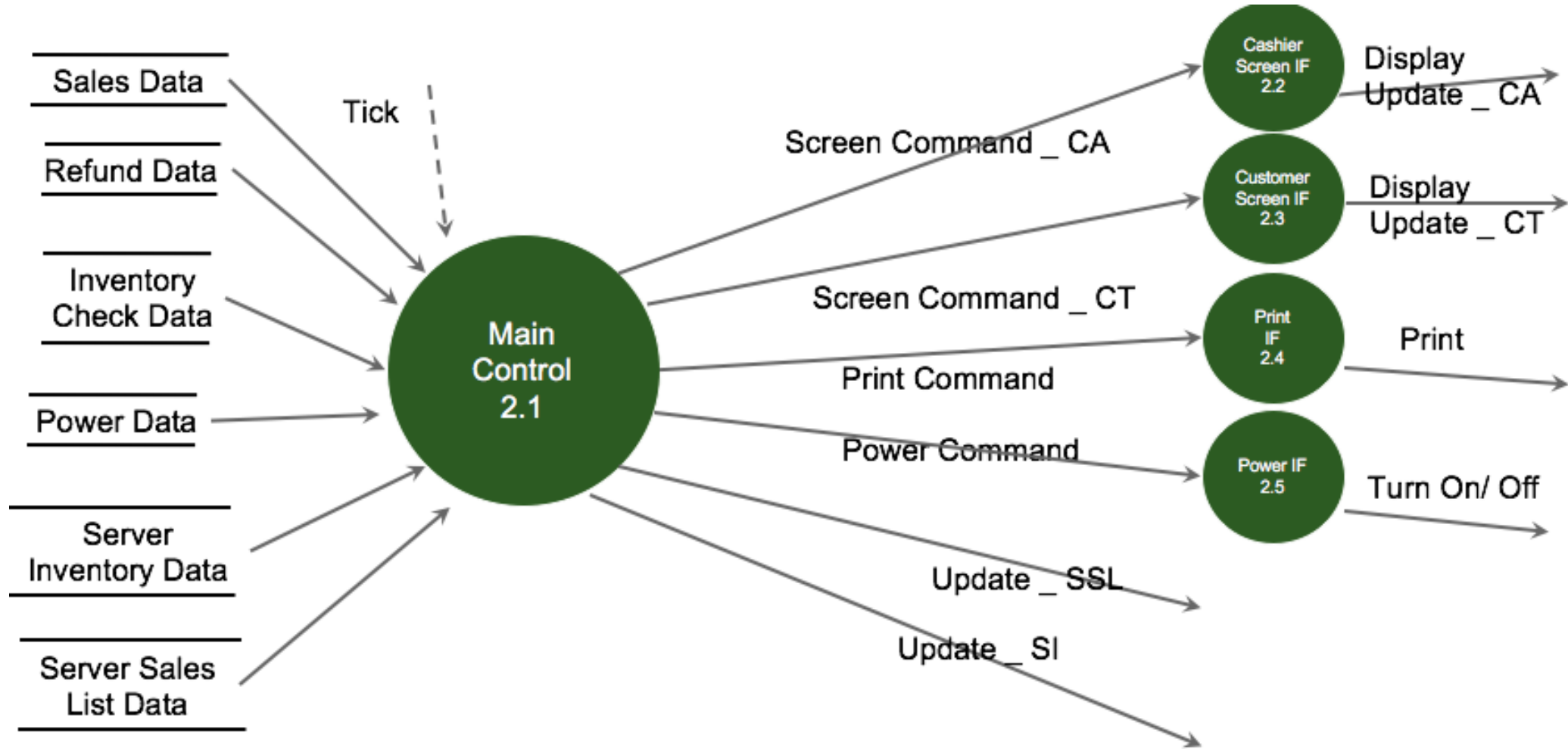
1. Revision

1.1 DFD Revision

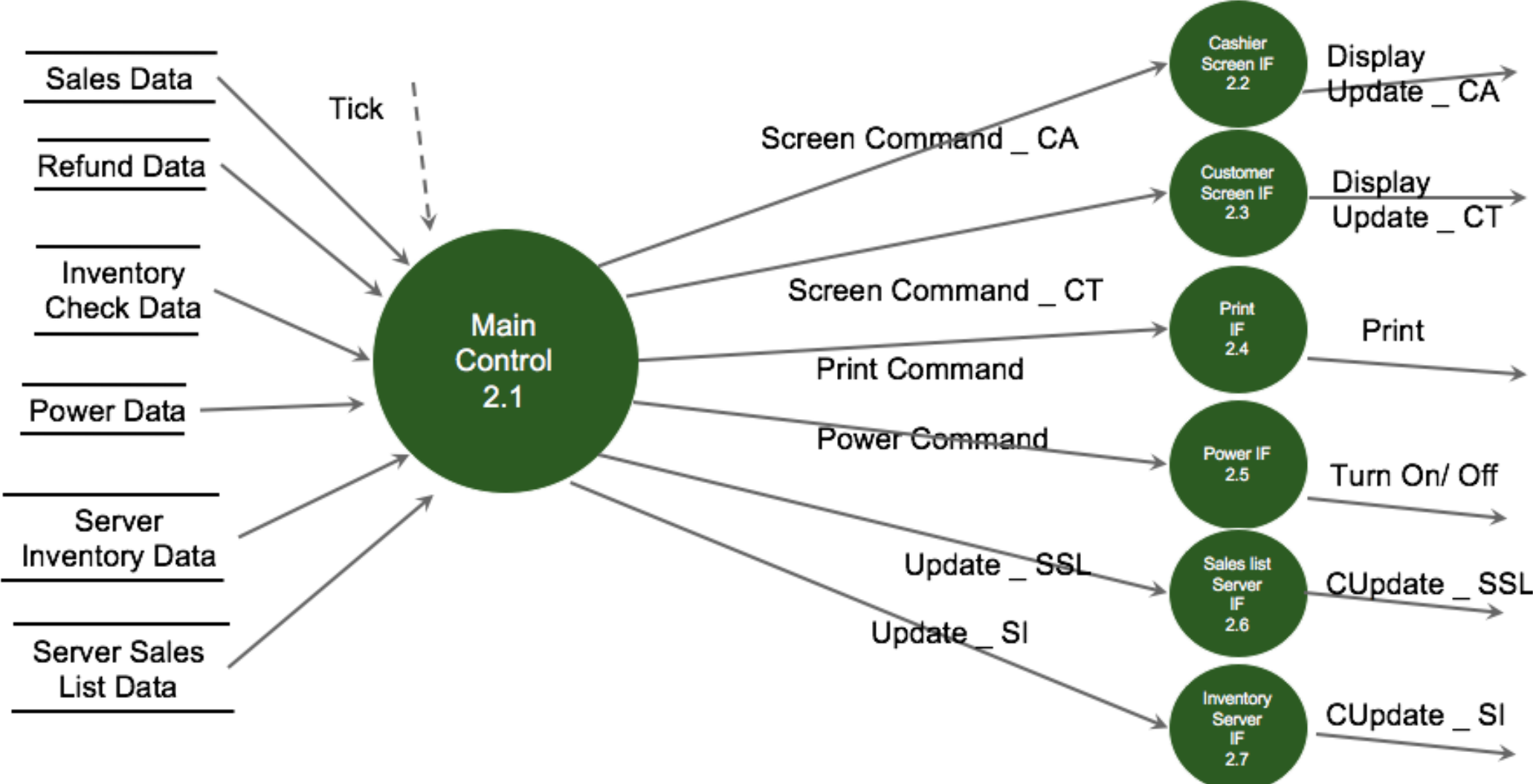
1.2 STD Revision

1.3 Structured Chart

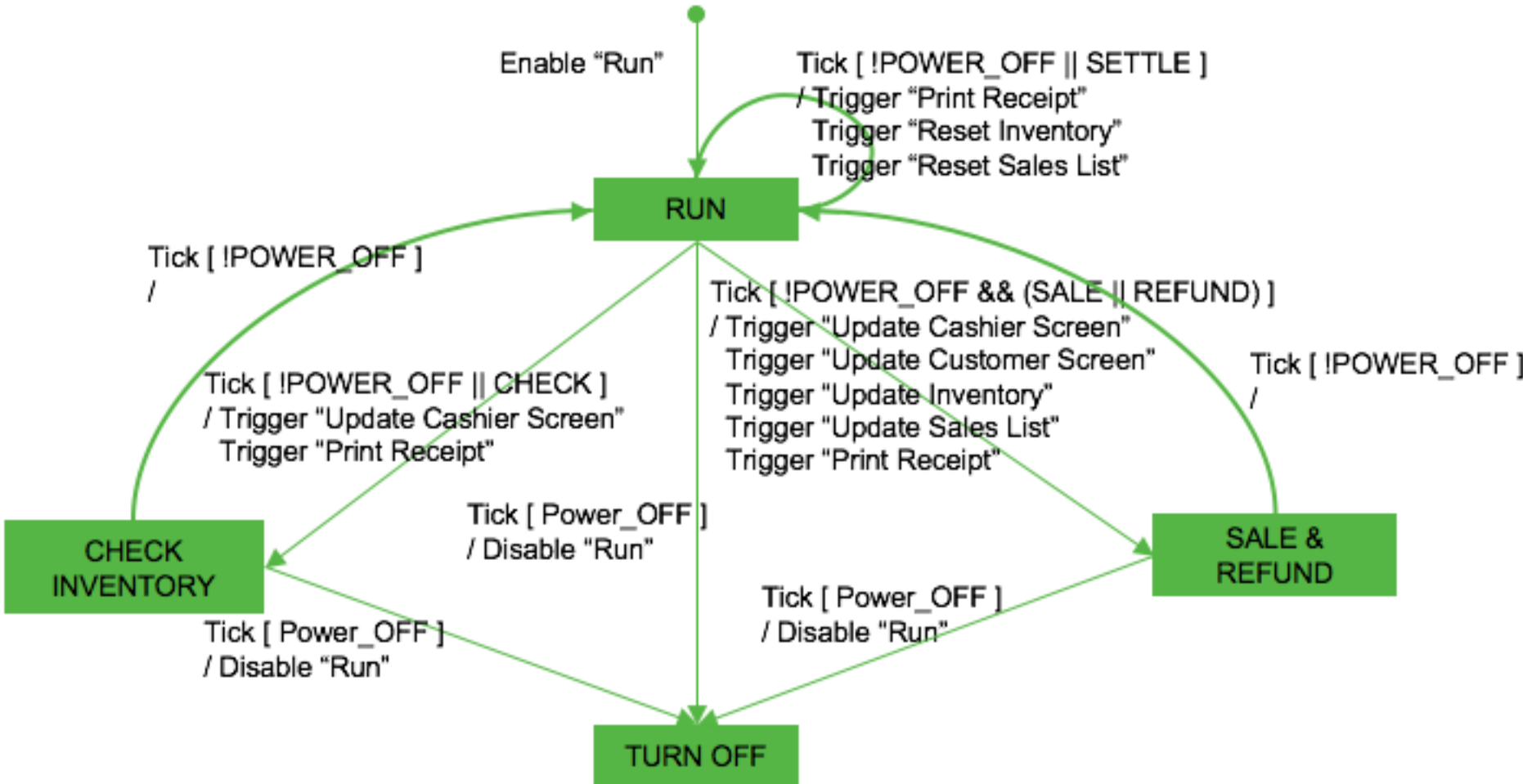
1.1 DFD Revision (Original)



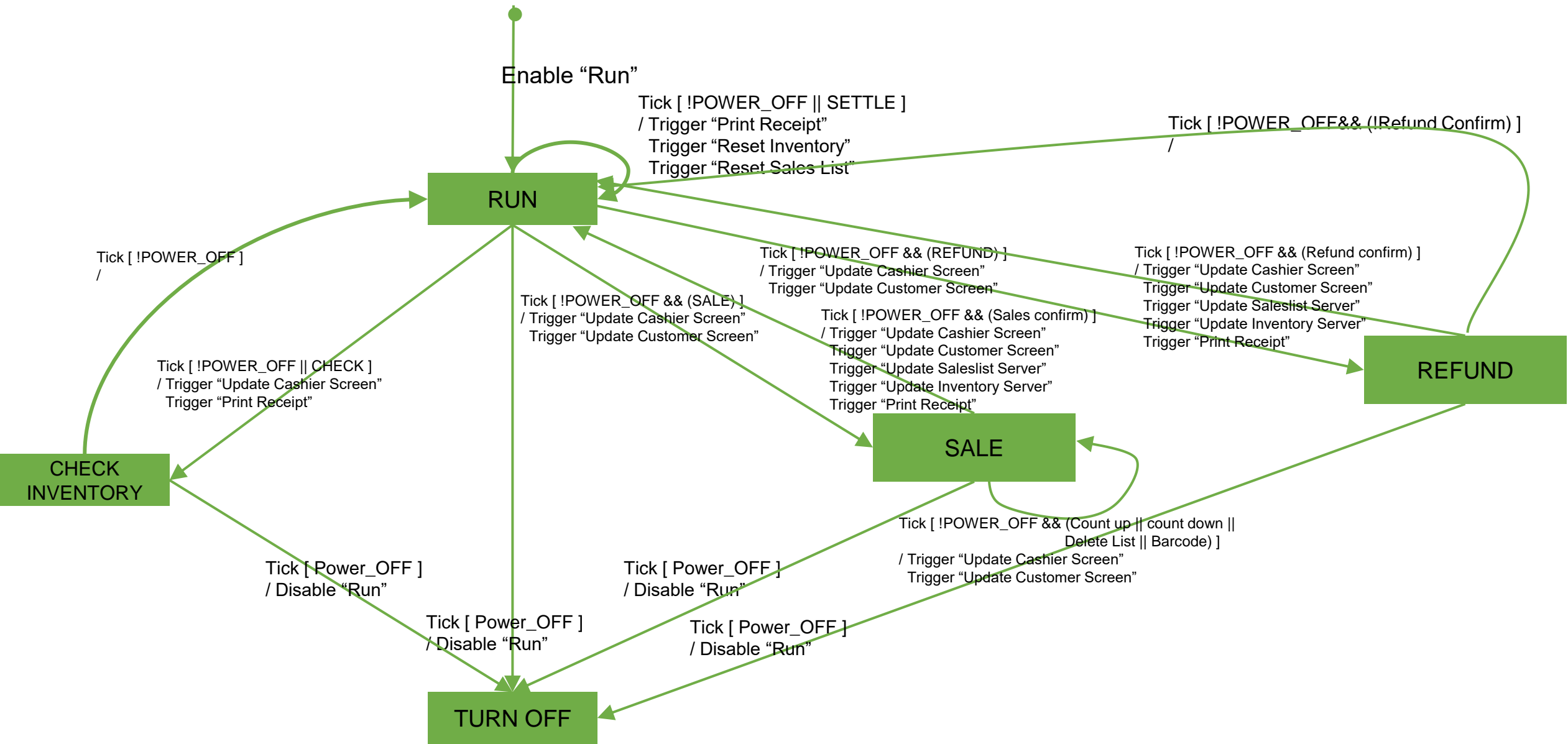
1.1 DFD Revision (Advanced)



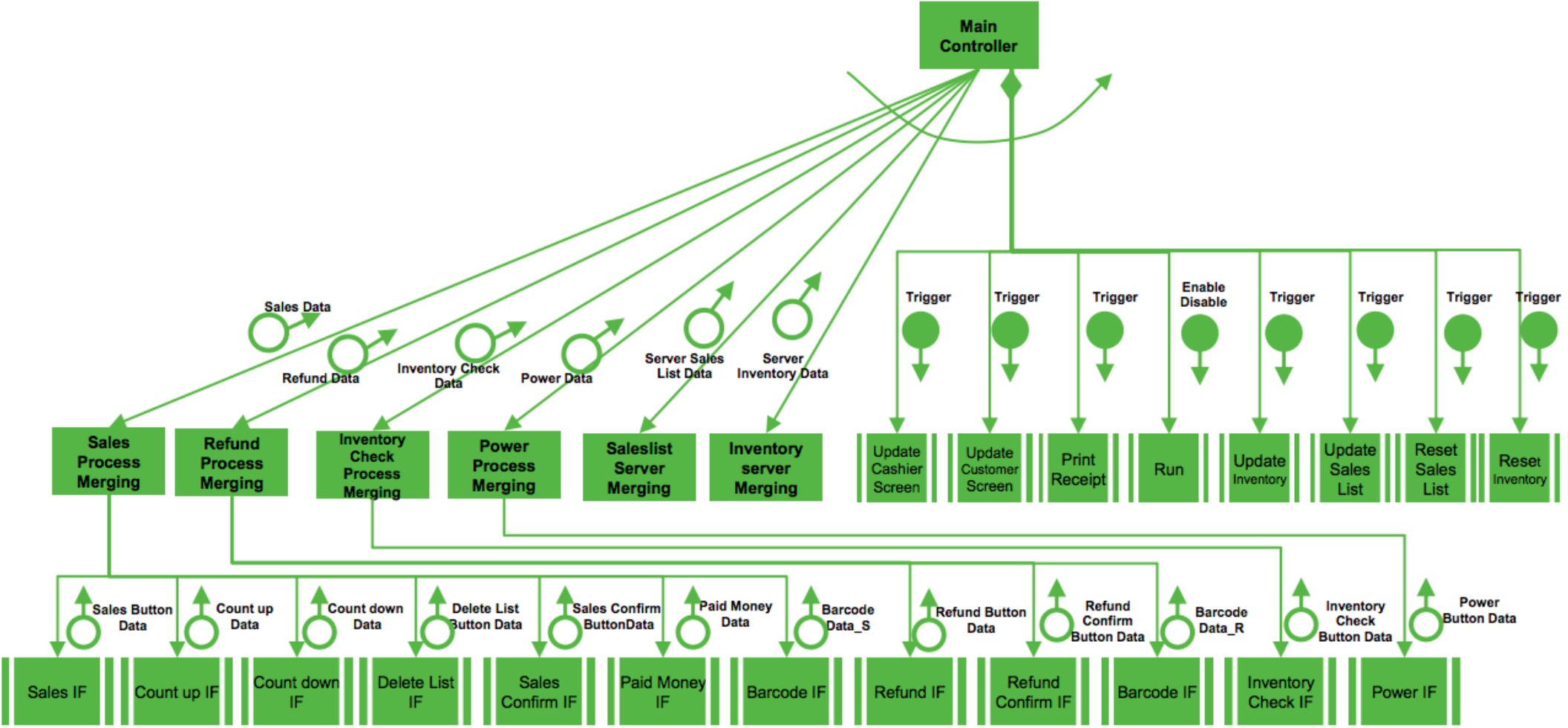
1.2 STD Revision (Original)



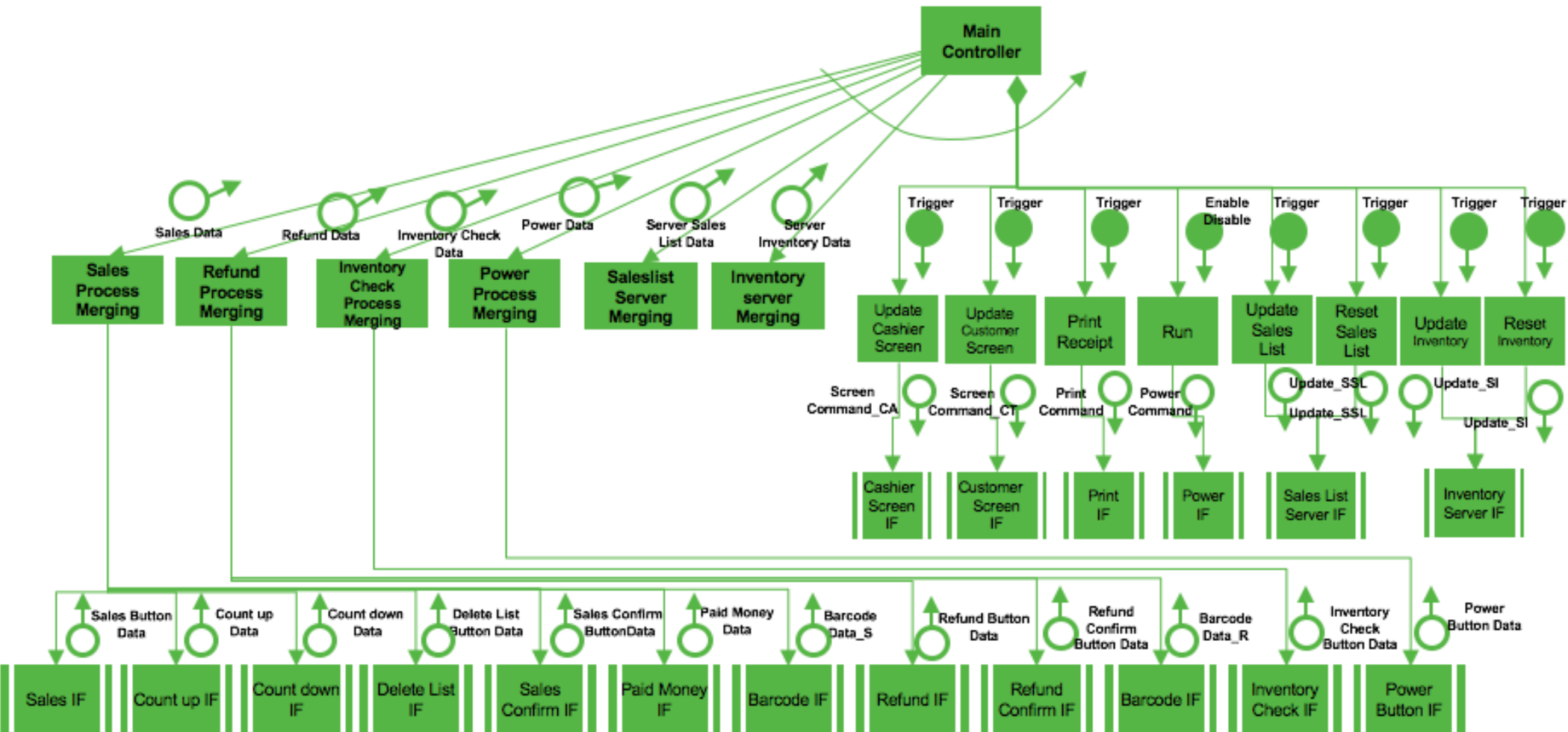
3.5. Data Flow Diagram _ Level 4



1.3 Structured Chart Revision (Original)



1.3 Structured Chart Revision (Advanced)



2. Implementation

Codes for POS

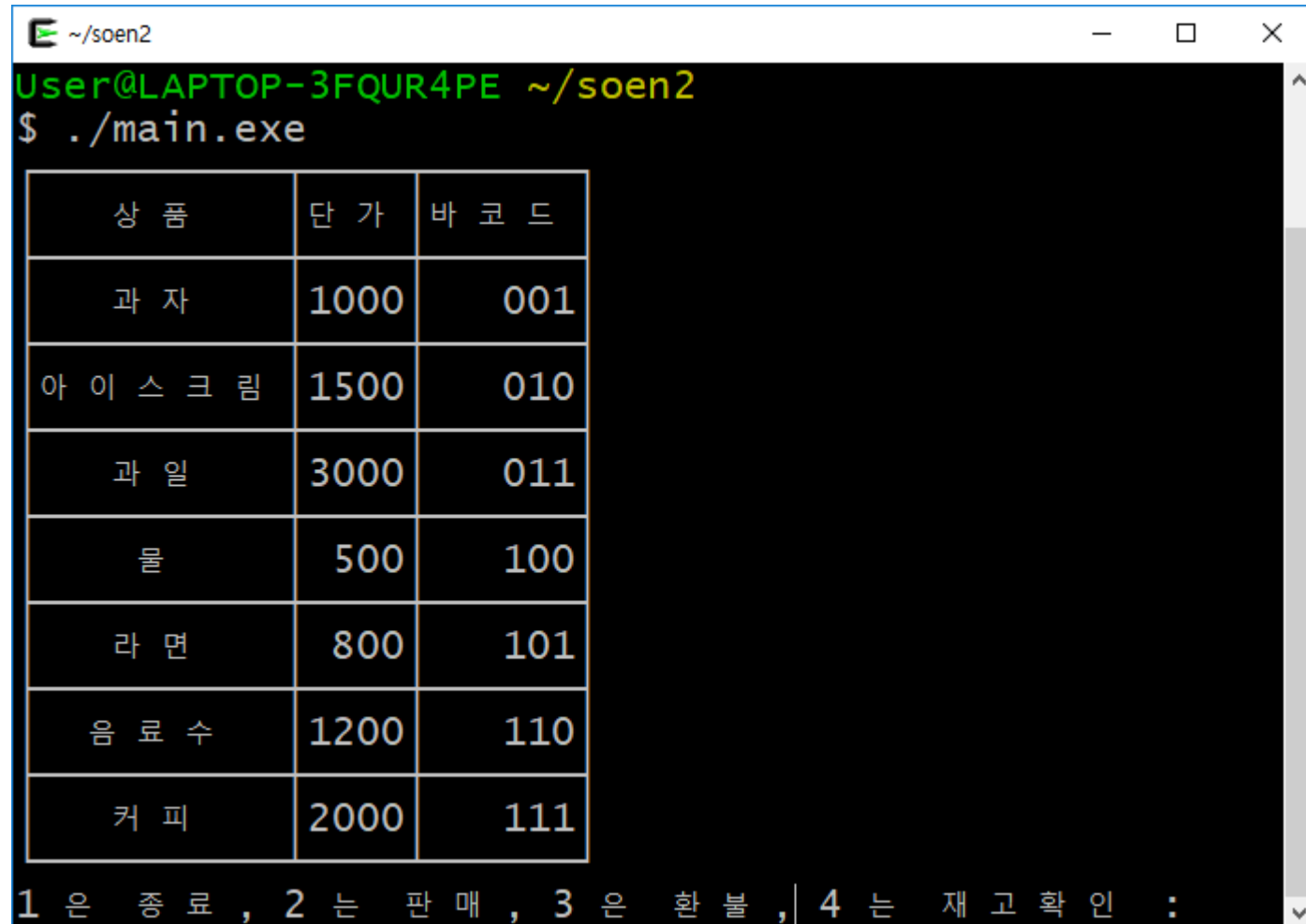
- checkInven.c
- checkInven.h
- dailySettlement.c
- dailySettlement.h
- deleteTXT.sh
- excute.sh
- initServer.c
- initServer.h
- main.c
- product.txt
- Readme.txt
- refundItem.c
- refundItem.h
- salesItem.c
- salesItem.h
- server_inventory.c
- server_inventory.h
- server_saleslist.c
- server_saleslist.h
- timeSet.c
- timeSet.h
- UpdateScreen.c
- UpdateScreen.h

- **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);
```



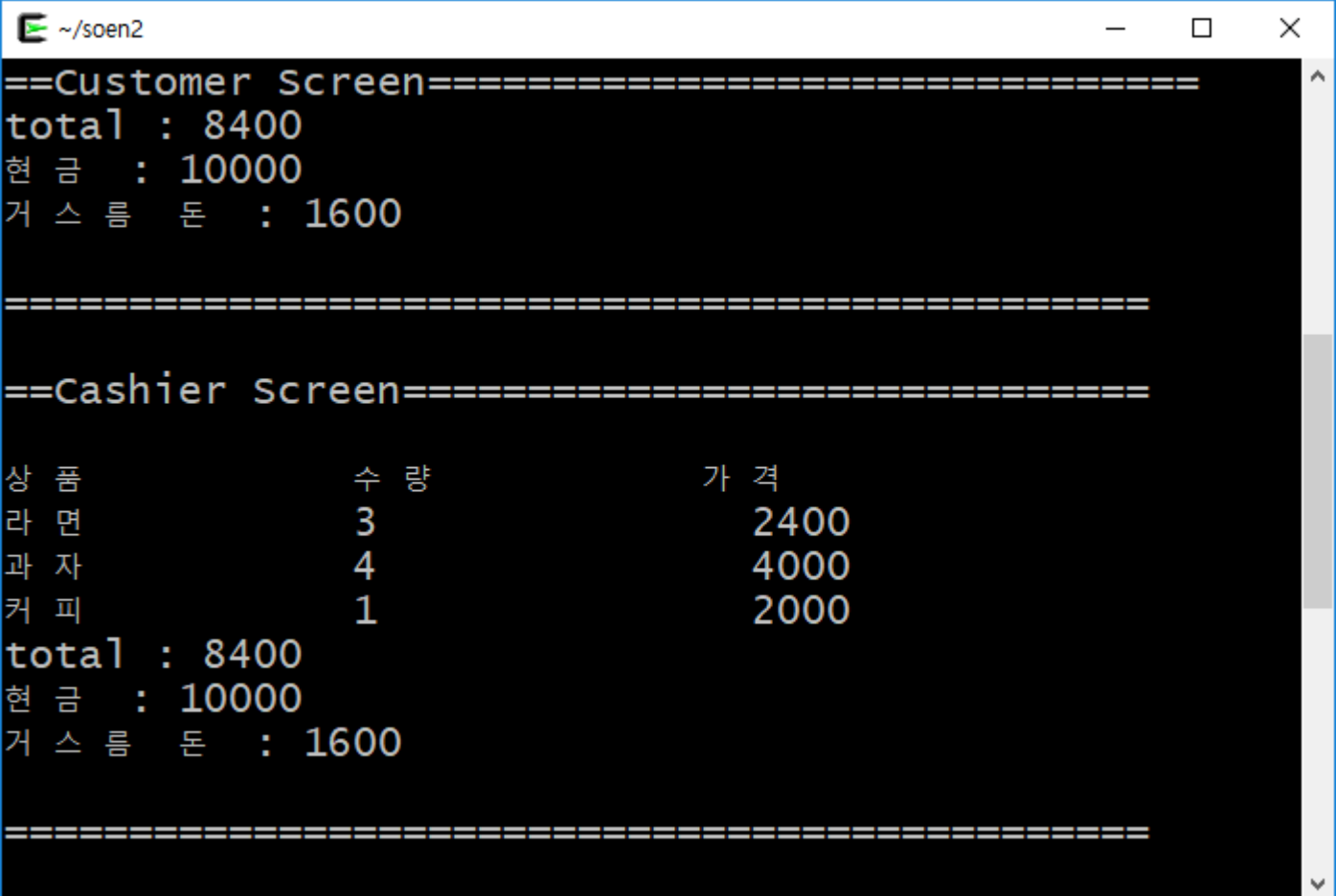
```
~/soen2
User@LAPTOP-3FQR4PE ~/.soen2
$ ./main.exe
```

| 상 품 | 단 가 | 바 코드 |
|-----------|------|------|
| 과 자 | 1000 | 001 |
| 아 이 스 크 림 | 1500 | 010 |
| 과 일 | 3000 | 011 |
| 물 | 500 | 100 |
| 라 면 | 800 | 101 |
| 음 료 수 | 1200 | 110 |
| 커피 | 2000 | 111 |

1 은 종 료 , 2 는 판 매 , 3 은 환 불 , 4 는 재 고 확 인 :

Sale

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



```
~/soen2  
==Customer Screen=====  
total : 8400  
현 금   : 10000  
거 스 름 돈   : 1600  
  
=====  
  
==Cashier screen=====  
  
상 품           수 량           가 격  
라 면           3           2400  
과 자           4           4000  
커피            1           2000  
total : 8400  
현 금   : 10000  
거 스 름 돈   : 1600  
  
=====
```

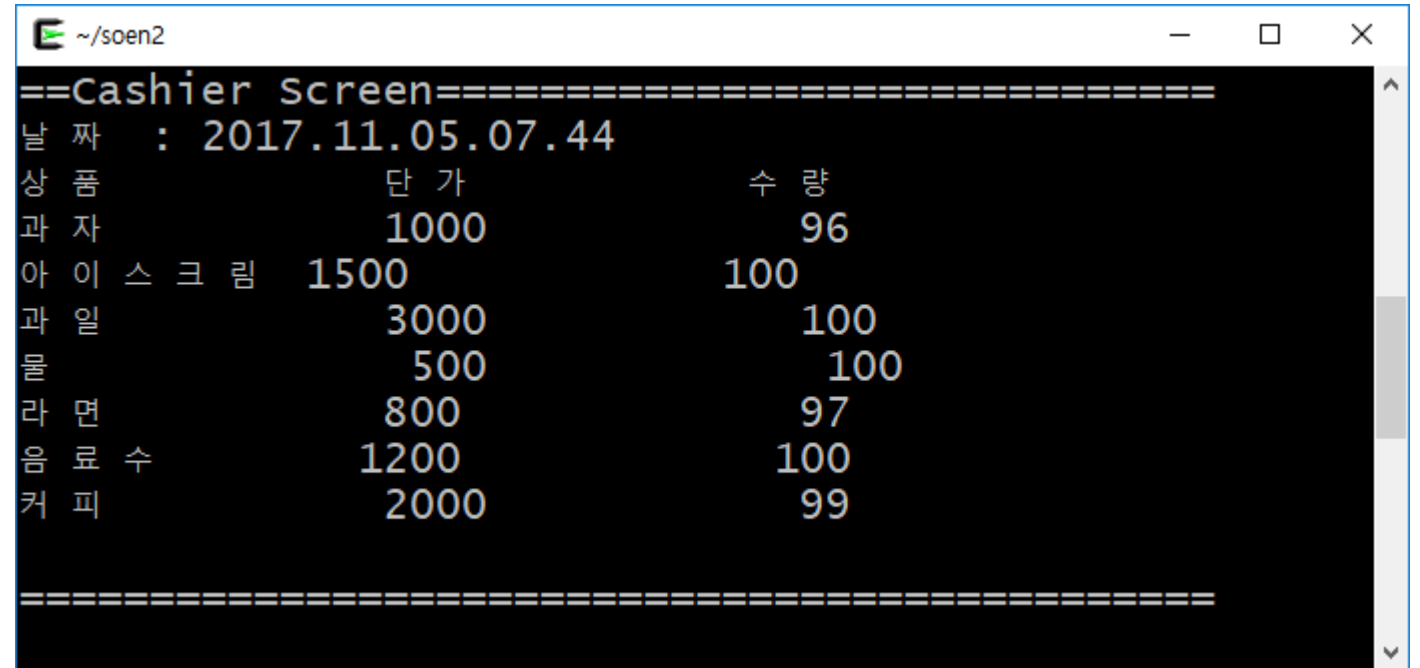
Refund

```
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);
```

```
~/soen2  
==Customer Screen=====^  
판 매 날 짜 : 2017년 11월 5일 0시 0분  
환 불 금 액 : 800  
=====^  
==Cashier Screen=====^  
판 매 날 짜 : 2017년 11월 5일 0시 0분  
상 품 수 량 단 가 금 액  
라 면 1 800 800  
환 불 금 액 : 800  
=====^  
환 불 을 정 말 진 행 하 시 겠 습 니 까 (y/n) : |
```

Check

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

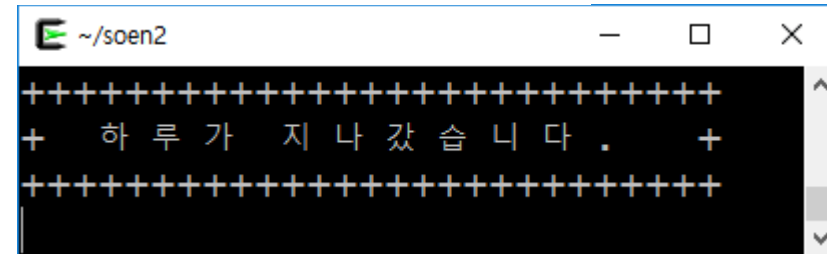


The screenshot shows a terminal window titled "~/soen2" with a black background and white text. The output is a receipt titled "Cashier Screen" with a date and time stamp "2017.11.05.07.44". Below the header is a table of items with columns for item name, unit price, and quantity. The items listed are: 과자 (1000), 아이스크림 (1500), 과일 (3000), 물 (500), 라면 (800), 음료수 (1200), and 커피 (2000). The quantities are: 96, 100, 100, 100, 97, 100, and 99 respectively. The terminal window has standard window controls (minimize, maximize, close) in the top right corner.

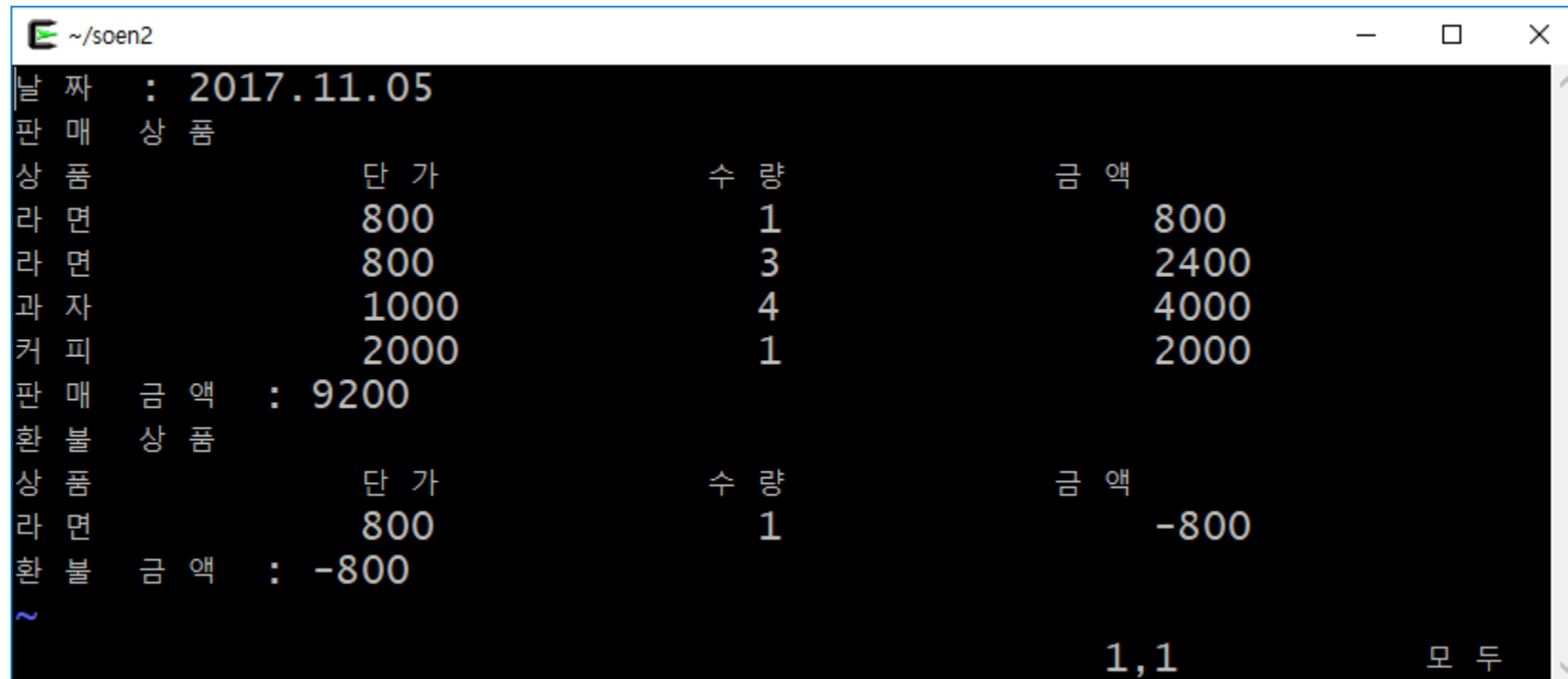
```
~/soen2
==Cashier Screen=====
날 짜   : 2017.11.05.07.44
상 품           단 가           수 량
과 자           1000           96
아 이 스 크 림  1500           100
과 일           3000           100
물              500           100
라 면           800            97
음 료 수       1200           100
커피            2000           99
=====
```

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);
```



```
~/soen2  
+++++  
+ 하루가 지나갔습니다. +  
+++++
```



```
~/soen2  
날 짜 : 2017.11.05  
판 매 상 품  
상 품          단 가          수 량          금 액  
라 면          800           1            800  
라 면          800           3            2400  
과 자          1000          4            4000  
커 피          2000          1            2000  
판 매 금 액 : 9200  
환 불 상 품  
상 품          단 가          수 량          금 액  
라 면          800           1            -800  
환 불 금 액 : -800  
~  
1,1          모 두
```


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);
```

A solid green shape that is a triangle pointing downwards, located on the left side of the slide.









2. Unit Test

Unit Test Plan

**Use
CuTest
Ver 1.5**

**For
Unit Test**

In Linux (Cygwin)

-  CuTest.c
-  CuTest.h
-  CuTestTest.c
-  index.html
-  license.txt
-  make-tests.sh
-  README.txt
-  style.css

Unit Test Plan

단순 **Data** 전달 **Process**나
Contol 전달 **Process**는 **Unit**
Test에서 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

```
~/soen
TestReset_Saleslist_00 Test Start
TestReset_Saleslist_01 Test Start
TestReset_Inventory_00 Test Start
TestReset_Inventory_01 Test Start
TestInventory_server_Merging_00 Test Start
TestInventory_server_Merging_01 Test Start
TestPrint_Receipt_00 Test Start
TestPrint_Receipt_01 Test Start
TestUpdate_Inventory_00 Test Start
TestUpdate_Inventory_01 Test Start
TestSaleslist_server_Merging_00 Test Start
TestSaleslist_server_Merging_01 Test Start
TestSaleslist_server_Merging_02 Test Start
TestUpdate_Sales_List_00 Test Start
TestUpdate_Sales_List_01 Test Start
TestUpdate_Cashier_Screen_00 Test Start
==Cashier Screen=====

상 품                수 량                가 격

=====

TestUpdate_Cashier_Screen_01 Test Start
TestUpdate_Customer_Screen_00 Test Start
==Customer Screen=====

판 매 날 짜   : 2017년  1월  1일  1시  1분

=====

TestUpdate_Customer_Screen_01 Test Start
.....

OK (19 tests)
```