1 Introduction

1.1 History

Version Control Previous Version : 1.0 Current Version : 1.1	
Release data (version 1.0) : 13.01.2021 Release data (version 1.1) : 26.01.2021	
Changes in Document	
Sections in Previous Document	Sections in current Document
Section 3 : The MCU continuously receives data from the GPS or GSM and writes to respective on board memory. Suppose there is an event from GSM for the track location(event X), the MCU reads the location data from memory and send back to the phone via GSM.	Section 3 : The MCU default state is ideal (MCU not doing any job), waiting for the track event from the phone, once event is received MCU reads the location data from the GPS and reads the network base station data from the GSM. MCU Writes the GPS data to the memory1 and GSM data to the memory2, and transmit the complete data to the mobile phone through GSM. After the complete process, MCU goes back to the ideal state.
Micro controller : This module is the main controlling unit of the system which receives the location data from the GPS and GSM and stores it in the memory. And wait for the wake up call (character X) from the phone.Once the character is received the processor read location data from memory and sends back to phone via GSM.	Micro controller : This module is the main controlling unit of the system which wait for the wake up call(x) from the phone, once it is received it read the GPS and GSM data and store it in the memory and the processor read location data from memory and sends back to phone via GSM.
Section 4.1.2 :	Section 4.1.2 : Added SPEC100, SPEC110, SPEC120
Section 4.1.2 :	Section 4.1.2 : Changed SPEC12, SPEC13
Section 4.2.1 :	Section 4.2.1: Added SPEC130, SPEC140, SPEC150
Section 4.1.2 :	Section 4.1.2 : Changed SPEC18, SPEC19
Section 5 :	Section 5 : Changed Data flow
Section 6 :	Section 5: Changed requirement and specification mapping
Figure 3,5,6,7	Figure 3,5,6,7 Changed

1.2 Identification

The specification document establishes the design requirements for the tracking system.

1.3 System Overview

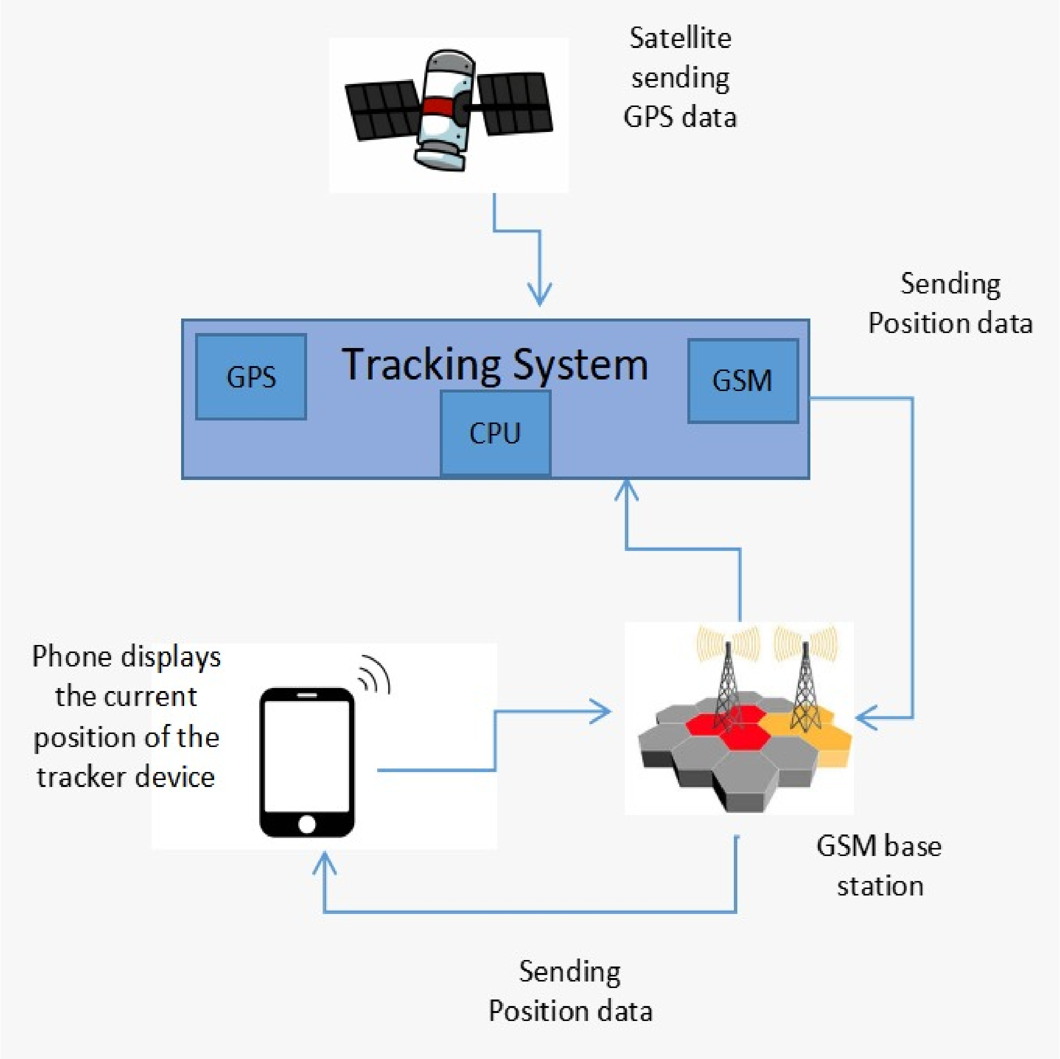

Figure 1: *system block diagram*

A tracking system consists of GPS and GSM modules to trace the current position of the vehicle on which the tracking system is mounted. The GUI (Graphical User Interface) is provided in the mobile phone for the user to initiate the tracking. The server is used to do the tracking calculation. The processor is used to read and transfer the position data to the server through GSM. Finally the mobile application conveys the position in the map.

1.4 Document Overview

This Specification document establishes the specification for the Tracking System. The document follows the format and the guidelines formulated by the professor.Dr.Ing.Andreas Siggelkow.

2 Requirements

Requirement	**ID**	**Importance**	**Verification**	**Description**
General				
Gen.: GPS	G01	HIGH	The GPS data format is generated and transmitted to processor in SystemC test bench.	The GPS reads the current position of the device from the satellite in the form of NMEA format.
Gen.: GSM	G02	HIGH	SystemC test bench	One GSM module is used to initiate the tracking event from the server and to receive the position data at the server from the device.
Memory				
Memory : Write GPS data to on board memory 1	Mem01	HIGH	SystemC Test bench	The microprocessor has to write the last updated position data (approx.1 to 2m accuracy)to the on board memory.

Memory : Write GSM base station network position to memory 2	Mem02	HIGH	SystemC Test bench	The microprocessor has to write the last updated network base station position (100 to 200 m accuracy)to the on board memory.
UART 1	UART01		Tested using FIFO	
UART 1: 115200		HIGH	SystemC Test bench	The speed of the serial transmission should be set to 115200 baud
UART 1: 8 bit		HIGH	SystemC Test bench	The data width of the serial transmission should be set to 8 bit.
UART 1: no parity		HIGH	SystemC Test bench	The serial transmission should not be verified with a parity bit.
UART 1: one stop bit		HIGH	SystemC Test bench	The serial transmission should end with the one end point.
UART 1: GPS Data		HIGH	SystemC Test bench	The GPS data has to be read by the processor from the GPS module through UART1.
UART 2	UART02		Tested using FIFO	

UART 2: 115200		HIGH	SystemC Test bench	The speed of the serial transmission should be set to 115200 baud
UART 2: 8 bit		HIGH	SystemC Test bench	The data width of the serial transmission should be set to 8 bit.
UART 2: no parity		HIGH	SystemC Test bench	The serial transmission should not be verified with a parity bit.
UART 2: one stop bit		HIGH	SystemC Test bench	The serial transmission should end with the one end point.
UART 2: GPS Data		HIGH	SystemC Test bench	UART2 is used by the GSM to send and receive data between MCU

3 Product Top level

The SystemC is the programming language which is used to design and the test the tracker system. There are totally five modules used to design the system and those are :

1. GPS : This module continuously sends the predefined GPS data to the processor through the FIFO1 (acts as UART1).

2. GSM : This module has two operation, one is to send the tracker system network base station data to the processor and other one is to send the track event(it is considered as a character X) to the processor (through FIFO 2 acts as a UART 2) which is generated by the phone and to receive the location data from the processor through FIFO 3 (acts as a UART 2).

3. Bus : This module just forms the data transmission bridge between master and slave. The data transmission is character by character (using TLM module).

4. Micro controller : This module is the main controlling unit of the system which wait for the wake up call(x) from the phone, once it is received it read the GPS and GSM data and store it in the memory and the processor read location data from memory and sends back to phone via GSM.

5. Memory : This module stores the GPS and GSM data at particular memory location. Whenever there is a read back request from the processor, memory sends back the stored data to the processor.

The Figure 2 shows the top level of the chip with systemC implementation ideas:

The MCU default state is ideal (MCU not doing any job), waiting for the track event from the phone, once event is received MCU reads the location data from the GPS and reads the network base station data from the GSM. MCU Writes the GPS data to the memory1 and GSM data to the memory2, and transmit the complete data to the mobile phone through GSM. After the complete process, MCU goes back to the ideal state.

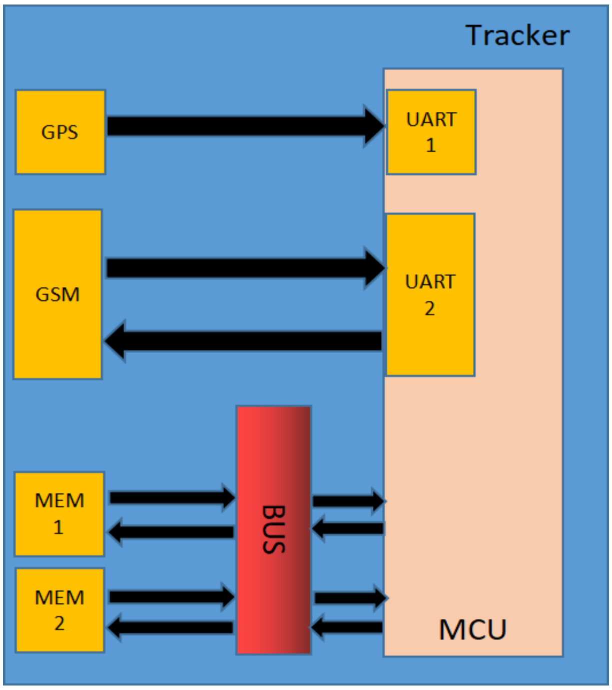

Figure 2: *Top level diagram*

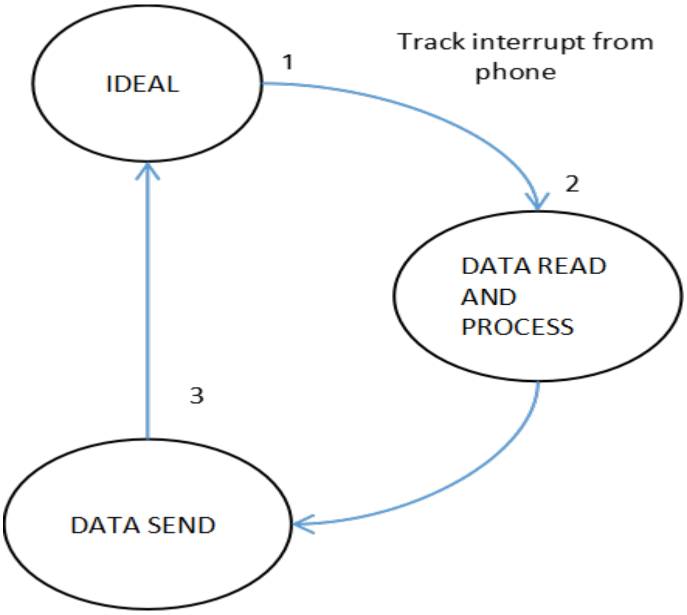

Figure 3: *The algorithm running on CPU*

IO Details

Pin	Direction	Width	Explanation
Clk_i	in	1	System clock
Rst_i	in	1	System reset, active low
Gps_i	in		Air interface to the satellites, represented by a FIFO for chars or unsigned int (template)
Gsm_i	In		Air interface from the GSM network, represented by a simple bus interface (SystemC / TLM)
Gsm_o	Out		Air interface to the GSM network, represented by a simple bus interface (SystemC / TLM)
Bus<1,2>		char or unsigned int	The bus has two contributors: • Master: the MCU • Slave: memory 1, for GPS data • Slave: memory 2, for GSM data

4 Architecture Concept

The task is to design the simple bus and master MCU which connects the different slave modules and is responsible for the data transmission between the connected modules. The Simple bus and the master uses the various sockets and interfaces, the figure 4 shows how the master is interfaced between the different slaves, similarly how the bus acts as a bridge between the master and the memory.

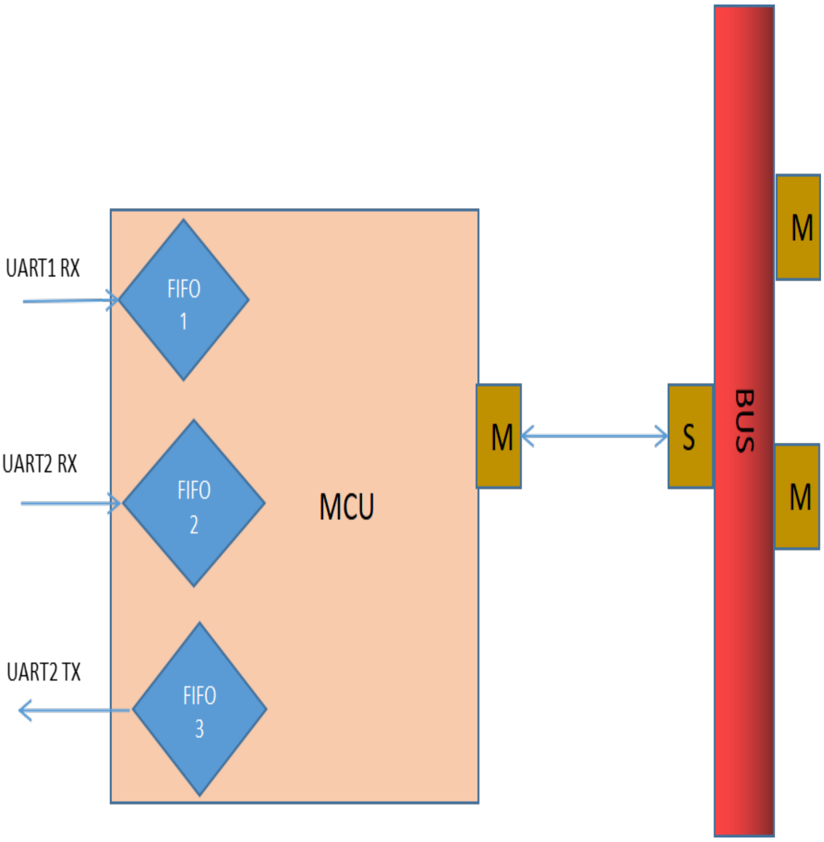

Figure 4: *MCU-Bus top level*

MCU use one master socket to communicate with the bus, where the bus use slave socket for the communication. MCU communicates with the GSM and GPS by the mean of FIFO (which acts as UART) which is a interface between MCU and both GPS and GSM. Bus uses two master socket in the other side to communicate with the two memories. In order to test the primary module (Master + Bus), the sub modules GPS,GSM and memory are used which are considered as a test bench for the primary module.

The sub-module interface are :

1. GPS and GSM connects with Master
2. Bus connects with two memories

4.1 Master and Bus

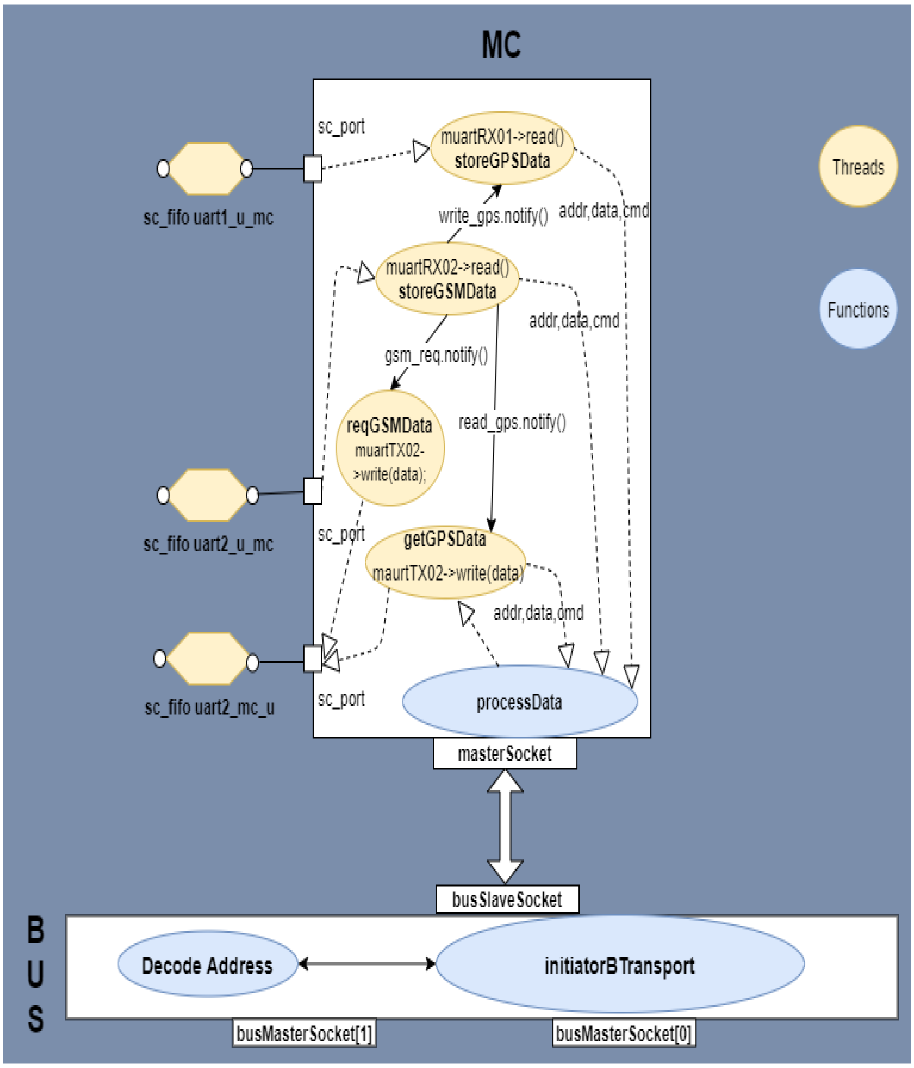

Figure 5: *Architecture of Master and bus*

The figure 5 show the complete architecture between the master and the bus, how the sockets and fifo's are incorporated in the primary module.

4.1.1 Bus

The Simple bus is a communication link between master and memory. Sockets and functions are used in bus which are shown in figure 5 and the explanations are given below.

Module	Bus
Type	Socket
Name	busSlaveSocket
Specification number	SPEC01
Description	Socket acts as slave socket in bus and the socket is connected to socket in master

Module	Bus
Type	Socket
Name	busMasterSocket[0]
Specification number	SPEC02
Description	Socket acts as master socket in bus and the socket is connected to slave 1 socket.

Module	Bus
Type	Socket
Name	busMasterSocket[1]
Specification number	SPEC03
Description	Socket acts as master socket in bus and the socket is connected to slave 2 socket.

Module	Bus
Type	Function
Name	void decode_address(const uint64 address, uint64& masked_address)
Specification number	SPEC05
Description	In this function port ID of the slave is determined from the upper 4 bytes[31-28] from 32-bit address and the address is determined from the lower 28 bytes[27-0].

Module	Bus
Type	Function
Name	void initiatorBTransport(int SocketId, payLoad_t& trans, sc_time& t)
Specification number	SPEC04
Description	The function initiatorBTransport() is registered as callback method to the slave socket. This will establish the communication link between the master CPU and memories. The first step in this function to get port ID of the slave and the address of the slave from the masked address which sent from the master CPU. Second step to send the data and address to the respective slave.

4.1.2 MCU

Master is responsible for read and write operation. When Master receives GPS and GSM data through port from respective module, master write the corresponding data to respective memory. If master receives a request to read GPS data from slave, it will read the data from the slave and forward data to GSM module through port. Usage of ports, sockets and functions as shown in figure 5 are explained below.

Module	Master
Type	Port
Name	muartRX01(template : unsigned char or int)
Specification number	SPEC06
Description	Port muartRX01 initialized as sc_fifo_in_if, which receives the GPS data from GPS module.

Module	Master
Type	Port
Name	muartRX02(template : unsigned char or int)
Specification number	SPEC07
Description	Port muartRX02 initialized as sc_fifo_in_if, which receives the GSM data to GSM module.

Module	Master
Type	Port
Name	muartTX02(template : unsigned char or int)
Specification number	SPEC08
Description	Port muartTX02 initialized as sc_fifo_out_if, which send the GPS data from GSM module.

Module	Master
Type	Socket
Name	masterSocket
Specification number	SPEC09
Description	The masterSocket is initiator socket, which is connected to the bus slave socket.

Module	Master
Type	Event
Name	read_gps
Specification number	SPEC10
Description	The event read_gps is triggered when the character 'x' is received from GSM module in function storeGSMData(). When event is triggered the GPS data read from the slave 1 and sent to GSM module through fifo in getGPSData().

Module	Master
Type	Event
Name	write_gps
Specification number	SPEC100
Description	The event write_gps is triggered when the character 'x' is received from GSM module in function storeGSMData(). When event is triggered the GPS data read from the fifo and sent to Memory1in storeGPSData().

Module	Master
Type	Event
Name	gsm_req
Specification number	SPEC110
Description	he event gsm_gps is triggered when the character ‘x’ is received from GSM module in function storeGSMData(). When event is triggered the master sends request(character ‘!’) to send GSM data to GSM module in reqGSMData().

Module	Master
Type	Function
Name	void processData(uint32_t address, char* data, int cmd)
Specification number	SPEC13
Description	Function receives the address, data and command as an input parameters. In this function generic payload is filled and sent through the socket b_transport.

Module	Master
Type	Thread
Name	void storeGPSData(void)
Specification number	SPEC12
Description	The storeGPSData() is registered as a SC_THREAD in the constructor. In this thread, wait until the write_gps event is occurred. Once event is notified data is read from the port muartRX01 and sent to memory1.

Module	Master
Type	Thread
Name	void storeGSMData(void)
Specification number	SPEC13
Description	The storeGSMData() is registered as a SC_THREAD in the constructor. The data is read from the port muartRX02 and sent to memory2 if character ‘x’ is not received. If ‘x’ is received, notify the read_gps, write_gps and req_gsm event. event.

Module	Master
Type	Thread
Name	void getGPSData(void)
Specification number	SPEC14
Description	The getGPSData() is registered as a SC_THREAD in the constructor. In this function, wait until the read_gps event is occurred. Once event is notified read GPS data from the memory 1 and sent to GSM module through port.

Module	Master
Type	Thread
Name	void reqGSMData(void)
Specification number	SPEC120
Description	The reqGSMData() is registered as a SC_THREAD in the constructor. In this thread wait unit req_gsm is occurred. Once event is triggered the master requests GSM module to send GSM data by sending character '!'.

4.2 Testing primary module

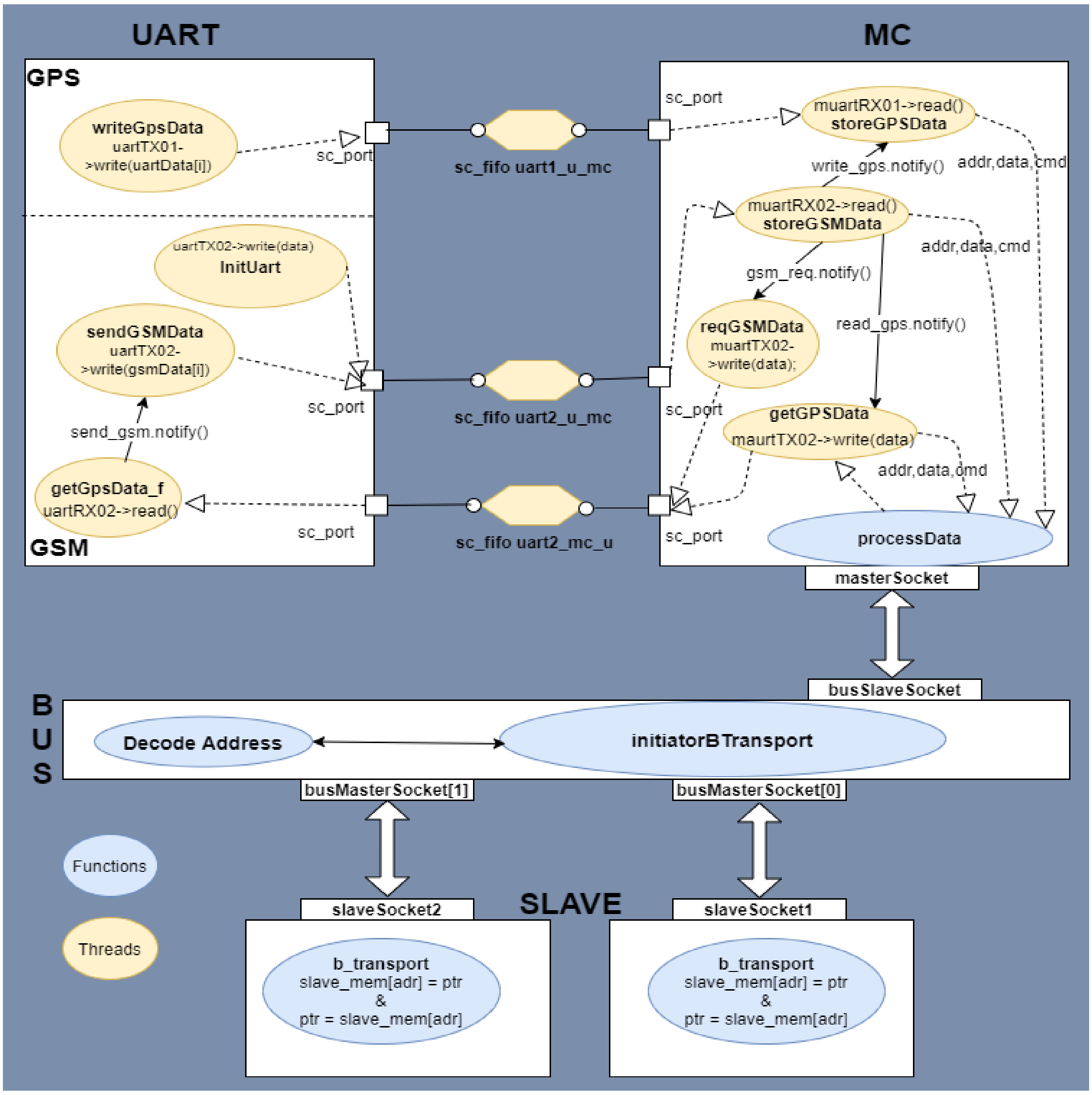
UART
MC
GPS
writeGpsData uartTX01->write(uartData[i])
sc_port
sc_fifo uart1_u_mc
sc_port
muartRX01->read() storeGPSData
write_gps.notify()
addr,data,cmd
uartTX02->write(data) InitUart
muartRX02->read() storeGSMData
addr,data,cmd
gsm_req.notify()
sendGSMData uartTX02->write(gsmData[i])
reqGSMData muartTX02->write(data);
read_gps.notify()
sc_fifo uart2_u_mc
sc_port
sc_port
send_gsm.notify()
getGPSData maurtTX02->write(data)
addr,data,cmd
getGpsData_f uartRX02->read()
sc_fifo uart2_mc_u
sc_port
sc_port
GSM
processData
masterSocket
busSlaveSocket
BUS
Decode Address
initiatorBTransport
busMasterSocket[1]
busMasterSocket[0]
Functions
Threads
SLAVE
slaveSocket2
slaveSocket1
b_transport slave_mem[adr] = ptr & ptr = slave_mem[adr]
b_transport slave_mem[adr] = ptr & ptr = slave_mem[adr]

Figure 6: *Architecture of complete primary and sub-module*

4.2.1 UART

UART module is a testing module where GPS and GSM data are predefined and sent to master through port and GSM receives the GPS data through port from Master. The explanation for the sockets and the functions which are shown in figure 6 are given below.

Module	UART
Type	Port
Name	uartTX01(template : unsigned int or char)
Specification number	SPEC15
Description	Port uartTX01 initialized as sc_fifo_out_if in GPS module, which sends the GPS data from GPS module to master CPU.

Module	UART
Type	Port
Name	uartTX02(template : unsigned int or char)
Specification number	SPEC16
Description	Port uartTX02 initialized as sc_fifo_out_if in GSM module, which sends the GSM data from GSM module to master CPU.

Module	UART
Type	Port
Name	uartRX02(template : unsigned int or char)
Specification number	SPEC17
Description	Port uartRX02 initialized as sc_fifo_in_if in GSM module, which receive the GSM data from master CPU.

Module	Master
Type	Event
Name	send_gsm
Specification number	SPEC130
Description	The send_gsm is triggered when character '!' is received in GSM module in getGpsData() thread. The GSM data is sent to master through fifo in sendGSMData().

Module	UART
Type	Thread
Name	void initUart(void)
Specification number	SPEC18
Description	The initUart() is registered as a SC_THREAD in the constructor. Tracking is started in this thread by sending character 'x' to master thread and simulation is stopped after tracking.

Module	UART
Type	Thread
Name	void getGpsData_f(void)
Specification number	SPEC19
Description	The getGpsData_f() is registered as a SC_THREAD in the constructor. If Character '!' is received from master, send_gsm event is notified to send the GSM data otherwise the data is considered as gps data and printed on terminal. .

Module	UART
Type	Thread
Name	void sendGSMData(void)
Specification number	SPEC140
Description	The sendGSMData() is registered as a SC_THREAD in the constructor. In this thread, wait unit the send_gsm event is notified and send the GSM data to master once event is notified.

Module	UART
Type	Thread
Name	void writeGpsData(void)
Specification number	SPEC150
Description	The WriteGpsData() is registered as a SC_THREAD in the constructor. This thread sends the GPS data to MCU with certain interval.

4.2.2 Memory

Memory is consider to be a slave, in write operation it stores the data and in the read operation copies the data back to payload. The explanation for the sockets and the functions which are shown in figure 6 are given below.

Module	Memory
Type	Socket
Name	slaveSocket
Specification number	SPEC20
Description	Slave socket is target socket, where it is connected to the target socket in the master CPU.

Module	Memory
Type	Function
Name	b_transport(tlm::tlm_generic_payload &trans, sc_time &delay)
Specification number	SPEC21
Description	The function b_transport is registered as callback method to the slave socket. In this function the read or write command is checked. If command is received as read, the data is copied from memory to payload or vice-versa.

5 Data Flow

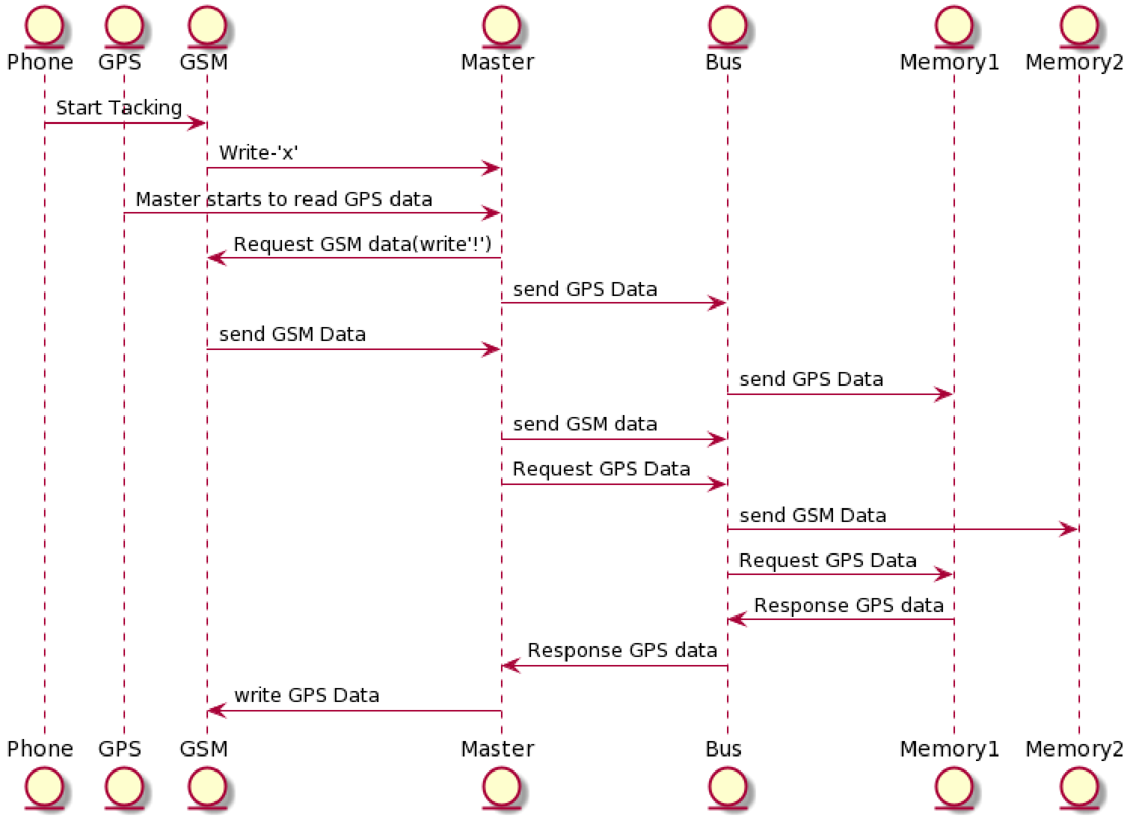

Figure 7: *Data transitions between modules*

Steps followed :

1.The data read operation is initiated by the processor if it receives the track event (x) from the phone via GSM.
2.Master reads the GPS data from the GPS module only if master receives the track request.
3.Similarly Master sends character '!' to GSM, indicates request for the network base station data.
4.Master reads the GSM network base station data.
5.Master sends both GPS and GSM data to the BUS.
6.BUS sends GPS data to memory 1 and GSM data to memory 2.
7.Master receives GPS data from the memory1 and sends back to the phone via GSM.

6 Requirement and specification traceability

Requirement ID number	Specification number
G01	[SPEC150]
G02	[SPEC140]
Mem01	[SPEC09] [SPEC01] [SPEC02] [SPEC20]
Mem02	[SPEC09] [SPEC01] [SPEC03] [SPEC20]
UART01	[SPEC15] [SPEC06]
UART02 UART02	[SPEC16] [SPEC07] [SPEC08] [SPEC17]
Bus01	[SPEC09] [SPEC01] [SPEC02] [SPEC03]