

MBSE WATERS CHALLENGE 2019

Generated by Doxygen 1.8.13

Sat May 16 2020 11:24:30

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	detectObject_t Struct Reference	5
3.1.1	Member Data Documentation	5
3.1.1.1	bboxDeviceDetection	5
3.1.1.2	bboxHostDetection	5
3.1.1.3	imageDeviceDetection	6
3.1.1.4	imageHostDetection	6
3.2	plannerData_t Struct Reference	6
3.2.1	Member Data Documentation	7
3.2.1.1	bBoxHost	7
3.2.1.2	canData	7
3.2.1.3	laneBoundary	7
3.2.1.4	matrixSFM	7
3.2.1.5	occupancyGrid	7
3.2.1.6	speedObjective	7
3.2.1.7	steerObjective	8
3.2.1.8	velocity	8
3.2.1.9	xCar	8
3.2.1.10	yawCar	8
3.2.1.11	yawRate	8
3.2.1.12	yCar	8
3.3	sfmData_t Struct Reference	9
3.3.1	Member Data Documentation	9
3.3.1.1	imageSFMHost	9
3.3.1.2	matrixSFMHost	9

4 File Documentation	11
4.1 inc/mbse.h File Reference	11
4.1.1 Detailed Description	13
4.1.2 Macro Definition Documentation	13
4.1.2.1 _GNU_SOURCE	13
4.1.2.2 MBSE_NUMBER_OF_THREADS	13
4.1.2.3 MBSE_THREAD_STACK_SIZE	13
4.1.2.4 MICRO_SECONDS	13
4.1.2.5 MILLI_SECONDS	14
4.1.2.6 NSEC_PER_SEC	14
4.1.2.7 SECONDS	14
4.1.3 Function Documentation	14
4.1.3.1 computeOSOverhead()	14
4.1.3.2 computeSpeedAndSteer()	14
4.1.3.3 error()	16
4.1.3.4 objDetectGetObject()	16
4.1.3.5 objDetectStructureFromMotion()	17
4.1.3.6 pathPlannerCalculation()	17
4.1.3.7 pathPlannerCanBusPolling()	17
4.1.3.8 shmemReadDetectionDataOutLabel()	19
4.1.3.9 shmemReadGridDataBufferLabel()	19
4.1.3.10 shmemReadLaneBoundaryBufferLabel()	20
4.1.3.11 shmemReadPlannerBufferLabel()	20
4.1.3.12 shmemReadSFMDDataOutLabel()	20
4.1.3.13 shmemReadSFMDetectionDataInLabel()	21
4.1.3.14 shmemWriteDetectionDataOutLabel()	21
4.1.3.15 shmemWriteGridDataBufferLabel()	22
4.1.3.16 shmemWriteLaneBoundaryBufferLabel()	22
4.1.3.17 shmemWritePlannerDataOutLabel()	22
4.1.3.18 shmemWriteSFMDDataOutLabel()	23

4.1.3.19	shmemWriteSFMDetectionDataInLabel()	23
4.1.3.20	utilAddDelay()	24
4.1.3.21	utilSetThreadPriority()	24
4.2	inc/mbseCuda.h File Reference	25
4.2.1	Detailed Description	26
4.2.2	Macro Definition Documentation	26
4.2.2.1	minVal	26
4.2.3	Typedef Documentation	26
4.2.3.1	detectObject	27
4.2.3.2	plannerData	27
4.2.3.3	sfmData	27
4.2.4	Function Documentation	27
4.2.4.1	addOSOverhead()	27
4.2.4.2	cuDetectObject()	27
4.2.4.3	cuObjDetectSFM()	28
4.2.4.4	cuPlannerFetchCanBusData()	28
4.2.4.5	cuPlannerInterpolatePath()	29
4.2.4.6	cuProcessDASM()	29
4.2.4.7	getCudaDeviceProperties()	30
4.3	src/computeSpeedAndSteer.c File Reference	30
4.3.1	Detailed Description	30
4.3.2	Function Documentation	31
4.3.2.1	computeOSOverhead()	31
4.3.2.2	computeSpeedAndSteer()	31
4.4	src/cuDASM.cu File Reference	31
4.4.1	Detailed Description	32
4.4.2	Function Documentation	33
4.4.2.1	addOSOverhead()	33
4.4.2.2	computeDASM()	33
4.4.2.3	cuProcessDASM()	33

4.4.2.4	osOverhead()	34
4.5	src/cudaUtils.cu File Reference	34
4.5.1	Detailed Description	35
4.5.2	Function Documentation	35
4.5.2.1	getCudaDeviceProperties()	35
4.6	src/cuObjDetection.cu File Reference	35
4.6.1	Detailed Description	36
4.6.2	Function Documentation	36
4.6.2.1	cuDetectObject()	36
4.6.2.2	cuObjDetectSFM()	37
4.6.2.3	processImage()	37
4.6.2.4	processSFMData()	37
4.7	src/cuPathPlanner.cu File Reference	38
4.7.1	Detailed Description	38
4.7.2	Function Documentation	39
4.7.2.1	cuPlannerFetchCanBusData()	39
4.7.2.2	cuPlannerInterpolatePath()	39
4.7.2.3	getCanBusData()	40
4.7.2.4	pathPlan()	40
4.8	src/main.c File Reference	40
4.8.1	Detailed Description	41
4.8.2	Typedef Documentation	41
4.8.2.1	threadPool_t	41
4.8.3	Function Documentation	41
4.8.3.1	main()	41
4.9	src/objectDetection.c File Reference	41
4.9.1	Detailed Description	42
4.9.2	Function Documentation	42
4.9.2.1	objDetectGetObject()	42
4.9.2.2	objDetectStructureFromMotion()	43

4.10	src/pathPlanner.c File Reference	43
4.10.1	Detailed Description	44
4.10.2	Function Documentation	44
4.10.2.1	pathPlannerCalculation()	44
4.10.2.2	pathPlannerCanBusPolling()	45
4.11	src/sharedMemory.c File Reference	45
4.11.1	Detailed Description	46
4.11.2	Macro Definition Documentation	46
4.11.2.1	DATA_GRID_BUFFER	47
4.11.2.2	DATA_IN_SFMDetection_BUFFER	47
4.11.2.3	DATA_LANE_BOUNDARY_BUFFER	47
4.11.2.4	DATA_OUT_DETECTION_BUFFER	47
4.11.2.5	DATA_OUT_PLANNER_BUFFER	47
4.11.2.6	DATA_OUT_SFMD_BUFFER	47
4.11.3	Function Documentation	47
4.11.3.1	shmemReadDetectionDataOutLabel()	47
4.11.3.2	shmemReadGridDataBufferLabel()	48
4.11.3.3	shmemReadLaneBoundaryBufferLabel()	48
4.11.3.4	shmemReadPlannerBufferLabel()	49
4.11.3.5	shmemReadSFMDDataOutLabel()	49
4.11.3.6	shmemReadSFMDetectionDataInLabel()	49
4.11.3.7	shmemWriteDetectionDataOutLabel()	50
4.11.3.8	shmemWriteGridDataBufferLabel()	50
4.11.3.9	shmemWriteLaneBoundaryBufferLabel()	51
4.11.3.10	shmemWritePlannerDataOutLabel()	51
4.11.3.11	shmemWriteSFMDDataOutLabel()	52
4.11.3.12	shmemWriteSFMDetectionDataInLabel()	52
4.12	src/utls.c File Reference	52
4.12.1	Detailed Description	53
4.12.2	Function Documentation	53
4.12.2.1	error()	53
4.12.2.2	utilAddDelay()	54
4.12.2.3	utilSetThreadPriority()	54

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

detectObject_t	5
plannerData_t	6
sfmData_t	9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

inc/ mbse.h	This header file used for all C specific source files	11
inc/ mbseCuda.h	This header file is used for declaring all CUDA specific source files.	25
src/ computeSpeedAndSteer.c	This file contains the basic implementation of the DASM and OS overhead tasks	30
src/ cuDASM.cu	This file contains the CUDA kernel implementation of the DASM and OS overhead tasks . . .	31
src/ cudaUtils.cu	This file contains the CUDA utility functions	34
src/ cuObjDetection.cu	This file contains the CUDA kernel implementation of Detection and Structure- from-motion tasks	35
src/ cuPathPlanner.cu	This file contains the CUDA kernel implementation of Planner and CAN Bus Polling tasks . . .	38
src/ main.c	This file contains the entry point of the applications and consists of the initialization of memory, threads of the application	40
src/ objectDetection.c	This file contains the basic implementation of the Detection and Structure from motion tasks .	41
src/ pathPlanner.c	This file contains the basic implementation of the Planner task and CAN Bus polling task . . .	43
src/ sharedMemory.c	This file declares and implement the read and write operation of all the shared memory	45
src/ utils.c	This file declares and implement the utility functions used in the application	52

Chapter 3

Class Documentation

3.1 detectObject_t Struct Reference

```
#include <mbseCuda.h>
```

Public Attributes

- int [bboxDeviceDetection](#)
Boundary box device detection.
- int [bboxHostDetection](#)
Boundary box host detection.
- int [imageHostDetection](#)
Image host detection.
- int [imageDeviceDetection](#)
Image device detection.

3.1.1 Member Data Documentation

3.1.1.1 bboxDeviceDetection

```
int detectObject_t::bboxDeviceDetection
```

Boundary box device detection.

3.1.1.2 bboxHostDetection

```
int detectObject_t::bboxHostDetection
```

Boundary box host detection.

3.1.1.3 imageDeviceDetection

```
int detectObject_t::imageDeviceDetection
```

Image device detection.

3.1.1.4 imageHostDetection

```
int detectObject_t::imageHostDetection
```

Image host detection.

The documentation for this struct was generated from the following file:

- [inc/mbseCuda.h](#)

3.2 plannerData_t Struct Reference

```
#include <mbseCuda.h>
```

Public Attributes

- int [canData](#)
Can polling data.
- int [yawRate](#)
Yaw rate in x-direction.
- int [velocity](#)
Velocity of the vehicle.
- int [xCar](#)
Position of the vehicle on x co-ordinate.
- int [yawCar](#)
Yaw rate in y-direction.
- int [yCar](#)
Position of the vehicle on y co-ordinate.
- int [matrixSFM](#)
Input SFM Data.
- int [bBoxHost](#)
Input detection data.
- int [occupancyGrid](#)
Occupancy Grid.
- int [laneBoundary](#)
Lane boundary.
- int [steerObjective](#)
Steer control data.
- int [speedObjective](#)

3.2.1 Member Data Documentation

3.2.1.1 bBoxHost

```
int plannerData_t::bBoxHost
```

Input detection data.

3.2.1.2 canData

```
int plannerData_t::canData
```

Can polling data.

3.2.1.3 laneBoundary

```
int plannerData_t::laneBoundary
```

Lane boundary.

3.2.1.4 matrixSFM

```
int plannerData_t::matrixSFM
```

Input SFM Data.

3.2.1.5 occupancyGrid

```
int plannerData_t::occupancyGrid
```

Occupancy Grid.

3.2.1.6 speedObjective

```
int plannerData_t::speedObjective
```

3.2.1.7 steerObjective

```
int plannerData_t::steerObjective
```

Steer control data.

3.2.1.8 velocity

```
int plannerData_t::velocity
```

Velocity of the vehicle.

3.2.1.9 xCar

```
int plannerData_t::xCar
```

Position of the vehicle on x co-ordinate.

3.2.1.10 yawCar

```
int plannerData_t::yawCar
```

Yaw rate in y-direction.

3.2.1.11 yawRate

```
int plannerData_t::yawRate
```

Yaw rate in x-direction.

3.2.1.12 yCar

```
int plannerData_t::yCar
```

Position of the vehicle on y co-ordinate.

The documentation for this struct was generated from the following file:

- [inc/mbseCuda.h](#)

3.3 sfmData_t Struct Reference

```
#include <mbseCuda.h>
```

Public Attributes

- int [matrixSFMHost](#)
SFM matrix host data.
- int [imageSFMHost](#)
SFM Image host data.

3.3.1 Member Data Documentation

3.3.1.1 imageSFMHost

```
int sfmData_t::imageSFMHost
```

SFM Image host data.

3.3.1.2 matrixSFMHost

```
int sfmData_t::matrixSFMHost
```

SFM matrix host data.

The documentation for this struct was generated from the following file:

- inc/[mbseCuda.h](#)

Chapter 4

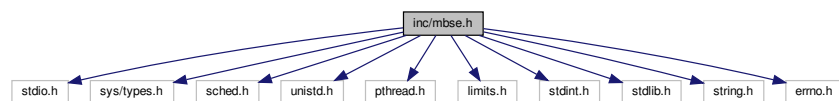
File Documentation

4.1 inc/mbse.h File Reference

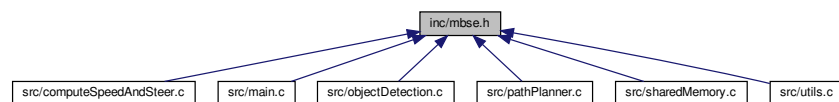
This header file used for all C specific source files.

```
#include <stdio.h>
#include <sys/types.h>
#include <sched.h>
#include <unistd.h>
#include <pthread.h>
#include <limits.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
```

Include dependency graph for mbse.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define _GNU_SOURCE`
- `#define MBSE_NUMBER_OF_THREADS 6`
- `#define MBSE_THREAD_STACK_SIZE (100 * 1024) /* 100 kB is enough for now. */`
- `#define NSEC_PER_SEC (1000 * 1000 * 1000)`
- `#define MICRO_SECONDS 1000`
- `#define MILLI_SECONDS (MICRO_SECONDS * 1000)`
- `#define SECONDS (MILLI_SECONDS * 1000)`

Functions

- void `error` (int at)
Function to print error message thrown from a particular point in application code.
- void `utilAddDelay` (uint32_t ms, struct timespec *deadline)
Function to add delay to the task.
- void `utilSetThreadPriority` (pthread_t threadId, int prio)
Function to set the thread priority.
- void * `pathPlannerCanBusPolling` (void *args)
This task is used perform the CAN Bus Polling task functionality.
- void * `pathPlannerCalculation` (void *args)
This task is used perform the Planner task functionality.
- void * `objDetectGetObject` (void *args)
This task is used perform the detection functionality.
- void * `objDetectStructureFromMotion` (void *args)
This task is used perform the Structure-From-Motion functionality.
- void * `computeSpeedAndSteer` (void *args)
This function implements the DASM task.
- void * `computeOSOverhead` (void *args)
This task is used to add the OS overhead to the overall tasks based on the Amalthea task model.
- int `shmemReadPlannerBufferLabel` (unsigned int index)
Function to read to the Planner Output data buffer.
- int `shmemReadLaneBoundaryBufferLabel` (unsigned int index)
Function to read to the lane boundary data buffer.
- int `shmemReadGridDataBufferLabel` (unsigned int index)
Function to read to the data grid buffer.
- void `shmemWritePlannerDataOutLabel` (int offset, int size, int data)
Function to write to the Planner data buffer.
- void `shmemWriteSFMDetectionDataInLabel` (int offset, int size, int data)
Function to write to the SFM and detection Input buffer.
- void `shmemWriteGridDataBufferLabel` (int offset, int size, int data)
Function to write to the data grid buffer.
- void `shmemWriteLaneBoundaryBufferLabel` (int offset, int size, int data)
Function to write to the lane boundary buffer.
- void `shmemWriteDetectionDataOutLabel` (int offset, int size, void *data)
Function to write to the Detection data buffer.
- int `shmemReadSFMDetectionDataInLabel` (unsigned int index)
Function to read to the Detection and SFM Input data buffer.
- void `shmemWriteSFMDDataOutLabel` (int offset, int size, void *data)
Function to write to the SFM data buffer.
- int `shmemReadSFMDDataOutLabel` (unsigned int index)
Function to read to the SFM Output data buffer.
- int `shmemReadDetectionDataOutLabel` (unsigned int index)
Function to read to the Detection Output data buffer.

4.1.1 Detailed Description

This header file used for all C specific source files.

Author

Anand Prakash

Date

22 April 2020

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>↵

4.1.2 Macro Definition Documentation

4.1.2.1 _GNU_SOURCE

```
#define _GNU_SOURCE
```

4.1.2.2 MBSE_NUMBER_OF_THREADS

```
#define MBSE_NUMBER_OF_THREADS 6
```

4.1.2.3 MBSE_THREAD_STACK_SIZE

```
#define MBSE_THREAD_STACK_SIZE (100 * 1024) /* 100 kB is enough for now. */
```

4.1.2.4 MICRO_SECONDS

```
#define MICRO_SECONDS 1000
```

4.1.2.5 MILLI_SECONDS

```
#define MILLI_SECONDS (MICRO_SECONDS * 1000)
```

4.1.2.6 NSEC_PER_SEC

```
#define NSEC_PER_SEC (1000 * 1000 * 1000)
```

4.1.2.7 SECONDS

```
#define SECONDS (MILLI_SECONDS * 1000)
```

4.1.3 Function Documentation

4.1.3.1 computeOSOverhead()

```
void* computeOSOverhead (
    void * args )
```

This task is used to add the OS overhead to the overall tasks based on the Amalthea task model.

This task adds an additional overhead to the overall application. It is used to simulate the overhead that occur in real scenario. It is executed on core number 3 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

4.1.3.2 computeSpeedAndSteer()

```
void* computeSpeedAndSteer (
    void * args )
```

This function implements the DASM task.

This task computes and establishes the speed and steer that must be effectively employed from the information that is provided by the Path Planner task. It is executed on core number 3 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

<i>in</i>	<i>args</i>	Optional argument. Currently not in use.
-----------	-------------	--

Returns

void

4.1.3.3 error()

```
void error (
    int at )
```

Function to print error message thrown from a particular point in application code.

This function prints the error code from which the error was thrown.

Parameters

<i>in</i>	<i>at</i>	Error code.
-----------	-----------	-------------

Returns

void

4.1.3.4 objDetectGetObject()

```
void* objDetectGetObject (
    void * args )
```

This task is used perform the detection functionality.

This task is responsible of detecting and classifying the objects in the road. All the objects detected are visualized and the information produced is sent to the Planner task. It is executed on core number 0 on Jetson TX2 ARM A57 core with the thread priority of 98.

Parameters

<i>in</i>	<i>args</i>	Optional argument. Currently not in use.
-----------	-------------	--

Returns

void

4.1.3.5 objDetectStructureFromMotion()

```
void* objDetectStructureFromMotion (
    void * args )
```

This task is used perform the Structure-From-Motion functionality.

Structure-From-Motion is a method for estimating 3-D structures (depth) from vehicle motion and sequences of 2-D images. This task returns a matrix of points representing the distance with respect the objects of the image. It is executed on core number 0 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

4.1.3.6 pathPlannerCalculation()

```
void* pathPlannerCalculation (
    void * args )
```

This task is used perform the Planner task functionality.

The main purpose of this component is to define and follow a given trajectory. This trajectory is defined as a spline, that is, a line built through polynomial interpolation at times on the map that represents at each point the position and orientation that the car will have to follow. The planner sends the goal state of the vehicle (i.e., target steer and speed) to the DASM task that is in charge of writing the commands in the CAN line the effective steer and speed to apply. It is executed on core number 5 on Jetson TX2 ARM A57 core with the thread priority of 98.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

4.1.3.7 pathPlannerCanBusPolling()

```
void* pathPlannerCanBusPolling (
    void * args )
```

This task is used perform the CAN Bus Polling task functionality.

This task snoops the key vehicle information (steer/wheel/break/acceleration status...) from the on-board CAN bus and sends it to the Localization, Planner and EKF tasks. It is executed on core number 5 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	<i>args</i>	Optional argument. Currently not in use.
----	-------------	--

Returns

void

4.1.3.8 shmemReadDetectionDataOutLabel()

```
int shmemReadDetectionDataOutLabel (  
    unsigned int index )
```

Function to read to the Detection Output data buffer.

This function read the Detection output data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.1.3.9 shmemReadGridDataBufferLabel()

```
int shmemReadGridDataBufferLabel (  
    unsigned int index )
```

Function to read to the data grid buffer.

This function read the data grid buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.1.3.10 shmemReadLaneBoundaryBufferLabel()

```
int shmemReadLaneBoundaryBufferLabel (
    unsigned int index )
```

Function to read to the lane boundary data buffer.

This function read the lane boundary data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.1.3.11 shmemReadPlannerBufferLabel()

```
int shmemReadPlannerBufferLabel (
    unsigned int index )
```

Function to read to the Planner Output data buffer.

This function read the Planner output data buffer. This buffer is used as an input to DASM task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.1.3.12 shmemReadSFMDDataOutLabel()

```
int shmemReadSFMDDataOutLabel (
    unsigned int index )
```

Function to read to the SFM Output data buffer.

This function read the SFM output data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.1.3.13 shmemReadSFMDetectionDataInLabel()

```
int shmemReadSFMDetectionDataInLabel (
    unsigned int index )
```

Function to read to the Detection and SFM Input data buffer.

This function read to the Detection and SFM Input data buffer. This buffer is used as an input to SFM and Detection task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.1.3.14 shmemWriteDetectionDataOutLabel()

```
void shmemWriteDetectionDataOutLabel (
    int offset,
    int size,
    void * data )
```

Function to write to the Detection data buffer.

This function writes to the Detection data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.1.3.15 shmemWriteGridDataBufferLabel()

```
void shmemWriteGridDataBufferLabel (
    int offset,
    int size,
    int data )
```

Function to write to the data grid buffer.

This function writes to the data grid buffer. This buffer is used as an input to Planner task

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.1.3.16 shmemWriteLaneBoundaryBufferLabel()

```
void shmemWriteLaneBoundaryBufferLabel (
    int offset,
    int size,
    int data )
```

Function to write to the lane boundary buffer.

This function writes to the lane boundary buffer. This buffer is used as an input to Planner task

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.1.3.17 shmemWritePlannerDataOutLabel()

```
void shmemWritePlannerDataOutLabel (
    int offset,
```

```
int size,  
int data )
```

Function to write to the Planner data buffer.

This function writes to the Planner data buffer. This buffer is used as an input to DASM task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.1.3.18 shmemWriteSFMDDataOutLabel()

```
void shmemWriteSFMDDataOutLabel (  
    int offset,  
    int size,  
    void * data )
```

Function to write to the SFM data buffer.

This function writes to the SFM data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.1.3.19 shmemWriteSFMDetectionDataInLabel()

```
void shmemWriteSFMDetectionDataInLabel (  
    int offset,  
    int size,  
    int data )
```

Function to write to the SFM and detection Input buffer.

This function writes to the SFM and detection Input buffer. This buffer is used as an input to SFM and detection task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.1.3.20 utilAddDelay()

```
void utilAddDelay (
    uint32_t ms,
    struct timespec * deadline )
```

Function to add delay to the task.

This function adds delay to the task by sleeping for the time provided in the argument in terms of millisecond.

Parameters

in	<i>ms</i>	Time in millisecond for which the thread needs to sleep.
in, out	<i>deadline</i>	Pointer to the timespec for the next deadline

Returns

void

4.1.3.21 utilSetThreadPriority()

```
void utilSetThreadPriority (
    pthread_t threadId,
    int customPrio )
```

Function to set the thread priority.

This task sets the thread priority based on the threadId and the customPrio provided The priority of the thread is set to max priority minus the custom priority

Parameters

in	<i>threadId</i>	Thread ID whose priority needs to be set.
in	<i>customPrio</i>	Priority offset of the thread from the maximum priority

Returns

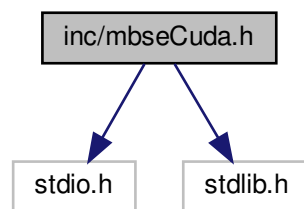
void

4.2 inc/mbseCuda.h File Reference

This header file is used for declaring all CUDA specific source files..

```
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for mbseCuda.h:



This graph shows which files directly or indirectly include this file:

**Classes**

- struct [detectObject_t](#)
- struct [sfmData_t](#)
- struct [plannerData_t](#)

Macros

- #define [minVal](#)(a, b) ((a > b) ? b : a)

Typedefs

- typedef struct [detectObject_t](#) [detectObject](#)
- typedef struct [sfmData_t](#) [sfmData](#)
- typedef struct [plannerData_t](#) [plannerData](#)

Functions

- void [cuDetectObject](#) (const char *function, [detectObject](#) *objdetected)
Function to process the Detection task.
- void [cuObjDetectSFM](#) (const char *func, [sfmData](#) *sfmInputData)
Function to process the SFM task.
- void [cuPlannerFetchCanBusData](#) (const char *func, int *hostCanPollingData)
Function to process the CAN bus polling task.
- void [cuPlannerInterpolatePath](#) (const char *func, [plannerData](#) *data)
Function to process the Planner task.
- void [cuProcessDASM](#) (const char *func, int *steer, int *speed)
Function to process the DASM task.
- void [addOSOverhead](#) (const char *func)
Function to process the OS overhead task.
- void [getCudaDeviceProperties](#) (void)
Function to get the CUDA device properties.

4.2.1 Detailed Description

This header file is used for declaring all CUDA specific source files..

Author

Anand Prakash

Date

22 April 2020

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>↵

4.2.2 Macro Definition Documentation

4.2.2.1 minVal

```
#define minVal(  
    a,  
    b ) ((a > b) ? b : a)
```

4.2.3 Typedef Documentation

4.2.3.1 detectObject

```
typedef struct detectObject_t detectObject
```

4.2.3.2 plannerData

```
typedef struct plannerData_t plannerData
```

4.2.3.3 sfmData

```
typedef struct sfmData_t sfmData
```

4.2.4 Function Documentation

4.2.4.1 addOSOverhead()

```
void addOSOverhead (
    const char * func )
```

Function to process the OS overhead task.

The functions adds some OS overhead to simulate the real scenario.

Parameters

<i>func[in]</i>	Function name
-----------------	---------------

Returns

void

4.2.4.2 cuDetectObject()

```
void cuDetectObject (
    const char * func,
    detectObject * objdetected )
```

Function to process the Detection task.

Function to detect the object and process the image. The output of this function is provided to the pathPlanner for further processing. It has three runnables.

Parameters

in	<i>func</i>	Function name
in, out	<i>objdetected</i>	Pointer to structure to detectObject input data

Returns

void

4.2.4.3 cuObjDetectSFM()

```
void cuObjDetectSFM (
    const char * func,
    sfmData * sfmInput )
```

Function to process the SFM task.

The functions process the data received from the input buffer and generates the input data for the planner task. It has three runnables.

Parameters

in	<i>func</i>	Function name
in, out	<i>sfmInput</i>	Pointer to structure to SFM input data

Returns

void

4.2.4.4 cuPlannerFetchCanBusData()

```
void cuPlannerFetchCanBusData (
    const char * func,
    int * hostCanPollingData )
```

Function to process the CAN bus polling task.

The functions process the data obtained from the global buffer. It provides this value as n input to Planner task.

Parameters

in	<i>func</i>	Function name
in, out	<i>hostCanPollingData</i>	Pointer to host can polling data.

Returns

void

4.2.4.5 cuPlannerInterpolatePath()

```
void cuPlannerInterpolatePath (
    const char * func,
    plannerData * data )
```

Function to process the Planner task.

The functions process the data received from the SFM, Detection, Can BUs Polling, grid data and lane boundary detection and process the data to generate the input to the DASM task.

Parameters

in	<i>func</i>	Function name
in, out	<i>data</i>	Pointer to structure to planner input data

Returns

void

4.2.4.6 cuProcessDASM()

```
void cuProcessDASM (
    const char * func,
    int * steer,
    int * speed )
```

Function to process the DASM task.

The functions process the data received from the planner task and generate the steer and speed output. The task consists of one runnable.

Parameters

<i>func[in]</i>	Function name
<i>steer[inout]</i>	Pointer to input steer from planner. Data is modified after processing.
<i>speed[inout]</i>	Pointer to input speed from planner. Data is modified after processing.

Returns

void

4.2.4.7 getCudaDeviceProperties()

```
void getCudaDeviceProperties (
    void )
```

Function to get the CUDA device properties.

The functions get the CUDA device count and prints the device properties of each CUDA device.

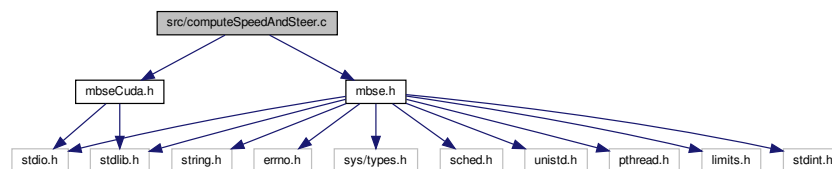
Returns

void

4.3 src/computeSpeedAndSteer.c File Reference

This file contains the basic implementation of the DASM and OS overhead tasks.

```
#include "mbse.h"
#include "mbseCuda.h"
Include dependency graph for computeSpeedAndSteer.c:
```



Functions

- void * [computeSpeedAndSteer](#) (void *args)
This function implements the DASM task.
- void * [computeOSOverhead](#) (void *args)
This task is used to add the OS overhead to the overall tasks based on the Amalthea task model.

4.3.1 Detailed Description

This file contains the basic implementation of the DASM and OS overhead tasks.

Author

Anand Prakash

Date

12 May 2020 This task computes and establishes the speed and steer that must be effectively employed from the information that is provided by the Path Planner task. It also implements a task which introduces the OS overhead that may occur during the execution.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.3.2 Function Documentation

4.3.2.1 computeOSOverhead()

```
void* computeOSOverhead (
    void * args )
```

This task is used to add the OS overhead to the overall tasks based on the Amalthea task model.

This task adds an additional overhead to the overall application. It is used to simulate the overhead that occur in real scenario. It is executed on core number 3 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

4.3.2.2 computeSpeedAndSteer()

```
void* computeSpeedAndSteer (
    void * args )
```

This function implements the DASM task.

This task computes and establishes the speed and steer that must be effectively employed from the information that is provided by the Path Planner task. It is executed on core number 3 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

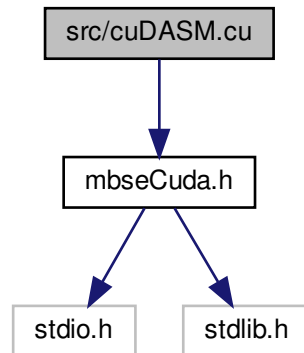
void

4.4 src/cuDASM.cu File Reference

This file contains the CUDA kernel implementation of the DASM and OS overhead tasks.

```
#include "mbseCuda.h"
```

Include dependency graph for cuDASM.cu:



Functions

- `__global__ void computeDASM (int *devSpeed, int *devSteer, int step, int size)`
Kernel that executes on the CUDA device to compute the speed and steer for DASM task.
- `void cuProcessDASM (const char *func, int *steer, int *speed)`
Function to process the DASM task.
- `__global__ void osOverhead (int *A, int *B, int *C, int size)`
Kernel that executes on the CUDA device the OS overhead task.
- `void addOSOverhead (const char *func)`
Function to process the OS overhead task.

4.4.1 Detailed Description

This file contains the CUDA kernel implementation of the DASM and OS overhead tasks.

Author

Anand Prakash

Date

12 May 2020 This file implements the runnables and used by the DASM and OS Overhead tasks.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.4.2 Function Documentation

4.4.2.1 addOSOverhead()

```
void addOSOverhead (
    const char * func )
```

Function to process the OS overhead task.

The functions adds some OS overhead to simulate the real scenario.

Parameters

<i>func[in]</i>	Function name
-----------------	---------------

Returns

void

4.4.2.2 computeDASM()

```
__global__ void computeDASM (
    int * devSpeed,
    int * devSteer,
    int step,
    int size )
```

Kernel that executes on the CUDA device to compute the speed and steer for DASM task.

4.4.2.3 cuProcessDASM()

```
void cuProcessDASM (
    const char * func,
    int * steer,
    int * speed )
```

Function to process the DASM task.

The functions process the data received from the planner task and generate the steer and speed output. The task consists of one runnable.

Parameters

<i>func[in]</i>	Function name
<i>steer[inout]</i>	Pointer to input steer from planner. Data is modified after processing.
<i>speed[inout]</i>	Pointer to input speed from planner. Data is modified after processing.

Returns

void

4.4.2.4 osOverhead()

```
__global__ void osOverhead (
    int * A,
    int * B,
    int * C,
    int size )
```

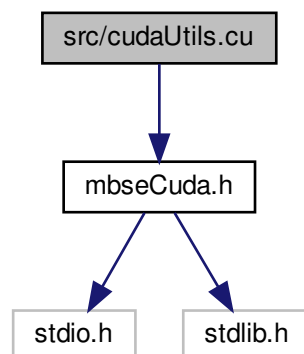
Kernel that executes on the CUDA device the OS overhead task.

4.5 src/cudaUtils.cu File Reference

This file contains the CUDA utility functions.

```
#include "mbseCuda.h"
```

Include dependency graph for cudaUtils.cu:

**Functions**

- void [getCudaDeviceProperties](#) (void)

Function to get the CUDA device properties.

4.5.1 Detailed Description

This file contains the CUDA utility functions.

Author

Anand Prakash

Date

01 May 2020 This file implements utility functions like getting the CUDA device properties. All the common CUDA specific implementation must be added in this file.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.5.2 Function Documentation

4.5.2.1 getCudaDeviceProperties()

```
void getCudaDeviceProperties (
    void )
```

Function to get the CUDA device properties.

The functions get the CUDA device count and prints the device properties of each CUDA device.

Returns

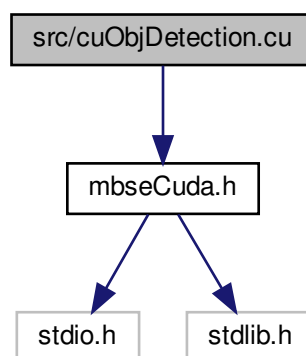
void

4.6 src/cuObjDetection.cu File Reference

This file contains the CUDA kernel implementation of Detection and Structure- from-motion tasks.

```
#include "mbseCuda.h"
```

Include dependency graph for cuObjDetection.cu:



Functions

- `__global__ void processImage` (int *hostBbox, int *devBbox, int *hostImage, int *devImage, int numElements)
CUDA Kernel Device code.
- void `cuDetectObject` (const char *func, `detectObject` *objdetected)
Function to process the Detection task.
- `__global__ void processSFMDData` (int *image, int *matrix, int nbin, int step, int nthreads, int nblocks)
Kernel that executes on the CUDA device to process the SFM data.
- void `cuObjDetectSFM` (const char *func, `sfmData` *sfmInput)
Function to process the SFM task.

4.6.1 Detailed Description

This file contains the CUDA kernel implementation of Detection and Structure- from-motion tasks.

Author

Anand Prakash

Date

17 April 2020 This file implements runnables executed for Detection and Structure-from-Motion tasks.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.6.2 Function Documentation

4.6.2.1 `cuDetectObject()`

```
void cuDetectObject (
    const char * func,
    detectObject * objdetected )
```

Function to process the Detection task.

Function to detect the object and process the image. The output of this function is provided to the pathPlanner for further processing. It has three runnables.

Parameters

in	<i>func</i>	Function name
in, out	<i>objdetected</i>	Pointer to structure to detectObject input data

Returns

void

4.6.2.2 cuObjDetectSFM()

```
void cuObjDetectSFM (
    const char * func,
    sfmData * sfmInput )
```

Function to process the SFM task.

The functions process the data received from the input buffer and generates the input data for the planner task. It has three runnables.

Parameters

in	func	Function name
in, out	sfmInput	Pointer to structure to SFM input data

Returns

void

4.6.2.3 processImage()

```
__global__ void processImage (
    int * hostBbox,
    int * devBbox,
    int * hostImage,
    int * devImage,
    int numElements )
```

CUDA Kernel Device code.

Runnable to Process the image to detect and classify the objects by creating the Boundary Box.

4.6.2.4 processSFMDData()

```
__global__ void processSFMDData (
    int * image,
    int * matrix,
    int nbin,
    int step,
    int nthreads,
    int nblocks )
```

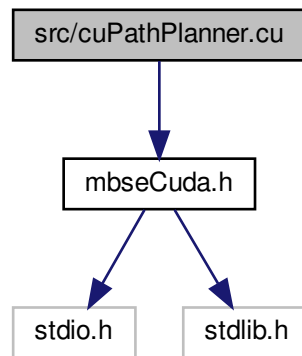
Kernel that executes on the CUDA device to process the SFM data.

4.7 src/cuPathPlanner.cu File Reference

This file contains the CUDA kernel implementation of Planner and CAN Bus Polling tasks.

```
#include "mbseCuda.h"
```

Include dependency graph for cuPathPlanner.cu:



Functions

- `__global__ void getCanBusData (int *canData, int size, int nthreads, int nblocks)`
Kernel that executes on the CUDA device to process the CAN Bus data.
- `void cuPlannerFetchCanBusData (const char *func, int *hostCanPollingData)`
Function to process the CAN bus polling task.
- `__global__ void pathPlan (int *devSpeed, int *devSteer, int size)`
Kernel that executes on the CUDA device to process the path planner task.
- `void cuPlannerInterpolatePath (const char *func, plannerData *data)`
Function to process the Planner task.

4.7.1 Detailed Description

This file contains the CUDA kernel implementation of Planner and CAN Bus Polling tasks.

Author

Anand Prakash

Date

12 May 2020 This file implements runnables executed for Planner and CAN Bus Polling tasks.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.7.2 Function Documentation

4.7.2.1 cuPlannerFetchCanBusData()

```
void cuPlannerFetchCanBusData (
    const char * func,
    int * hostCanPollingData )
```

Function to process the CAN bus polling task.

The functions process the data obtained from the global buffer. It provides this value as n input to Planner task.

Parameters

in	<i>func</i>	Function name
in, out	<i>hostCanPollingData</i>	Pointer to host can polling data.

Returns

void

4.7.2.2 cuPlannerInterpolatePath()

```
void cuPlannerInterpolatePath (
    const char * func,
    plannerData * data )
```

Function to process the Planner task.

The functions process the data received from the SFM, Detection, Can BUs Polling, grid data and lane boundary detection and process the data to generate the input to the DASM task.

Parameters

in	<i>func</i>	Function name
in, out	<i>data</i>	Pointer to structure to planner input data

Returns

void

4.7.2.3 getCanBusData()

```
__global__ void getCanBusData (
    int * canData,
    int size,
    int nthreads,
    int nblocks )
```

Kernel that executes on the CUDA device to process the CAN Bus data.

4.7.2.4 pathPlan()

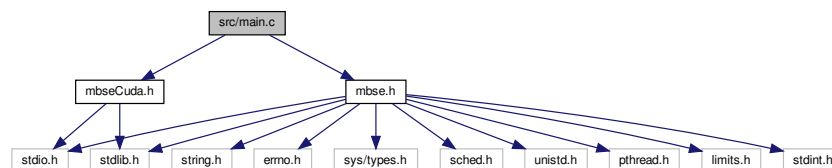
```
__global__ void pathPlan (
    int * devSpeed,
    int * devSteer,
    int size )
```

Kernel that executes on the CUDA device to process the path planner task.

4.8 src/main.c File Reference

This file contains the entry point of the applications and consists of the initialization of memory, threads of the application.

```
#include "mbse.h"
#include "mbseCuda.h"
Include dependency graph for main.c:
```



Typedefs

- typedef void [*\(* threadPool_t\)](#) (void *)

Functions

- int [main](#) (int argc, char *argv[])

4.8.1 Detailed Description

This file contains the entry point of the applications and consists of the initialization of memory, threads of the application.

Author

Anand Prakash

Date

11 April 2020 This is the entry point of the application. The application can be executed only with root/superuser privileges and get the CUDA device properties along with initializing the threads and start their execution. The tasks are executed on three ARM A57 cores and uses RMS scheduling approach.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.8.2 Typedef Documentation

4.8.2.1 threadPool_t

```
typedef void*(* threadPool_t) (void *)
```

4.8.3 Function Documentation

4.8.3.1 main()

```
int main (
    int argc,
    char * argv[] )
```

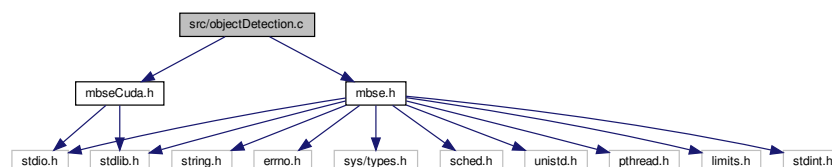
4.9 src/objectDetection.c File Reference

This file contains the basic implementation of the Detection and Structure from motion tasks.

```
#include "mbse.h"
```

```
#include "mbseCuda.h"
```

Include dependency graph for objectDetection.c:



Functions

- void * [objDetectGetObject](#) (void *args)
This task is used perform the detection functionality.
- void * [objDetectStructureFromMotion](#) (void *args)
This task is used perform the Structure-From-Motion functionality.

4.9.1 Detailed Description

This file contains the basic implementation of the Detection and Structure from motion tasks.

Author

Anand Prakash

Date

22 April 2020.

The detection task is responsible of detecting and classifying the objects in the road. It uses a machine learning approach. All the objects detected are visualized and the information produced is sent to the Planner task.

Structure-From-Motion task is a method for estimating 3-D structures (depth) from vehicle motion and sequences of 2-D images. This task returns a matrix of points representing the distance with respect the objects of the image.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.↵html>

4.9.2 Function Documentation

4.9.2.1 objDetectGetObject()

```
void* objDetectGetObject (
    void * args )
```

This task is used perform the detection functionality.

This task is responsible of detecting and classifying the objects in the road. All the objects detected are visualized and the information produced is sent to the Planner task. It is executed on core number 0 on Jetson TX2 ARM A57 core with the thread priority of 98.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

4.9.2.2 objDetectStructureFromMotion()

```
void* objDetectStructureFromMotion (
    void * args )
```

This task is used perform the Structure-From-Motion functionality.

Structure-From-Motion is a method for estimating 3-D structures (depth) from vehicle motion and sequences of 2-D images. This task returns a matrix of points representing the distance with respect the objects of the image. It is executed on core number 0 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

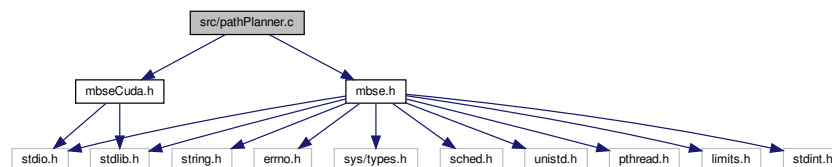
4.10 src/pathPlanner.c File Reference

This file contains the basic implementation of the Planner task and CAN Bus polling task.

```
#include "mbse.h"
```

```
#include "mbseCuda.h"
```

Include dependency graph for pathPlanner.c:



Functions

- void * [pathPlannerCalculation](#) (void *args)
This task is used perform the Planner task functionality.
- void * [pathPlannerCanBusPolling](#) (void *args)
This task is used perform the CAN Bus Polling task functionality.

4.10.1 Detailed Description

This file contains the basic implementation of the Planner task and CAN Bus polling task.

Author

Anand Prakash

Date

22 April 2020 The main purpose of Planner component is to define and follow a given trajectory. This trajectory is defined as a spline, that is, a line built through polynomial interpolation at times on the map that represents at each point the position and orientation that the car will have to follow. The spline can be enriched with additional information such as speed to hold, stop, priorities, etc. The planner sends the goal state of the vehicle (i.e., target steer and speed) to the DASM task that is in charge of writing the commands in the CAN line the effective steer and speed to apply.

The CAN Bus polling task snoops the key vehicle information (steer/wheel/break/acceleration status...) from the on-board CAN bus and sends it to the Planner task.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>↵

4.10.2 Function Documentation

4.10.2.1 pathPlannerCalculation()

```
void* pathPlannerCalculation (
    void * args )
```

This task is used perform the Planner task functionality.

The main purpose of this component is to define and follow a given trajectory. This trajectory is defined as a spline, that is, a line built through polynomial interpolation at times on the map that represents at each point the position and orientation that the car will have to follow. The planner sends the goal state of the vehicle (i.e., target steer and speed) to the DASM task that is in charge of writing the commands in the CAN line the effective steer and speed to apply. It is executed on core number 5 on Jetson TX2 ARM A57 core with the thread priority of 98.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

void

4.10.2.2 pathPlannerCanBusPolling()

```
void* pathPlannerCanBusPolling (
    void * args )
```

This task is used perform the CAN Bus Polling task functionality.

This task snoops the key vehicle information (steer/wheel/break/acceleration status...) from the on-board CAN bus and sends it to the Localization, Planner and EKF tasks. It is executed on core number 5 on Jetson TX2 ARM A57 core with the thread priority of 99.

Parameters

in	args	Optional argument. Currently not in use.
----	------	--

Returns

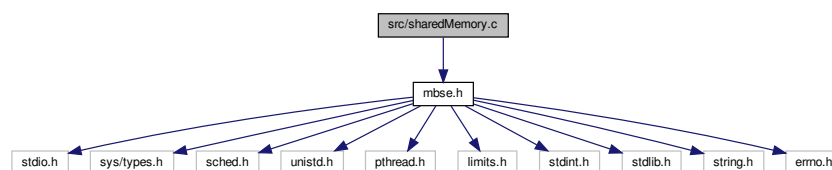
void

4.11 src/sharedMemory.c File Reference

This file declares and implement the read and write operation of all the shared memory.

```
#include "mbse.h"
```

Include dependency graph for sharedMemory.c:



Macros

- `#define DATA_IN_SFM_DETECTION_BUFFER ((2 * 1024 * 1024) / 4)`
- `#define DATA_OUT_SFM_BUFFER ((24 * 1024) / 4)`
- `#define DATA_OUT_DETECTION_BUFFER ((750 * 1024) / 4)`
- `#define DATA_OUT_PLANNER_BUFFER ((1 * 1024) / 4)`
- `#define DATA_GRID_BUFFER ((1024 * 512) / 4)`
- `#define DATA_LANE_BOUNDARY_BUFFER (256 / 4)`

Functions

- int [shmemReadSFMDetectionDataInLabel](#) (unsigned int index)
Function to read to the Detection and SFM Input data buffer.
- int [shmemReadSFMDataOutLabel](#) (unsigned int index)
Function to read to the SFM Output data buffer.
- int [shmemReadDetectionDataOutLabel](#) (unsigned int index)
Function to read to the Detection Output data buffer.
- int [shmemReadPlannerBufferLabel](#) (unsigned int index)
Function to read to the Planner Output data buffer.
- int [shmemReadGridDataBufferLabel](#) (unsigned int index)
Function to read to the data grid buffer.
- int [shmemReadLaneBoundaryBufferLabel](#) (unsigned int index)
Function to read to the lane boundary data buffer.
- void [shmemWriteDetectionDataOutLabel](#) (int offset, int size, void *data)
Function to write to the Detection data buffer.
- void [shmemWriteSFMDataOutLabel](#) (int offset, int size, void *data)
Function to write to the SFM data buffer.
- void [shmemWritePlannerDataOutLabel](#) (int offset, int size, int data)
Function to write to the Planner data buffer.
- void [shmemWriteSFMDetectionDataInLabel](#) (int offset, int size, int data)
Function to write to the SFM and detection Input buffer.
- void [shmemWriteGridDataBufferLabel](#) (int offset, int size, int data)
Function to write to the data grid buffer.
- void [shmemWriteLaneBoundaryBufferLabel](#) (int offset, int size, int data)
Function to write to the lane boundary buffer.

4.11.1 Detailed Description

This file declares and implement the read and write operation of all the shared memory.

Author

Anand Prakash

Date

05 May 2020 It consists a common input buffer for Detection and Structure for Motion task which is 2MB in size. Output data buffer for Structure for motion and Detection task with size 24KB and 750KB respectively. The planner, data grid and lane boundary buffers are 1KB, 512KB and 256 bytes in size respectively.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>↵

4.11.2 Macro Definition Documentation

4.11.2.1 DATA_GRID_BUFFER

```
#define DATA_GRID_BUFFER ((1024 * 512) / 4)
```

4.11.2.2 DATA_IN_SFM_DETECTION_BUFFER

```
#define DATA_IN_SFM_DETECTION_BUFFER ((2 * 1024 * 1024) / 4)
```

4.11.2.3 DATA_LANE_BOUNDARY_BUFFER

```
#define DATA_LANE_BOUNDARY_BUFFER (256 / 4)
```

4.11.2.4 DATA_OUT_DETECTION_BUFFER

```
#define DATA_OUT_DETECTION_BUFFER ((750 * 1024) / 4)
```

4.11.2.5 DATA_OUT_PLANNER_BUFFER

```
#define DATA_OUT_PLANNER_BUFFER ((1 * 1024) / 4)
```

4.11.2.6 DATA_OUT_SFM_BUFFER

```
#define DATA_OUT_SFM_BUFFER ((24 * 1024) / 4)
```

4.11.3 Function Documentation

4.11.3.1 shmemReadDetectionDataOutLabel()

```
int shmemReadDetectionDataOutLabel (  
    unsigned int index )
```

Function to read to the Detection Output data buffer.

This function read the Detection output data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.11.3.2 shmemReadGridDataBufferLabel()

```
int shmemReadGridDataBufferLabel (
    unsigned int index )
```

Function to read to the data grid buffer.

This function read the data grid buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.11.3.3 shmemReadLaneBoundaryBufferLabel()

```
int shmemReadLaneBoundaryBufferLabel (
    unsigned int index )
```

Function to read to the lane boundary data buffer.

This function read the lane boundary data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.11.3.4 shmemReadPlannerBufferLabel()

```
int shmemReadPlannerBufferLabel (
    unsigned int index )
```

Function to read to the Planner Output data buffer.

This function read the Planner output data buffer. This buffer is used as an input to DASM task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.11.3.5 shmemReadSFMDDataOutLabel()

```
int shmemReadSFMDDataOutLabel (
    unsigned int index )
```

Function to read to the SFM Output data buffer.

This function read the SFM output data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.11.3.6 shmemReadSFMDetectionDataInLabel()

```
int shmemReadSFMDetectionDataInLabel (
    unsigned int index )
```

Function to read to the Detection and SFM Input data buffer.

This function read to the Detection and SFM Input data buffer. This buffer is used as an input to SFM and Detection task.

Parameters

in	<i>index</i>	Index at which the data is to be written.
----	--------------	---

Returns

The value at the requested index, -1 in case the index is out of bounds.

4.11.3.7 shmemWriteDetectionDataOutLabel()

```
void shmemWriteDetectionDataOutLabel (
    int offset,
    int size,
    void * data )
```

Function to write to the Detection data buffer.

This function writes to the Detection data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.11.3.8 shmemWriteGridDataBufferLabel()

```
void shmemWriteGridDataBufferLabel (
    int offset,
    int size,
    int data )
```

Function to write to the data grid buffer.

This function writes to the data grid buffer. This buffer is used as an input to Planner task

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.11.3.9 shmemWriteLaneBoundaryBufferLabel()

```
void shmemWriteLaneBoundaryBufferLabel (
    int offset,
    int size,
    int data )
```

Function to write to the lane boundary buffer.

This function writes to the lane boundary buffer. This buffer is used as an input to Planner task

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.11.3.10 shmemWritePlannerDataOutLabel()

```
void shmemWritePlannerDataOutLabel (
    int offset,
    int size,
    int data )
```

Function to write to the Planner data buffer.

This function writes to the Planner data buffer. This buffer is used as an input to DASM task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.11.3.11 shmemWriteSFMDDataOutLabel()

```
void shmemWriteSFMDDataOutLabel (
    int offset,
    int size,
    void * data )
```

Function to write to the SFM data buffer.

This function writes to the SFM data buffer. This buffer is used as an input to Planner task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

void

4.11.3.12 shmemWriteSFMDetectionDataInLabel()

```
void shmemWriteSFMDetectionDataInLabel (
    int offset,
    int size,
    int data )
```

Function to write to the SFM and detection Input buffer.

This function writes to the SFM and detection Input buffer. This buffer is used as an input to SFM and detection task.

Parameters

in	<i>offset</i>	Offset of the buffer.
in	<i>size</i>	Length of data.
in	<i>data</i>	Actual data that needs to be copied.

Returns

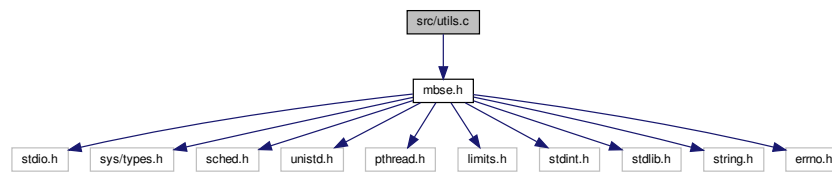
void

4.12 src/utls.c File Reference

This file declares and implement the utility functions used in the application.

```
#include "mbse.h"
```

Include dependency graph for utlis.c:



Functions

- void [utilSetThreadPriority](#) (pthread_t threadId, int customPrio)
Function to set the thread priority.
- void [utilAddDelay](#) (uint32_t ms, struct timespec *deadline)
Function to add delay to the task.
- void [error](#) (int at)
Function to print error message thrown from a particular point in application code.

4.12.1 Detailed Description

This file declares and implement the utility functions used in the application.

Author

Anand Prakash

Date

22 April 2020 It consists of functions setting the thread priority, adding delay to the tasks and printing errors. All the C utility functions must be added to this file.

See also

<https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/index.html>

4.12.2 Function Documentation

4.12.2.1 error()

```
void error (
    int at )
```

Function to print error message thrown from a particular point in application code.

This function prints the error code from which the error was thrown.

Parameters

in	<i>at</i>	Error code.
----	-----------	-------------

Returns

void

4.12.2.2 utilAddDelay()

```
void utilAddDelay (
    uint32_t ms,
    struct timespec * deadline )
```

Function to add delay to the task.

This function adds delay to the task by sleeping for the time provided in the argument in terms of millisecond.

Parameters

in	<i>ms</i>	Time in millisecond for which the thread needs to sleep.
in, out	<i>deadline</i>	Pointer to the timespec for the next deadline

Returns

void

4.12.2.3 utilSetThreadPriority()

```
void utilSetThreadPriority (
    pthread_t threadId,
    int customPrio )
```

Function to set the thread priority.

This task sets the thread priority based on the threadId and the customPrio provided The priority of the thread is set to max priority minus the custom priority

Parameters

in	<i>threadId</i>	Thread ID whose priority needs to be set.
in	<i>customPrio</i>	Priority offset of the thread from the maximum priority

Returns

void

Index

- [_GNU_SOURCE](#)
 - [mbse.h, 13](#)
- [addOSOverhead](#)
 - [cuDASM.cu, 33](#)
 - [mbseCuda.h, 27](#)
- [bBoxHost](#)
 - [plannerData_t, 7](#)
- [bboxDeviceDetection](#)
 - [detectObject_t, 5](#)
- [bboxHostDetection](#)
 - [detectObject_t, 5](#)
- [canData](#)
 - [plannerData_t, 7](#)
- [computeDASM](#)
 - [cuDASM.cu, 33](#)
- [computeOSOverhead](#)
 - [computeSpeedAndSteer.c, 31](#)
 - [mbse.h, 14](#)
- [computeSpeedAndSteer](#)
 - [computeSpeedAndSteer.c, 31](#)
 - [mbse.h, 14](#)
- [computeSpeedAndSteer.c](#)
 - [computeOSOverhead, 31](#)
 - [computeSpeedAndSteer, 31](#)
- [cuDASM.cu](#)
 - [addOSOverhead, 33](#)
 - [computeDASM, 33](#)
 - [cuProcessDASM, 33](#)
 - [osOverhead, 34](#)
- [cuDetectObject](#)
 - [cuObjDetection.cu, 36](#)
 - [mbseCuda.h, 27](#)
- [cuObjDetectSFM](#)
 - [cuObjDetection.cu, 37](#)
 - [mbseCuda.h, 28](#)
- [cuObjDetection.cu](#)
 - [cuDetectObject, 36](#)
 - [cuObjDetectSFM, 37](#)
 - [processImage, 37](#)
 - [processSFMDData, 37](#)
- [cuPathPlanner.cu](#)
 - [cuPlannerFetchCanBusData, 39](#)
 - [cuPlannerInterpolatePath, 39](#)
 - [getCanBusData, 39](#)
 - [pathPlan, 40](#)
- [cuPlannerFetchCanBusData](#)
 - [cuPathPlanner.cu, 39](#)
- [mbseCuda.h, 28](#)
- [cuPlannerInterpolatePath](#)
 - [cuPathPlanner.cu, 39](#)
 - [mbseCuda.h, 29](#)
- [cuProcessDASM](#)
 - [cuDASM.cu, 33](#)
 - [mbseCuda.h, 29](#)
- [cudaUtils.cu](#)
 - [getCudaDeviceProperties, 35](#)
- [DATA_GRID_BUFFER](#)
 - [sharedMemory.c, 46](#)
- [DATA_IN_SFM_DETECTION_BUFFER](#)
 - [sharedMemory.c, 47](#)
- [DATA_LANE_BOUNDARY_BUFFER](#)
 - [sharedMemory.c, 47](#)
- [DATA_OUT_DETECTION_BUFFER](#)
 - [sharedMemory.c, 47](#)
- [DATA_OUT_PLANNER_BUFFER](#)
 - [sharedMemory.c, 47](#)
- [DATA_OUT_SFM_BUFFER](#)
 - [sharedMemory.c, 47](#)
- [detectObject](#)
 - [mbseCuda.h, 26](#)
- [detectObject_t, 5](#)
 - [bboxDeviceDetection, 5](#)
 - [bboxHostDetection, 5](#)
 - [imageDeviceDetection, 5](#)
 - [imageHostDetection, 6](#)
- [error](#)
 - [mbse.h, 16](#)
 - [utils.c, 53](#)
- [getCanBusData](#)
 - [cuPathPlanner.cu, 39](#)
- [getCudaDeviceProperties](#)
 - [cudaUtils.cu, 35](#)
 - [mbseCuda.h, 29](#)
- [imageDeviceDetection](#)
 - [detectObject_t, 5](#)
- [imageHostDetection](#)
 - [detectObject_t, 6](#)
- [imageSFMHost](#)
 - [sfmData_t, 9](#)
- [inc/mbse.h, 11](#)
- [inc/mbseCuda.h, 25](#)
- [laneBoundary](#)
 - [plannerData_t, 7](#)

MBSE_NUMBER_OF_THREADS
 mbse.h, 13
 MBSE_THREAD_STACK_SIZE
 mbse.h, 13
 MICRO_SECONDS
 mbse.h, 13
 MILLI_SECONDS
 mbse.h, 13
 main
 main.c, 41
 main.c
 main, 41
 threadPool_t, 41
 matrixSFMHost
 sfmData_t, 9
 matrixSFM
 plannerData_t, 7
 mbse.h
 _GNU_SOURCE, 13
 computeOSOverhead, 14
 computeSpeedAndSteer, 14
 error, 16
 MBSE_NUMBER_OF_THREADS, 13
 MBSE_THREAD_STACK_SIZE, 13
 MICRO_SECONDS, 13
 MILLI_SECONDS, 13
 NSEC_PER_SEC, 14
 objDetectGetObject, 16
 objDetectStructureFromMotion, 16
 pathPlannerCalculation, 17
 pathPlannerCanBusPolling, 17
 SECONDS, 14
 shmemReadDetectionDataOutLabel, 19
 shmemReadGridDataBufferLabel, 19
 shmemReadLaneBoundaryBufferLabel, 19
 shmemReadPlannerBufferLabel, 20
 shmemReadSFMDDataOutLabel, 20
 shmemReadSFMDetectionDataInLabel, 21
 shmemWriteDetectionDataOutLabel, 21
 shmemWriteGridDataBufferLabel, 21
 shmemWriteLaneBoundaryBufferLabel, 22
 shmemWritePlannerDataOutLabel, 22
 shmemWriteSFMDDataOutLabel, 23
 shmemWriteSFMDetectionDataInLabel, 23
 utilAddDelay, 24
 utilSetThreadPriority, 24
 mbseCuda.h
 addOSOverhead, 27
 cuDetectObject, 27
 cuObjDetectSFM, 28
 cuPlannerFetchCanBusData, 28
 cuPlannerInterpolatePath, 29
 cuProcessDASM, 29
 detectObject, 26
 getCudaDeviceProperties, 29
 minVal, 26
 plannerData, 27
 sfmData, 27
 minVal
 mbseCuda.h, 26
 NSEC_PER_SEC
 mbse.h, 14
 objDetectGetObject
 mbse.h, 16
 objectDetection.c, 42
 objDetectStructureFromMotion
 mbse.h, 16
 objectDetection.c, 43
 objectDetection.c
 objDetectGetObject, 42
 objDetectStructureFromMotion, 43
 occupancyGrid
 plannerData_t, 7
 osOverhead
 cuDASM.cu, 34
 pathPlan
 cuPathPlanner.cu, 40
 pathPlanner.c
 pathPlannerCalculation, 44
 pathPlannerCanBusPolling, 44
 pathPlannerCalculation
 mbse.h, 17
 pathPlanner.c, 44
 pathPlannerCanBusPolling
 mbse.h, 17
 pathPlanner.c, 44
 plannerData
 mbseCuda.h, 27
 plannerData_t, 6
 bBoxHost, 7
 canData, 7
 laneBoundary, 7
 matrixSFM, 7
 occupancyGrid, 7
 speedObjective, 7
 steerObjective, 7
 velocity, 8
 xCar, 8
 yCar, 8
 yawCar, 8
 yawRate, 8
 processImage
 cuObjDetection.cu, 37
 processSFMDData
 cuObjDetection.cu, 37
 SECONDS
 mbse.h, 14
 sfmData
 mbseCuda.h, 27
 sfmData_t, 9
 imageSFMHost, 9
 matrixSFMHost, 9
 sharedMemory.c

- DATA_GRID_BUFFER, [46](#)
- DATA_IN_SFM_DETECTION_BUFFER, [47](#)
- DATA_LANE_BOUNDARY_BUFFER, [47](#)
- DATA_OUT_DETECTION_BUFFER, [47](#)
- DATA_OUT_PLANNER_BUFFER, [47](#)
- DATA_OUT_SFM_BUFFER, [47](#)
- shmemReadDetectionDataOutLabel, [47](#)
- shmemReadGridDataBufferLabel, [48](#)
- shmemReadLaneBoundaryBufferLabel, [48](#)
- shmemReadPlannerBufferLabel, [48](#)
- shmemReadSFMDDataOutLabel, [49](#)
- shmemReadSFMDetectionDataInLabel, [49](#)
- shmemWriteDetectionDataOutLabel, [50](#)
- shmemWriteGridDataBufferLabel, [50](#)
- shmemWriteLaneBoundaryBufferLabel, [51](#)
- shmemWritePlannerDataOutLabel, [51](#)
- shmemWriteSFMDDataOutLabel, [51](#)
- shmemWriteSFMDetectionDataInLabel, [52](#)
- shmemReadDetectionDataOutLabel
 - mbse.h, [19](#)
 - sharedMemory.c, [47](#)
- shmemReadGridDataBufferLabel
 - mbse.h, [19](#)
 - sharedMemory.c, [48](#)
- shmemReadLaneBoundaryBufferLabel
 - mbse.h, [19](#)
 - sharedMemory.c, [48](#)
- shmemReadPlannerBufferLabel
 - mbse.h, [20](#)
 - sharedMemory.c, [48](#)
- shmemReadSFMDDataOutLabel
 - mbse.h, [20](#)
 - sharedMemory.c, [49](#)
- shmemReadSFMDetectionDataInLabel
 - mbse.h, [21](#)
 - sharedMemory.c, [49](#)
- shmemWriteDetectionDataOutLabel
 - mbse.h, [21](#)
 - sharedMemory.c, [50](#)
- shmemWriteGridDataBufferLabel
 - mbse.h, [21](#)
 - sharedMemory.c, [50](#)
- shmemWriteLaneBoundaryBufferLabel
 - mbse.h, [22](#)
 - sharedMemory.c, [51](#)
- shmemWritePlannerDataOutLabel
 - mbse.h, [22](#)
 - sharedMemory.c, [51](#)
- shmemWriteSFMDDataOutLabel
 - mbse.h, [23](#)
 - sharedMemory.c, [51](#)
- shmemWriteSFMDetectionDataInLabel
 - mbse.h, [23](#)
 - sharedMemory.c, [52](#)
- speedObjective
 - plannerData_t, [7](#)
- src/computeSpeedAndSteer.c, [30](#)
- src/cuDASM.cu, [31](#)
- src/cuObjDetection.cu, [35](#)
- src/cuPathPlanner.cu, [38](#)
- src/cudaUtils.cu, [34](#)
- src/main.c, [40](#)
- src/objectDetection.c, [41](#)
- src/pathPlanner.c, [43](#)
- src/sharedMemory.c, [45](#)
- src/Utils.c, [52](#)
- steerObjective
 - plannerData_t, [7](#)
- threadPool_t
 - main.c, [41](#)
- utilAddDelay
 - mbse.h, [24](#)
 - utils.c, [54](#)
- utilSetThreadPriority
 - mbse.h, [24](#)
 - utils.c, [54](#)
- utils.c
 - error, [53](#)
 - utilAddDelay, [54](#)
 - utilSetThreadPriority, [54](#)
- velocity
 - plannerData_t, [8](#)
- xCar
 - plannerData_t, [8](#)
- yCar
 - plannerData_t, [8](#)
- yawCar
 - plannerData_t, [8](#)
- yawRate
 - plannerData_t, [8](#)