[MDA-EFSM model for gas-pump]

Malhar Patwari- A20410420

CS586 Software System Architecture

Date: **4/25/2018**

PART 1: MDA-EFSM model for the GasPump components

(I) A list of meta events for the MDA-EFSM

Activate()

Start()

PayType(int a) //credit: a=1; cash: a=2; debit: a=3

Reject()

Cancel()

Approved()

StartPump()

Pump()

StopPump()

SelectGas(int i) // Regular: i=1; Super: i=2; Premium: i=3; Diesel: i=4

Receipt()

NoReceipt()

CorrectPin()

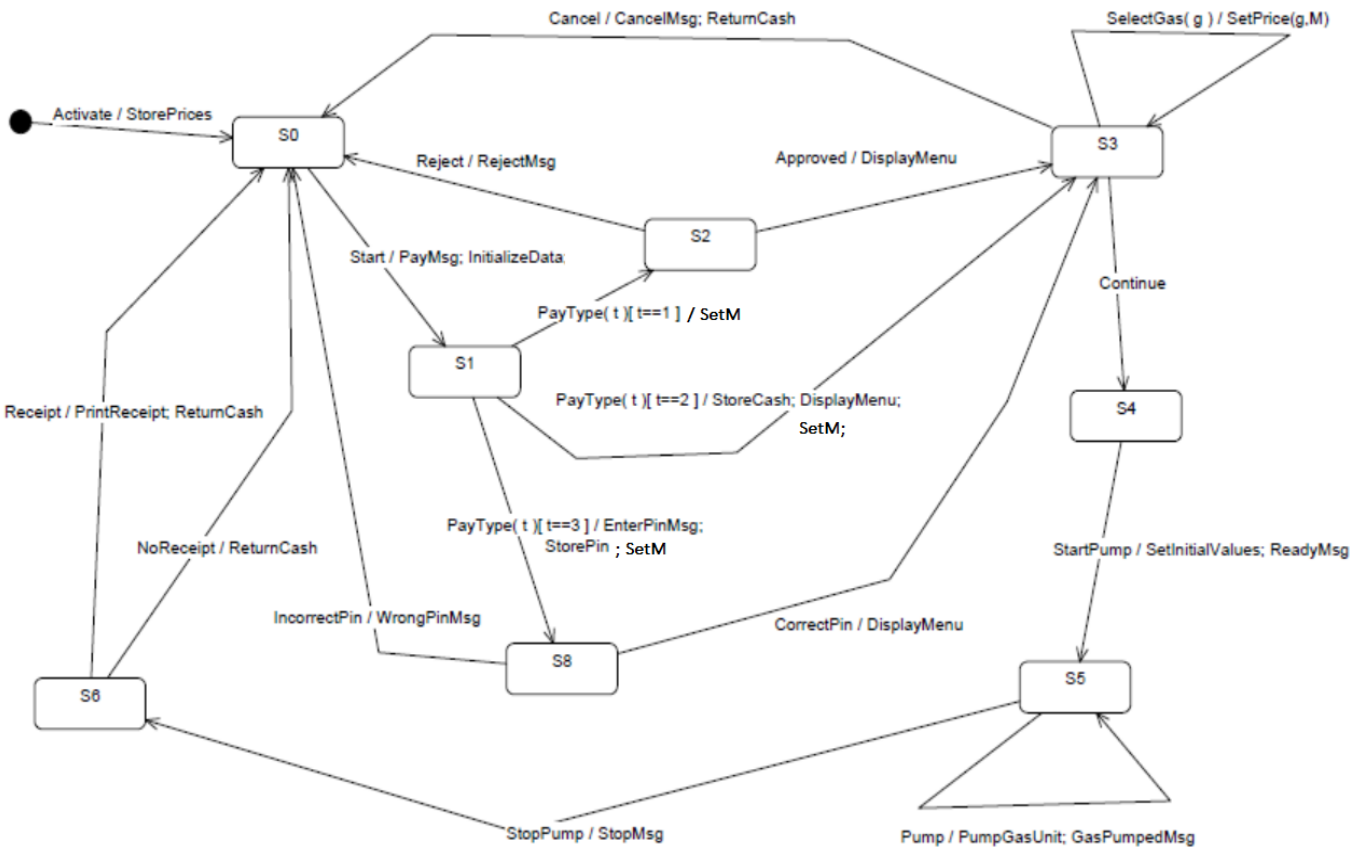
IncorrectPin()

Continue()

(II) A list of meta actions for the MDA-EFSM

StorePrices	// stores price(s) for the gas from the temporary data store
PayMsg	// displays a type of payment method
StoreCash	// stores cash from the temporary data store
DisplayMenu	// display a menu with a list of selections
RejectMsg	// displays credit card not approved message
SetPrice(int g)	// set the price for the gas identified by g identifier as in
SetM	//set M value according to type of payment
ReadyMsg	// displays the ready for pumping message
SetInitialValues	// set G (or L) and total to 0;
PumpGasUnit	// disposes unit of gas and counts # of units disposed
GasPumpedMsg	// displays the amount of disposed gas
StopMsg	// stop pump message
PrintReceipt	// print a receipt
CancelMsg	// displays a cancellation message
ReturnCash	// returns the remaining cash
WrongPinMsg	// displays incorrect pin message
StorePin	// stores the pin from the temporary data store
EnterPinMsg	// displays a message to enter pin
InitializeData	// set the value of price and cash to 0

(iii) A state diagram of the MDA-EFSM



MDA-EFSM for Gas Pumps

(iv) Pseudo-code of all operations of Input Processors of GasPump-1 and GasPump-2

GasPump1:

```
Activate(float a, float b) {
        if ((a>0)&&(b>0)) {
                d->temp_a=a;
                d->temp_b=b;
                m->Activate()
        }
}

Start() {
        m->Start();
}

PayCredit() {
        d->temp_m = 1;
        m->PayType(1);
}

Reject() {
        m->Reject();
}

PayDebit(string p) {
        d->temp_m =  2;
        d->temp_p=p;
        m->PayType(3);
}

Pin(string x) {
        if (d->pin==x){
                m->CorrectPin();
        }
        else {
                m->InCorrectPin();
        }
}

Cancel() {
        m->Cancel();
}
```

```
Approved() {
        m->Approved();
}

Diesel() {
        m->SelectGas(4)
}

Regular() {
        m->SelectGas(1)
}

StartPump() {
        if (d->price>0) {
                m->Continue();
                m->StartPump();
        }
}

PumpGallon() {
        m->Pump();
}

StopPump() {
        m->StopPump();
        m->Receipt();
}

FullTank() {
        m->StopPump();
        m->Receipt();
}
```

Notice:

m: is a pointer to the MDA object

d: is a pointer to the Data Store object

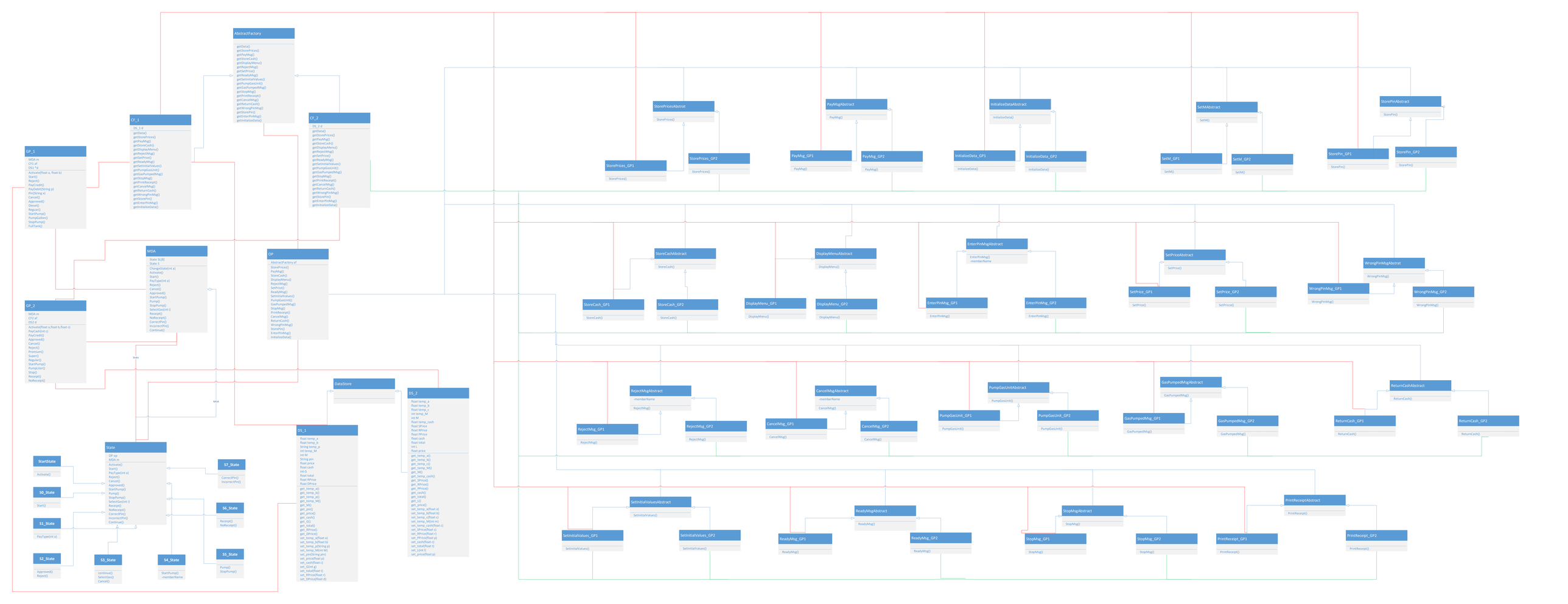
GasPump2:

```
Activate(float a, float b, float c) {
        if ((a>0)&&(b>0)&&(c>0)) {
                d->temp_a=a;
                d->temp_b=b;
                d->temp_c=c
                m->Activate()
        }
        else{
                print("Not a valid price");
        }
}

PayCash(float c) {
        if (c>0) {
                d->temp_cash=c;
                d->temp_m = 0;
                m->start();
                m->PayType(2);
        }
}

PayCredit() {
        d->temp_m = 1;
        m->start();
        m->PayType(1);
}

Reject() {
        m->Reject();
}

Approved() {
        m-> Approved();
}

Cancel() {
        m->Cancel();
}

Super() {
        m->SelectGas(2);
        m->Continue();
}
```

```
Premium() {
        m->SelectGas(3);
        m->Continue();
}

Regular() {
        m->SelectGas(1);
        m->Continue();
}

StartPump() {
        m->StartPump();
}

PumpLiter() {
        if (d->cash>0)&&(d->cash < d->price*(d->L+1))
                m->StopPump();
        }
        Else{
                m->Pump();
        }
}

Stop() {
        m->StopPump();
}

Receipt() {
        m->Receipt();
}

NoReceipt() {
        m->NoReceipt();
}
```

Notice:
cash: contains the value of cash deposited
price: contains the price of the selected gas
L: contains the number of liters already pumped
cash , L, price are in the data store
m: is a pointer to the MDA object
d: is a pointer to the Data Store object

CLASS DIAGRAM


AbstractFactory
getData()
getStorePrices()
getPayMsg()
getStoreCash()
getDisplayMenu()
getRejectMsg()
getSetPrice()
getReadyMsg()
getSetInitialValues()
getPumpGasUnit()
getGasPumpedMsg()
getStopMsg()
getPrintReceipt()
getCancelMsg()
getReturnCash()
getWrongPinMsg()
getStorePins()
getEnterPinMsg()
getInitializeData()
CF_1
DS_1 d
getData()
getStorePrices()
getPayMsg()
getStoreCash()
getDisplayMenu()
getRejectMsg()
getSetPrice()
getReadyMsg()
getSetInitialValues()
getPumpGasUnit()
getGasPumpedMsg()
getStopMsg()
getPrintReceipt()
getCancelMsg()
getReturnCash()
getWrongPinMsg()
getStorePins()
getEnterPinMsg()
getInitializeData()
CF_2
DS_2 d
getData()
getStorePrices()
getPayMsg()
getStoreCash()
getDisplayMenu()
getRejectMsg()
getSetPrice()
getReadyMsg()
getSetInitialValues()
getPumpGasUnit()
getGasPumpedMsg()
getStopMsg()
getPrintReceipt()
getCancelMsg()
getReturnCash()
getWrongPinMsg()
getStorePins()
getEnterPinMsg()
getInitializeData()
GP_1
MDA m
CF1 af
DS1 *d
Activate(float a, float b)
Start()
Reject()
PayCredit()
PayDebit(String p)
Pin(String x)
Cancel()
Approved()
Diesel()
Regular()
StartPump()
PumpStation()
StopPump()
FullTank()
GP_2
MDA m
CF2 af
DS2 d
Activate(float a,float b,float c)
PayCash(int c)
PayCredit()
Approved()
Cancel()
Reject()
Premium()
Super()
Regular()
StartPump()
PumpLiter()
Stop()
Receipt()
NoReceipt()
MDA
State S1[6]
State S
ChangeState(int a)
Activate()
Start()
PayType(int a)
PayMsg()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int i)
Receipt()
NoReceipt()
CorrectPin()
IncorrectPin()
Continue()
OP
AbstractFactory af
StorePrices()
PayMsg()
StoreCash()
DisplayMenu()
RejectMsg()
SetPrice()
ReadyMsg()
SetInitialValues()
PumpGasUnit()
GasPumpedMsg()
StopMsg()
PrintReceipt()
CancelMsg()
ReturnCash()
WrongPinMsg()
StorePins()
EnterPinMsg()
InitializeData()
DataStore
DS_2
float temp_a
float temp_b
float temp_c
int temp_M
int M
float temp_cash
float $Price
float RPrice
float PPrice
float cash
float total
int L
float price
get_temp_a()
get_temp_b()
get_temp_c()
get_temp_M()
get_M()
get_cash()
get_total()
get_$Price()
get_RPrice()
get_PPrice()
get_cash()
get_total()
get_L()
get_price()
set_temp_a(float a)
set_temp_b(float b)
set_temp_c(float c)
set_temp_M(int m)
set_temp_cash(float c)
set_$Price(float s)
set_RPrice(float r)
set_PPrice(float p)
set_cash(float c)
set_total(float t)
set_L(int l)
set_price(float p)
DS_1
float temp_a
float temp_b
String temp_p
int temp_M
int M
String pin
float price
float cash
int G
float total
float RPrice
float DPrice
get_temp_a()
get_temp_b()
get_temp_p()
get_temp_M()
get_M()
get_pin()
get_price()
get_cash()
get_G()
get_total()
get_RPrice()
get_DPrice()
set_temp_a(float a)
set_temp_b(float b)
set_temp_p(String p)
set_temp_M(int M)
set_pin(String pin)
set_price(float p)
set_cash(float c)
set_G(int g)
set_total(float t)
set_RPrice(float r)
set_DPrice(float d)
State
OP op
MDA m
Activate()
Start()
PayType(int a)
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int i)
Receipt()
NoReceipt()
CorrectPin()
IncorrectPin()
Continue()
StartState
Activate()
S0_State
Start()
S1_State
PayType(int a)
S2_State
Approved()
Reject()
S3_State
continue()
SelectGas()
Cancel()
S4_State
StartPump()
-memberName
S5_State
Pump()
StopPump()
S6_State
Receipt()
NoReceipt()
S7_State
CorrectPin()
IncorrectPin()
StorePricesAbstract
StorePrices()
StorePrices_GP1
StorePrices()
StorePrices_GP2
StorePrices()
PayMsgAbstract
PayMsg()
PayMsg_GP1
PayMsg()
PayMsg_GP2
PayMsg()
InitializeDataAbstract
InitializeData()
InitializeData_GP1
InitializeData()
InitializeData_GP2
InitializeData()
SetMAbstract
SetM()
SetM_GP1
SetM()
SetM_GP2
SetM()
StorePinAbstract
StorePins()
StorePin_GP1
StorePins()
StorePin_GP2
StorePins()
StoreCashAbstract
StoreCash()
StoreCash_GP1
StoreCash()
StoreCash_GP2
StoreCash()
DisplayMenuAbstract
DisplayMenu()
DisplayMenu_GP1
DisplayMenu()
DisplayMenu_GP2
DisplayMenu()
EnterPinMsgAbstract
EnterPinMsg()
-memberName
EnterPinMsg_GP1
EnterPinMsg()
EnterPinMsg_GP2
EnterPinMsg()
SetPriceAbstract
SetPrice()
SetPrice_GP1
SetPrice()
SetPrice_GP2
SetPrice()
WrongPinMsgAbstract
WrongPinMsg()
WrongPinMsg_GP1
WrongPinMsg()
WrongPinMsg_GP2
WrongPinMsg()
RejectMsgAbstract
-memberName
RejectMsg()
RejectMsg_GP1
RejectMsg()
RejectMsg_GP2
RejectMsg()
CancelMsgAbstract
-memberName
CancelMsg()
CancelMsg_GP1
CancelMsg()
CancelMsg_GP2
CancelMsg()
PumpGasUnitAbstract
PumpGasUnit()
PumpGasUnit_GP1
PumpGasUnit()
PumpGasUnit_GP2
PumpGasUnit()
GasPumpedMsgAbstract
GasPumpedMsg()
GasPumpedMsg_GP1
GasPumpedMsg()
GasPumpedMsg_GP2
GasPumpedMsg()
ReturnCashAbstract
ReturnCash()
ReturnCash_GP1
ReturnCash()
ReturnCash_GP2
ReturnCash()
SetInitialValuesAbstract
SetInitialValues()
SetInitialValues_GP1
SetInitialValues()
SetInitialValues_GP2
SetInitialValues()
ReadyMsgAbstract
ReadyMsg()
ReadyMsg_GP1
ReadyMsg()
ReadyMsg_GP2
ReadyMsg()
StopMsgAbstract
StopMsg()
StopMsg_GP1
StopMsg()
StopMsg_GP2
StopMsg()
PrintReceiptAbstract
PrintReceipt()
PrintReceipt_GP1
PrintReceipt()
PrintReceipt_GP2
PrintReceipt()

PART 2: Responsibilities of Classes and Operations:

Package Main:

(1) Class GP_1:

Responsibility- This Class is an implementation of Gas pump 1 with its specific types of input operations and supports functionalities specific to Gas Pump1.

Supported Operations –

MDA m;

CF_1 cf;

DS_1 ds;

Operation	Description
GP_1()	Initialize instances for MDA, CF_1 and DS_1
Activate(float,float)	Stores Data in to temporary variables in DS_1 and calls Activate() method of MDA.
Start()	Calls Start() method of MDA
Reject()	Calls Reject() method of MDA
PayCredit()	Stores 1 in temporary M variable of DS_1 and calls PayType(1) of MDA
PayDebit(String)	Stores pin and temporary variable of DS_1, Stores 2 in temporary M variable of DS_1 and calls PayType(3) of MDA
Pin(String)	Validates Pin. If matched, calls CorrectPin() of MDA otherwise calls IncorrectPin() of MDA
Cancel()	Calls Cancel() method of MDA
Approved()	Calls Approved() method of MDA

Diesel()	Calls SelectGas(4) of MDA
Regular()	Calls SelectGas(1) of MDA
StartPump()	If set price is positive, calls Continue() and StartPump() of MDA
PumpGallon()	Calls Pump() method of MDA
StopPump()	Calls StopPump() and Receipt() method of MDA
FullTank()	Calls StopPump() and Receipt() method of MDA

(2) Class GP_2:

Responsibility- This Class is an implementation of Gas pump 2 with its specific types of input operations and supports functionalities specific to Gas Pump2.

Supported Operations –

MDA m;

CF_2 cf;

DS_2 ds;

Operation	Description
GP_2()	Initialize instances for MDA, CF_2 and DS_2
Activate(float,float,float)	Stores Data in to temporary variables in DS_2 and calls Activate() method of MDA.
Reject()	Calls Reject() method of MDA
PayCredit()	Stores 1 in temporary M variable of DS_2 and calls PayType(1) of MDA
PayCash(String)	If cah is positive, Stores cash and temporary variable of DS_2, Stores 0 in temporary M variable of DS_2 and calls PayType(2) of MDA
Cancel()	Calls Cancel() method of MDA

Approved()	Calls Approved() method of MDA
Premium()	Calls SelectGas(3) and Continue() of MDA
Regular()	Calls SelectGas(1) and Continue() of MDA
Super()	Calls SelectGas(2) and Continue() of MDA
StartPump()	Calls StartPump() of MDA
PumpLiter()	If Cash is not insufficient, Calls Pump() of MDA. otherwise StopPump() of MDA
Stop()	Calls StopPump() and Receipt() method of MDA
Receipt()	Calls Receipt() method of MDA
NoReceipt()	Calls Receipt() method of MDA

(3) Class MDA:

Responsibility- This Class contains Meta events. Which are platform Independent and compatible to handle different client functionalities.

Supported Operations –

State S;

State List[8];

Operation	Description
MDA()	Initialize all State objects and store it in to arraylist
ChangeState(int)	Changes current State
setOP()	Store Output processor instance
getOP()	Return Output Processor Instance
Activate()	Calls Activate() method to Current State Object

Start()	Calls Start() method to Current State Object
PayType(int)	CallsPayType() method to Current State Object
Reject()	Calls Reject() method to Current State Object
Cancel()	Calls Cancel() method to Current State Object
Approved()	Calls Approved() method to Current State Object
StartPump()	Calls StartPump() method to Current State Object
Pump()	Calls Pump() method to Current State Object
StopPump()	Calls StopPump() method to Current State Object
SelectGas(int)	CallsSelectGas() method to Current State Object
CorrectPin()	Calls CorrectPin() method to Current State Object
IncorrectPin()	Calls IncorrectPin() method to Current State Object
Receipt()	Calls Receipt() method to Current State Object
NoReceipt()	Calls NoReceipt() method to Current State Object
continue()	Calls continue() method to Current State Object

(4) Class OP:

Responsibility- This Class contains handles Meta Actions. This Class is the client of various action strategies.

Each Actions having more than one Strategy and has an Abstract class associated with the OP.

Supported Operations -

AF af;

Operation	Description
OP()	Initialize instance of AbstractFactory
StorePrices()	Calls StorePrices() action from Strategy StorePrces
PayMsg()	Calls PayMsg() action from Strategy payMsg
StoreCash()	Calls StoreCash() action from Strategy StoreCash
DisplayMenu()	Calls DisplayMenu() action from Strategy DisplayMenu
RejectMsg()	Calls RejectMsg() action from Strategy RejectMsg
SetInitialValues()	Calls StorePrices() action from Strategy StorePrces
ReadyMsg()	Calls ReadyMsg() action from Strategy ReadyMsg
PumpGasUnit()	Calls PumpGasUnit() action from Strategy PumpGasUnit
GasPumpedMsg()	Calls GasPumpedMsg() action from Strategy GasPumpedMsg
StopMsg()	Calls StopMsg () action from Strategy StopMsg
Printreceipt()	Calls Printreceipt () action from Strategy Printreceipt
CancelMsg()	Calls CancelMsg () action from Strategy CancelMsg
ReturnCash()	Calls ReturnCash () action from Strategy ReturnCash
WrongPinMsg()	Calls WrongPinMsg () action from Strategy WrongPinMsg
StorePin()	Calls StorePin () action from Strategy StorePin
EnterPinMsg()	Calls EnterPinMsg () action from Strategy EnterPinMsg
InitializeData()	Calls InitializeData () action from Strategy InitializeData
SetPrice(int)	Calls SetPrice () action from Strategy SetPrice
SetM()	Calls SetM () action from Strategy SetM

Package Factories:

(5) Class AbstractFactory:

Responsibility- This class is an abstract class of Factories. It is used to group the various concrete Factories CF_1 and CF_2.

Supported Operations -

Operation	Description
getData()	Abstract Method
getStorePrices()	Abstract Method
getPayMsg()	Abstract Method
getStoreCash()	Abstract Method
getDisplayMenu()	Abstract Method
getRejectMsg()	Abstract Method
getSetInitialValues()	Abstract Method
getReadyMsg()	Abstract Method
getPumpGasUnit()	Abstract Method
getGasPumpedMsg()	Abstract Method
getStopMsg()	Abstract Method
getPrintreceipt()	Abstract Method
getCancelMsg()	Abstract Method
getReturnCash()	Abstract Method

getWrongPinMsg()	Abstract Method
getStorePin()	Abstract Method
getEnterPinMsg()	Abstract Method
getInitializeData()	Abstract Method
getSetPrice(int)	Abstract Method
getSetM()	Abstract Method

(6) Class CF_1:

Responsibility- This class is a concrete factory class and responsible for creating objects of strategy classes for GP1.

Supported Operations -

DS_1 ds;

Operation	Description
CF_1()	Create new instance of DS_1
getData()	Return instance of DS_1
getStorePrices()	Return a new instance of StorePrices_GP1()
getPayMsg()	Return a new instance of PayMsg_GP1()
getStoreCash()	Return a new instance of StoreCash_GP1()
getDisplayMenu()	Return a new instance of DislayMenu_GP1()
getRejectMsg()	Return a new instance of RwjwctMsg_GP1()
getSetInitialValues()	Return a new instance of SetInitialValues_GP1()
getReadyMsg()	Return a new instance of ReadyMsg_GP1()
getPumpGasUnit()	Return a new instance of PumpGasUNit_GP1()

getGasPumpedMsg()	Return a new instance of GasPumpedMsg_GP1()
getStopMsg()	Return a new instance of StopMsg_GP1()
getPrintReceipt()	Return a new instance of PrintReceipt_GP1()
getCancelMsg()	Return a new instance of CancelMsg_GP1()
getReturnCash()	Return a new instance of ReturnCash_GP1()
getWrongPinMsg()	Return a new instance of WrongPinMsg_GP1()
getStorePin()	Return a new instance of StorePin_GP1()
getEnterPinMsg()	Return a new instance of EnterPinMsg_GP1()
getInitializeData()	Return a new instance of InitializeData_GP1()
getSetPrice(int)	Return a new instance of SetPrice_GP1()
getSetM()	Return a new instance of SetM_GP1()

(7) Class CF_2:

Responsibility- This class is a concrete factory class and responsible for creating objects of strategy classes for GP2.

Supported Operations -

DS_2 ds;

Operation	Description
CF_2()	Create new instance of DS_2
getData()	Return instance of DS_2
getStorePrices()	Return a new instance of StorePrices_GP2()
getPayMsg()	Return a new instance of PayMsg_GP2()
getStoreCash()	Return a new instance of StoreCash_GP2()

getDisplayMenu()	Return a new instance of DislayMenu_GP2()
getRejectMsg()	Return a new instance of RwjwctMsg_GP2()
getSetInitialValues()	Return a new instance of SetInitialValues_GP2()
getReadyMsg()	Return a new instance of ReadyMsg_GP2()
getPumpGasUnit()	Return a new instance of PumpGasUNit_GP2()
getGasPumpedMsg()	Return a new instance of GasPumpedMsg_GP2()
getStopMsg()	Return a new instance of StopMsg_GP2()
getPrintReceipt()	Return a new instance of PrintReceipt_GP2()
getCancelMsg()	Return a new instance of CancelMsg_GP2()
getReturnCash()	Return a new instance of ReturnCash_GP2()
getWrongPinMsg()	Return a new instance of WrongPinMsg_GP2()
getStorePin()	Return a new instance of StorePin_GP2()
getEnterPinMsg()	Return a new instance of EnterPinMsg_GP2()
getInitializeData()	Return a new instance of InitializeData_GP2()
getSetPrice(int)	Return a new instance of SetPrice_GP2()
getSetM()	Return a new instance of SetM_GP2()

Package Data:

(8) Class DataStore:

Responsibility- This is the abstract class inherited by DS_1 and DS_2.

(9) Class DS_1:

Responsibility- This Class stores data related to GP1.

Supported Operations –

Float temp_a, temp_b, price, cash, total, RPrice, DPrice

String temp_p, pin

Int temp_M, M,G

Operation	Description
Set_temp_a(float)	Sets argument value in to temp_a
Set_temp_b(float)	Sets argument value in to temp_b
Set_temp_p(String)	Sets argument value in to temp_p
Set_temp_M(int)	Sets argument value in to temp_M
Set_pin(String)	Sets argument value in to pin
Set_price(float)	Sets argument value in to price
Set_cash(foat)	Sets argument value in to cash
Set_G(int)	Sets argument value in to G
Set_total(float)	Sets argument value in to total
Set_RPrice(float)	Sets argument value in to RPrice
Set_DPrice(float)	Sets argument value in to DPrice
get_temp_a()	Return temp_a
get_temp_b()	Return temp_b
get_temp_p()	Return temp_p
get_temp_M()	Return temp_M
get_pin()	Return pin
get_price()	Return price
get_cash()	Return cash

get_G()	Return G
get_total()	Return total
get_RPrice()	Return RPrice
get_DPrice()	Return DPrice

(10) Class DS_2:

Responsibility- This Class stores data related to GP2.

Supported Operations –

Float temp_a, temp_b, temp_c, price, temp_cash, cash, total, RPrice, PPrice, SPrice

Int temp_M, M,L

Operation	Description
Set_temp_a(float)	Sets argument value in to temp_a
Set_temp_b(float)	Sets argument value in to temp_b
Set_temp_c(String)	Sets argument value in to temp_c
Set_temp_M(int)	Sets argument value in to temp_M
Set_temp_cash(float)	Sets argument value in to cash
Set_price(float)	Sets argument value in to price
Set_cash(foat)	Sets argument value in to cash
Set_L(int)	Sets argument value in to L
Set_total(float)	Sets argument value in to total
Set_RPrice(float)	Sets argument value in to RPrice
Set_PPrice(float)	Sets argument value in to PPrice

Set_SPrice(float)	Sets argument value in to SPrice
get_temp_a()	Return temp_a
get_temp_b()	Return temp_b
get_temp_c()	Return temp_c
get_temp_M()	Return temp_M
get_M()	Return M
get_temp_cash()	Return temp_cash
get_price()	Return price
get_cash()	Return cash
get_L()	Return L
get_total()	Return total
get_RPrice()	Return RPrice
get_SPrice()	Return SPrice
Get_PPrice()	Return PPrice

Package States:

(11) Class State:

Responsibility- This class is like an abstract class. It does not have implementations of the actions.

Supported Operations –

MDA m;

OP op;

Operation	Description
State(MDA m)	Initialize MDA object
Activate()	No Implementation
Start()	No Implementation
PayType(int)	No Implementation
Reject()	No Implementation
Cancel()	No Implementation
Approved()	No Implementation
StartPump()	No Implementation
Pump()	No Implementation
StopPump()	No Implementation
SelectGas(int)	No Implementation
CorrectPin()	No Implementation
IncorrectPin()	No Implementation
Receipt()	No Implementation
NoReceipt()	No Implementation
continue()	No Implementation

(12) Class Start State:

Responsibility- This class is a concrete class of class State. It is having only Activate() method to activate GP.

Supported Operations –

MDA m;

OP op;

Operation	Description
StartState(MDA m)	Initialize MDA object
Activate()	Calls StorePrices() of OP and calls ChangeState(1) of MDA

(13) Class S0_State:

Responsibility- This class is a concrete class of class State. It is having only Start() method to initialize GP.

Supported Operations –

MDA m;

OP op;

Operation	Description
S0_State(MDA m)	Initialize MDA object
Activate()	Calls PayMsg() and initializeData() of OP and calls ChangeState(2) of MDA

(14) Class S1_State:

Responsibility- This class is a concrete class of class State. This state has only PayType() method.

Supported Operations –

MDA m;

OP op;

Operation	Description
S1_State(MDA m)	Initialize MDA object
PayType(a)	if a==1, calls setM of OP, calls changestate(3) of MDA

	if a==2 ,calls storecash() ,displaymenu(), setM() of OP, calls changestate(4) of MDA if a==3, calls Enterpinmsg(),StorePin(),SetM() of OP, calls ChangeState(8) of MDA.

(15) Class S2_State:

Responsibility- This class is a concrete class of class State. It handles Approve() and reject() methods.

Supported Operations –

MDA m;

OP op;

Operation	Description
S2_State(MDA m)	Initialize MDA object
Approved()	Calls DisplauMenu() of OP and calls ChangeState(4) of MDA
Reject()	Calls RejectMsg() of OP and calls ChangeState(1) of MDA

(16) Class S3_State:

Responsibility- This class is a concrete class of class State. It handles SelectGas() and Cancel() methods.

Supported Operations –

MDA m;

OP op;

Operation	Description
S3_State(MDA m)	Initialize MDA object
SelectGas(int a)	It calls SetPrice(g) of OP, where g is type of gas Calls ChangeState(4) of MDA.

Cancel()	Calls CancelMsg(), ReturnCash() of OP and ChangeState(1) of MDA

(17) Class S4_State:

Responsibility- This class is a concrete class of class State. It handles only StartPump() method.

Supported Operations –

MDA m;

OP op;

Operation	Description
S4_State(MDA m)	Initialize MDA object
StartPump()	Calls SetInitialValues() ,ReadyMsg() of OP and calls ChangeState(6) of MDA

(18) Class S5_State:

Responsibility- This class is a concrete class of class State. It is having Pump(), StopPump() methods.

Supported Operations –

MDA m;

OP op;

Operation	Description
S5_State(MDA m)	Initialize MDA object.
Pump()	Calls PumpGasUnit(),GasPumpedMsg() of OP and calls ChangeState(6) of MDA.
StopPump()	Calls StopMsg() of OP and calls ChangeState(7) of MDA.

(19) Class S6_State:

Responsibility- This class is a concrete class of class State. It is having Receipt() and NoReceipt() methods.

Supported Operations –

MDA m;

OP op;

Operation	Description
S6_State(MDA m)	Initialize MDA object.
Receipt()	Calls PrintReceipt(), ReturnCash() of OP and calls ChangeState(1) of MDA.
NoReceipt()	Calls ReturnCash() of OP and calls ChangeState(1) of MDA.

(20) Class S7_State:

Responsibility- This class is a concrete class of class State. It has CorrectPin() and InCorrectPin() methods.

Supported Operations –

MDA m;

OP op;

Operation	Description
S6_State(MDA m)	Initialize MDA object.
CorrectPin()	Calls DisplayMenu() of OP and calls ChangeState(4) of MDA.
InCorrectPin()	Calls WrongPinMsg() of OP and calls ChangeState(1) of MDA.

Package Strategy:

(21) Class CancelMsgAbstract:

Responsibility- This class is an abstract class of strategy method CancelMsg.

Supported Operations :-

Operation	Description
CancelMsg()	No Implementation

(22) Class CancelMsg_GP1:

Responsibility- This class is a Concrete class of strategy method CancelMsg for GP1.

Supported Operations :-

Operation	Description
CancelMsg()	Print Cancellation Message for GP1

(23) Class CancelMsg_GP2:

Responsibility- This class is a Concrete class of strategy method CancelMsg for GP2.

Supported Operations :-

Operation	Description
CancelMsg()	Print Cancellation Message for GP2

(24) Class DisplayMenuAbstract:

Responsibility- This class is an abstract class of strategy method DisplayMenu().

Supported Operations :-

Operation	Description
DisplayMenu()	No Implementation

(25) Class DisplayMenu_GP1:

Responsibility- This class is a Concrete class of strategy method DisplayMenu for GP1.

Supported Operations :-

Operation	Description
DisplayMenu()	Display Menu for GP1.

(26) Class CancelMsg_GP2:

Responsibility- This class is a Concrete class of strategy method DisplayMenu for GP2.

Supported Operations :-

Operation	Description
DisplayMenu()	Display Menu for GP2.

(27) Class EnterPinMsgAbstract:

Responsibility- This class is an abstract class of strategy method EnterPinMsg().

Supported Operations :-

Operation	Description
EnterPinMsg()	No Implementation

(28) Class EnterPinMsg_GP1:

Responsibility- This class is a Concrete class of strategy method EnterPinMsg() for GP1.

Supported Operations :-

Operation	Description
EnterPinMsg()	Enter Pin Message for GP1

(29) Class EnterPinMsg_GP2:

Responsibility- This class is a Concrete class of strategy method EnterPinMsg for GP2.

Supported Operations :-

Operation	Description
EnterPinMsg()	No Implementation

(30) Class GasPumpedMsgAbstract:

Responsibility- This class is an abstract class of strategy method GasPumpedMsg().

Supported Operations :-

Operation	Description

GasPumpedMsg()	No Implementation

(31) Class GasPumpedMsg_GP1:

Responsibility- This class is a Concrete class of strategy method GasPumpedMsg() for GP1.

Supported Operations :-

Operation	Description
GasPumpedMsg ()	Prints gas pumped message for GP1

(32) Class GasPumpedMsg_GP2:

Responsibility- This class is a Concrete class of strategy method GasPumpedMsg for GP2.

Supported Operations :-

Operation	Description
GasPumpedMsg()	Gas Pumped Message for GP2.

(33) Class InitializeDataAbstract:

Responsibility- This class is an abstract class of strategy method InitializeData().

Supported Operations :-

Operation	Description
InitializeData()	No Implementation

(34) Class InitializeData_GP1:

Responsibility- This class is a Concrete class of strategy method InitializeData() for GP1.

Supported Operations :-

Operation	Description
InitializeData()	Set price and cash to 0 for GP1.

(35) Class InitializeData_GP2:

Responsibility- This class is a Concrete class of strategy method InitializeData for GP2.

Supported Operations :-

Operation	Description
InitializedData()	Set price and cash for GP2.

(36) Class PayMsgAbstract:

Responsibility- This class is an abstract class of strategy method PayMsg().

Supported Operations :-

Operation	Description
PayMsg()	No Implementation

(37) Class PayMsg _GP1:

Responsibility- This class is a Concrete class of strategy method PayMsg() for GP1.

Supported Operations :-

Operation	Description
PayMsg()	Display Pay message for GP1.

(38) Class PayMsg_GP2:

Responsibility- This class is a Concrete class of strategy method PayMsg for GP2.

Supported Operations :-

Operation	Description
PayMsg()	Display Pay message for GP2.

(39) Class PrintReceiptAbstract:

Responsibility- This class is an abstract class of strategy method PrintReceipt().

Supported Operations :-

Operation	Description
PrintReceipt()	No Implementation

(40) Class PrintReceipt _GP1:

Responsibility- This class is a Concrete class of strategy method PrintReceipt() for GP1.

Supported Operations :-

Operation	Description
PrintReceipt()	Print Receipt for GP1..

(41) Class PrintReceipt_GP2:

Responsibility- This class is a Concrete class of strategy method PrintReceipt() for GP2.

Supported Operations :-

Operation	Description
PrintReceipt()	Print Receipt for GP2.

(42) Class PumpGasUnitAbstract:

Responsibility- This class is an abstract class of strategy method PumpGasUnit().

Supported Operations :-

Operation	Description
PumpGasUnit()	No Implementation

(43) Class PumpGasUnit _GP1:

Responsibility- This class is a Concrete class of strategy method PumpGasUnit() for GP1.

Supported Operations :-

Operation	Description
PumpGasUnit()	It increases the gas disposed counter by one and compute the total And store it in DS_1 for GP1.

(44) Class PumpGasUnit_GP2:

Responsibility- This class is a Concrete class of strategy method PumpGasUnit() for GP2.

Supported Operations :-

Operation	Description
PumpGasUnit()	It increases the gas disposed counter by one and compute the total And store it in DS_2 for GP2.

(45) Class ReadyMsgAbstract:

Responsibility- This class is an abstract class of strategy method ReadyMsg().

Supported Operations :-

Operation	Description
ReadyMsg()	No Implementation

(46) Class ReadyMsg _GP1:

Responsibility- This class is a Concrete class of strategy method ReadyMsg() for GP1.

Supported Operations :-

Operation	Description
ReadyMsg()	It prints the ready Message for GP1.

(47) Class ReadyMsg_GP2:

Responsibility- This class is a Concrete class of strategy method ReadyMsg() for GP2.

Supported Operations :-

Operation	Description
ReadyMsg()	It prints the ready Message for GP2.

(48) Class RejectMsgAbstract:

Responsibility- This class is an abstract class of strategy method RejectMsg().

Supported Operations :-

Operation	Description
RejectMsg()	No Implementation

(49) Class RejectMsg _GP1:

Responsibility- This class is a Concrete class of strategy method RejectMsg() for GP1.

Supported Operations :-

Operation	Description
RejectMsg()	It prints the reject Message for GP1.

(50) Class RejectMsg_GP2:

Responsibility- This class is a Concrete class of strategy method RejectMsg() for GP2.

Supported Operations :-

Operation	Description
RejectMsg()	It prints the reject Message for GP2.

(51) Class ReturnCashAbstract:

Responsibility- This class is an abstract class of strategy method ReturnCash().

Supported Operations :-

Operation	Description
ReturnCash()	No Implementation

(52) Class ReturnCash_GP1:

Responsibility- This class is a Concrete class of strategy method ReturnCash() for GP1.

Supported Operations :-

Operation	Description
ReturnCash()	No Implementation as This Gaspump does not support cash Transaction

(53) Class ReturnCash_GP2:

Responsibility- This class is a Concrete class of strategy method ReturnCash() for GP2.

Supported Operations :-

Operation	Description
ReturnCash()	It returns remaining cash.

(54) Class SetInitialValuesAbstract:

Responsibility- This class is an abstract class of strategy method SetInitialValues().

Supported Operations :-

Operation	Description
SetInitialValues()	No Implementation

(55) Class SetInitialValues_GP1:

Responsibility- This class is a Concrete class of strategy method SetInitialValues() for GP1.

Supported Operations :-

Operation	Description
SetInitialValues()	It initializes Gas Unit, price and total price for GP1.

(56) Class SetInitialValues_GP2:

Responsibility- This class is a Concrete class of strategy method SetInitialValues() for GP2.

Supported Operations :-

Operation	Description
SetInitialValues()	It initializes Gas Unit, price and total price for GP2. If M==1, It updates price.

(57) Class SetMAbstract:

Responsibility- This class is an abstract class of strategy method SetM().

Supported Operations :-

Operation	Description
SetM()	No Implementation

(58) Class SetM_GP1:

Responsibility- This class is a Concrete class of strategy method SetM() for GP1.

Supported Operations :-

Operation	Description
SetM()	It sets M value from temp_M to M in DS_1 for GP1

(59) Class SetM_GP2:

Responsibility- This class is a Concrete class of strategy method SetM() for GP2.

Supported Operations :-

Operation	Description
SetM()	It sets M value from temp_M to M in DS_2 for GP2

(60) Class SetPriceAbstract:

Responsibility- This class is an abstract class of strategy method SetPrice().

Supported Operations :-

Operation	Description
SetPrice(int)	No Implementation

(61) Class SetPrice_GP1:

Responsibility- This class is a Concrete class of strategy method SetPrice() for GP1.

Supported Operations :-

Operation	Description
SetPrice(int g)	It sets prices from RPrice or DPrice based on g value in DS_1 for GP_1.

(62) Class SetPrice_GP2:

Responsibility- This class is a Concrete class of strategy method SetPrice() for GP2.

Supported Operations :-

Operation	Description
SetPrice()	It sets prices from RPrice,PPrice or SPrice based on g value in DS_2 for GP_2.

(63) Class StopMsgAbstract:

Responsibility- This class is an abstract class of strategy method StopMsg().

Supported Operations :-

Operation	Description
StopMsg()	No Implementation

(64) Class StopMsg _GP1:

Responsibility- This class is a Concrete class of strategy method StopMsg() for GP1.

Supported Operations :-

Operation	Description
StopMsg()	Prints Stop Pump Message for GP_1

(65) Class StopMsg_GP2:

Responsibility- This class is a Concrete class of strategy method StopMsg() for GP2.

Supported Operations :-

Operation	Description
StopMsg()	Prints Stop Pump Message for GP_2

(66) Class StoreCashAbstract:

Responsibility- This class is an abstract class of strategy method StoreCash().

Supported Operations :-

Operation	Description
StoreCash()	No Implementation

(67) Class StoreCash_GP1:

Responsibility- This class is a Concrete class of strategy method StoreCash() for GP1.

Supported Operations :-

Operation	Description
StoreCash()	No implementation as GP1 does not support Cash Transactions.

(68) Class StoreCash_GP2:

Responsibility- This class is a Concrete class of strategy method StoreCash() for GP2.

Supported Operations :-

Operation	Description
StoreCash()	Stores Cash from temp_cash to Cash in DS_2 for GP2.

(69) Class StorePinAbstract:

Responsibility- This class is an abstract class of strategy method StorePin().

Supported Operations :-

Operation	Description
StorePin()	No Implementation

(70) Class StorePin_GP1:

Responsibility- This class is a Concrete class of strategy method StorePin() for GP1.

Supported Operations :-

Operation	Description
StorePin()	Stores Pin from temp_p to pin in DS_1 for GP1.

(71) Class StorePin_GP2:

Responsibility- This class is a Concrete class of strategy method StorePin() for GP2.

Supported Operations :-

Operation	Description
StorePin()	No Implementation as GP2 does not support Debit Transactions.

(72) Class StorePricesAbstract:

Responsibility- This class is an abstract class of strategy method StorePrices().

Supported Operations :-

Operation	Description
StorePrices()	No Implementation

(73) Class StorePrices_GP1:

Responsibility- This class is a Concrete class of strategy method StorePrices() for GP1.

Supported Operations :-

Operation	Description
StorePrices()	Stores Prices of RPrice,DPrices from temp_a,temp_b in DS_1 for GP1.

(74) Class StorePrices_GP2:

Responsibility- This class is a Concrete class of strategy method StorePrices() for GP2.

Supported Operations :-

Operation	Description
StorePrices()	Stores Prices of RPrice,SPrices,PPrice from temp_a,temp_b,temp_c in DS_2 for GP2.

(75) Class WrongPinMsg Abstract:

Responsibility- This class is an abstract class of strategy method WrongPinMsg().

Supported Operations :-

Operation	Description
WrongPinMsg()	No Implementation

(76) Class WrongPinMsg_GP1:

Responsibility- This class is a Concrete class of strategy method WrongPinMsg() for GP1.

Supported Operations :-

Operation	Description
WrongPinMsg()	Displays Wrong Pin Message For GP1.

(77) Class WrongPinMsg_GP2:

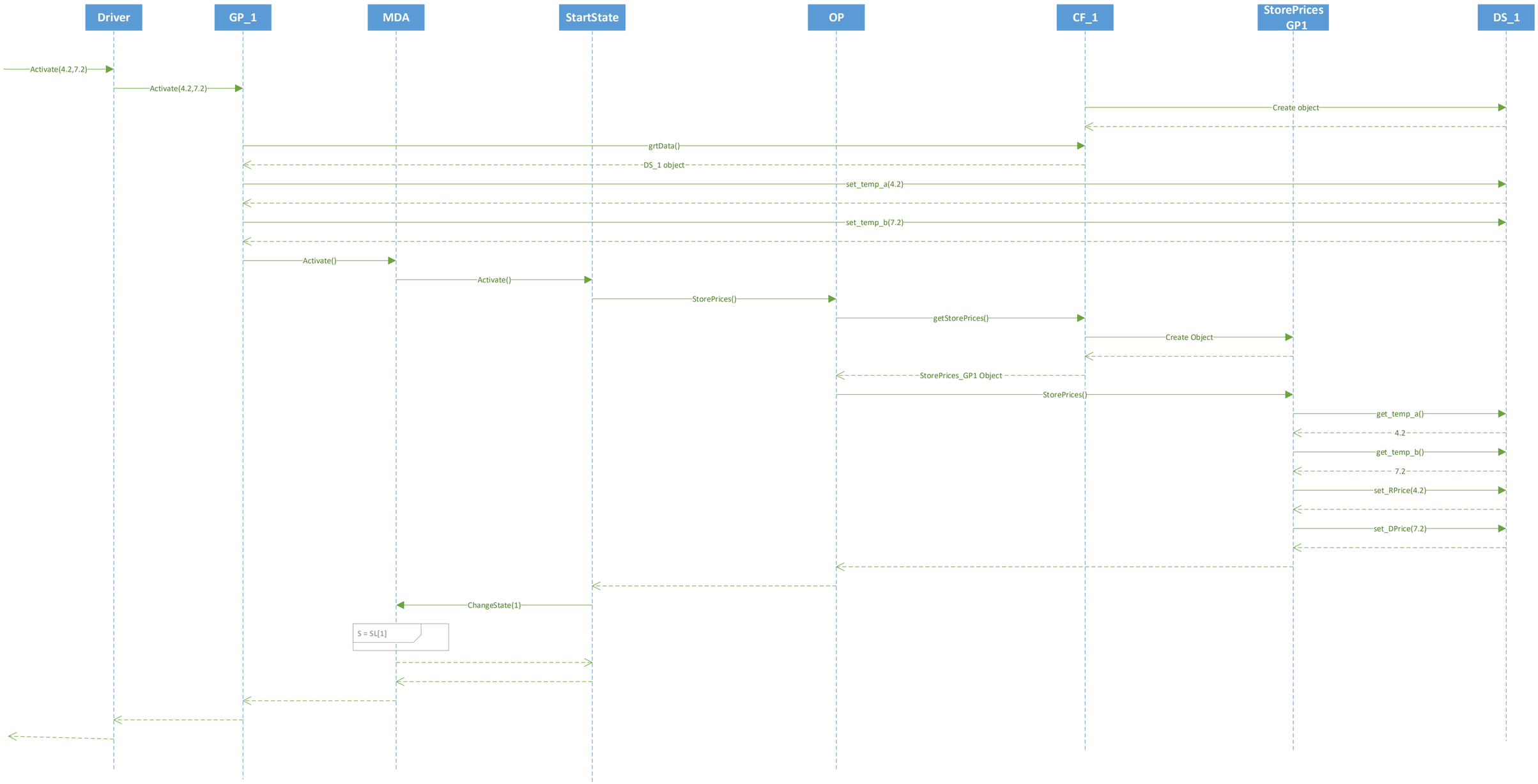
Responsibility- This class is a Concrete class of strategy method WrongPinMsg() for GP2.

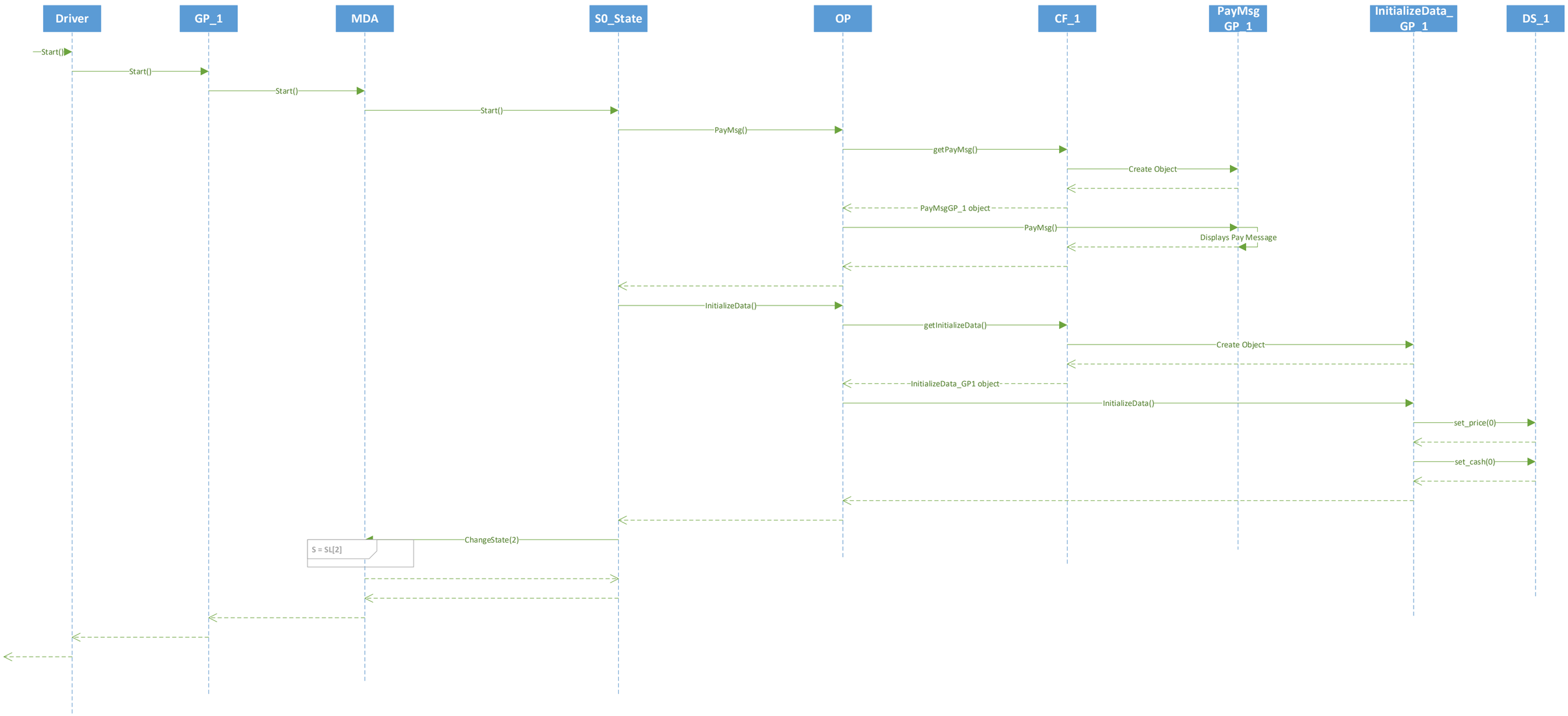
Supported Operations :-

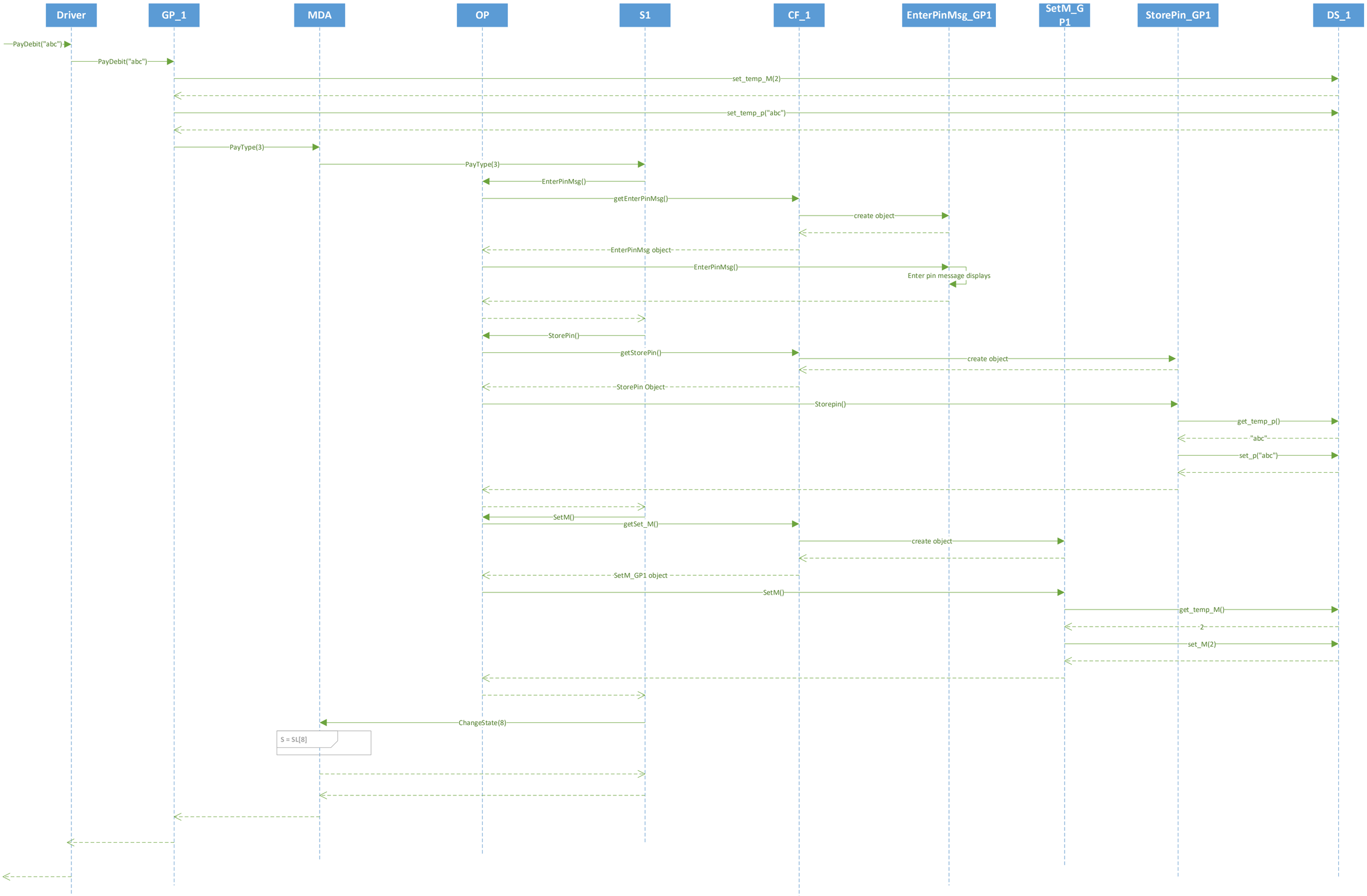
Operation	Description
WrongPinMsg()	No Implementation as GP2 Does not support Debit Transaction.

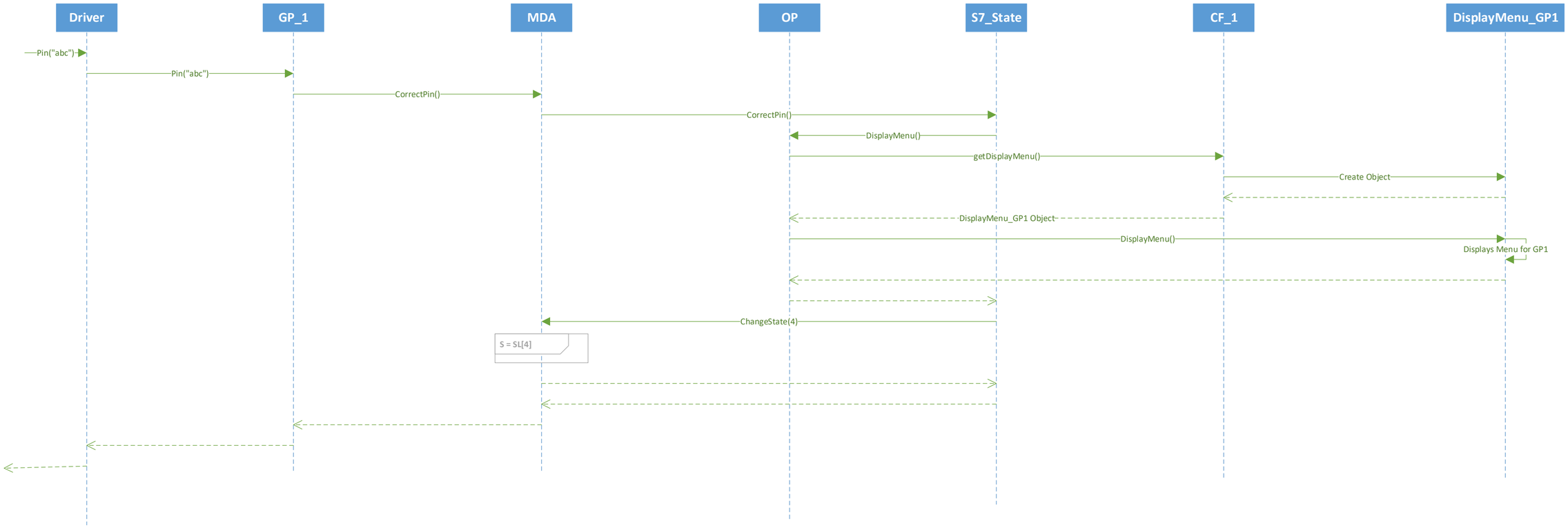
SCENARIO – 1

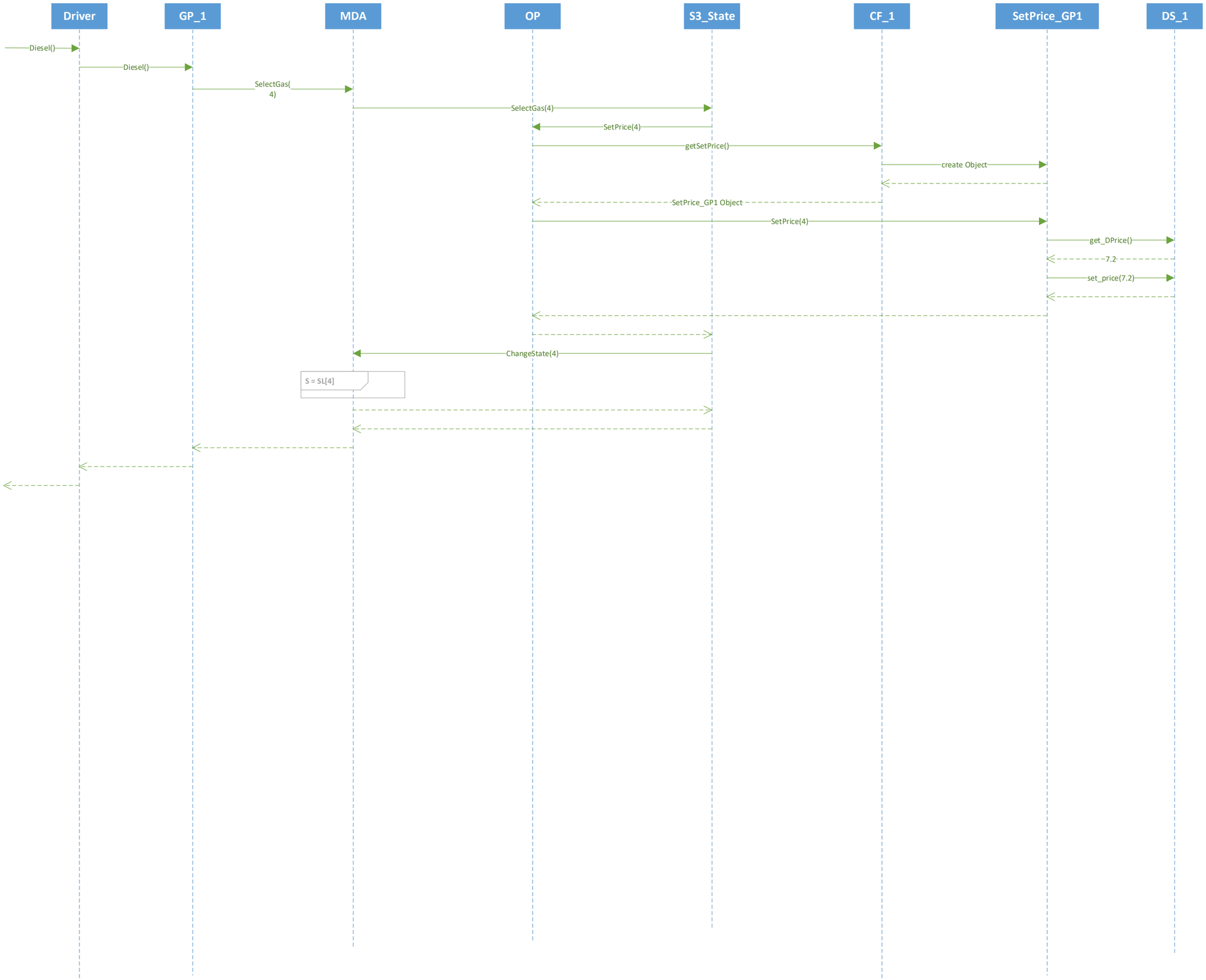
Sequence Diagram

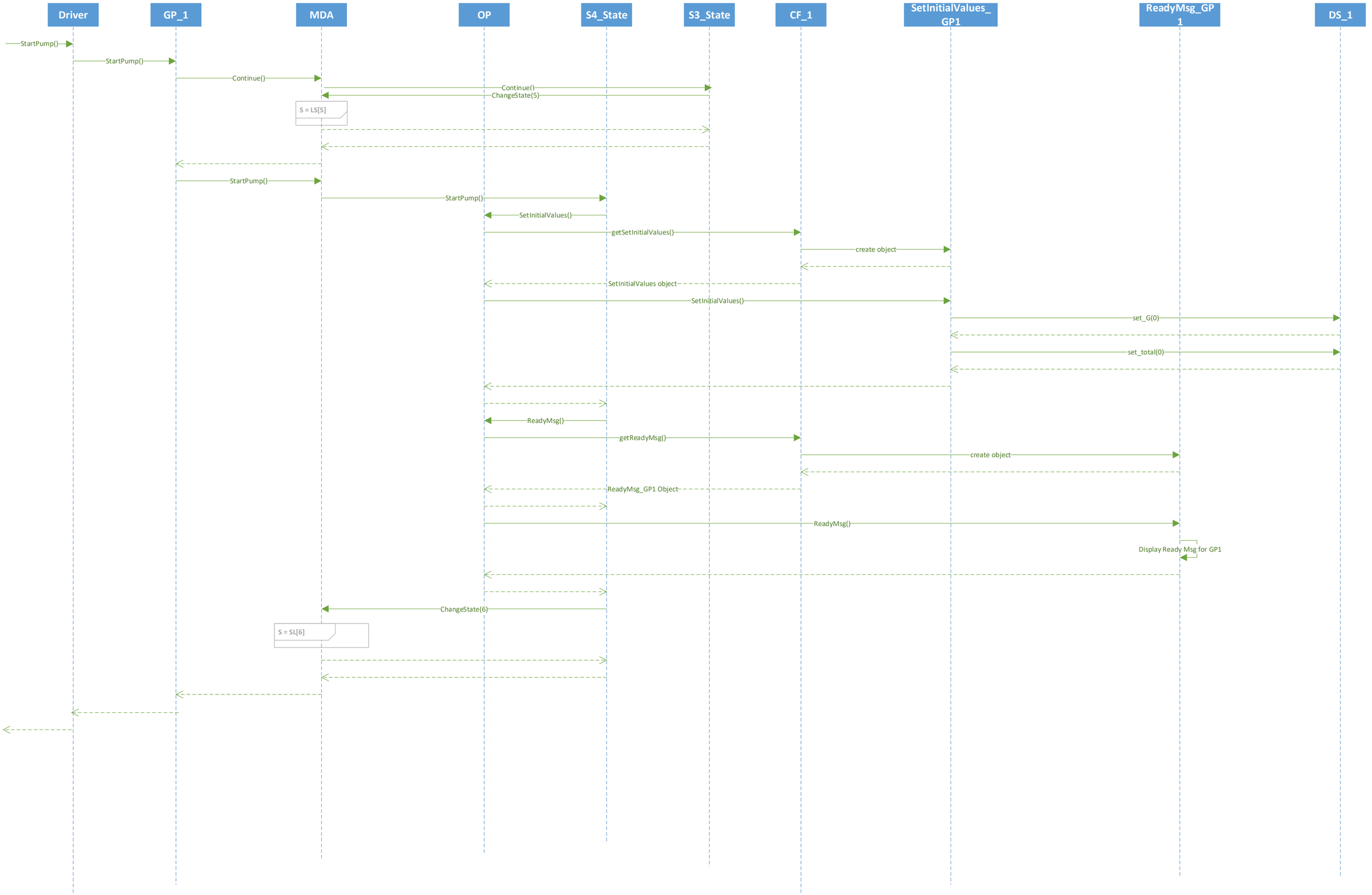

Driver
GP_1
MDA
StartState
OP
CF_1
StorePrices GP1
DS_1
Activate(4.2,7.2)
Activate(4.2,7.2)
Create object
grtData()
DS_1 object
set_temp_a(4.2)
set_temp_b(7.2)
Activate()
Activate()
StorePrices()
getStorePrices()
Create Object
StorePrices_GP1 Object
StorePrices()
get_temp_a()
4.2
get_temp_b()
7.2
set_RPrice(4.2)
set_DPrice(7.2)
ChangeState(1)
S = SL[1]

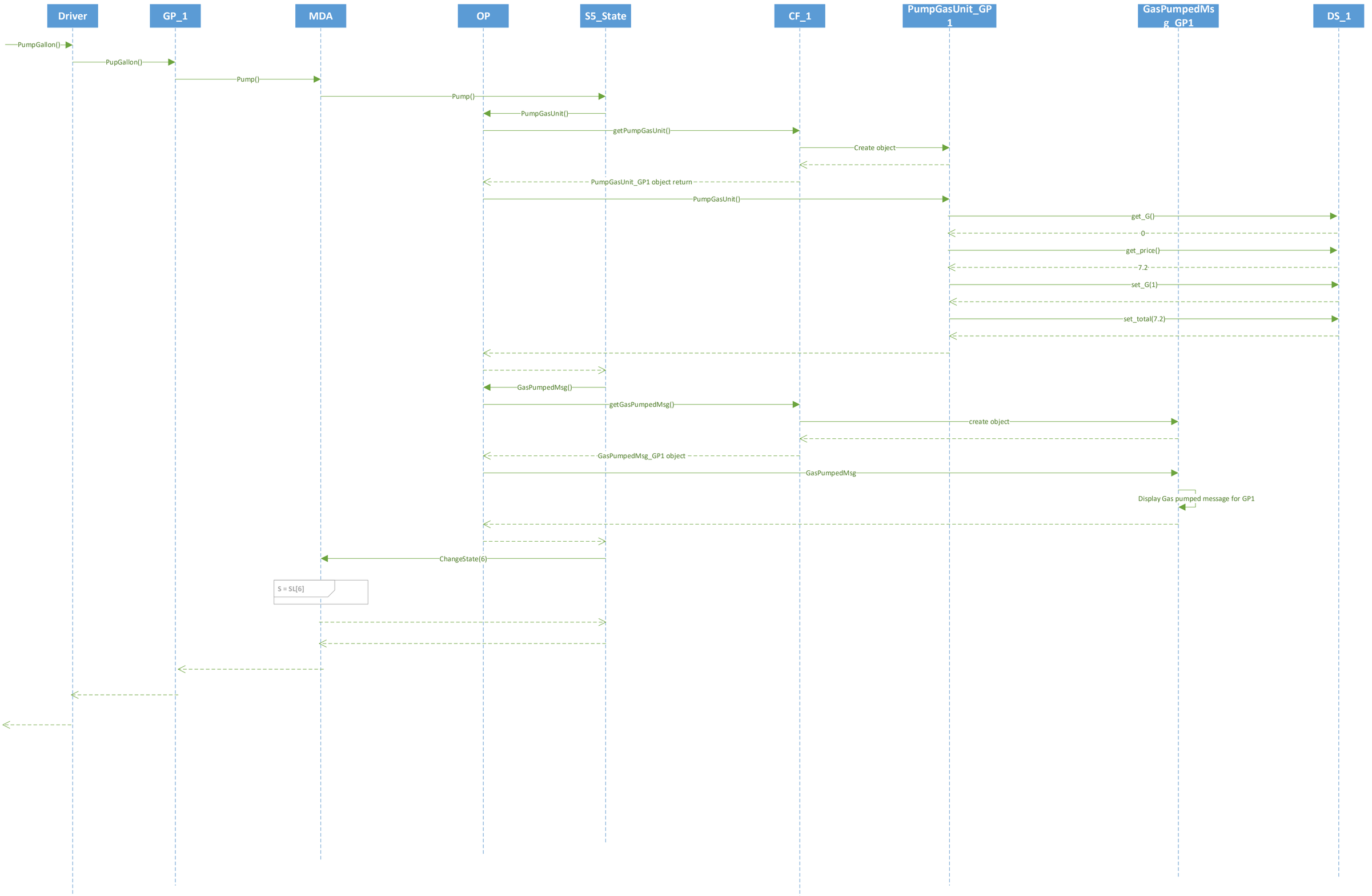

Driver
GP_1
MDA
S0_State
OP
CF_1
PayMsg GP_1
InitializeData_ GP_1
DS_1
Start()
Start()
Start()
Start()
PayMsg()
getPayMsg()
Create Object
PayMsgGP_1 object
PayMsg()
Displays Pay Message
InitializeData()
getInitializeData()
Create Object
InitializeData_GP1 object
InitializeData()
set_price(0)
set_cash(0)
ChangeState(2)
S = SL[2]

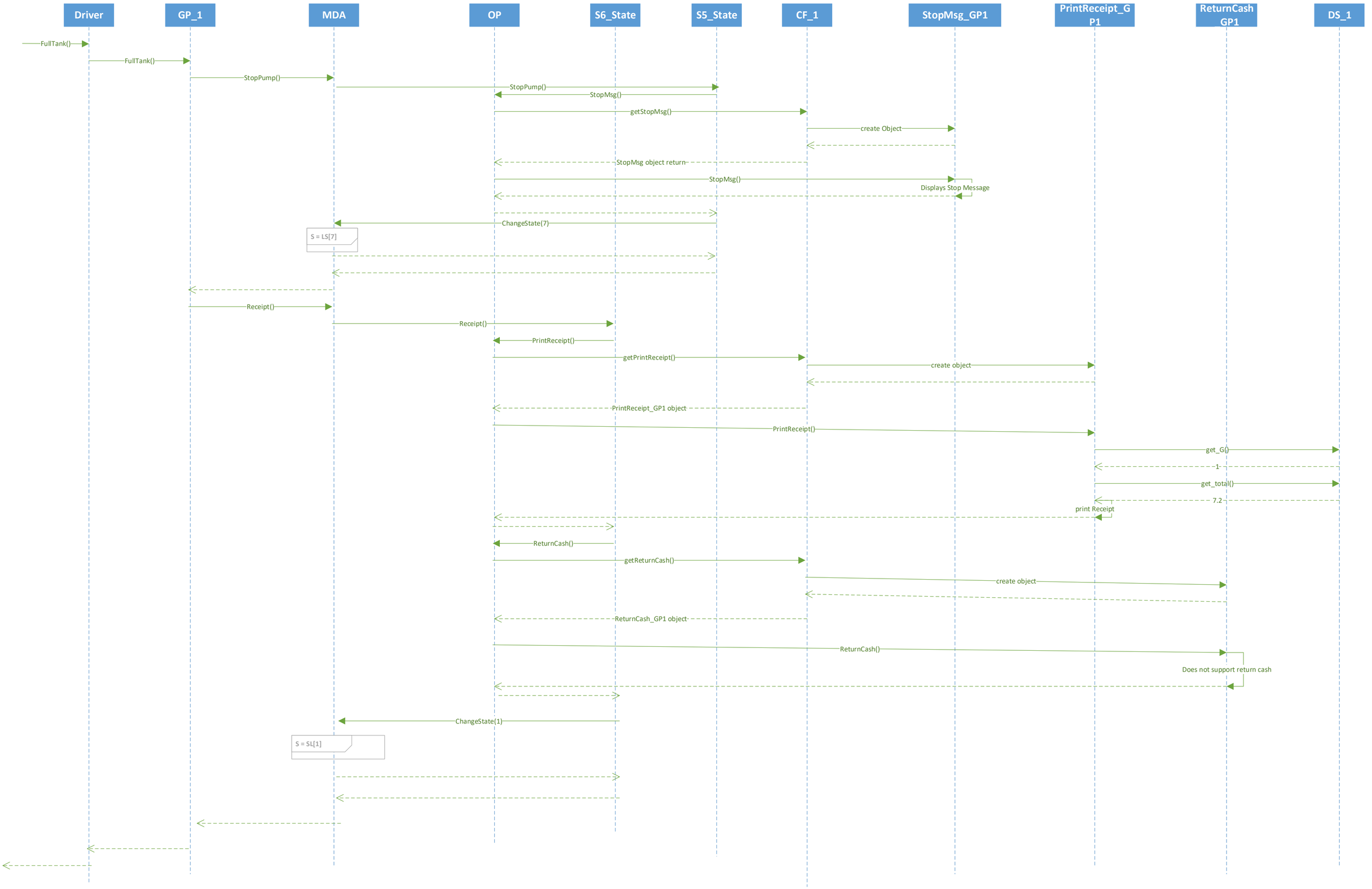

Driver
GP_1
MDA
OP
S1
CF_1
EnterPinMsg_GP1
SetM_GP1
StorePin_GP1
DS_1
PayDebit("abc")
PayDebit("abc")
set_temp_M(2)
set_temp_p("abc")
PayType(3)
PayType(3)
EnterPinMsg()
getEnterPinMsg()
create object
EnterPinMsg object
EnterPinMsg()
Enter pin message displays
StorePin()
getStorePin()
create object
StorePin Object
Storepin()
get_temp_p()
"abc"
set_p("abc")
SetM()
getSet_M()
create object
SetM_GP1 object
SetM()
get_temp_M()
2
set_M(2)
ChangeState(8)
S = SL[8]


Driver
GP_1
MDA
OP
S7_State
CF_1
DisplayMenu_GP1
Pin("abc")
Pin("abc")
CorrectPin()
CorrectPin()
DisplayMenu()
getDisplayMenu()
Create Object
DisplayMenu_GP1 Object
DisplayMenu()
Displays Menu for GP1
ChangeState(4)
S = SL[4]


Driver
GP_1
MDA
OP
S3_State
CF_1
SetPrice_GP1
DS_1
Diesel()
Diesel()
SelectGas(
4)
SelectGas(4)
SetPrice(4)
getSetPrice()
create Object
SetPrice_GP1 Object
SetPrice(4)
get_DPrice()
7.2
set_price(7.2)
ChangeState(4)
S = SL[4]

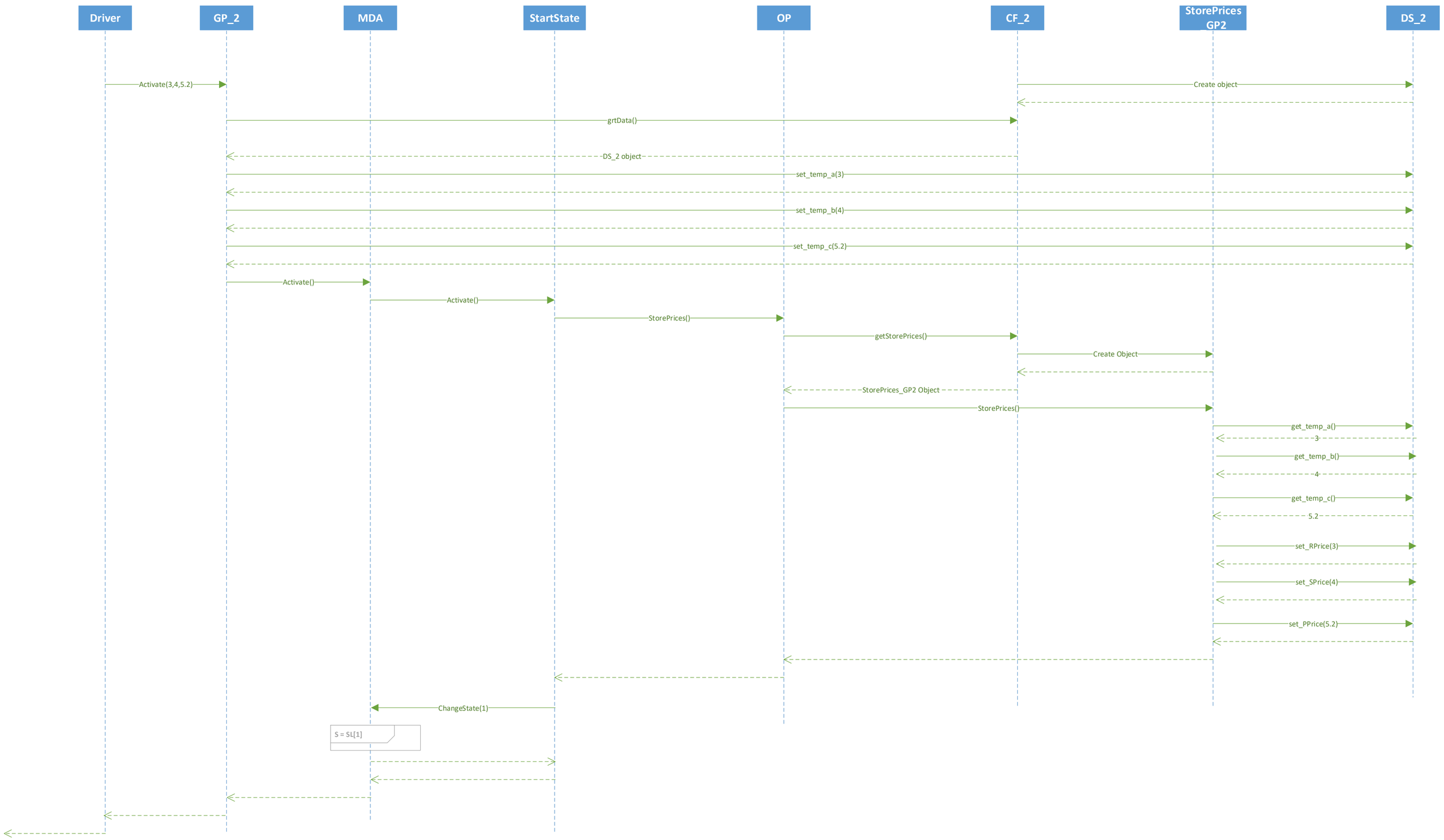

Driver
GP_1
MDA
OP
S4_State
S3_State
CF_1
SetInitialValues_GP1
ReadyMsg_GP1
DS_1
StartPump()
StartPump()
Continue()
Continue()
ChangeState(5)
S = LS[5]
StartPump()
StartPump()
SetInitialValues()
getSetInitialValues()
create object
SetInitialValues object
SetInitialValues()
set_G(0)
set_total(0)
ReadyMsg()
getReadyMsg()
create object
ReadyMsg_GP1 Object
ReadyMsg()
Display Ready Msg for GP1
ChangeState(6)
S = SL[6]

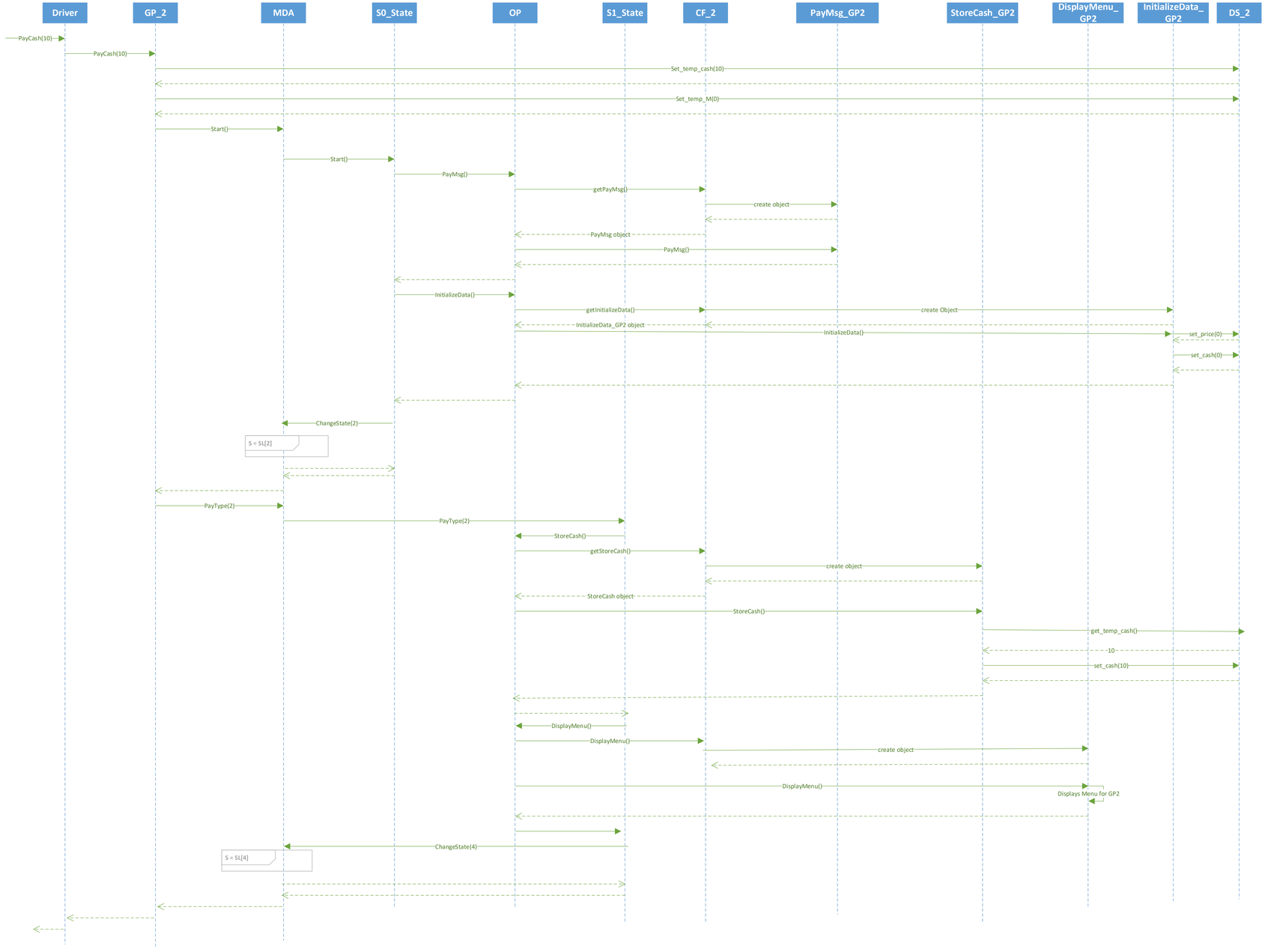

Driver
GP_1
MDA
OP
S5_State
CF_1
PumpGasUnit_GP1
GasPumpedMsg_GP1
DS_1
PumpGallon()
PupGallon()
Pump()
Pump()
PumpGasUnit()
getPumpGasUnit()
Create object
PumpGasUnit_GP1 object return
PumpGasUnit()
get_G()
0
get_price()
7.2
set_G(1)
set_total(7.2)
GasPumpedMsg()
getGasPumpedMsg()
create object
GasPumpedMsg_GP1 object
GasPumpedMsg
Display Gas pumped message for GP1
ChangeState(6)
S = SL[6]

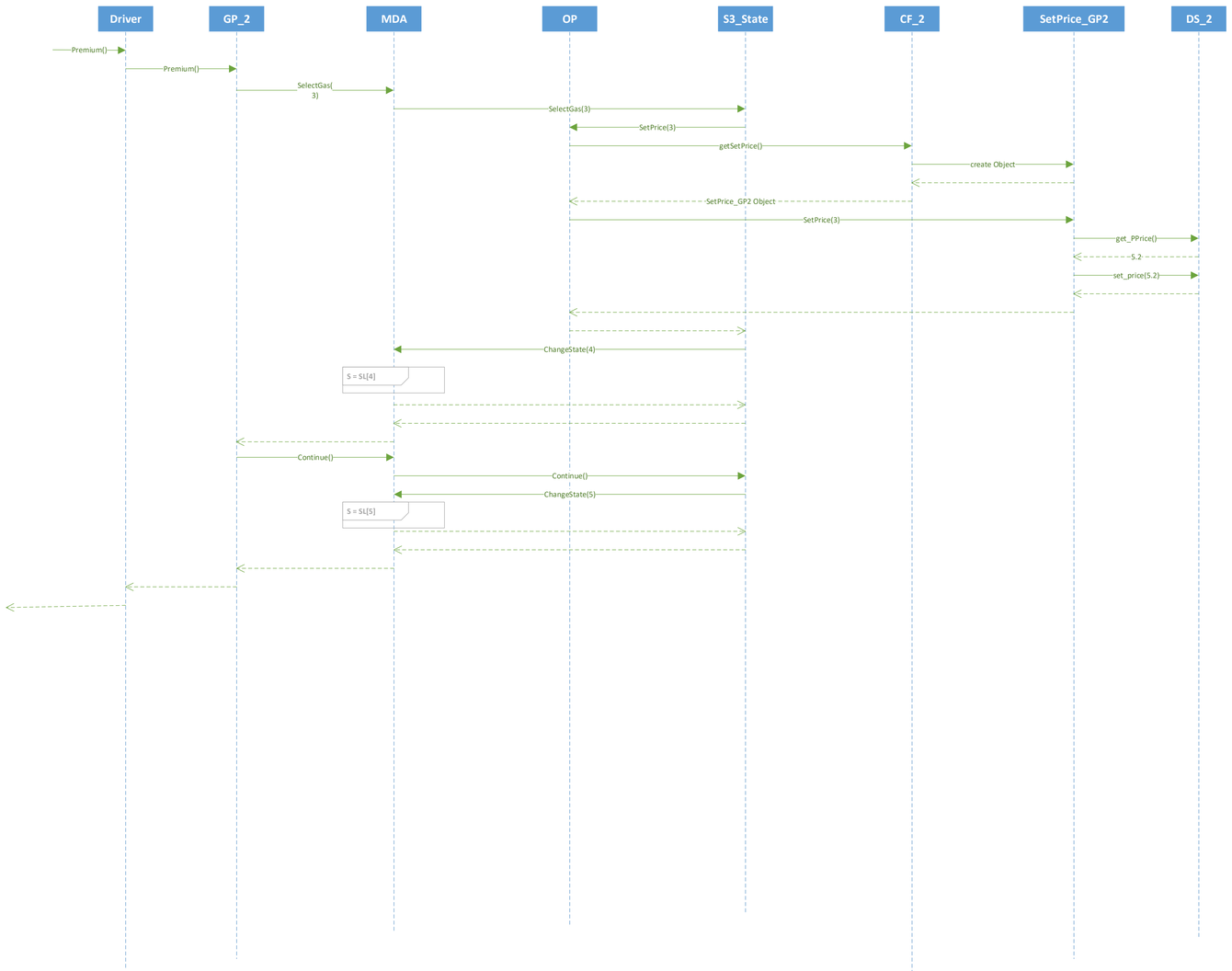

Driver
GP_1
MDA
OP
S6_State
S5_State
CF_1
StopMsg_GP1
PrintReceipt_GP1
ReturnCash_GP1
DS_1
FullTank()
FullTank()
StopPump()
StopPump()
StopMsg()
getStopMsg()
create Object
StopMsg object return
StopMsg()
Displays Stop Message
ChangeState(7)
S = LS[7]
Receipt()
Receipt()
PrintReceipt()
getPrintReceipt()
create object
PrintReceipt_GP1 object
PrintReceipt()
get_G()
1
get_total()
7.2
print Receipt
ReturnCash()
getReturnCash()
create object
ReturnCash_GP1 object
ReturnCash()
Does not support return cash
ChangeState(1)
S = SL[1]

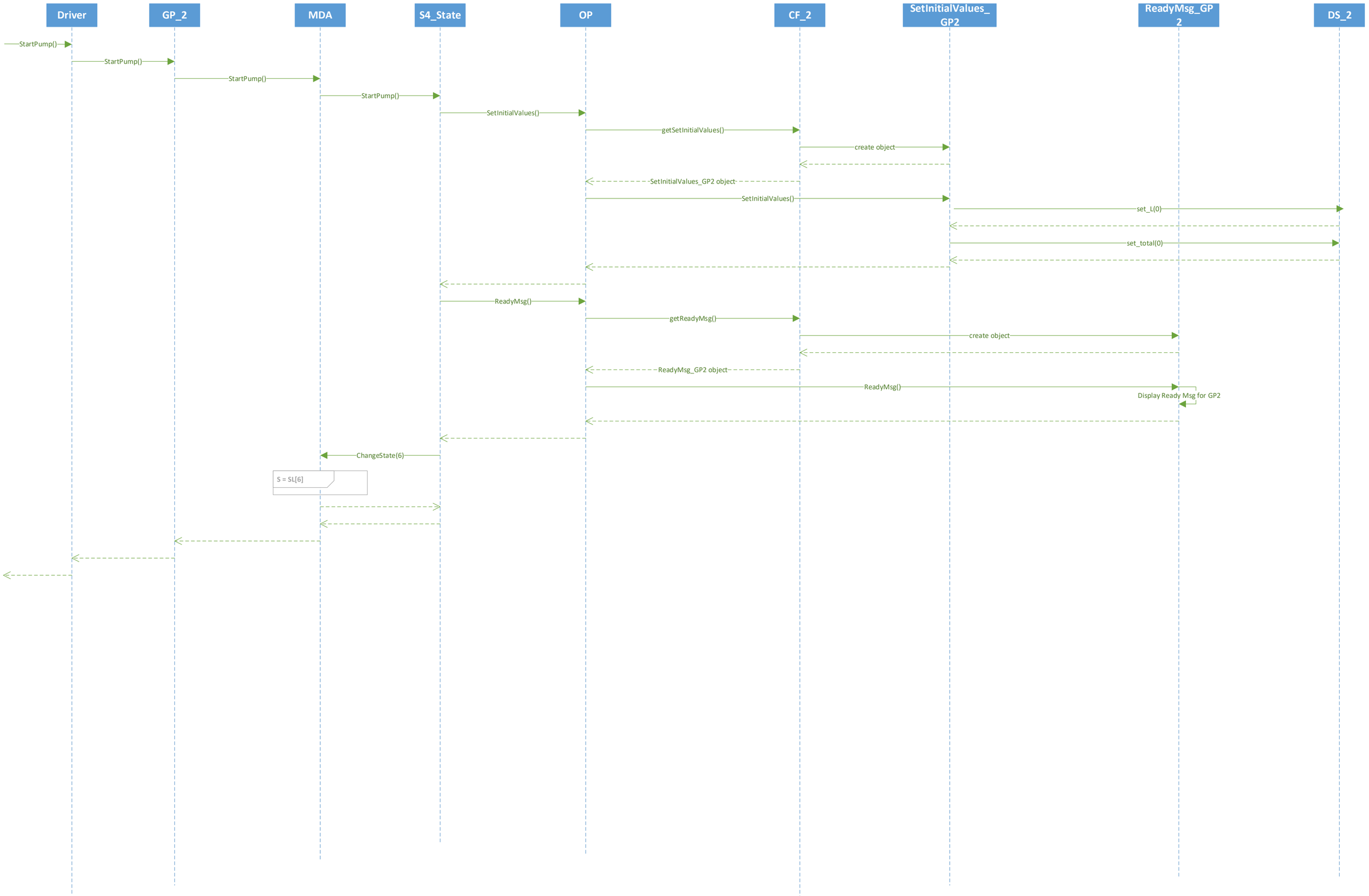
SCENARIO – 2

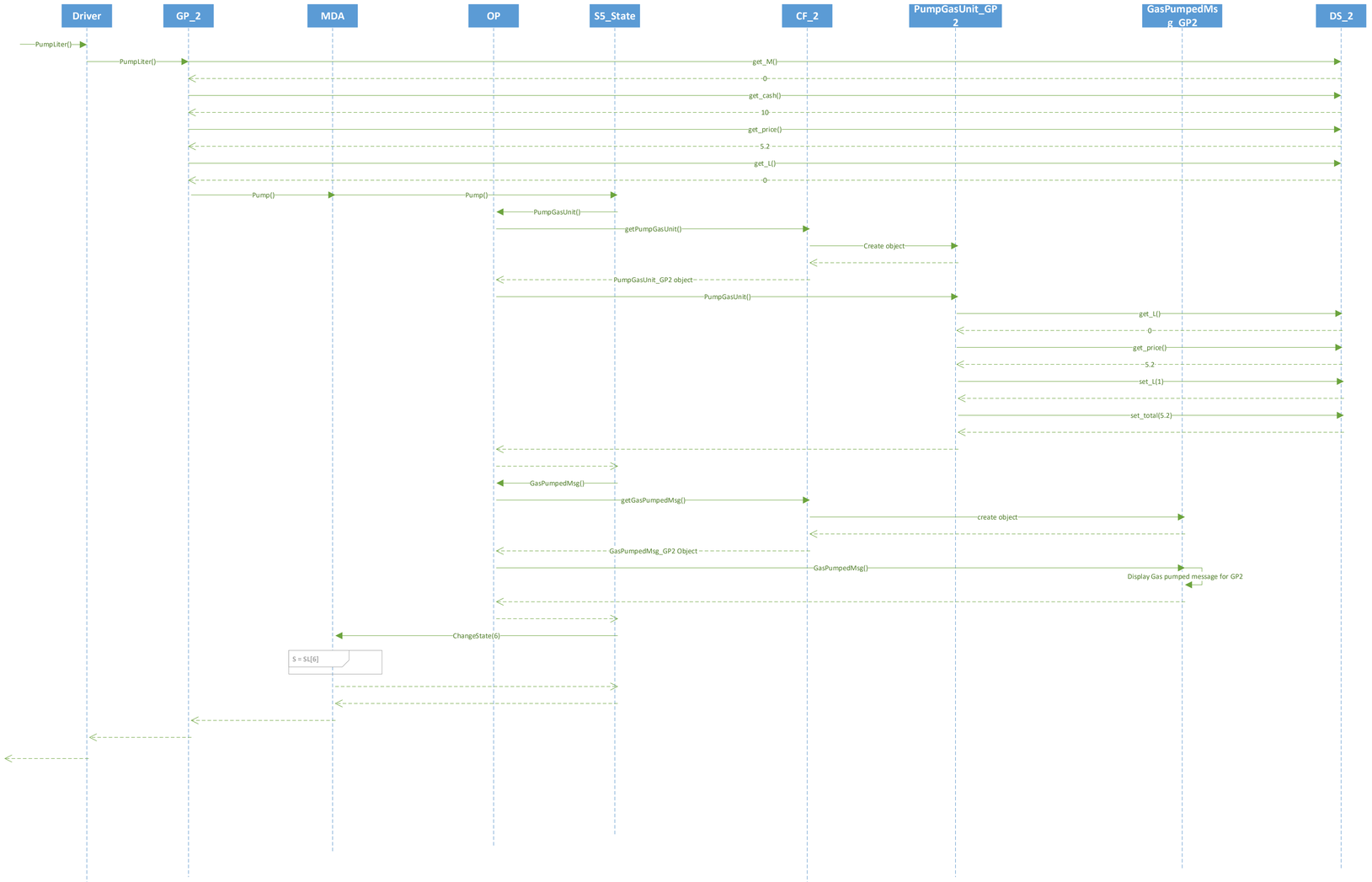
Sequence Diagram

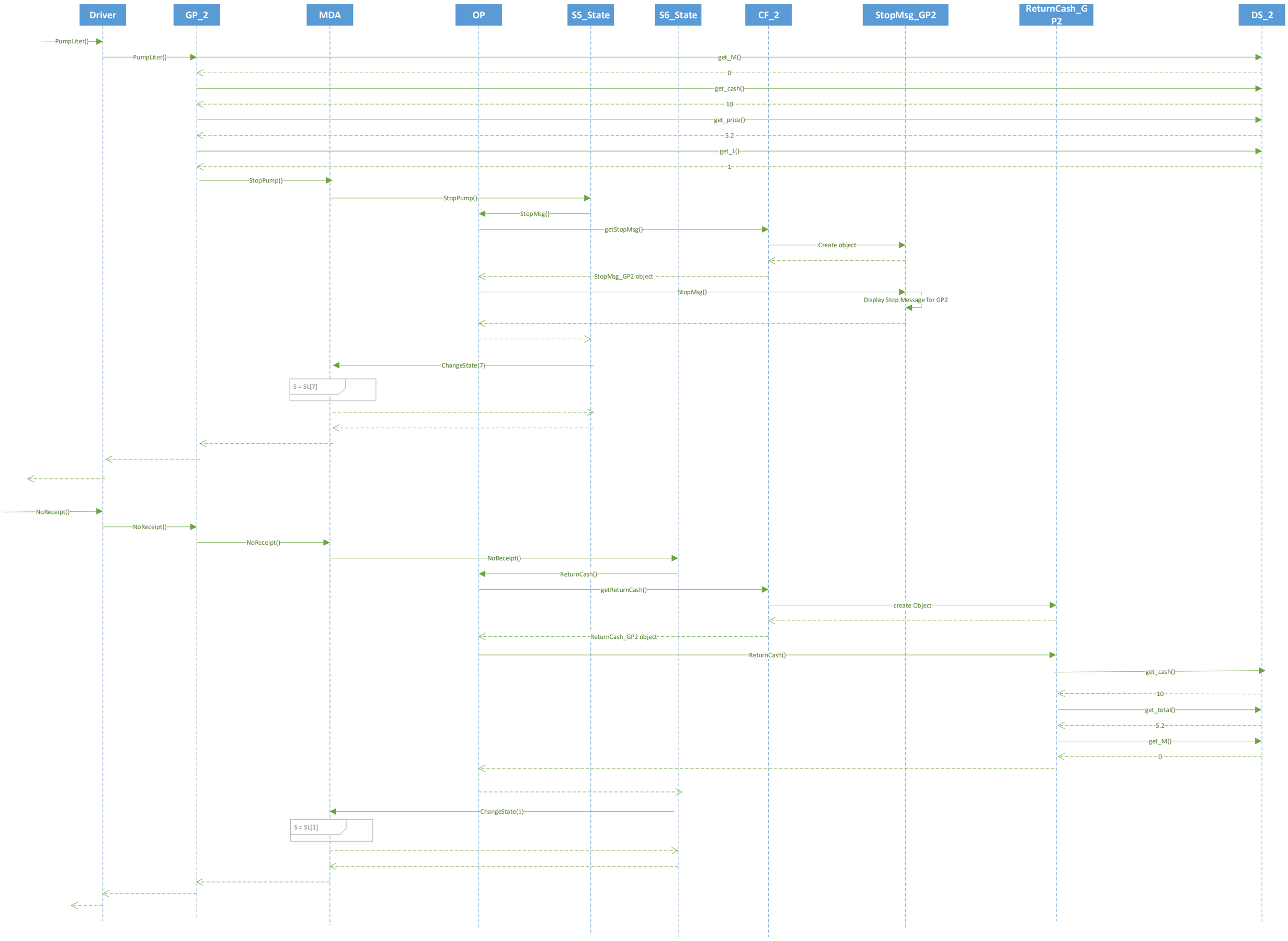

Driver
GP_2
MDA
StartState
OP
CF_2
StorePrices GP2
DS_2
Activate(3,4,5.2)
Create object
grtData()
DS_2 object
set_temp_a(3)
set_temp_b(4)
set_temp_c(5.2)
Activate()
Activate()
StorePrices()
getStorePrices()
Create Object
StorePrices_GP2 Object
StorePrices()
get_temp_a()
3
get_temp_b()
4
get_temp_c()
5.2
set_RPrice(3)
set_SPrice(4)
set_PPrice(5.2)
ChangeState(1)
S = SL[1]


Driver
GP_2
MDA
S0_State
OP
S1_State
CF_2
PayMsg_GP2
StoreCash_GP2
DisplayMenu_GP2
InitializeData_GP2
DS_2
PayCash(10)
PayCash(10)
Set_temp_cash(10)
Set_temp_M(0)
Start()
Start()
PayMsg()
getPayMsg()
create object
PayMsg object
PayMsg()
InitializeData()
getInitializeData()
create Object
InitializeData_GP2 object
InitializeData()
set_price(0)
set_cash(0)
ChangeState(2)
S = SL[2]
PayType(2)
PayType(2)
StoreCash()
getStoreCash()
create object
StoreCash object
StoreCash()
get_temp_cash()
10
set_cash(10)
DisplayMenu()
DisplayMenu()
create object
DisplayMenu()
Displays Menu for GP2
ChangeState(4)
S = SL[4]


Driver
GP_2
MDA
OP
S3_State
CF_2
SetPrice_GP2
DS_2
Premium()
Premium()
SelectGas(
3)
SelectGas(3)
SetPrice(3)
getSetPrice()
create Object
SetPrice_GP2 Object
SetPrice(3)
get_PPrice()
5.2
set_price(5.2)
ChangeState(4)
S = SL[4]
Continue()
Continue()
ChangeState(5)
S = SL[5]


Driver
GP_2
MDA
S4_State
OP
CF_2
SetInitialValues_GP2
ReadyMsg_GP2
DS_2
StartPump()
StartPump()
StartPump()
StartPump()
SetInitialValues()
getSetInitialValues()
create object
SetInitialValues_GP2 object
SetInitialValues()
set_L(0)
set_total(0)
ReadyMsg()
getReadyMsg()
create object
ReadyMsg_GP2 object
ReadyMsg()
Display Ready Msg for GP2
ChangeState(6)
S = SL[6]


Driver
GP_2
MDA
OP
S5_State
CF_2
PumpGasUnit_GP2
GasPumpedMsg_GP2
DS_2
PumpLiter()
PumpLiter()
get_M()
0
get_cash()
10
get_price()
5.2
get_L()
0
Pump()
Pump()
PumpGasUnit()
getPumpGasUnit()
Create object
PumpGasUnit_GP2 object
PumpGasUnit()
get_L()
0
get_price()
5.2
set_L(1)
set_total(5.2)
GasPumpedMsg()
getGasPumpedMsg()
create object
GasPumpedMsg_GP2 Object
GasPumpedMsg()
Display Gas pumped message for GP2
ChangeState(6)
S = S1[6]


Driver
GP_2
MDA
OP
S5_State
S6_State
CF_2
StopMsg_GP2
ReturnCash_GP2
DS_2
PumpLiter()
PumpLiter()
get_M()
0
get_cash()
10
get_price()
5.2
get_L()
1
StopPump()
StopPump()
StopMsg()
getStopMsg()
Create object
StopMsg_GP2 object
StopMsg()
Display Stop Message for GP2
ChangeState(7)
S = SL[7]
NoReceipt()
NoReceipt()
NoReceipt()
NoReceipt()
ReturnCash()
getReturnCash()
create Object
ReturnCash_GP2 object
ReturnCash()
get_cash()
10
get_total()
5.2
get_M()
0
ChangeState(1)
S = SL[1]