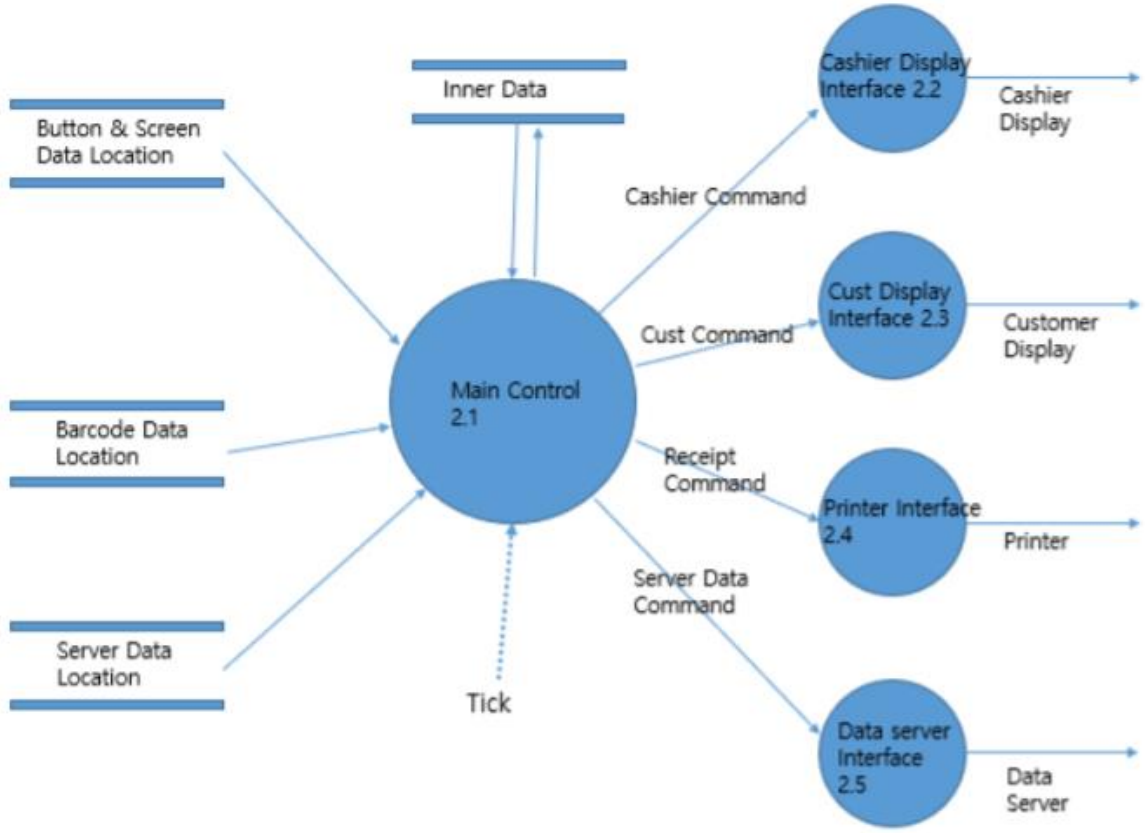
Team 1

20131272 나경수
20141276 서수빈
20141313 장진서
20161293 전다윤

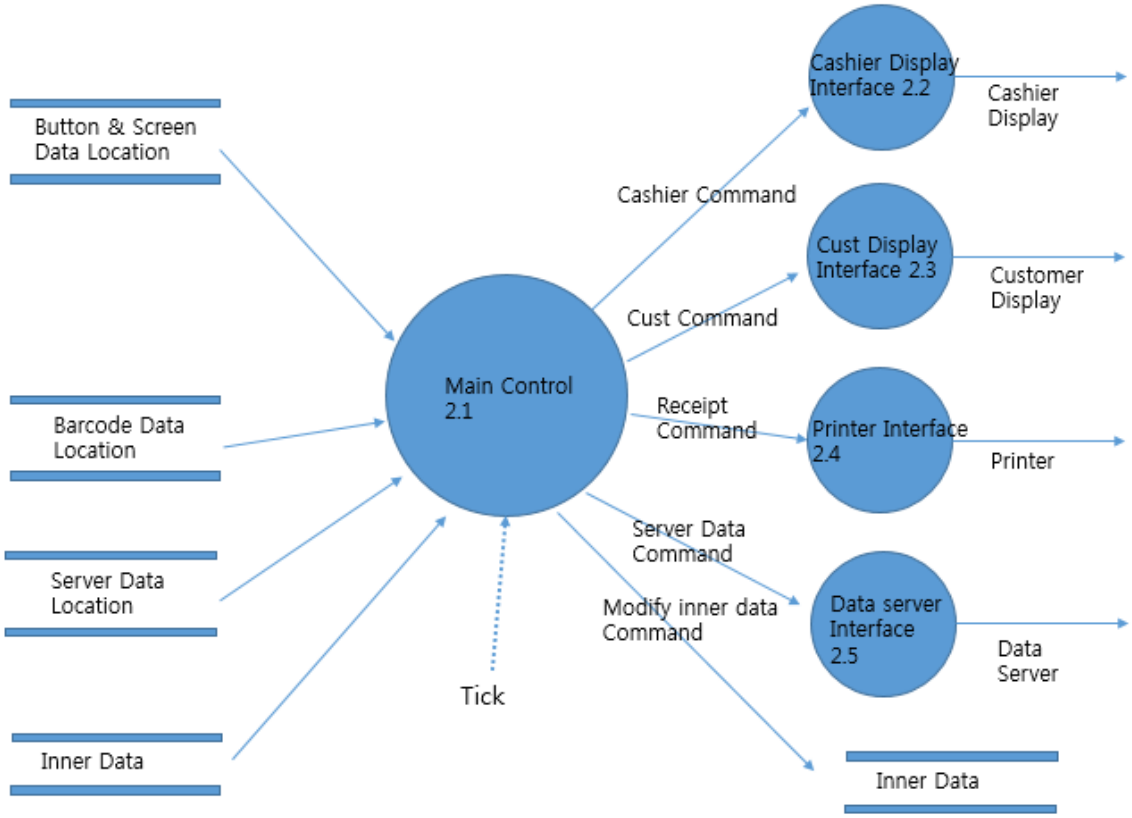
Contents

01 DFD Level 2 (ver2)



VOYAGE

01 DFD Level 2 (ver3)



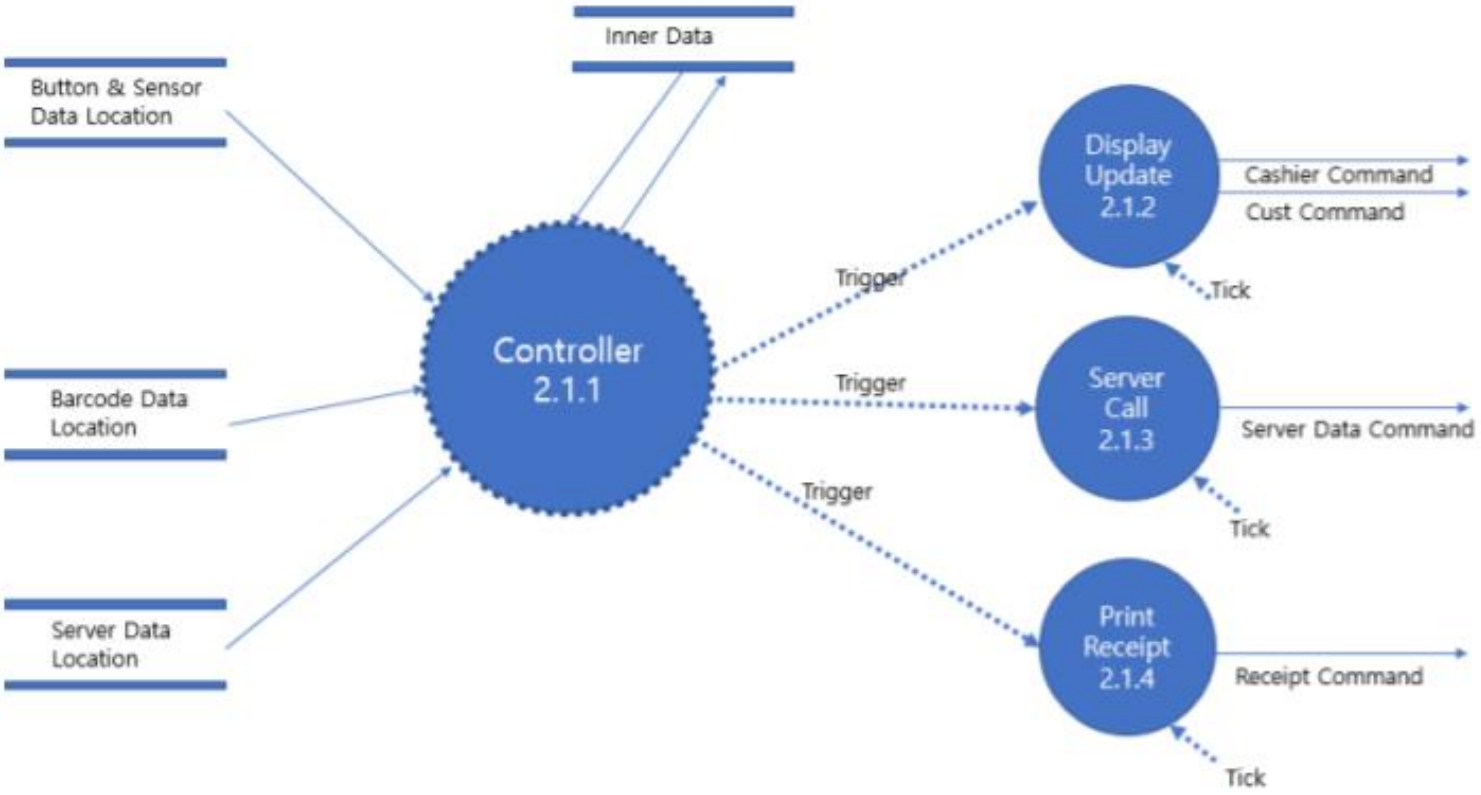
VOYAGE

01 Data Dictionary (DFD Level2)

Button & Screen Data Location	**NoG : Num of Goods/수량 변경 버튼을 입력받으면 True가 된다.**	**True/False**
	Home : 홈 버튼을 눌렀을 때 True가 된다.	True/False
	P : 전원 버튼을 눌렀을 때 True/False가 된다.	True/False
	FS : 판매완료 버튼을 눌렀을 때 True가 된다.	True/False
	InMoney : 구매자로부터 받은 금액에 대한 정보	Integer
	CS : Check Stock/재고확인 버튼을 눌렀을 때 True가 된다.	True/False
	Return : 환불 버튼을 누르면 True가 된다.	True/False
	YN : 환불수행여부를 예(1) 아니오(-1) 형태로 입력받는다.	Integer

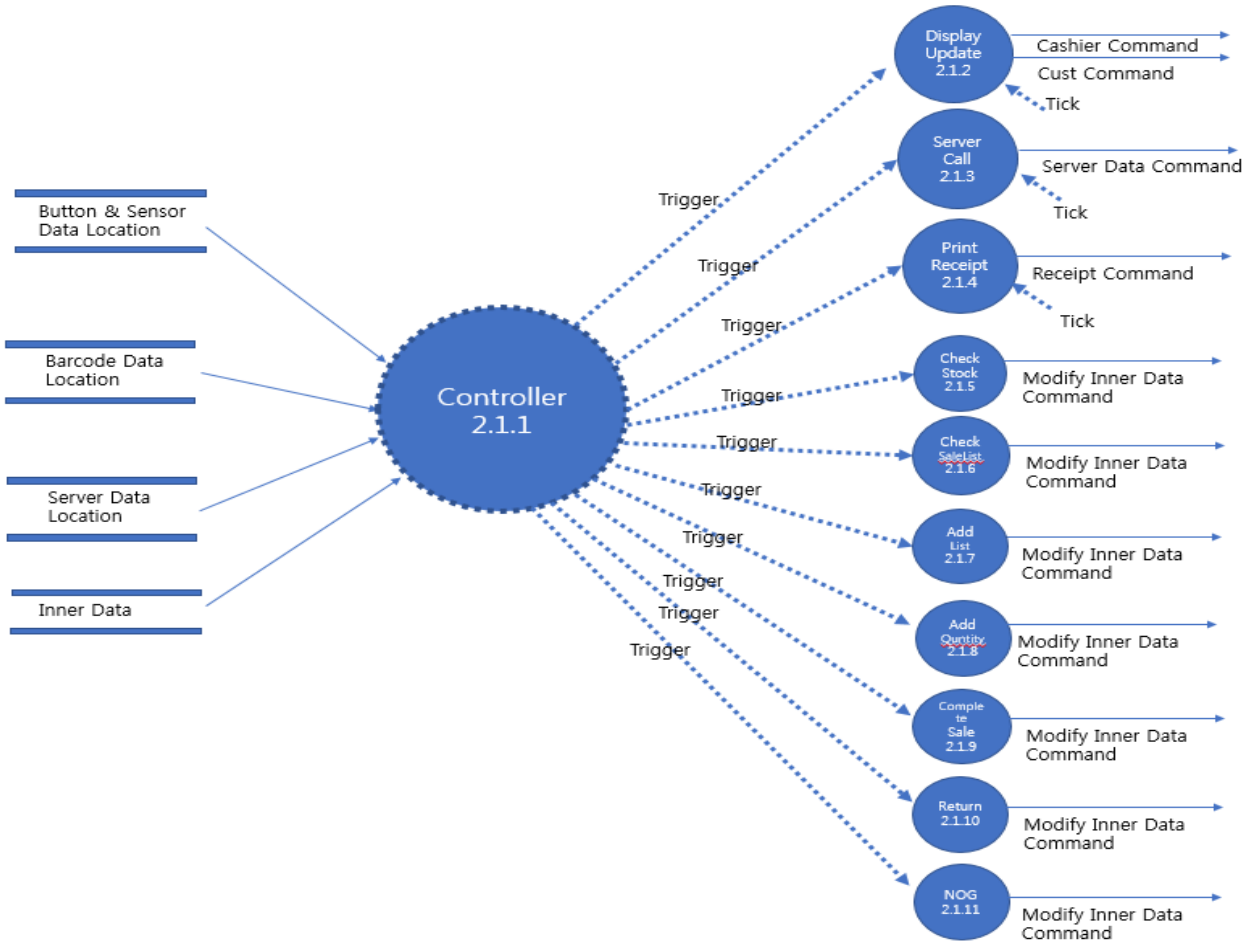
Inner Data	**DayChange : Time Controller로부터 일이 바뀔 때마다 True를 입력받는다.**	**True/False**
	IsExist : 해당 제품이 존재하는 지 여부를 알려준다.	True/False
	IsOnList : 해당 제품이 판매목록상에 있는지 여부	True/False
	Num : 구매 수량에 대한 정보	Integer
	Total :결제금액에 대한 정보	Integer
	St : 해당 제품의 재고 정보를 알려준다.	Integer

01 DFD Level 3 (ver2)



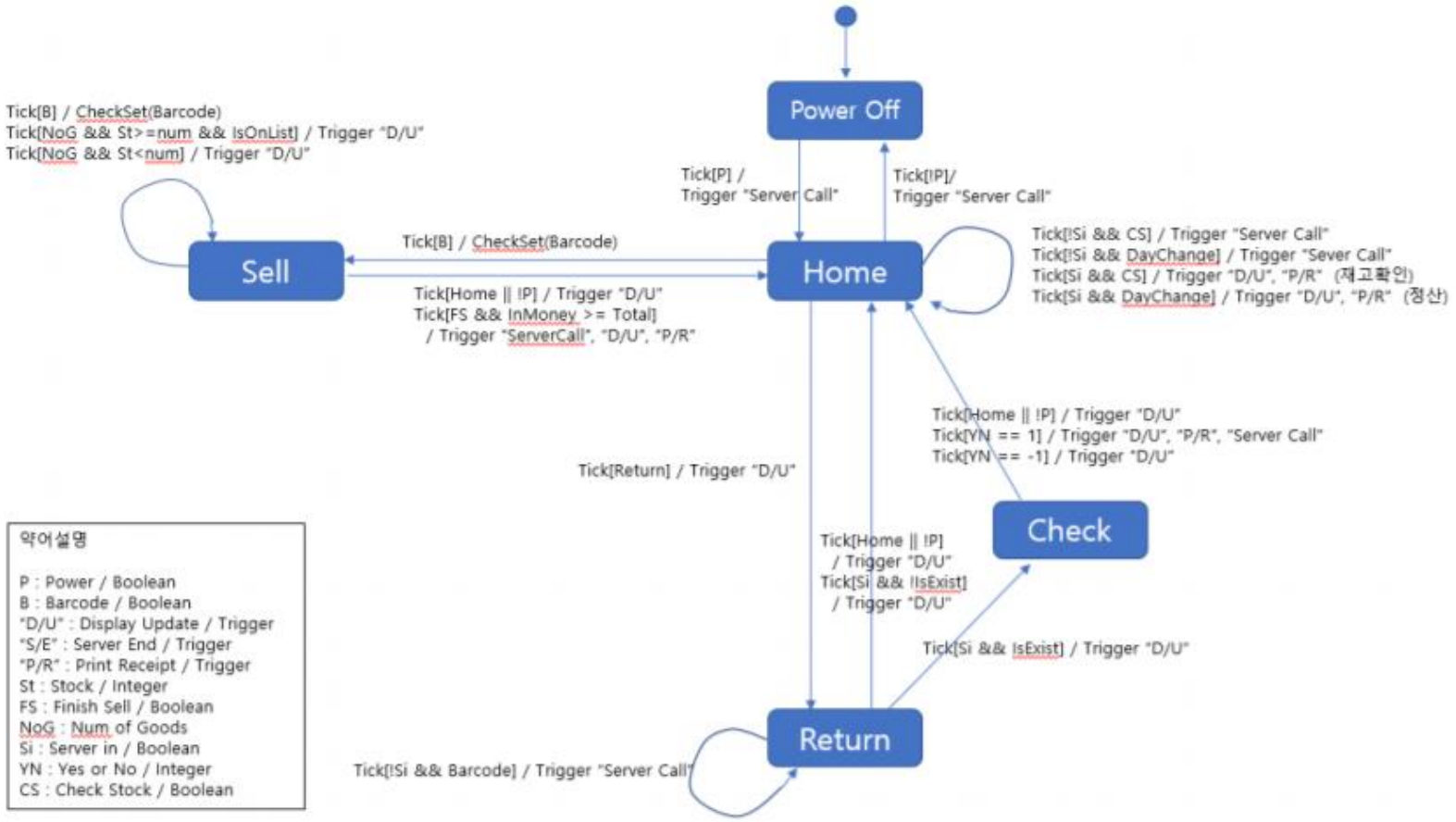
VOYAGE

01 DFD Level 3 (ver3)



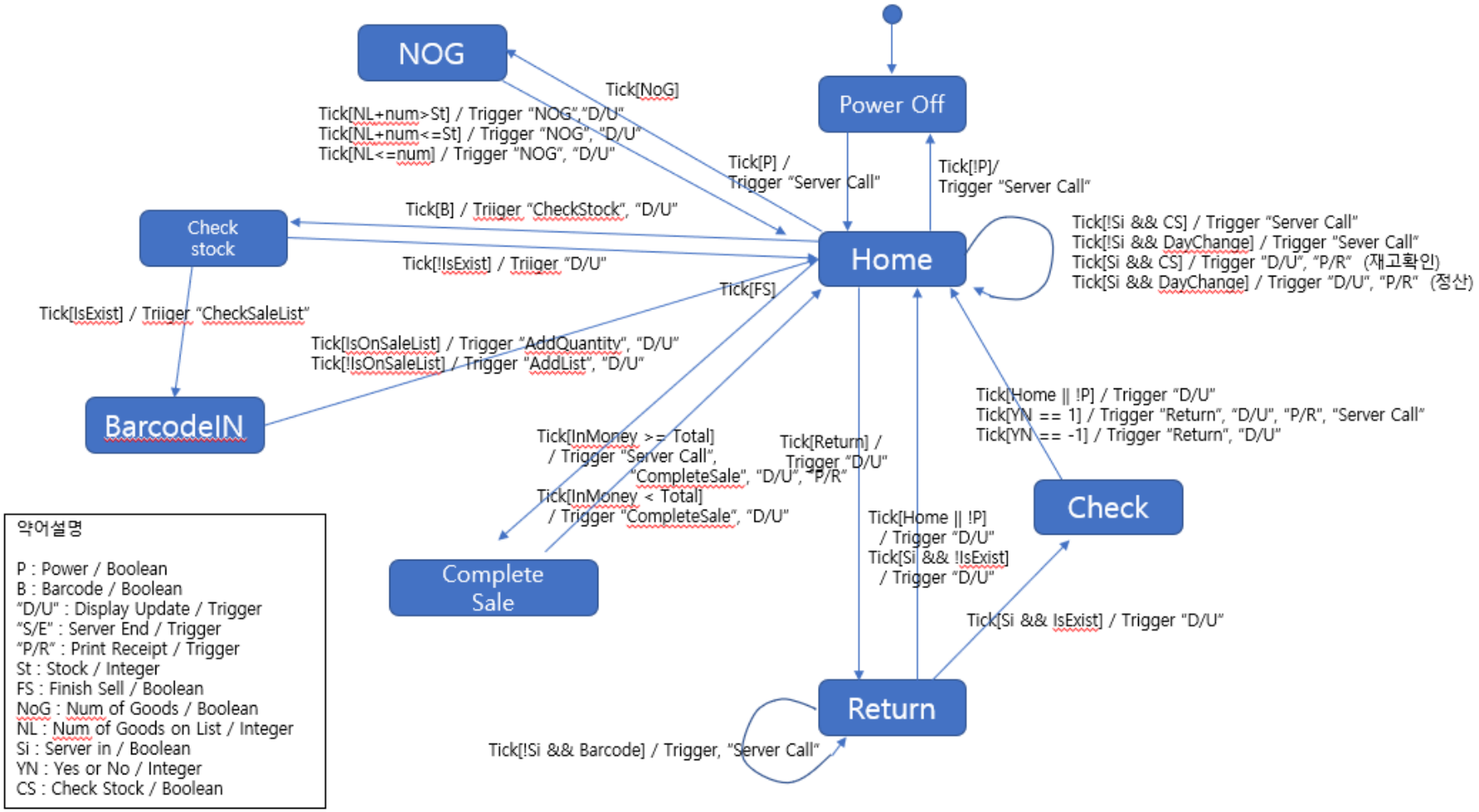
VOYAGE

01 State Chart Diagram (ver 1)



VOYAGE

01 State Chart Diagram(ver2)

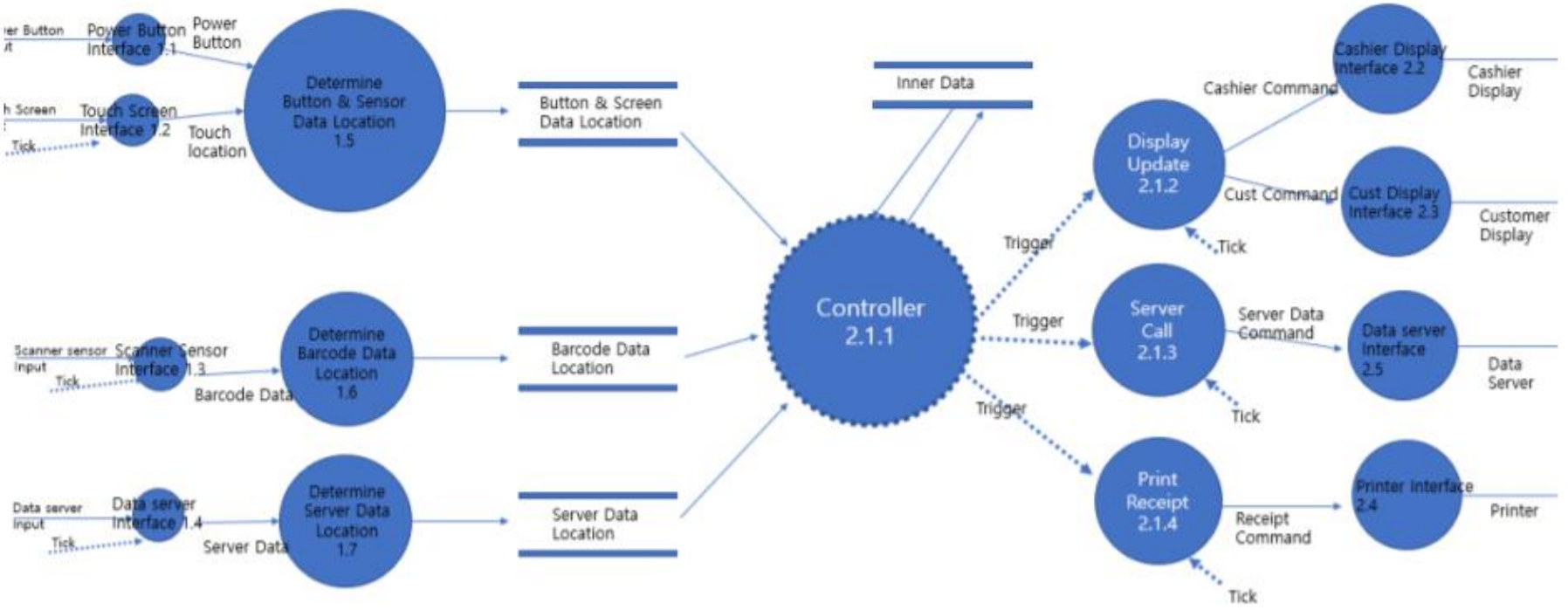


약어설명

P : Power / Boolean
B : Barcode / Boolean
"D/U" : Display Update / Trigger
"S/E" : Server End / Trigger
"P/R" : Print Receipt / Trigger
St : Stock / Integer
FS : Finish Sell / Boolean
NoG : Num of Goods / Boolean
NL : Num of Goods on List / Integer
Si : Server in / Boolean
YN : Yes or No / Integer
CS : Check Stock / Boolean

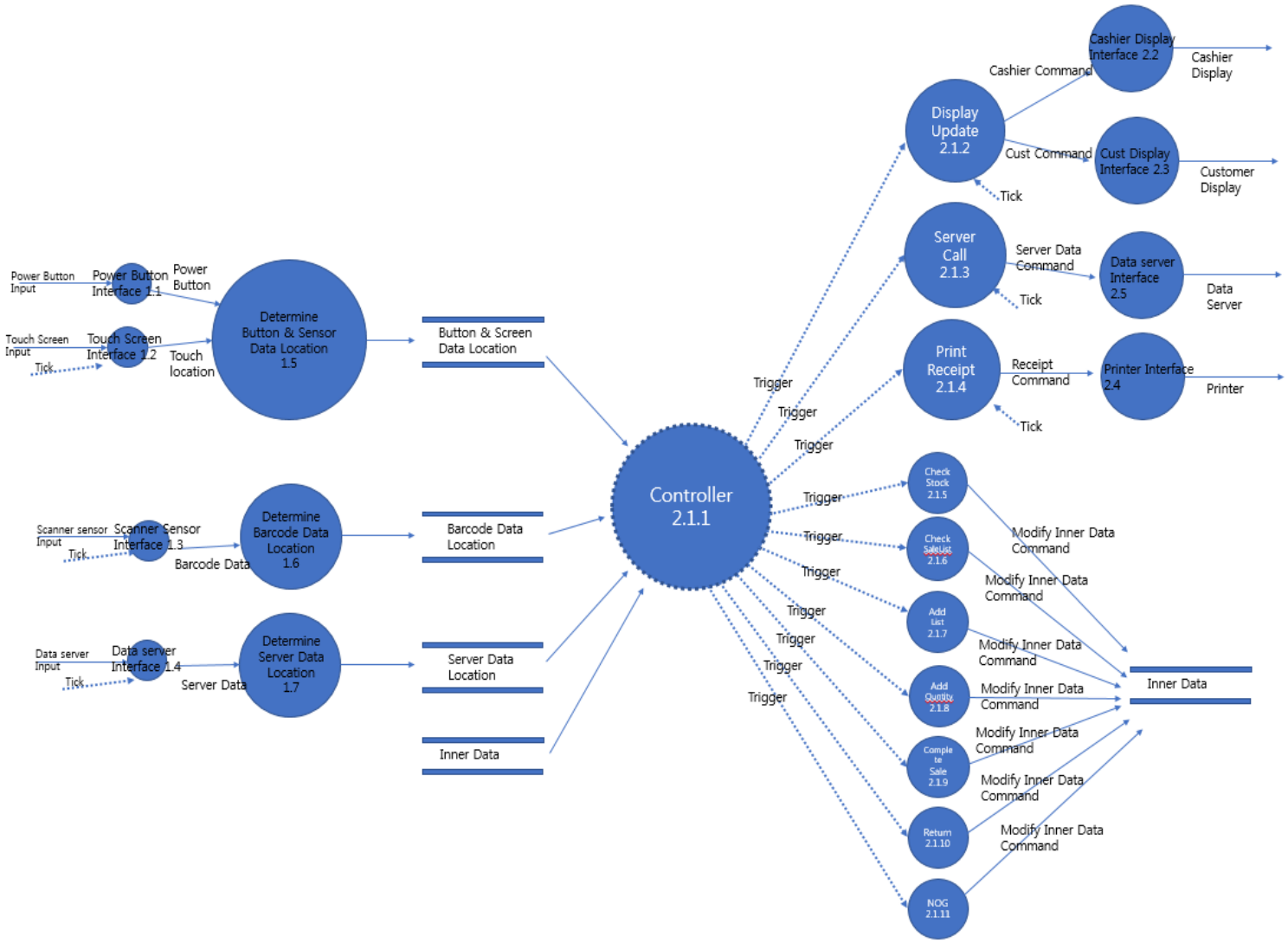
VOYAGE

01 Overall DFD(ver1)



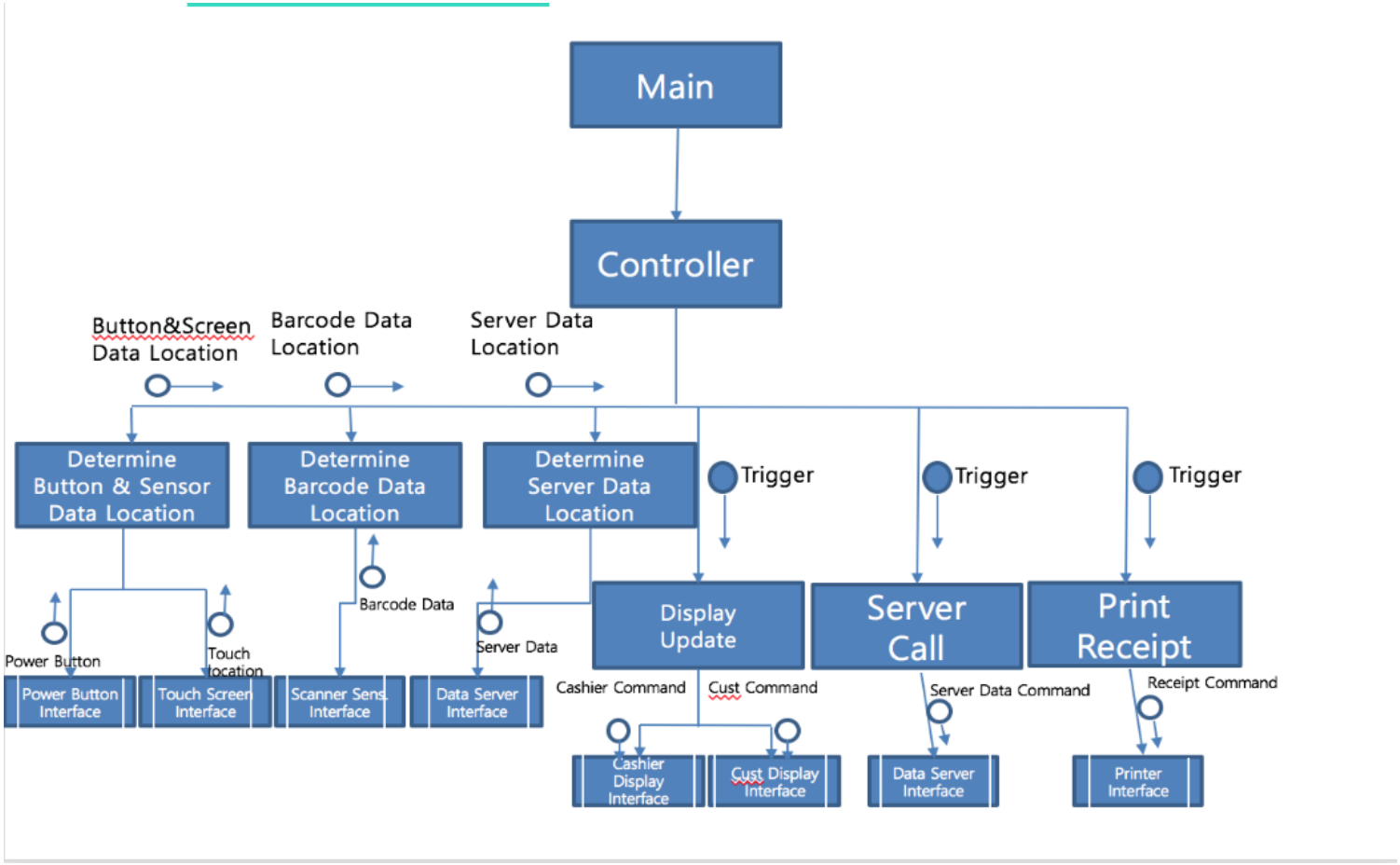
VOYAGE

01 Overall DFD(ver2)



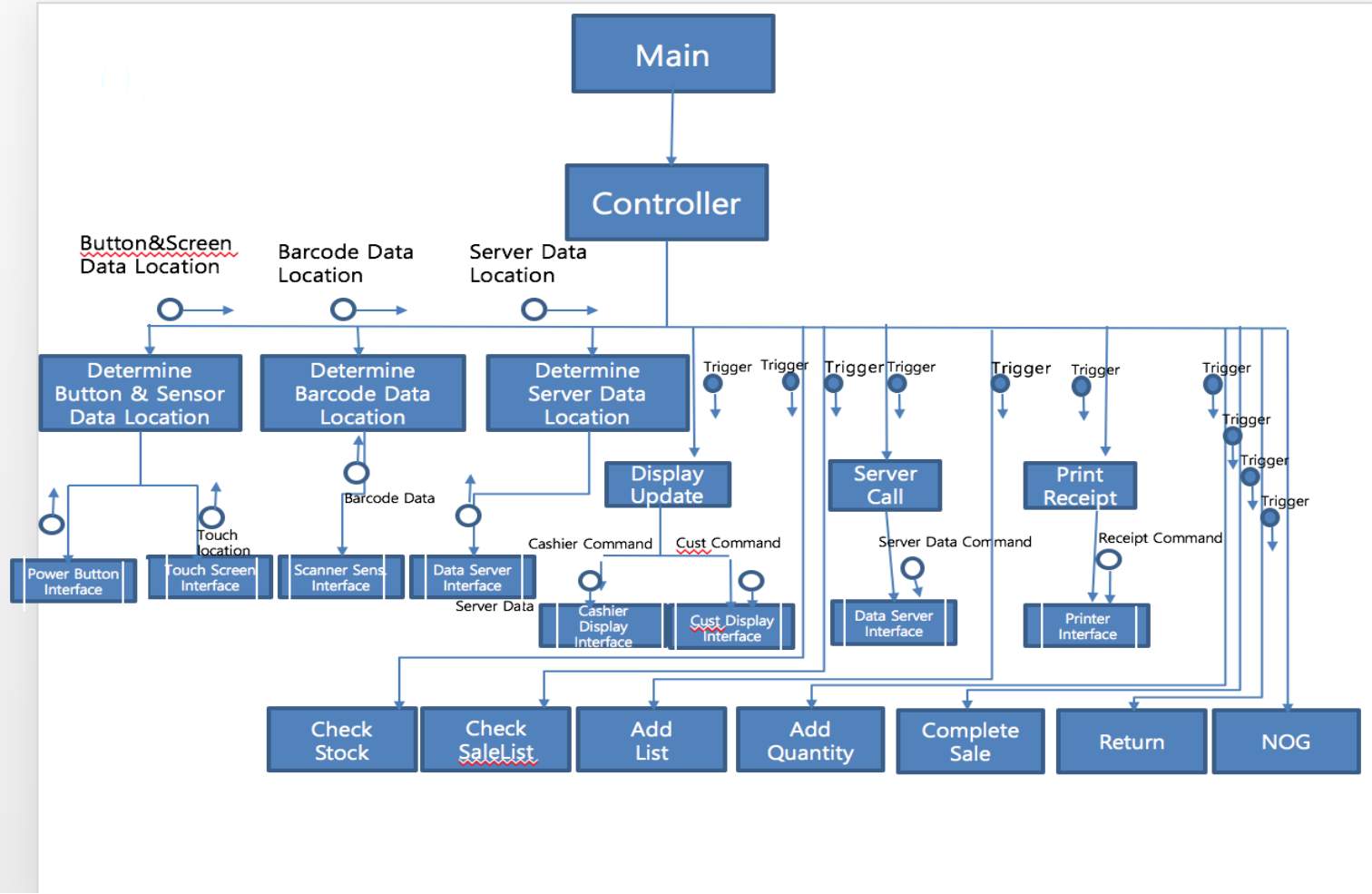
VOYAGE

01 Structured Charts (Basic) (ver 1)



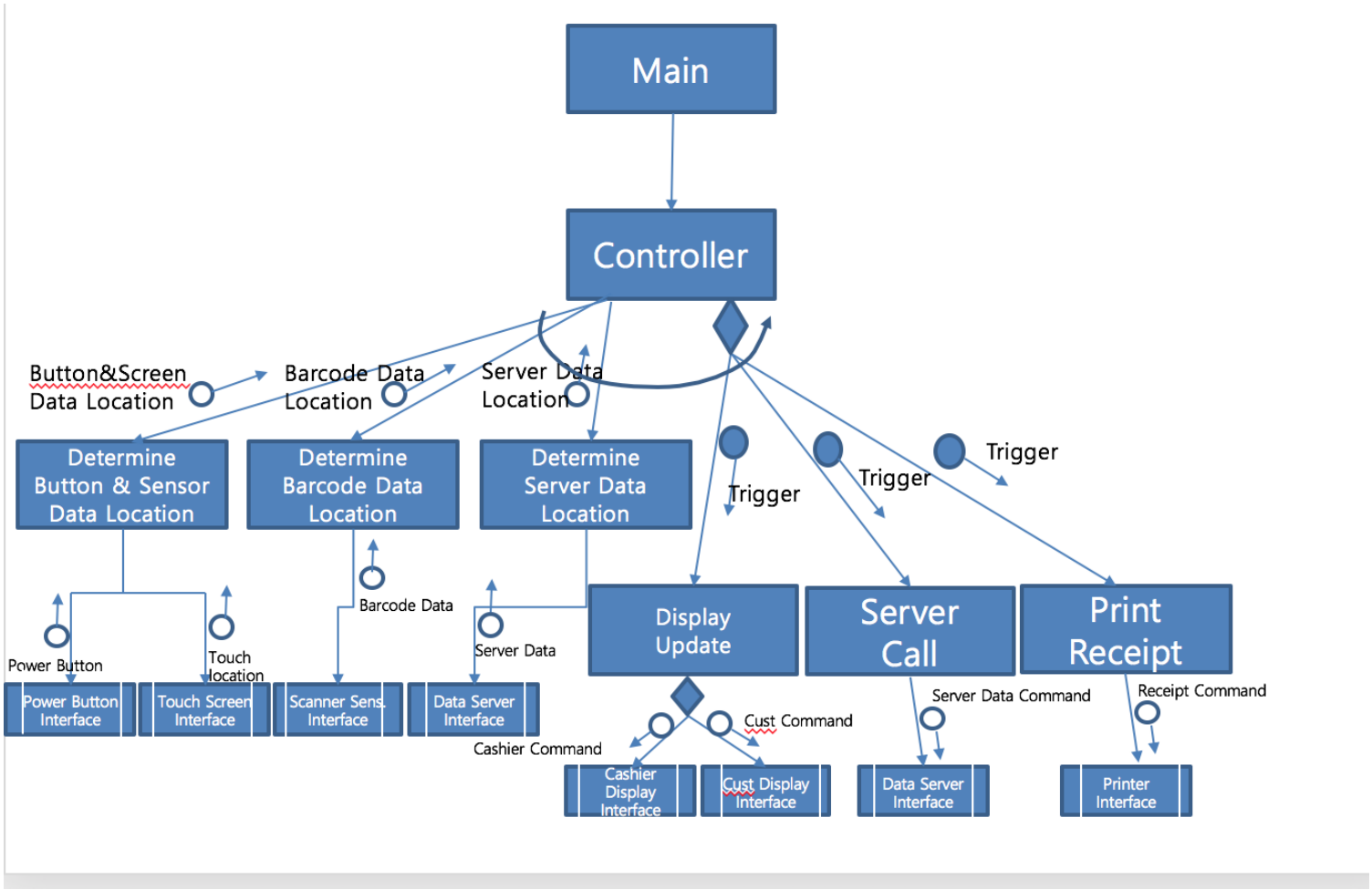
VOYAGE

01 Structured Charts (Basic) (ver 2)



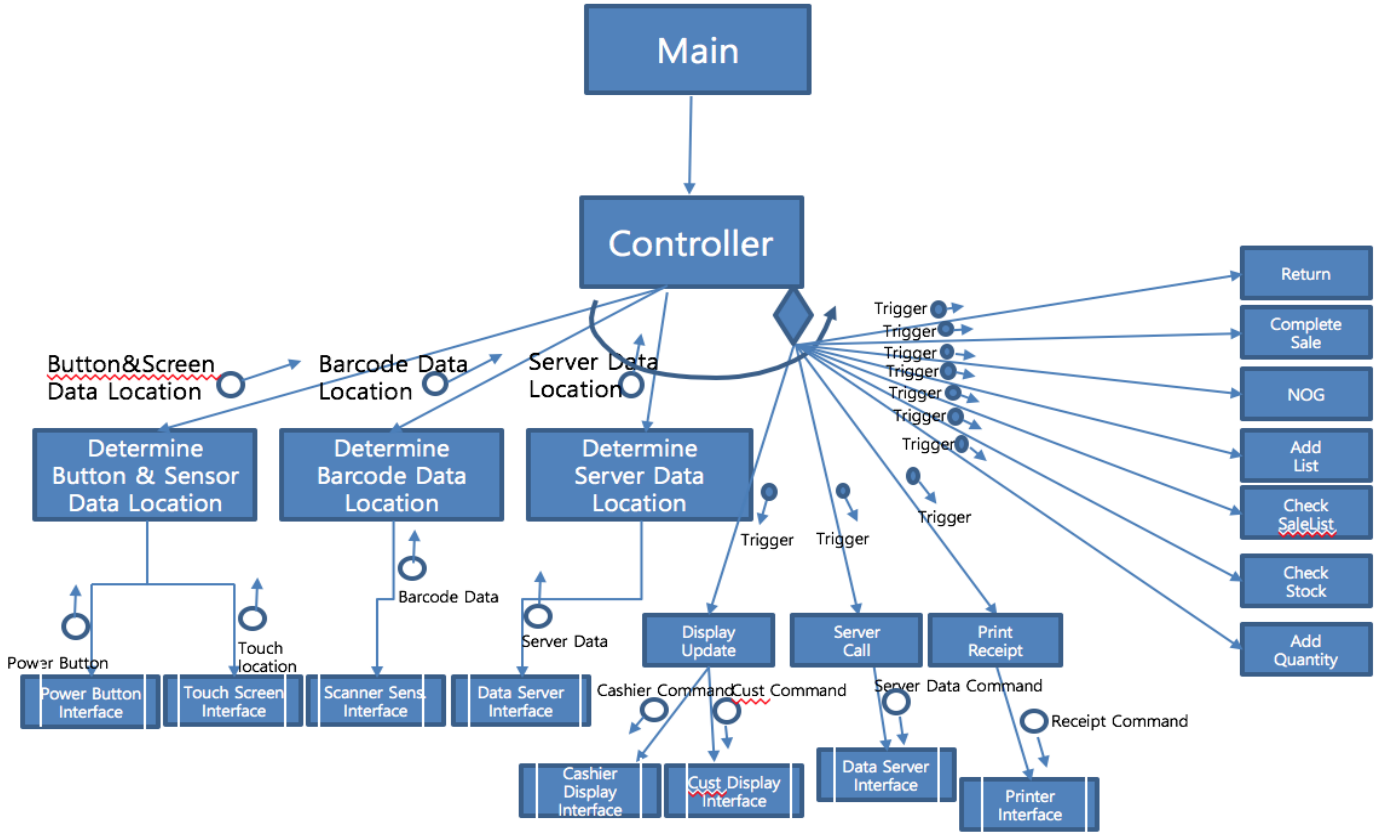
VOYAGE

01 Structured Charts (Advanced) (ver 1)



VOYAGE

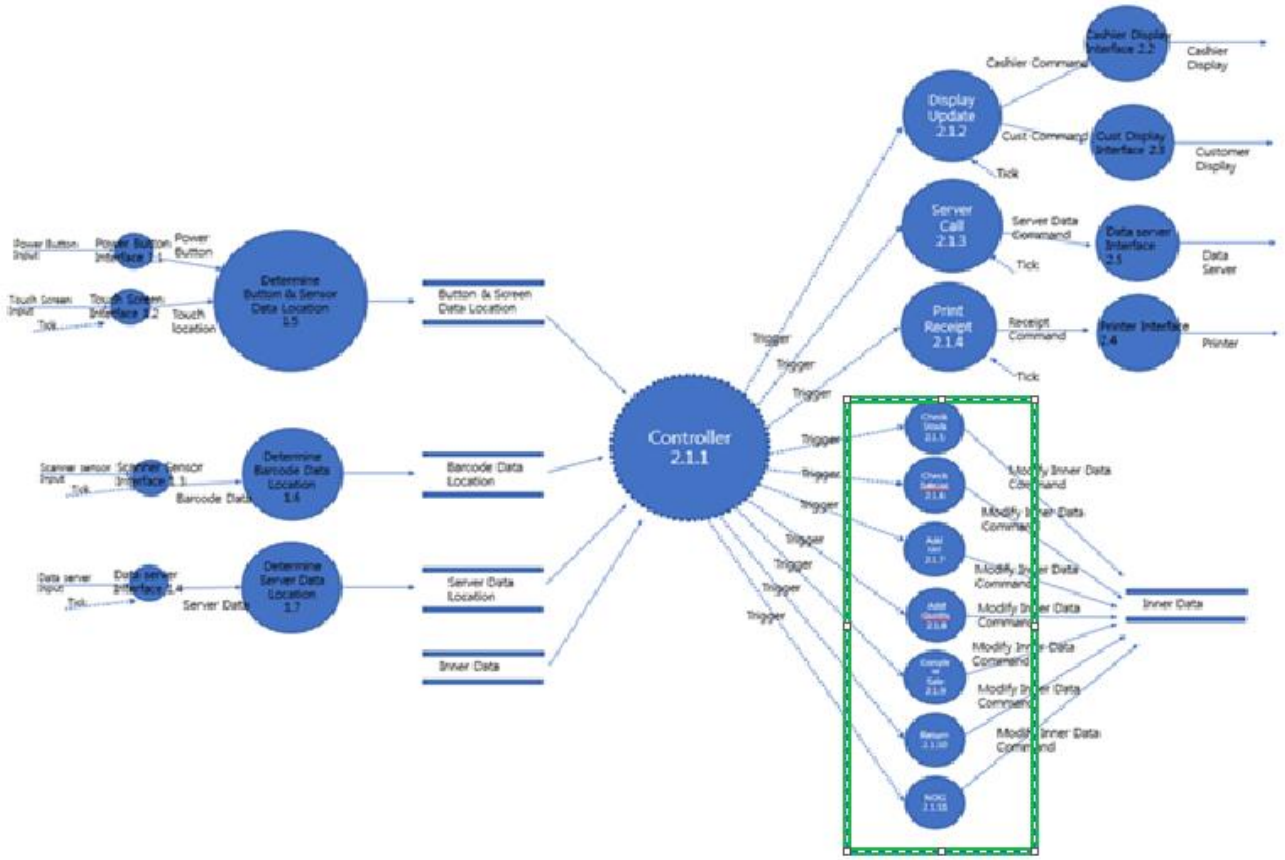
01 Structured Charts (Advanced) (ver 2)



VOYAGE

02 Features to be tested

Reference – Overall DFD



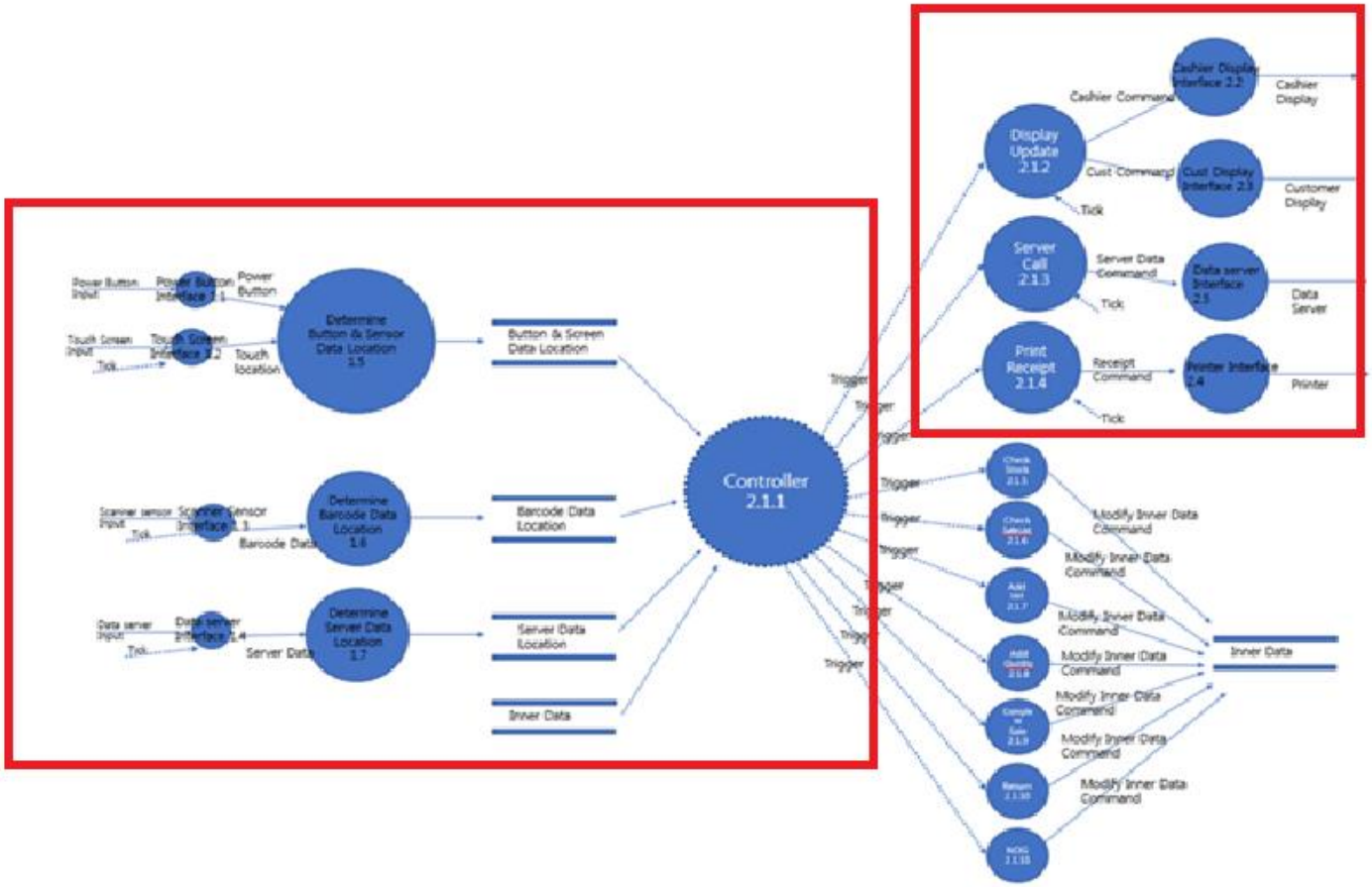
VOYAGE

02 Features to be tested

ID	Name	Description
2.1.5	Check Stock	남은 재고 수량을 check 하고 Inner Data를 바꾼다.
2.1.6	CheckSaleLIst	찍은 바코드가 현재 판매목록에 있는지 check하고 Inner Data를 바꾼다.
2.1.7	AddList	새로운 제품이 찍히면 현재 판매 목록에 추가시킨다. Inner Data에 판매 목록 바꾸기.
2.1.8	AddQuantity	현재 판매 목록의 제품 수량을 증감시키기 위해 Inner Data를 바꾼다.
2.1.9	CompleteSale	판매 완료를 하기위해 현재 판매목록을 지우고 재고 수량을 고친다.
2.1.10	Return	환불 기능을 하기 위해 Inner Data의 재고 수량을 바꾼다.
2.1.11	NOG	현재 판매목록의 제품 수량을 입력된 수 만큼 증감시키기 위해 Inner Data를 바꾼다.

02 Features not to be tested

Reference – Overall DFD



VOYAGE

02 Test Identification

Identifier	Feature	Valid/Invalid value
T1_POS_000_000	2.1.5 check stock	valid
T1_POS_000_001	2.1.5 check stock	valid
T1_POS_000_002	2.1.5 check stock	valid
T1_POS_000_003	2.1.5 check stock	invalid
T1_POS_001_000	2.1.8 AddQuantity	valid
T1_POS_001_001	2.1.8 AddQuantity	invalid
T1_POS_002_000	2.1.11 NOG	valid
T1_POS_002_001	2.1.11 NOG	invalid
T1_POS_003_000	2.1.9 Complete Sale	valid
T1_POS_003_001	2.1.9 Complete Sale	invalid
T1_POS_004_000	2.1.10 Return	valid
T1_POS_004_001	2.1.10 Return	invalid
T1_POS_005_000	2.1.6 CheckSaleList	valid
T1_POS_005_001	2.1.6 CheckSaleList	invalid

02 Test Case Specification Identifier

Test case Identifier	Input Specification (input // 주어진 환경)	Output Specification
T1_POS_000_000	stockList, "001", &a, name (재고struct,바코드,제품가격,제품명)' // stockList has Item("001") ,Item("010")	True
T1_POS_000_001	stockList, "010", &a, name // 위와 동일환경	True
T1_POS_000_002	stockList, "010", &a, name // 위와 동일환경	True
T1_POS_000_003	stockList, "011", &a, name // 위와 동일환경	False
T1_POS_001_000	saleList, price, "001" (현재판매목록, 상품가격, 바코드) // saleList has Item("001")*2 saleList has Item("010")*1	3
T1_POS_001_001	saleList, price, "010" // 위와 동일환경	2

02 Test Case Specification

Test case Identifier	Input Specification	Output Specification
T1_POS_002_000	1,2, saleList,stockList (current fruit 1, increment currnet fruit by 2) // stockList has Item("fruit")*20 saleList has Item("fruit")*1	3
T1_POS_002_001	1,1000, saleList,stockList // 위와 동일환경	-1
T1_POS_002_002	1,-500, saleList,stockList // 위와 동일환경	0
T1_POS_003_000	saleList, stockList // saleList has Item("001")*1 saleList has Item("001")*1 StockList has Item("001")*100 StockList has Item("010")*100	99
T1_POS_003_001	stockList, "010" // 위와 동일환경	99
T1_POS_004_000	stockList, "001" (product current quantity, product barcode) // StockList has Item("fruit")*100 ReturnList has Item("fruit")*3	103

02 Test Case Specification

Test case Identifier	Input Specification	Output Specification
T1_POS_005_000	saleList, "001" (현재판매목록struct, 바코드) SaleList has Item("001")*1 // saleList has Item("fruit")*1	1
T1_POS_005_001	saleList, "002"	-1
T1_POS_006_000	"001" (추가한 상품 바코드"001") // productList has Item("001")*2	True"
T1_POS_006_001	"000" // 위와 동일환경	False

03 Unit Test

```
START_TESTS()

START_TEST("checkStock Test")
    StockList *stockList = getStockList("stock/20170927_product.txt", 27);
    int a = 500;
    char *name;
    ASSERT(checkStockList(stockList, "001", &a, name));
    ASSERT(checkStockList(stockList, "010", &a, name));
    ASSERT(checkStockList(stockList, "010", &a, name));
    ASSERT_NOT_EQUALS(checkStockList(stockList, "011", &a, name),1);
END_TEST()

START_TEST("IsOnSaleList Test")
    SaleList *saleList = (SaleList*)malloc(sizeof(SaleList));
    initSaleList(saleList);
    ProductList *pl = saleList->productList;
    Product *newProduct;

    newProduct = makeProduct("001", 1,100, "fruit");
    addProduct(pl, newProduct);

    ASSERT_EQUALS(isOnSaleList(saleList, "001"),1);
    ASSERT_NOT_EQUALS(isOnSaleList(saleList, "002"), 1);
END_TEST()
```

03 Unit Test

```
START_TEST("sale Test")
    SaleList *saleList = (SaleList*)malloc(sizeof(SaleList));
    initSaleList(saleList);
    ProductList *pl = saleList->productList;
    Product *newProduct;

    newProduct = makeProduct("001", 1,100, "fruit");
    addProduct(pl, newProduct);
    newProduct = makeProduct("010", 1,150, "beer");
    addProduct(pl, newProduct);

    StockList *stockList = getStockList("stock/20170927_product.txt", 27);
    StockList *beforeList = getStockList("stock/20170927_product.txt", 27);


    calcControl(saleList, stockList);

    int expect = findBarcode(beforeList, "001")->quantity;
    int expect2 = findBarcode(beforeList, "010")->quantity;

    ASSERT_EQUALS(findBarcode(stockList, "001")->quantity, expect-1);
    ASSERT_EQUALS(findBarcode(stockList, "010")->quantity, expect2-1);
END_TEST()
```

03 Unit Test

```
START_TEST("refund Test")

StockList* refundList = (StockList*)malloc(sizeof(StockList));
initStockList(refundList);


Stock *newStock = makeStock("fruit", "", 3, 0); // fruit 3
addStock(refundList, newStock);
newStock = makeStock("beer", "", 2, 0);
addStock(refundList, newStock);


StockList *stockList = getStockList("stock/20170927_product.txt", 27);
StockList *beforeList = getStockList("stock/20170927_product.txt", 27);



fixStockFile(refundList, stockList, "stock/20170927_product.txt");

int expect = findBarcode(beforeList, "001")->quantity;
int expect2 = findBarcode(beforeList, "010")->quantity;


ASSERT_EQUALS(findBarcode(stockList, "001")->quantity, expect+3); //fruit 3
ASSERT_EQUALS(findBarcode(stockList, "010")->quantity, expect2+2);
END_TEST()
```

04 Testing result

```
jins@DESKTOP-CS1DLTU ~/seproject2
$ ./a
> checkStock Test...
> IsOnSaleList Test...
> AddQuantiy Test...
> AddList Test...
> sale Test...
> refund Test...
> NOG Test...

--- Results ---
Tests run:    7
Passes:      17
Failures:     0

jins@DESKTOP-CS1DLTU ~/seproject2
$
```

04 Testing result

Identifier	Output Specification	Result	Pass/Non Pass
T1_POS_000_000	True	True	Pass
T1_POS_000_001	True	True	Pass
T1_POS_000_002	True	True	Pass
T1_POS_000_003	False	False	Pass
T1_POS_001_000	3	3	Pass
T1_POS_001_001	2	2	Pass
T1_POS_002_000	3	3	Pass
T1_POS_002_001	-1	-1	Pass
T1_POS_002_002	0	0	Pass
T1_POS_003_000	99	99	Pass
T1_POS_003_001	99	99	Pass
T1_POS_004_000	103	103	Pass
T1_POS_004_001	102	102	Pass
T1_POS_005_000	1	1	Pass

Q & A