SPRINTS

Challenge Requirements Document

Wave	**4**		**Trainer Name**	**Hossam Adel**
Week Number	**6**		**Challenge Name**	**Basic Com Module**
Duration			**Type**	**Project**

Team Size
2

SW/HW environment:
UART driver based on the ISR and Dev Board.

Restrictions (peripherals, configurations, what to use and not to use):
We going to Design the software using whatever we had in our HW to get the highest throughput.

Challenge Requirements

AGILE_REQ1:
Implement BCM (Basic Com Module), Module has a capability to receive and send stream of data without reach 100% cpu load.

AGILE_REQ2:
Implement BCM_Init by following the next table :

Function Name	BCM_Init
Syntax	EnmBCMError_t BCM_Init (const BCM_ConfigType * ConfigPtr)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in):	ConfigPtr Pointer to a selected configuration structure
Parameters (out):	None
Parameters (inOut):	None
Return:	EnmBCMError_t | one of predefine enumeration number

AGILE_REQ3:

The function BCM_Init shall initialized the hardware UART module.

AGILE_REQ4:

Implement BCM_DeInit by following the next table :

Function Name	BCM_DeInit
Syntax	EnmBCMError_t BCM_DeInit (void)
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in):	None
Parameters (out):	None
Parameters (inOut):	None
Return:	EnmBCMError_t | one of predefine enumeration number

AGILE_REQ5:

The function BCM_DeInit shall uninitialized the hardware UART module.

AGILE_REQ6:

Implement BCM_RxMainFunction by following the next table :
<Part of my training let the Sprinters imaging how to implement the function from scratch>

Hint: This table need to be filled be the Sprinters.

Function Name	BCM_RxDispatch
Syntax	void BCM_RxDispatch(void)
Sync/Async	ASynchronous
Reentrancy	Non Reentrant
Parameters (in):	None
Parameters (out):	None
Parameters (inOut):	None
Return:	None

AGILE_REQ7:

The function BCM_RxMainFunction shall Run the basic logic code to move the check the status machine code of the Receiving bytes and reconstruct the packet to extract only the user data.

This function part of the super loop code.

AGILE_REQ8:

Implement BCM_TxMainFunction by following the next table :
<Part of my training let the Sprinters imaging how to implement the function from scratch>

Hint: This table need to be filled be the Sprinters.

Function Name	BCM_TxDispatch
Syntax	`void BCM_TxDispatch(void)`
Sync/Async	ASynchronous
Reentrancy	Non Reentrant
Parameters (in):	None
Parameters (out):	None
Parameters (inOut):	None
Return:	None

AGILE_REQ9:

The function BCM_TxMainFunction shall Run the basic logic code to move the check the status machine code of the transmitting bytes and construct the packet (Header – Data – Check SUM), and the Header contain Command and the data size.

This function part of the super loop code.

AGILE_REQ10:
Implement BCM_Send by following the next table :
<Part of my training let the Sprinters imaging how to implement the function from scratch>

Hint: This table need to be filled be the Sprinters.

Function Name	BCM_Send
Syntax	`EnmBCMError_t BCM_Send(uint8 * COPY_ptrData,uint16 COPY_u16BufferSize, BCM_ptrToFuncTX COPY_BCM_ptrConsumerFunc)`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in):	Pointer to buffer address, Buffer Size
Parameters (out):	Pointer to Consumer
Parameters (inOut):	None
Return:	`EnmBCMError_t`

AGILE_REQ11:

The function BCM_Send shall invoked by the user to send the stream of data.

This function part of the super loop code.

AGILE_REQ12:

Implement BCM_Receive by following the next table :
<Part of my training let the Sprinters imaging how to implement the function from scratch>

Hint: This table need to be filled be the Sprinters.

Function Name	**BCM_SetupRxBuffer**
Syntax	`EnmBCMError_t BCM_SetupRxBuffer(uint8* COPY_ptrRxBuffer,uint16 COPY_u16BufferSize,BCM_ptrToFuncRX COPY_BCM_ptrConsumerFunc)`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in):	Pointer to buffer address, Buffer Size
Parameters (out):	Pointer to Consumer
Parameters (inOut):	None
Return:	`EnmBCMError_t`

AGILE_REQ13:

The function BCM_Receive shall invoked by the user to receive the stream of data.

This function part of the super loop code.