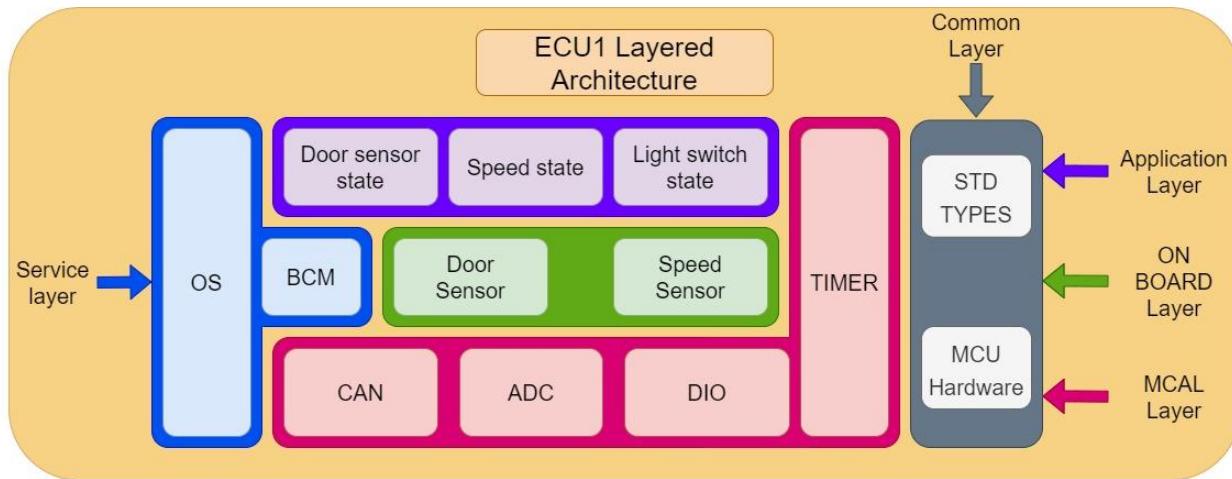


Embedded Software Engineering

Static Design

ECU1:

layered architecture:



Full API details and File structure:

1.Car Speed state module APIs:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "EMBEDDED SYSTEM DESIGN". The "Car_Speed_State.h" file is selected, displaying its contents.
- Code Editor:** Displays the "Car_Speed_State.h" header file with the following code:

```
ECU1 > src > APP > Inc > C_Car_Speed_State.h ...
1 #include <Std_Types.h>
2 #define Speed_Refresh_Rate 5
3
4 extern uint32 currentSpeed;
5
6 /*****
7 * Syntax      : void Upload_Car_Speed(uint32 currentSpeed)
8 * Description : upload the car speed every 5 ms using the BCM to ECU2
9 *
10 * Async/Async   : Synchronous
11 * Reentrancy    : Non Reentrant
12 * Parameters (in) : currentSpeed the current car speed reading
13 * Parameters (out): None
14 * Return value : None
15 *****/
16
17 void Upload_Car_Speed(uint32 currentSpeed);
18
19
20
21
```

The status bar at the bottom shows: Activate Windows Go to Settings to activate Windows. Line 20, Col 1 Spaces: 4 UTF-8 C++ Win32 Prettier 24% ENG 6:19 PM 07-Sep-22

2. Door state module APIs:

The screenshot shows the Visual Studio Code interface with the file `Door_State.h` open. The code defines a function `Upload_Door_State` with annotations for syntax, description, and parameters.

```
ECU1 > src > APP > Inc > C_Door_State.h ...
1 #include <Std_types.h>
2 #include <Door_Sensor.h>
3 #define DOOR_STATE_REFRESH_RATE 10
4
5 extern Door_State currentDoorState;
6
7
8
9
10 ****
11 * \Syntax      : void Upload_Door_State(Door_State currentDoorState)
12 * \Description : upload the Door state every 10 ms using the BCM to ECU
13 *
14 *
15 * \Sync\Async   : Synchronous
16 * \Reentrancy  : Non Reentrant
17 * \Parameters (in) : currentDoorState the current Door state reading
18 * \Parameters (out): None
19 * \Return value : None
20 ****
21 void Upload_Door_State(Door_State currentDoorState);
```

Activate Windows
Go to Settings to activate Windows.

File Edit Selection View Go Run Terminal Help Door_State.h - Embedded system design - Visual Studio Code

EXPLORER CAN.h C_DAC.h C_DIO.h C_Timer.h C_Buzzer.h C_LED.h C_OS.h C_Car_Speed_State.h C_Door_State.h ...

OPEN EDITORS C_Door_State.h ECU1\src\APP\Inc

EMBEDDED SYSTEM DESIGN ECU1\src APP Inc C_Car_Speed_State.h C_Door_State.h C_Switch_State.h C_Car_Speed_State.c C_Door_State.c C_Switch_State.c Common MCAL Inc C_DAC.h C_CAN.h C_DIO.h C_Timer.h C_ADC.c C_CAN.c C_DIO.c C_Timer.c

OUTLINE TIMELINE

Ln 13, Col 44 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier 619 PM 07-Sep-22 24% ENG

3. Switch state module APIs:

The screenshot shows the Visual Studio Code interface with the file `Switch_State.h` open. The code defines a function `Upload_Switch_State` with annotations for syntax, description, and parameters. It also includes a definition for `Attach_Switch`.

```
ECU1 > src > APP > Inc > C_Switch_State.h ...
4
5 #define SWITCH_STATE_REFRESH_RATE 20
6
7 extern uint8 currentSwitchState;
8
9 ****
10 * \Syntax      : void Upload_Switch_State(uint8 currentSwitchState,uint32 Switch_ID)
11 * \Description : upload the Switch state every 20 ms using the BCM to ECU
12 *
13 *
14 * \Sync\Async   : Synchronous
15 * \Reentrancy  : Non Reentrant
16 * \Parameters (in) : currentSwitchState the current switch state reading
17 * \Parameters (out): None
18 * \Return value : None
19 ****
20 void Upload_Switch_State(uint8 currentSwitchState,uint32 Switch_ID);
21
22
23 ****
24 * \Syntax      : uint32 Attach_Switch(Pin_Num pin , Port_Num port);
25 * \Description : initialize a switch with port and pin numbers
26 *
27 *
28 * \Sync\Async   : Synchronous
29 * \Reentrancy  : Non Reentrant
30 * \Parameters (in) : pin switch pin number
31 * \Parameters (out): None
32 * \Return value : Switch id
33 ****
34 uint32 Attach_Switch(Pin_Num pin , Port_Num port);
```

Activate Windows
Go to Settings to activate Windows.

File Edit Selection View Go Run Terminal Help Switch_State.h - Embedded system design - Visual Studio Code

EXPLORER C_DAC.h C_DIO.h C_Timer.h C_Buzzer.h C_LED.h C_OS.h C_Car_Speed_State.h C_Door_State.h C_Switch_State.h ...

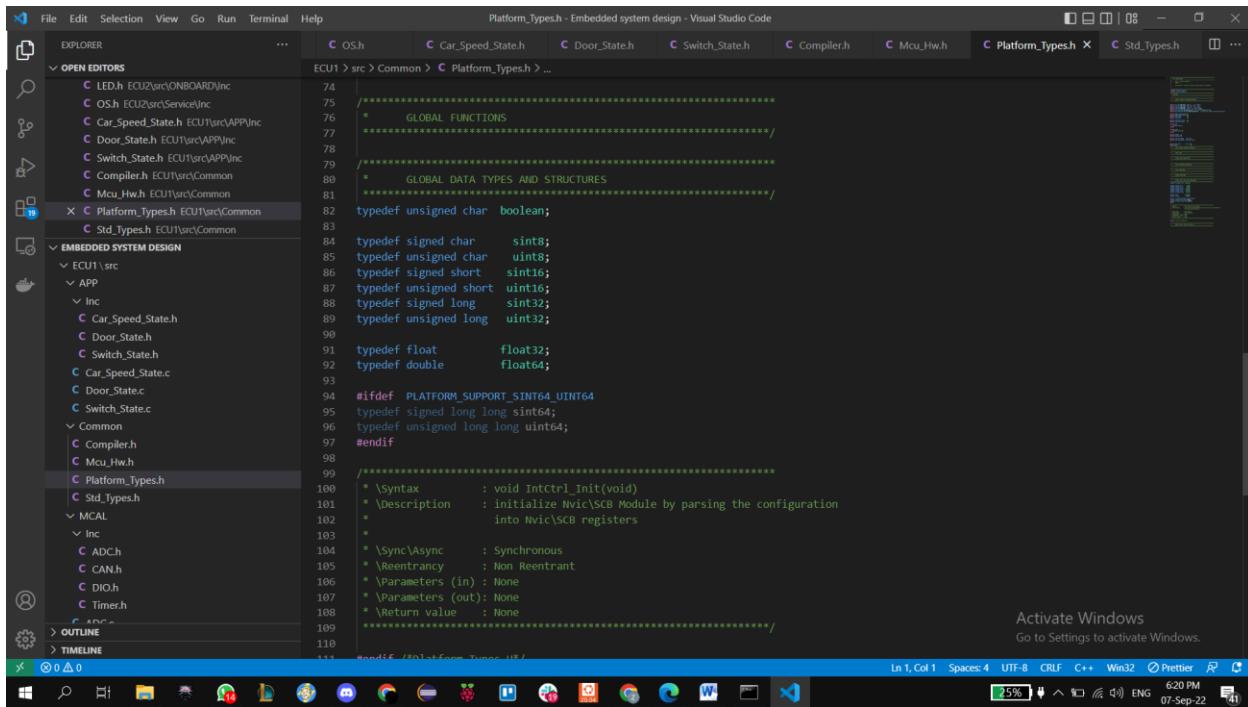
OPEN EDITORS C_Switch_State.h ECU1\src\APP\Inc

EMBEDDED SYSTEM DESIGN ECU1\src APP Inc C_Car_Speed_State.h C_Door_State.h C_Switch_State.h C_Car_Speed_State.c C_Door_State.c C_Switch_State.c Common MCAL Inc C_DAC.h C_CAN.h C_DIO.h C_Timer.h C_ADC.c C_CAN.c C_DIO.c C_Timer.c

OUTLINE TIMELINE

Ln 33, Col 29 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier 619 PM 07-Sep-22 24% ENG

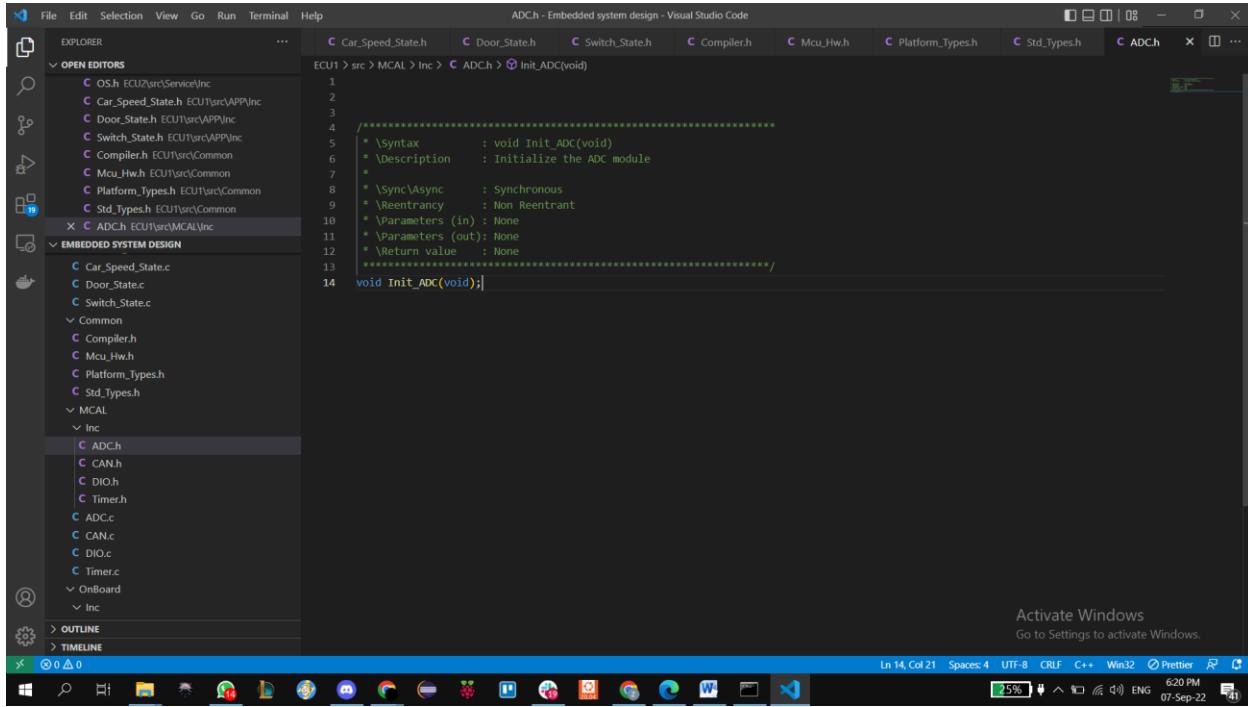
4. Std Types Module APIs:



The screenshot shows the Visual Studio Code interface with the file `Platform.Types.h` open. The code defines various global functions and data types, including boolean, sint8, uint8, sint16, uint16, sint32, uint32, float32, and float64. It also includes a section for the `PLATFORM_SUPPORT_SINT64_UINT64` macro, defining signed long long and unsigned long long types. The code is annotated with comments for syntax, description, reentrancy, parameters, and return value.

```
74 //*****  
75 * GLOBAL FUNCTIONS  
76 *****/  
77 /*  
80 * GLOBAL DATA TYPES AND STRUCTURES  
81 *****/  
82 typedef unsigned char boolean;  
83  
84 typedef signed char sint8;  
85 typedef unsigned char uint8;  
86 typedef signed short sint16;  
87 typedef unsigned short uint16;  
88 typedef signed long sint32;  
89 typedef unsigned long uint32;  
90  
91 typedef float float32;  
92 typedef double float64;  
93  
94 #ifndef PLATFORM_SUPPORT_SINT64_UINT64  
95 typedef signed long long sint64;  
96 typedef unsigned long long uint64;  
97 #endif  
98  
99 /*  
100 * Syntax : void IntCtrl1_Init(void)  
101 * Description : Initialize Nvic\SCB Module by parsing the configuration  
102 * into Nvic\SCB registers  
103 *  
104 * \Sync\Async : Synchronous  
105 * \Reentrancy : Non Reentrant  
106 * \Parameters (in) : None  
107 * \Parameters (out) : None  
108 * \Return value : None  
109 *****/  
110  
111
```

5. ADC Module APIs:



The screenshot shows the Visual Studio Code interface with the file `ADC.h` open. The code contains a single function declaration for `Init_ADC`, which is marked as synchronous and non-reentrant. It takes no parameters and returns no value. The code is annotated with comments for syntax, description, reentrancy, parameters, and return value.

```
1  
2  
3  
4 //*****  
5 * Syntax : void Init_ADC(void)  
6 * Description : Initialize the ADC module  
7 *  
8 * \Sync\Async : Synchronous  
9 * \Reentrancy : Non Reentrant  
10 * \Parameters (in) : None  
11 * \Parameters (out) : None  
12 * \Return value : None  
13 *  
14 void init_ADC(void);
```

6.CAN Module APIs:

CAN.h - Embedded system design - Visual Studio Code

File Edit Selection View Go Run Terminal Help CAN.h > src > MCAL > Inc > CAN.h ...

EXPLORER OPEN EDITORS ECU1 > src > MCAL > Inc > CAN.h ...

Car_Speed_State.h ECUT\src\APP\Inc
Door_State.h ECUT\src\APP\Inc
Switch_State.h ECUT\src\APP\Inc
Compiler.h ECUT\src\Common
Mcu_Hw.h ECUT\src\Common
Platform_Types.h ECUT\src\Common
Std_Types.h ECUT\src\Common
ADC.h ECUT\src\MCAL\Inc
CAN.h ECUT\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN

Car_Speed_State.c
Door_State.c
Switch_State.c
Common
Compiler.h
Mcu_Hw.h
Platform_Types.h
Std_Types.h
MCAL
Inc
ADC.h
CAN.h
DIO.h
Timer.h
ADC.c
CAN.c
DIO.c
Timer.c
Onboard
Inc

OUTLINE
TIMELINE

18 * \Syntax : void Write_CAN(uint32 value)
19 * \Description : Write a word on the CAN bus
20
21 * \Sync\Async : Synchronous
22 * \Reentrancy : Non Reentrant
23 * \Parameters (in) : value the word to be written on CAN bus
24 * \Parameters (out) : None
25 * \Return value : None

27 void Write_CAN(uint32 value);
28
29 ****
30 * \Syntax : uint32 Read_CAN(void)
31 * \Description : read a word from the CAN bus
32
33 * \Sync\Async : Synchronous
34 * \Reentrancy : Non Reentrant
35 * \Parameters (in) : None
36 * \Parameters (out) : None
37 * \Return value : word received on CAN bus
38 *****
39 uint32 Read_CAN(void);

Activate Windows
Go to Settings to activate Windows.

In 36 Col 15 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22

0 0 △ 0

7.DIO Module APIs:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists the project structure under "OPEN EDITORS". The "DIO.h ECU\src\MCAL\Inc" file is currently selected.
- Editor:** The main pane displays the content of the DIO.h header file. The code defines several pin enumeration types and a mode enumeration type.
- Bottom Status Bar:** Shows the current file is "DIO.h", line 62, column 68, with 38 lines of code. It also shows settings like "Spaces: 4", "UTF-8", "C++", "Win32", and "Prettier".
- Bottom Taskbar:** Shows icons for various applications including File Explorer, Task Manager, and a terminal window.

```
File Edit Selection View Go Run Terminal Help DIO.h - Embedded system design - Visual Studio Code
EXPLORER OPEN EDITORS
C Switch_State.h C Compiler.h C Mcu_Hw.h C Platform_Types.h C Std_Types.h C ADCh C CANh ECU\...
C Door_State.h ECU\src\APP\Inc C DIO.h > ...
C Switch_State.h ECU\src\APP\Inc
C Compiler.h ECU\src\Common
C Mcu_Hw.h ECU\src\Common
C Platform_Types.h ECU\src\Common
C Std_Types.h ECU\src\MCAL\Inc
C ADC.h ECU\src\MCAL\Inc
C CAN.h ECU\src\MCAL\Inc
X C DIO.h ECU\src\MCAL\Inc
EMBEDDED SYSTEM DESIGN
C Car_Speed_State.c
C Door_State.c
C Switch_State.c
Common
C Compiler.h
C Mcu_Hw.h
C Platform_Types.h
C Std_Types.h
MCAL
Inc
C ADC.h
C CAN.h
C DIO.h
C Timer.h
C ADC.c
C CAN.c
C DIO.c
C Timer.c
Onboard
Inc
OUTLINE
TIMELINE

DIO.h > src > MCAL > Inc > C DIO.h > ...
1 #include <std_types.h>
2
3 typedef enum{
4     Pin0,
5     Pin1,
6     Pin2,
7     Pin3,
8     Pin4,
9     Pin5,
10    Pin6,
11    Pin7
12 }Pin_Num;
13
14 typedef enum{
15     Pin0,
16     Pin1,
17     Pin2,
18     Pin3,
19     Pin4,
20     Pin5,
21     Pin6,
22     Pin7
23 }Port_Num;
24
25 typedef enum{
26     INPUT,
27     OUTPUT
28 }Mode;
29
30
31 *****
32 * \Syntax      : void Set_Pin_Mode(Pin_Num pin , Port_Num port , Mode mode)
33 * \Description : set the pin direction as input or output
34 *
35 * \Sync\Async   : Synchronous
36 * \Reentrancy  : Non Reentrant
37 * \Parameters (in) : pin the pin number
38 *                   port the port number
Activate Windows
Go to Settings to activate Windows.
Ln 62 Col 68 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier
25% ENG 07-Sep-22
```

File Edit Selection View Go Run Terminal Help DIO.h - Embedded system design - Visual Studio Code

EXPLORER C Switch_State.h C Compiler.h C Mcu_Hw.h C Platform_Types.h C Std.Types.h C ADC.h C CAN.h ECU1\src\MCAL\Inc C DIO.h ECU1\src\MCAL\Inc

OPEN EDITORS C Door_State.h ECU1\src\APP\Inc C Switch_State.h ECU1\src\APP\Inc C Compiler.h ECU1\src\Common C Mcu_Hw.h ECU1\src\Common C Platform_Types.h ECU1\src\Common C Std.Types.h ECU1\src\Common C ADC.h ECU1\src\MCAL\Inc C CAN.h ECU1\src\MCAL\Inc C DIO.h ECU1\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN C Car_Speed.State.c C Door.State.c C Switch.State.c Common Compiler.h Mcu.Hw.h Platform.Types.h Std.Types.h MCAL Inc ADC.h CAN.h DIO.h Timer.h ADC.c CAN.c DIO.c Timers.c OnBoard Inc

OUTLINE TIMELINE

```
28 }Mode;
29
30
31 /**
32 * \Syntax : void Set_Pin_Mode(Pin_Num pin , Port_Num port , Mode mode)
33 * \Description : set the pin direction as input or output
34 *
35 * \Sync\Async : Synchronous
36 * \Reentrancy : Non Reentrant
37 * \Parameters (in) : pin the pin number
38 *                   port the port number
39 *                   mode the state of pin input or output
40 * \Parameters (out): None
41 * \Return value : None
42 */
43 void Set_Pin_Mode(Pin_Num pin , Port_Num port , Mode mode);
44
45
46
47 /**
48 * \Syntax : void Write_Pin(Pin_Num pin , Port_Num port , uint8 Level)
49 * \Description : write a given level value on a chosen channel
50 *
51 * \Sync\Async : Synchronous
52 * \Reentrancy : Non Reentrant
53 * \Parameters (in) : Level the value to be written in the channel
54 *                   pin the pin number
55 *                   port the port number
56 * \Parameters (out): None
57 * \Return value : None
58 */
59 void Write_Pin(Pin_Num pin , Port_Num port , uint8 value);
60
61
62 /**
63 * \Syntax : uint8 Read_Pin(Pin_Num pin , Port_Num port )
64 * \Description : returns the value on a given channel
65
66 * \Sync\Async : Synchronous
67 * \Reentrancy : Non Reentrant
68 * \Parameters (in) : pin the pin number
69 *                   port the port number
70 * \Parameters (out): None
71 * \Return value : uint8 the value on pin
72 */
73 uint8 Read_Pin(Pin_Num pin , Port_Num port );
74
75
76
77 /**
78 * \Syntax : void Write_Port(Port_Num PortId, uint32 Level )
79 * \Description : writes a value on a chosen port in the gpio
80 *
81 * \Sync\Async : Synchronous
82 * \Reentrancy : Non Reentrant
83 * \Parameters (in) : PortId the port number to write on
84 *                   Level the value to write on the desired port
85 * \Parameters (out): None
86 * \Return value : None
87 */
88 void Write_Port(Port_Num PortId, uint32 Level );
89
90
91 /**
92 * \Syntax : uint32 Read_Port(Port_Num PortId);
93 * \Description : returns the value on a chosen port in the gpio
94 *
95 * \Sync\Async : Synchronous
96 * \Reentrancy : Non Reentrant
97 * \Parameters (in) : PortId the port id to read from
98 */
99
```

Activate Windows
Go to Settings to activate Windows.

Ln 62, Col 68 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help DIO.h - Embedded system design - Visual Studio Code

EXPLORER C Switch_State.h C Compiler.h C Mcu_Hw.h C Platform_Types.h C Std.Types.h C ADC.h C CAN.h ECU1\src\MCAL\Inc C DIO.h ECU1\src\MCAL\Inc

OPEN EDITORS C Door_State.h ECU1\src\APP\Inc C Switch_State.h ECU1\src\APP\Inc C Compiler.h ECU1\src\Common C Mcu_Hw.h ECU1\src\Common C Platform_Types.h ECU1\src\Common C Std.Types.h ECU1\src\Common C ADC.h ECU1\src\MCAL\Inc C CAN.h ECU1\src\MCAL\Inc C DIO.h ECU1\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN C Car_Speed.State.c C Door.State.c C Switch.State.c Common Compiler.h Mcu.Hw.h Platform.Types.h Std.Types.h MCAL Inc ADC.h CAN.h DIO.h Timer.h ADC.c CAN.c DIO.c Timers.c OnBoard Inc

OUTLINE TIMELINE

```
61
62 /**
63 * \Syntax : uint8 Read_Pin(Pin_Num pin , Port_Num port )
64 * \Description : returns the value on a given channel
65
66 * \Sync\Async : Synchronous
67 * \Reentrancy : Non Reentrant
68 * \Parameters (in) : pin the pin number
69 *                   port the port number
70 * \Parameters (out): None
71 * \Return value : uint8 the value on pin
72 */
73 uint8 Read_Pin(Pin_Num pin , Port_Num port );
74
75
76
77 /**
78 * \Syntax : void Write_Port(Port_Num PortId, uint32 Level )
79 * \Description : writes a value on a chosen port in the gpio
80 *
81 * \Sync\Async : Synchronous
82 * \Reentrancy : Non Reentrant
83 * \Parameters (in) : PortId the port number to write on
84 *                   Level the value to write on the desired port
85 * \Parameters (out): None
86 * \Return value : None
87 */
88 void Write_Port(Port_Num PortId, uint32 Level );
89
90
91 /**
92 * \Syntax : uint32 Read_Port(Port_Num PortId);
93 * \Description : returns the value on a chosen port in the gpio
94 *
95 * \Sync\Async : Synchronous
96 * \Reentrancy : Non Reentrant
97 * \Parameters (in) : PortId the port id to read from
98 */
99
```

Activate Windows
Go to Settings to activate Windows.

Ln 62, Col 68 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22

DIO.h - Embedded system design - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER C Switch_State.h C Compiler.h C Mcu_Hw.h C Platform_Types.h C Std.Types.h C ADC.h C CAN.h ECU1\src\MCAL\Inc\...
OPEN EDITORS C DIO.h ECU1\src\MCAL\Inc\...
EMBEDDED SYSTEM DESIGN C Car_Speed.State.c C Door.State.c C Switch.State.c
C Common C Compiler.h C Mcu.Hw.h C Platform.Types.h C Std.Types.h
C MCAL C Inc C ADC.h C CAN.h C DIO.h
C Timer.h C ADC.c C CAN.c C DIO.c C Timers.c
C OnBoard C Inc
OUTLINE
TIMELINE
0 0 0
Activate Windows
Go to Settings to activate Windows.
Ln 62, Col 68 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22 41

```

8.Timer Module APIs:

Timer.h - Embedded system design - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER C Compiler.h C Mcu_Hw.h C Platform_Types.h C Std.Types.h C ADC.h C CAN.h ECU1\src\MCAL\Inc\...
OPEN EDITORS C Timer.h ECU1\src\MCAL\Inc\...
EMBEDDED SYSTEM DESIGN C Car_Speed.State.c C Door.State.c C Switch.State.c
C Common C Compiler.h C Mcu.Hw.h C Platform.Types.h C Std.Types.h
C MCAL C Inc C ADC.h C CAN.h C DIO.h
C Timer.h C ADC.c C CAN.c C DIO.c C Timers.c
C OnBoard C Inc
OUTLINE
TIMELINE
0 0 0
Activate Windows
Go to Settings to activate Windows.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22 41

```

File Edit Selection View Go Run Terminal Help

Timer.h - Embedded system design - Visual Studio Code

EXPLORER ...

OPEN EDITORS Compiler.h Mcu_Hw.h Platform.Types.h Std.Types.h ADC.h CAN.h DIO.h Timer.h ECU1\src\MCAL\Inc

ECU1 > src > MCAL > Inc > Timer.h > ...

```
68 } gpt_ChannelType;
69 /*gpt_Mode_Type*/
70
71 typedef enum
72 {
73
74     GPT_CH_MODE_ONESHOT = 0x1,
75     GPT_CH_MODE_CONTINUOUS = 0x2
76
77 } Gpt_Channel_Mode_Type;
78
79 /*gpt_ModeType*/
80
81
82 typedef enum
83 {
84     GPT_MODE_NORMAL = 0,
85     GPT_MODE_SLEEP
86
87 } Gpt_Mode_Type;
88
89 /*gpt_PredefTimerType*/
90
91
92 typedef struct
93 {
94     /*TODO SEE IF YOU WILL ADD TICK FREQUENCY*/
95     boolean GPT_PREDEF_TIMER_100US_32BIT;
96     boolean GPT_PREDEF_TIMER_US_16BIT;
97     boolean GPT_PREDEF_TIMER_1US_24BIT;
98     boolean GPT_PREDEF_TIMER_1US_32BIT;
99
100 } Gpt_Driver_Cfg_Type;
101
102 /*Gpt_ConfigType*/
103
104     Gpt_ChannelType channelId;
```

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help

Timer.h - Embedded system design - Visual Studio Code

EXPLORER ...

OPEN EDITORS Compiler.h Mcu_Hw.h Platform.Types.h Std.Types.h ADC.h CAN.h DIO.h Timer.h ECU1\src\MCAL\Inc

ECU1 > src > MCAL > Inc > Timer.h > ...

```
91     typedef struct
92     {
93         /*TODO SEE IF YOU WILL ADD TICK FREQUENCY*/
94         boolean GPT_PREDEF_TIMER_100US_32BIT;
95         boolean GPT_PREDEF_TIMER_US_16BIT;
96         boolean GPT_PREDEF_TIMER_1US_24BIT;
97         boolean GPT_PREDEF_TIMER_1US_32BIT;
98
99     } Gpt_Driver_Cfg_Type;
100
101 /*Gpt_ConfigType*/
102
103
104     Gpt_ChannelType channelId;
105     Gpt_ValueType tickValue;
106     Gpt_ValueType maxTickCount;
107     Gpt_Channel_Mode_Type channelMode;
108     gptNotification gptNotification;
109
110 } Gpt_ConfigType;
111
112 /**
113 * \Syntax      : void Gpt_Init(const Gpt_ConfigType *ConfigPtr)
114 * \Description : Initialize the general purpose timers according
115 *                 to user configurations
116 *
117 * \Sync\Async   : Synchronous
118 * \Reentrancy  : Non Reentrant
119 * \Parameters (in) : ConfigPtr the array of user configurations for each selected timer
120 * \Parameters (out): None
121 * \Return value : None
122 */
123
124 void Gpt_Init(const Gpt_ConfigType *ConfigPtr);
125
126 /**
127 * \Syntax      : void Gpt_DisableNotification(Gpt_ChannelType Channel)
```

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 620 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help

Timer.h - Embedded system design - Visual Studio Code

OPEN EDITORS

- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- ADC.h
- CAN.h
- DIO.h
- Timer.h

EMBEDDED SYSTEM DESIGN

- Car_Speed_State.c
- Door_State.c
- Switch_State.c
- Common
- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
- ADC.h
- CAN.h
- DIO.h
- Timer.h
- ADC.c
- CAN.c
- DIO.c
- Timer.c

OnBoard

INC

OUTLINE

TIMELINE

```
125 ****
126 /**
127 * \Syntax      : void Gpt_DisableNotification(Gpt_ChannelType Channel)
128 * \Description : disables the Interrupt on a given channel
129 *
130 * \Sync\Async   : Synchronous
131 * \Reentrancy  : Non Reentrant
132 * \Parameters (in) : Channel channel to disable its interrupts
133 * \Parameters (out): None
134 * \Return value : None
135 ****
136
137 void Gpt_DisableNotification(Gpt_ChannelType Channel);
138
139 ****
140 /**
141 * \Syntax      : void Gpt_EnableNotification(Gpt_ChannelType Channel)
142 * \Description : enables the Interrupt on a given channel
143 *
144 * \Sync\Async   : Synchronous
145 * \Reentrancy  : Non Reentrant
146 * \Parameters (in) : Channel channel to enable its interrupts
147 * \Parameters (out): None
148 * \Return value : None
149 ****
150 void Gpt_EnableNotification(Gpt_ChannelType Channel);
151
152 ****
153 /**
154 * \Syntax      : void Gpt_StartTimer(Gpt_ChannelType Channel,Gpt_ValueType Value)
155 * \Description : enables the timer TEN field to start the timer
156 *
157 * \Sync\Async   : Synchronous
158 * \Reentrancy  : Non Reentrant
159 * \Parameters (in) : Channel the channel number to start
160 *                   Value the preload value to start the timer with
161 * \Parameters (out): None
162 * \Return value : None
163 ****
164 void Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value);
165
166 ****
167 /**
168 * \Syntax      : void Gpt_StopTimer(Gpt_ChannelType Channel)
169 * \Description : disables the timer TEN field to stop the timer
170 *
171 * \Sync\Async   : Synchronous
172 * \Reentrancy  : Non Reentrant
173 * \Parameters (in) : Channel the channel number to stop
174 * \Parameters (out): None
175 * \Return value : None
176 ****
177 void Gpt_StopTimer(Gpt_ChannelType Channel);
178
179 ****
180 /**
181 * \Syntax      : Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel)
182 * \Description : returns the number of counted ticks from the start of the timer
183 *
184 * \Sync\Async   : Synchronous
185 * \Reentrancy  : Non Reentrant
186 * \Parameters (in) : Channel the channel id to get its elapsed tick value
187 * \Parameters (out): Gpt_ValueType the elapsed tick value
188 * \Return value : None
189 ****
```

Activate Windows
Go to Settings to activate Windows.

In 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 621 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help

Timer.h - Embedded system design - Visual Studio Code

OPEN EDITORS

- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- ADC.h
- CAN.h
- DIO.h
- Timer.h

EMBEDDED SYSTEM DESIGN

- Car_Speed_State.c
- Door_State.c
- Switch_State.c
- Common
- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
- ADC.h
- CAN.h
- DIO.h
- Timer.h
- ADC.c
- CAN.c
- DIO.c
- Timer.c

OnBoard

INC

OUTLINE

TIMELINE

```
151 ****
152 /**
153 * \Syntax      : void Gpt_StartTimer(Gpt_ChannelType Channel,Gpt_ValueType Value)
154 * \Description : enables the timer TEN field to start the timer
155 *
156 * \Sync\Async   : Synchronous
157 * \Reentrancy  : Non Reentrant
158 * \Parameters (in) : Channel the channel number to start
159 *                   Value the preload value to start the timer with
160 * \Parameters (out): None
161 * \Return value : None
162 ****
163
164 void Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value);
165
166 ****
167 /**
168 * \Syntax      : void Gpt_StopTimer(Gpt_ChannelType Channel)
169 * \Description : disables the timer TEN field to stop the timer
170 *
171 * \Sync\Async   : Synchronous
172 * \Reentrancy  : Non Reentrant
173 * \Parameters (in) : Channel the channel number to stop
174 * \Parameters (out): None
175 * \Return value : None
176 ****
177 void Gpt_StopTimer(Gpt_ChannelType Channel);
178
179 ****
180 /**
181 * \Syntax      : Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel)
182 * \Description : returns the number of counted ticks from the start of the timer
183 *
184 * \Sync\Async   : Synchronous
185 * \Reentrancy  : Non Reentrant
186 * \Parameters (in) : Channel the channel id to get its elapsed tick value
187 * \Parameters (out): Gpt_ValueType the elapsed tick value
188 * \Return value : None
189 ****
```

Activate Windows
Go to Settings to activate Windows.

In 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 621 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help

Timer.h - Embedded system design - Visual Studio Code

OPEN EDITORS

- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- ADC.h
- CAN.h
- DIO.h
- Timer.h**

EMBEDDED SYSTEM DESIGN

- Car_Speed_State.c
- Door_State.c
- Switch_State.c
- Common
- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
 - ADC.h
 - CAN.h
 - DIO.h
 - Timer.h**
- ADC.c
- CAN.c
- DIO.c
- Timer.c
- OnBoard
 - Inc

OUTLINE

TIMELINE

```
ECU1 > src > MCAL > Inc > Timer.h > ...
```

175 **Gpt_StopTimer(Gpt_ChannelType Channel);**
176 *********
177 *** \Syntax : Gpt_ValueType Gpt_StopTimer(Gpt_ChannelType Channel)**
178 *** \Description : returns the number of counted ticks from the start of the timer**
179 *** **
180 *** \Sync\Async : Synchronous**
181 *** \Reentrancy : Non Reentrant**
182 *** \Parameters (in) : Channel the channel id to get its elapsed tick value**
183 *** \Parameters (out) : Gpt_ValueType the elapsed tick value**
184 *** \Return value : None**
185 *********
186 **Gpt_ValueType Gpt_StopTimer(Gpt_ChannelType Channel);**
187 *********
188 *** \Syntax : Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel)**
189 *** \Description : returns the number of ticks still remaining for the timer to reset**
190 *** **
191 *** \Sync\Async : Synchronous**
192 *** \Reentrancy : Non Reentrant**
193 *** \Parameters (in) : Channel the channel id to get its remaining tick value**
194 *** \Parameters (out) : Gpt_ValueType the remaining tick value**
195 *** \Return value : None**
196 *********
197 **Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel);**
198 *********
199 *** \Syntax : Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel)**
200 *** \Description : returns the number of ticks still remaining for the timer to reset**
201 *** **
202 *** \Sync\Async : Synchronous**
203 *** \Reentrancy : Non Reentrant**
204 *** \Parameters (in) : Channel the channel id to get its remaining tick value**
205 *** \Parameters (out) : Gpt_ValueType the remaining tick value**
206 *** \Return value : None**
207 *********
208 **GLOBAL DATA PROTOTYPES**
209 **extern const Gpt_Configtype gpt_cfg[];**
210 **extern const Gpt_Driver_Cfg_Type gpt_driver_cfg[];**
211 **extern boolean Timer0Flag;**
212 **#endif /*GPT_H*/**
213 *********
214 **END OF FILE: gpt.h**
215 *********

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 621 PM 07-Sep-22 26% 🔍 ENG

File Edit Selection View Go Run Terminal Help

Timer.h - Embedded system design - Visual Studio Code

OPEN EDITORS

- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- ADC.h
- CAN.h
- DIO.h
- Timer.h**

EMBEDDED SYSTEM DESIGN

- Car_Speed_State.c
- Door_State.c
- Switch_State.c
- Common
- Compiler.h
- Mcu_Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
 - ADC.h
 - CAN.h
 - DIO.h
 - Timer.h**
- ADC.c
- CAN.c
- DIO.c
- Timer.c
- OnBoard
 - Inc

OUTLINE

TIMELINE

```
ECU1 > src > MCAL > Inc > Timer.h > ...
```

185 *** \Parameters (in) : Channel the channel id to get its elapsed tick value**
186 *** \Parameters (out) : Gpt_ValueType the elapsed tick value**
187 *** \Return value : None**
188 *********
189 **Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel);**
190 *********
191 *** \Syntax : Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel)**
192 *** \Description : returns the number of ticks still remaining for the timer to reset**
193 *** **
194 *** \Sync\Async : Synchronous**
195 *** \Reentrancy : Non Reentrant**
196 *** \Parameters (in) : Channel the channel id to get its remaining tick value**
197 *** \Parameters (out) : Gpt_ValueType the remaining tick value**
198 *** \Return value : None**
199 *********
200 **Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel);**
201 *********
202 *** GLOBAL DATA PROTOTYPES**
203 **extern const Gpt_Configtype gpt_cfg[];**
204 **extern const Gpt_Driver_Cfg_Type gpt_driver_cfg[];**
205 **extern boolean Timer0Flag;**
206 **#endif /*GPT_H*/**
207 *********
208 **END OF FILE: gpt.h**
209 *********

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 621 PM 07-Sep-22 26% 🔍 ENG

9.Door Sensor Module APIs:

The screenshot shows the Visual Studio Code interface with the file `Door_Sensor.h` open. The code defines two functions: `Init_Door_Sensor(void)` and `Door_State Get_Door_State(void)`. The `Init_Door_Sensor` function initializes the door sensor module. The `Get_Door_State` function returns the current sensor reading as a `Door_State` type.

```
#include <Std_Types.h>
typedef enum{
    CLOSE,
    OPEN
}Door_State;

void Init_Door_Sensor(void);
Door_State Get_Door_State(void);
```

10.Speed Sensor Module APIs:

The screenshot shows the Visual Studio Code interface with the file `Speed_Sensor.h` open. The code defines one function: `Get_Speed(void)`. This function gets the current speed reading in km/h.

```
#include <Std_Types.h>
#define Speed uint32

void Init_Speed_Sensor(void);
Speed Get_Speed(void);
```

11.BCM Module APIs:

The screenshot shows the Visual Studio Code interface with the file `OS.h` open. The code defines the `create_Task` function from the `OS.h` header. The function takes parameters for task name, stack depth, priority, and a task handle. It also includes documentation for syntax, description, and parameters.

```
#include <Std_types.h>
#define configSTACK_DEPTH_TYPE uint16
typedef void (* TaskFunction_t)(void);
typedef struct tsTaskControlBlock * TaskHandle_t;
typedef unsigned long UBaseType_t;

//*********************************************************************
/* \Syntax      : void create_Task(TaskFunction_t pxTaskCode,
                                     const char * const pcName,
                                     const configSTACK_DEPTH_TYPE usStackDepth,
                                     void * const pvParameters,
                                     UBaseType_t uxPriority,
                                     TaskHandle_t * const pxCreatedTask)
   * \Description : create a task in the RTOS with a set op parameters
   * \Sync\Async  : Synchronous
   * \Reentrancy : Non Reentrant
   * \Parameters {in} : pxTaskCode the pointer to function that will execute as a task
                     pcName the function name as string to represent the task
                     usStackDepth
                     pvParameters task parameters
                     uxPriority task priority
                     pxCreatedTask task handler
   * \Parameters {out}: None
   * \Return value : None
******/
void create_Task(TaskFunction_t pxTaskCode, const char * const pcName,
                 const configSTACK_DEPTH_TYPE usStackDepth,
                 void * const pvParameters,
                 UBaseType_t uxPriority,
                 TaskHandle_t * const pxCreatedTask);

//*********************************************************************
/* \Syntax      : void Start_Scheduler(void)
   * \Description : start the scheduler to run the tasks and schedual them
   * \Sync\Async  : Synchronous
   * \Reentrancy : Non Reentrant
   * \Parameters {in} : None
   * \Parameters {out}: None
   * \Return value : None
******/
void Start_Scheduler(void);
```

12.OS APIs:

The screenshot shows the Visual Studio Code interface with the file `OS.h` open. The code defines the `create_Task` function from the `OS.h` header. The function takes parameters for task name, stack depth, priority, and a task handle. It also includes documentation for syntax, description, and parameters.

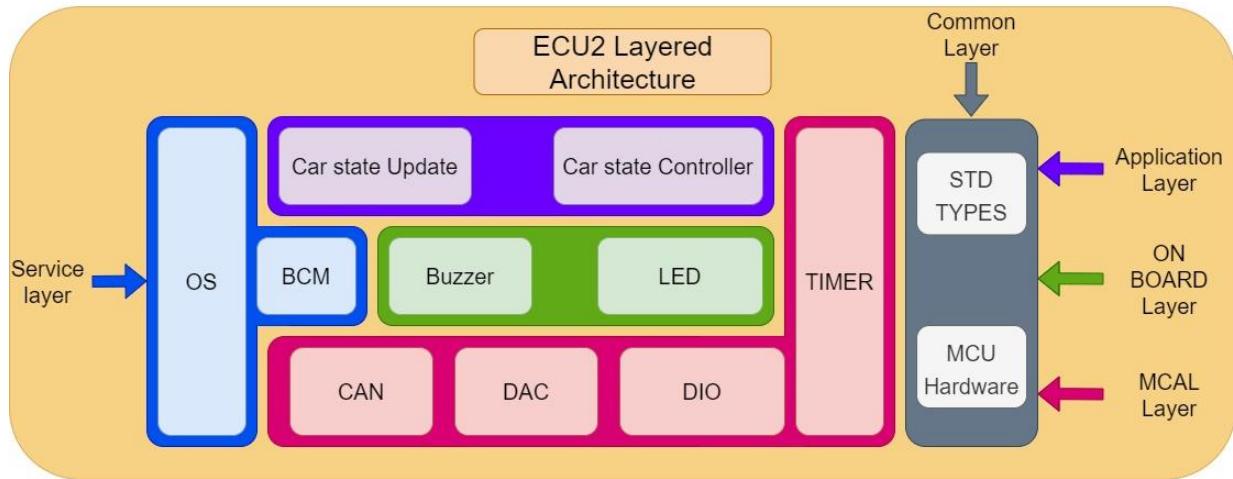
```
#include <Std_types.h>
#define configSTACK_DEPTH_TYPE uint16
typedef void (* TaskFunction_t)(void);
typedef struct tsTaskControlBlock * TaskHandle_t;
typedef unsigned long UBaseType_t;

//*********************************************************************
/* \Syntax      : void create_Task(TaskFunction_t pxTaskCode,
                                     const char * const pcName,
                                     const configSTACK_DEPTH_TYPE usStackDepth,
                                     void * const pvParameters,
                                     UBaseType_t uxPriority,
                                     TaskHandle_t * const pxCreatedTask)
   * \Description : create a task in the RTOS with a set op parameters
   * \Sync\Async  : Synchronous
   * \Reentrancy : Non Reentrant
   * \Parameters {in} : pxTaskCode the pointer to function that will execute as a task
                     pcName the function name as string to represent the task
                     usStackDepth
                     pvParameters task parameters
                     uxPriority task priority
                     pxCreatedTask task handler
   * \Parameters {out}: None
   * \Return value : None
******/
void create_Task(TaskFunction_t pxTaskCode, const char * const pcName,
                 const configSTACK_DEPTH_TYPE usStackDepth,
                 void * const pvParameters,
                 UBaseType_t uxPriority,
                 TaskHandle_t * const pxCreatedTask);

//*********************************************************************
/* \Syntax      : void Start_Scheduler(void)
   * \Description : start the scheduler to run the tasks and schedual them
   * \Sync\Async  : Synchronous
   * \Reentrancy : Non Reentrant
   * \Parameters {in} : None
   * \Parameters {out}: None
   * \Return value : None
******/
void Start_Scheduler(void);
```

ECU2:

layered architecture:



Full API details and File structure:

1.Car Speed Controller Module APIs:

```

File Edit Selection View Go Run Terminal Help Car_State_Controller.h - Embedded system design - Visual Studio Code
EXPLORER OPEN EDITORS ECU2 > src > APP > Inc > C_Car_State_Controller.h ...
C_Car_State_Controller.h ECU2\src\APP\Inc
C_Car_State_Update.h ECU2\src\APP\Inc
C_CAN.h ECU2\src\MCAL\Inc
C_DAC.h ECU2\src\MCAL\Inc
C_DIO.h ECU2\src\MCAL\Inc
C_Timer.h ECU2\src\MCAL\Inc
C_Buzzer.h ECU2\src\ONBOARD\Inc
C_LED.h ECU2\src\ONBOARD\Inc
C_OS.h ECU2\src\Service\Inc
EMBEDDED SYSTEM DESIGN ECU1\src APP MCAL OnBoard Service ECU2\src APP Inc C_Car_State_Controller.h
C_Car_State_Controller.h ECU2\src\APP\Inc
C_Car_State_Update.h ECU2\src\APP\Inc
C_Car_State_Controller.c ECU2\src\APP\Inc
C_Car_State_Update.c ECU2\src\APP\Inc
Common MCAL ONBOARD Service project3_diagrams
OUTLINE TIMELINE
Activate Windows Go to Settings to activate Windows.
Ln 26, Col 52 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 6:40 PM 07-Sep-22

```

The screenshot shows the Visual Studio Code interface with the file `Car_State_Controller.h` open. The code defines two main functions: `void Update_Buzzer(State state);` and `void Update_LED(State state);`. Both functions are annotated with detailed comments specifying their syntax, description, parameters, and return value.

```

1 #include <Car_State_Update.h>
2 #include <Std_Types.h>
3
4
5 /**
6  * @Syntax : void Update_Buzzer(State state)
7  * @Description : update the buzzer state with its new state
8  *                based on the speed , switch and door readings
9  * @Sync\Async : Synchronous
10 * @Reentrancy : Non Reentrant
11 * @Parameters (in) : state the car global state variable
12 * @Parameters (out): None
13 * @Return value : None
14 */
15 void Update_Buzzer(State state);
16
17 /**
18  * @Syntax : void Update_LED(State state)
19  * @Description : update the LED state variable with its new state
20  *                based on the speed , switch and door readings
21  * @Sync\Async : Synchronous
22  * @Reentrancy : Non Reentrant
23  * @Parameters (in) : state the car global state variable
24  * @Parameters (out): None
25  * @Return value : None
26 */
27 void Update_LED(State state);
28
29

```

2. Car Status Update Module APIs:

Car_Status_Update.h - Embedded system design - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... C Car_State_Controller.h C Car_Status_Update.h C CAN.h ECU2... C DACH C DIO.h ECU2... C Timer.h ECU2... C Buzzer.h C LED.h ...
OPEN EDITORS ECU2 > src > APP > Inc > C Car_Status_Update.h ...
1 #include <Std_types.h>
2
3 typedef struct{
4     uint8 LED_State;
5     uint8 Door_State;
6     uint8 Switch_State;
7     uint32 Current_Speed;
8 }State;
9
10
11 ****
12 /**
13 * \Syntax : void Update_Speed(uint32 Current_Speed)
14 * \Description : update the speed state variable with its new state
15 *                 received on the CAN bus via the BCM from ECU1
16 * \Sync\Async : Synchronous
17 * \Reentrancy : Non Reentrant
18 * \Parameters (in) : Current_Speed the current speed reading
19 * \Parameters (out): None
20 * \Return value : None
21 ****
22 void Update_Speed(uint32 Current_Speed);
23 ****
24 /**
25 * \Syntax : void Update_Door_State(uint8 Door_State)
26 * \Description : update the door state variable with its new state
27 *                 received on the CAN bus via the BCM from ECU1
28 * \Sync\Async : Synchronous
29 * \Reentrancy : Non Reentrant
30 * \Parameters (in) : Door_State the current door state
31 * \Parameters (out): None
32 * \Return value : None
33 ****
34
35 /**
36 * \Syntax : void Update_Switch_State(uint8 Switch_State)
37 * \Description : update the switch state variable with its new state
38 *                 received on the CAN bus via the BCM from ECU1
39 * \Sync\Async : Synchronous
40 * \Reentrancy : Non Reentrant
41 * \Parameters (in) : Switch_State the current switch state
42 * \Parameters (out): None
43 * \Return value : None
44 ****
45 void Update_Door_State(uint8 Door_State);
46
47
48 /**
49 * \Syntax : State Refresh_Car_Status(void)
50 * \Description : this function receives any message from the BCM
51 *                 and decodes it knowing which sensor state the BCM sent
52 *                 and updates the sensor returning the new state
53 *                 and updating the global state variables
54 * \Sync\Async : Synchronous
55 * \Reentrancy : Non Reentrant
56 * \Parameters (in) : None
57 * \Parameters (out): None
58 * \Return value : State a copy of the current state
59 ****
60 State Refresh_Car_Status(void);
61
62
63 /**
64 * \Syntax : State Get_Car_Status(void)
65 * \Description : returns a copy of the global current state
66 * \Sync\Async : Synchronous
67 * \Reentrancy : Non Reentrant
68

```

Activate Windows
Go to Settings to activate Windows.

Ln 66, Col 65 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 640 PM 07-Sep-22

Car_Status_Update.h - Embedded system design - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... C Car_Status_Update.h C CAN.h ECU2... C DACH C DIO.h ECU2... C Timer.h ECU2... C Buzzer.h C LED.h ...
OPEN EDITORS ECU2 > src > APP > Inc > C Car_Status_Update.h ...
31 * \Return value : None
32 ****
33 void Update_Door_State(uint8 Door_State);
34
35
36 /**
37 * \Syntax : void Update_Switch_State(uint8 Switch_State)
38 * \Description : update the switch state variable with its new state
39 *                 received on the CAN bus via the BCM from ECU1
40 * \Sync\Async : Synchronous
41 * \Reentrancy : Non Reentrant
42 * \Parameters (in) : Switch_State the current switch state
43 * \Parameters (out): None
44 * \Return value : None
45 ****
46 void Update_Switch_State(uint8 Switch_State);
47
48
49 /**
50 * \Syntax : State Refresh_Car_Status(void)
51 * \Description : this function receives any message from the BCM
52 *                 and decodes it knowing which sensor state the BCM sent
53 *                 and updates the sensor returning the new state
54 *                 and updating the global state variables
55 * \Sync\Async : Synchronous
56 * \Reentrancy : Non Reentrant
57 * \Parameters (in) : None
58 * \Parameters (out): None
59 * \Return value : State a copy of the current state
60 ****
61 State Refresh_Car_Status(void);
62
63
64 /**
65 * \Syntax : State Get_Car_Status(void)
66 * \Description : returns a copy of the global current state
67 * \Sync\Async : Synchronous
68 * \Reentrancy : Non Reentrant

```

Activate Windows
Go to Settings to activate Windows.

Ln 66, Col 65 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 640 PM 07-Sep-22

Car_State_Update.h - Embedded system design - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER Car_State_Controller.h Car_State_Update.h CAN.h ECU2...
OPEN EDITORS C Car_State_Controller.h ECU2\src\APP\Inc
  X C Car_State_Update.h ECU2\src\APP\Inc
  C CAN.h ECU2\src\MCAL\Inc
  C DACH ECU2\src\MCAL\Inc
  C DIO.h ECU2\src\MCAL\Inc
  C Timer.h ECU2\src\MCAL\Inc
  C Buzzer.h ECU2\src\ONBOARD\Inc
  C LED.h ECU2\src\ONBOARD\Inc
  C OS.h ECU2\src\Service\Inc
EMBEDDED SYSTEM DESIGN
  ECU1\src
    > APP
    > Common
    > MCAL
    > OnBoard
    > Service
  ECU2\src
    > APP
      > Inc
        C Car.State.Controller.h
        X C Car.State.Update.h
        C Car.State_Controller.c
        C Car.State_Update.c
      > Common
      > MCAL
      > ONBOARD
      > Service
    project3_diagrams
OUTLINE
TIMELINE
Ln 66, Col 65  Spaces: 4  UTF-8  CRLF  C++  Win32  Prettier  640 PM  07-Sep-22  42%  Activate Windows
Go to Settings to activate Windows.

```

3. Std Stypes Module APIs:

Platform_Types.h - Embedded system design - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER Car.State_Controller.h Car.State.Update.h Compiler.h ECU2...
OPEN EDITORS C Car.State_Controller.h ECU2\src\APP\Inc
  C Car.State.Update.h ECU2\src\APP\Inc
  X C Compiler.h ECU2\src\Common
  C Platform.Types.h ECU2\src\Common
  C CAN.h ECU2\src\MCAL\Inc
  C DACH ECU2\src\MCAL\Inc
  C DIO.h ECU2\src\MCAL\Inc
  C Timer.h ECU2\src\MCAL\Inc
  C Buzzer.h ECU2\src\ONBOARD\Inc
EMBEDDED SYSTEM DESIGN
  ECU1\src
  ECU2\src
    > APP
      > Inc
        C Car.State.Controller.h
        C Car.State.Update.h
        C Car.State_Controller.c
        C Car.State_Update.c
      > Common
        C Compiler.h
        C Mcu_Hw.h
        X C Platform.Types.h
        C Std.Types.h
      > MCAL
      > ONBOARD
      > Service
    project3_diagrams
OUTLINE
TIMELINE
Ln 1, Col 1  Spaces: 4  UTF-8  CRLF  C++  Win32  Prettier  641 PM  07-Sep-22  42%  Activate Windows
Go to Settings to activate Windows.

```

4. CAN Module APIs:

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists project files under "OPEN EDITORS" and "EMBEDDED SYSTEM DESIGN".
- Code Editor:** The main area displays the content of `CAN.h`. The code defines two functions: `Init_CAN` and `Write_CAN`, both annotated with UML-style documentation blocks.
- Status Bar:** At the bottom, it shows "Activate Windows" with a link to settings, and various status icons like battery level (42%), signal strength, and system status.

```
File Edit Selection View Go Run Terminal Help CAN.h - Embedded system design - Visual Studio Code

EXPLORER
  OPEN EDITORS
    Car_State_Controller.h ECU2\src\APP\Inc
    Car_State_Update.h ECU2\src\APP\Inc
    Compiler.h ECU2\src\Common
    Platform.Types.h ECU2\src\Common
    Std.Types.h ECU2\src\Common
    X CAN.h ECU2\src\MCAL\Inc
    DAC.h ECU2\src\MCAL\Inc
    DIO.h ECU2\src\MCAL\Inc
    Timer.h ECU2\src\MCAL\Inc

  EMBEDDED SYSTEM DESIGN
    Inc
      Car_State_Controller.h
      Car_State_Update.h
      Car_State_Controller.c
      Car_State_Update.c
    Common
      Compiler.h
      Mcu_Hw.h
      Platform.Types.h
      Std.Types.h
    MCAL
      Inc
        CAN.h
        DAC.h
        DIO.h
        Timer.h
        CAN.c
        DAC.c
        DIO.c
        Timer.c

OUTLINE
TIMELINE

CAN.h ECU2\src\MCAL\Inc
1 #include <std_types.h>
2
3
4 /*****
5 * \Syntax      : void Init_CAN(void)
6 * \Description : Initialize the CAN module
7 *
8 * \Sync\Async   : Synchronous
9 * \Reentrancy  : Non Reentrant
10 * \Parameters (in) : None
11 * \Parameters (out): None
12 * \Return value : None
13 *****/
14 void Init_CAN(void);
15
16 /
17 /*****
18 * \Syntax      : void Write_CAN(uint32 value)
19 * \Description : Write a word on the CAN bus
20 *
21 * \Sync\Async   : Synchronous
22 * \Reentrancy  : Non Reentrant
23 * \Parameters (in) : value the word to be written on CAN bus
24 * \Parameters (out): None
25 * \Return value : None
26 *****/
27 void Write_CAN(uint32 value);
28
29 /*****
30 * \Syntax      : uint32 Read_CAN(void)
31 * \Description : read a word from the CAN bus
32 *
33 * \Sync\Async   : Synchronous
34 * \Reentrancy  : Non Reentrant
35 * \Parameters (in) : None
36 * \Parameters (out): None
37 * \Return value : word received on CAN bus
38 *****/
40

Activate Windows
Go to Settings to activate Windows.

In 16, Col 1  Spaces: 4  UTF-8  CRLF  C++  Win32  Pretty
41 42% 641 PM  ENG 97.6% 20
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "EMBEDDED SYSTEM DESIGN".
- Open Editors:** Shows multiple files including `CAN.h`, `Car_State_Controller.h`, `Car_State_Update.h`, `Compiler.h`, `Platform_Types.h`, `Std_Types.h`, and `DAC.h`.
- Code Editor:** The main editor window displays the `CAN.h` header file. The code includes two function declarations: `void Init_CAN(void)` and `void Write_CAN(uint32 value)`. Each function has detailed documentation comments above it.
- Bottom Status Bar:** Shows the current file is `CAN.h`, line 24, column 30, with 641 PM as the time.
- Bottom Icons:** Includes icons for file operations like Open, Save, Find, and Run.

5. DAC Module APIs:

The screenshot shows the Visual Studio Code interface with the title "DACH - Embedded system design - Visual Studio Code". The Explorer sidebar on the left shows the project structure under "OPEN EDITORS". The main editor window displays the code for "rate_Controller.h". The code defines a function `void Init_DAC(void);` with detailed documentation comments.

```
rate_Controller.h  Car.State.Update.h  Compiler.h ECU2\src\APP\Inc  Platform.Types.h ECU2\src\  Std.Types.h ECU2\src\  CAN.h ECU2\src\MCAL\Inc  DIO.h ECU2\src\MCAL\Inc  DAC.h ECU2\src\MCAL\Inc  Car.State.Controller.h  Car.State.Update.h  Car.State.Controller.c  Car.State.Update.c  Common  Compiler.h  Mcu.Hw.h  Platform.Types.h  Std.Types.h  MCAL  Inc  CAN.h  DIO.h  DAC.h  CAN.c  DAC.c  DIO.c  Timer.c
```

```
ECU2 > src > MCAL > Inc > C DACH > ...
1  #include <Std_types.h>
2
3
4
5  ****
6  * Syntax      : void Init_DAC(void)
7  * Description : Initialize the DAC module
8  *
9  * \Sync\Async  : Synchronous
10 * \Reentrancy : Non Reentrant
11 * \Parameters (in) : None
12 * \Parameters (out) : None
13 * \Return value : None
14 ****
15 void Init_DAC(void);
16
```

Activate Windows
Go to Settings to activate Windows.

Ln 7, Col 41 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 641 PM 07-Sep-22

6. DIO Module APIs:

The screenshot shows the Visual Studio Code interface with the title "DIO.h - Embedded system design - Visual Studio Code". The Explorer sidebar on the left shows the project structure under "OPEN EDITORS". The main editor window displays the code for "Car.State.Update.h". The code defines several enum types for pin and port numbers, as well as a function `Set_Pin_Mode`.

```
Car.State.Update.h  Compiler.h ECU2\src\APP\Inc  Platform.Types.h ECU2\src\  Std.Types.h ECU2\src\  CAN.h ECU2\src\MCAL\Inc  DIO.h ECU2\src\MCAL\Inc  DAC.h ECU2\src\MCAL\Inc  Car.State.Controller.h  Car.State.Update.h  Car.State.Controller.c  Car.State.Update.c  Common  Compiler.h  Mcu.Hw.h  Platform.Types.h  Std.Types.h  MCAL  Inc  CAN.h  DIO.h  DAC.h  Timer.h  CAN.c  DAC.c  DIO.c  Timer.c
```

```
ECU2 > src > MCAL > Inc > C DIO.h > ..._unnamed_enum_08ce_3
1  #include <Std_types.h>
2
3  typedef enum{
4      Pin0,
5      Pin1,
6      Pin2,
7      Pin3,
8      Pin4,
9      Pin5,
10     Pin6,
11     Pin7;
12 }Pin_Num;
13
14 typedef enum{
15     Pin0,
16     Pin1,
17     Pin2,
18     Pin3,
19     Pin4,
20     Pin5,
21     Pin6,
22     Pin7;
23 }Port_Num;
24
25 typedef enum{
26     INPUT,
27     OUTPUT,
28 }Mode;
29
30 ****
31  * Syntax      : void Set_Pin_Mode(Pin_Num pin , Port_Num port , Mode mode)
32  * Description : set the pin direction as input or output
33  *
34  * \Sync\Async  : Synchronous
35  * \Reentrancy : Non Reentrant
36  * \Parameters (in) : pin the pin number
37  *                   port the port number
38 ****
```

Activate Windows
Go to Settings to activate Windows.

Ln 25, Col 14 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 641 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help DIO.h - Embedded system design - Visual Studio Code

EXPLORER OPEN EDITORS 1 unsaved

- Car_State_Controller.h ECU2\src\APP\Inc
- Car_State_Update.h ECU2\src\APP\Inc
- Compiler.h ECU2\src\Common
- Platform.Types.h ECU2\src\Common
- Std.Types.h ECU2\src\Common
- CAN.h ECU2\src\MCAL\Inc
- DAC.h ECU2\src\MCAL\Inc
- DIO.h ECU2\src\MCAL\Inc**
- Timer.h ECU2\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN

- Inc
- Car.State.Controller.h
- Car.State.Update.h
- Car.State_Controller.c
- Car.State_Update.c
- Common
- Compiler.h
- Mcu.Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
- CAN.h
- DAC.h
- DIO.h**
- Timer.h
- CAN.c
- DAC.c
- DIO.c
- Timer.c

> OUTLINE > TIMELINE

```
28     * \Mode;
29
30
31     *****
32     * \Syntax      : void Set_Pin_Mode(Pin_Num pin , Port_Num port , Mode mode)
33     * \Description   : set the pin direction as input or output
34
35     * \Sync\Async    : Synchronous
36     * \Reentrancy    : Non Reentrant
37     * \Parameters (in) : pin the pin number
38             port the port number
39             mode the state of pin input or output
40     * \Parameters (out): None
41     * \Return value   : None
42     *****
43 void Set_Pin_Mode(Pin_Num pin , Port_Num port , Mode mode);
44
45
46     *****
47     * \Syntax      : void Write_Pin(Pin_Num pin , Port_Num port , uint8 Level)
48     * \Description   : write a given level value on a chosen channel
49
50
51     * \Sync\Async    : Synchronous
52     * \Reentrancy    : Non Reentrant
53     * \Parameters (in) : Level the value to be written in the channel
54             pin the pin number
55             port the port number
56     * \Parameters (out): None
57     * \Return value   : None
58     *****
59 void Write_Pin(Pin_Num pin , Port_Num port , uint8 value);
60
61
62     *****
63     * \Syntax      : uint8 Read_Pin(Pin_Num pin , Port_Num port )
64     * \Description   : returns the value on a given channel
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
```

Activate Windows
Go to Settings to activate Windows.

Ln 25, Col 14 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 641 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help DIO.h - Embedded system design - Visual Studio Code

EXPLORER OPEN EDITORS 1 unsaved

- Car.State_Update.h ECU2\src\APP\Inc
- Car.State_Update.h ECU2\src\APP\Inc
- Compiler.h ECU2\src\Common
- Platform.Types.h ECU2\src\Common
- Std.Types.h ECU2\src\Common
- CAN.h ECU2\src\MCAL\Inc
- DAC.h ECU2\src\MCAL\Inc
- DIO.h ECU2\src\MCAL\Inc**
- Timer.h ECU2\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN

- Inc
- Car.State_Controller.h
- Car.State.Update.h
- Car.State_Controller.c
- Car.State_Update.c
- Common
- Compiler.h
- Mcu.Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
- CAN.h
- DAC.h
- DIO.h**
- Timer.h
- CAN.c
- DAC.c
- DIO.c
- Timer.c

> OUTLINE > TIMELINE

```
58     * \ValueIn Value : WRITE
59 void Write_Pin(Pin_Num pin , Port_Num port , uint8 value);
60
61
62     *****
63     * \Syntax      : uint8 Read_Pin(Pin_Num pin , Port_Num port )
64     * \Description   : returns the value on a given channel
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
```

Activate Windows
Go to Settings to activate Windows.

Ln 25, Col 14 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 641 PM 07-Sep-22

```
DIO.h - Embedded system design - Visual Studio Code

File Edit Selection View Go Run Terminal Help DIO.h - Embedded system design - Visual Studio Code

EXPLORER OPEN EDITORS 1 unsaved
Car_State_Update.h Compiler.h ECU2\src\APP\Inc Platform_Types.h ECU2\src\APP\Inc Std_Types.h ECU2\src\APP\Inc CAN.h ECU2\src\MCAL\Inc DAC.h ECU2\src\MCAL\Inc DIO.h ECU2\src\MCAL\Inc Timer.h ECU2\src\MCAL\Inc
ECU2 > src > MCAL > Inc > DIO.h > _unnamed_enum_08ce_3
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

void Write_Port(Port_Num PortId, uint32 Level );
*****/void Write_Port(Port_Num PortId, uint32 Level );

*****/uint32 Read_Port(Port_Num PortId);
*****/uint32 Read_Port(Port_Num PortId);

Activate Windows
Go to Settings to activate Windows.

In 25, Col 14 Spaces: 4 UTF-8 CRLF C++ Win32 Prettier 641 PM 07-Sep-22
```

7. Timer Module APIs:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "OPEN EDITORS". The "Timer.h" file is selected.
- Editor:** Displays the content of the "Timer.h" file, which includes definitions for Gpt_ValueType, Gpt_Notification_Type, Gpt_ChannelType, and Gpt_Mode_Type.
- Status Bar:** Shows "Activate Windows" and "Go to Settings to activate Windows."
- Bottom Taskbar:** Includes icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and various system status indicators like battery level and signal strength.

```
File Edit Selection View Go Run Terminal Help
File Explorer OPEN EDITORS 1 unsaved ...
C Compiler.h ECU2\src\APP\Inc
C Platform_Types.h ECU2\src\APP\Inc
C Std.Types.h ECU2\src\APP\Inc
C CAN.h ECU2\src\MCAL\Inc
C DAC.h ECU2\src\MCAL\Inc
C DIO.h ECU2\src\MCAL\Inc
C Timer.h ECU2\src\MCAL\Inc

C Car_State_Controller.h ECU2\src\APP\Inc
C Car_State_Update.h ECU2\src\APP\Inc
C Compiler.h ECU2\src\Common
C Platform_Types.h ECU2\src\Common
C Std.Types.h ECU2\src\Common
● C CAN.h ECU2\src\MCAL\Inc
● C DAC.h ECU2\src\MCAL\Inc
● C DIO.h ECU2\src\MCAL\Inc
X C Timer.h ECU2\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN
  + Inc
    C Car_State_Controller.h
    C Car_State_Update.h
    C Car_State_Controller.c
    C Car_State_Update.c
  + Common
    C Compiler.h
    C Mcu.Hw.h
    C Platform_Types.h
    C Std.Types.h
  + MCAL
    + Inc
      C CAN.h
      C DAC.h
      C DIO.h
    C Timer.h
      C CAN.c
      C DAC.c
      C DIO.c
      C Timer.c

OUTLINE
TIMELINE
```

```
*****/*Gpt_Tick_ValueType*/
typedef uint32 Gpt_ValueType;
/*Gpt_Notification_Type*/
typedef void (*GptNotification)(void);
/*Gpt_ChannelType*/
typedef enum
{
    Timer_0_16_32_Bit = 0,
    Timer_1_16_32_Bit,
    Timer_2_16_32_Bit,
    Timer_3_16_32_Bit,
    Timer_4_16_32_Bit,
    Timer_5_16_32_Bit,
    Wide_Timer_0_32_64_Bit,
    Wide_Timer_1_32_64_Bit,
    Wide_Timer_2_32_64_Bit,
    Wide_Timer_3_32_64_Bit,
    Wide_Timer_4_32_64_Bit,
    Wide_Timer_5_32_64_Bit
} Gpt_ChannelType;
/*Gpt_Mode_Type*/
typedef enum
{
    GPT_CH_MODE_ONESHOT = 0x1,
    GPT_CH_MODE_CONTINUOUS = 0x2
}
```

Activate Windows
Go to Settings to activate Windows.

In 121 Col 27 Spaces: 4 UTF-8 CR/LF C++ Win32 ⚡ Prettier 641 PM 07-Sep-22

The screenshot shows the Visual Studio Code interface with the Timer.h file open in the editor. The code defines several enums and structures related to GPT (General Purpose Timer) configurations and modes.

```
ECU2 > src > MCAL > Inc > Timer.h ...
71     typedef enum
72     {
73         GPT_CH_MODE_ONESHOT = 0x1,
74         GPT_CH_MODE_CONTINUOUS = 0x2
75     } gpt_Channel_Mode_Type;
76
77     /*Gpt_ModeType*/
78
79     typedef enum
80     {
81         GPT_MODE_NORMAL = 0,
82         GPT_MODE_SLEEP
83     } gpt_Mode_Type;
84
85     /*Gpt_PredefTimerType*/
86
87     typedef struct
88     {
89         /*TODO SEE IF YOU WILL ADD TICK FREQUENCY*/
90         boolean GPT_PREDEF_TIMER_100US_32BIT;
91         boolean GPT_PREDEF_TIMER_US_16BIT;
92         boolean GPT_PREDEF_TIMER_US_24BIT;
93         boolean GPT_PREDEF_TIMER_IUS_32BIT;
94     } gpt_Driver_Cfg_Type;
95
96     /*Gpt_ConfigType*/
97     typedef struct
98     {
99         gpt_ChannelType channelId;
100        Gpt_ValueType tickValue;
101        Gpt_ValueType maxTickValue;
102        gpt_Channel_Mode_Type channelMode;
103    } gpt_ConfigType;
```

The screenshot shows the Visual Studio Code interface with the Timer.h file open in the editor. The code includes detailed comments for the Gpt_Init and Gpt_DisableNotification functions.

```
ECU2 > src > MCAL > Inc > Timer.h ...
100     /*Gpt_ConfigType*/
101     typedef struct
102     {
103         gpt_ChannelType channelId;
104         Gpt_ValueType tickValue;
105         Gpt_ValueType maxTickValue;
106         gpt_Channel_Mode_Type channelMode;
107         gptNotification gptNotification;
108     } gpt_ConfigType;
109
110     ****
111     * \Syntax      : void Gpt_Init(const Gpt_ConfigType *ConfigPtr)
112     * \Description : Initialize the general purpose timers according
113     *                to user configurations
114     *
115     * \Sync\Async   : Synchronous
116     * \Reentrancy  : Non Reentrant
117     * \Parameters (in) : ConfigPtr the array of user configurations for each selected timer
118     * \Parameters (out): None
119     * \Return value : None
120     ****
121
122     void Gpt_Init(const gpt_ConfigType *ConfigPtr);
123
124     ****
125     * \Syntax      : void Gpt_DisableNotification(Gpt_ChannelType Channel)
126     * \Description : disables the Interrupt on a given channel
127     *
128     * \Sync\Async   : Synchronous
129     * \Reentrancy  : Non Reentrant
130     * \Parameters (in) : Channel channel to disable its interrupts
131     * \Parameters (out): None
132     * \Return value : None
133     ****
134
135     void Gpt_DisableNotification(Gpt_ChannelType Channel);
```

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS 1 unsaved

- Car_State_Controller.h ECU2\src\APP\Inc
- Car_State_Update.h ECU2\src\APP\Inc
- Compiler.h ECU2\src\Common
- Platform.Types.h ECU2\src\Common
- Std.Types.h ECU2\src\Common
- CAN.h ECU2\src\MCAL\Inc
- DAC.h ECU2\src\MCAL\Inc
- DIO.h ECU2\src\MCAL\Inc
- Timer.h ECU2\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN

- Inc
- Car.State.Controller.h
- Car.State.Update.h
- Car.State_Controller.c
- Car.State_Update.c
- Common
- Compiler.h
- Mcu.Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
- CAN.h
- DAC.h
- DIO.h
- Timer.h
- CAN.c
- DAC.c
- DIO.c
- Timer.c

OUTLINE **TIMELINE**

```
126 //*****
127 * \Syntax      : void Gpt_DisableNotification(Gpt_ChannelType Channel)
128 * \Description : disables the interrupt on a given channel
129 *
130 * \Sync\Async   : Synchronous
131 * \Reentrancy   : Non Reentrant
132 * \Parameters (in) : Channel channel to disable its interrupts
133 * \Parameters (out): None
134 * \Return value : None
135 *****/
137 void Gpt_DisableNotification(Gpt_ChannelType Channel);
138
139 //*****
140 * \Syntax      : void Gpt_EnableNotification(Gpt_ChannelType Channel)
141 * \Description : enables the interrupt on a given channel
142 *
143 * \Sync\Async   : Synchronous
144 * \Reentrancy   : Non Reentrant
145 * \Parameters (in) : Channel channel to enable its interrupts
146 * \Parameters (out): None
147 * \Return value : None
148 *****/
149 void Gpt_EnableNotification(Gpt_ChannelType Channel);
150
151 //*****
152 * \Syntax      : void Gpt_StartTimer(Gpt_ChannelType Channel,Gpt_ValueType Value)
153 * \Description : enables the timer TEN field to start the timer
154 *
155 * \Sync\Async   : Synchronous
156 * \Reentrancy   : Non Reentrant
157 * \Parameters (in) : Channel the channel number to start
158 *                   Value the preload value to start the timer with
159 * \Parameters (out): None
160 * \Return value : None
161 *****/
162
```

Activate Windows
Go to Settings to activate Windows.

Ln 121, Col 27 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 642 PM 07-Sep-22

File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS 1 unsaved

- Car.State_Controller.h ECU2\src\APP\Inc
- Car.State_Update.h ECU2\src\APP\Inc
- Compiler.h ECU2\src\Common
- Platform.Types.h ECU2\src\Common
- Std.Types.h ECU2\src\Common
- CAN.h ECU2\src\MCAL\Inc
- DAC.h ECU2\src\MCAL\Inc
- DIO.h ECU2\src\MCAL\Inc
- Timer.h ECU2\src\MCAL\Inc

EMBEDDED SYSTEM DESIGN

- Inc
- Car.State_Controller.h
- Car.State.Update.h
- Car.State_Controller.c
- Car.State_Update.c
- Common
- Compiler.h
- Mcu.Hw.h
- Platform.Types.h
- Std.Types.h
- MCAL
- Inc
- CAN.h
- DAC.h
- DIO.h
- Timer.h
- CAN.c
- DAC.c
- DIO.c
- Timer.c

OUTLINE **TIMELINE**

```
152 //*****
153 * \Syntax      : void Gpt_StartTimer(Gpt_ChannelType Channel,Gpt_ValueType Value)
154 * \Description : enables the timer TEN field to start the timer
155 *
156 * \Sync\Async   : Synchronous
157 * \Reentrancy   : Non Reentrant
158 * \Parameters (in) : Channel the channel number to start
159 *                   Value the preload value to start the timer with
160 * \Parameters (out): None
161 * \Return value : None
162 *****/
164 void Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value);
165
166 //*****
167 * \Syntax      : void Gpt_StopTimer(Gpt_ChannelType Channel)
168 * \Description : disables the timer TEN field to stop the timer
169 *
170 * \Sync\Async   : Synchronous
171 * \Reentrancy   : Non Reentrant
172 * \Parameters (in) : Channel the channel number to stop
173 * \Parameters (out): None
174 * \Return value : None
175 *****/
176 void Gpt_StopTimer(Gpt_ChannelType Channel);
177
178 //*****
179 * \Syntax      : Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel)
180 * \Description : returns the number of counted ticks from the start of the timer
181 *
182 * \Sync\Async   : Synchronous
183 * \Reentrancy   : Non Reentrant
184 * \Parameters (in) : Channel the channel id to get its elapsed tick value
185 * \Parameters (out): Gpt_ValueType the elapsed tick value
186 * \Return value : None
187 *****/
188
```

Activate Windows
Go to Settings to activate Windows.

Ln 121, Col 27 Spaces: 4 UTF-8 CRLF C++ Win32 ⚡ Prettier 642 PM 07-Sep-22

The screenshot shows the Visual Studio Code interface with the Timer.h file open. The code defines two functions: Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel) and Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel). Both functions are marked as synchronous and non-reentrant. The code also includes global data prototypes for gpt_cfg[] and gpt_driver_cfg[].

```
179 //*****
180 * \Syntax      : Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel)
181 * \Description  : returns the number of counted ticks from the start of the timer
182 *
183 * \Sync\Async   : Synchronous
184 * \Reentrancy   : Non Reentrant
185 * \Parameters (in) : Channel the channel id to get its elapsed tick value
186 * \Parameters (out): gpt_ValueType the elapsed tick value
187 * \Return value  : None
188 *****/
189 Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType Channel);
190
191 //*****
192 * \Syntax      : Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel)
193 * \Description  : returns the number of ticks still remaining for the timer to reset
194 *
195 * \Sync\Async   : Synchronous
196 * \Reentrancy   : Non Reentrant
197 * \Parameters (in) : Channel the channel id to get its remaining tick value
198 * \Parameters (out): Gpt_ValueType the remaining tick value
199 * \Return value  : None
200 *****/
201 Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel);
202
203 //*****
204 * GLOBAL DATA PROTOTYPES
205 *****/
206 extern const gpt_ConfigType gpt_cfg[];
207 extern const gpt_Driver_Cfg_Type gpt_Driver_Cfg[];
208 extern boolean Timer0Flag;
209
210 #endif /*GPT_H*/
211
212 //*****
213 * END OF FILE: Gpt.h
214 *****/
215
```

This screenshot is identical to the one above, showing the Timer.h file in Visual Studio Code. The code defines the same two functions and includes the same global data prototypes. The interface and file content are identical to the first screenshot.

```
192 //*****
193 * \Syntax      : Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel)
194 * \Description  : returns the number of ticks still remaining for the timer to reset
195 *
196 * \Sync\Async   : Synchronous
197 * \Reentrancy   : Non Reentrant
198 * \Parameters (in) : Channel the channel id to get its remaining tick value
199 * \Parameters (out): Gpt_ValueType the remaining tick value
200 * \Return value  : None
201 *****/
202 Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel);
203
204 //*****
205 * GLOBAL DATA PROTOTYPES
206 *****/
207
208 extern const gpt_ConfigType gpt_cfg[];
209 extern const gpt_Driver_Cfg_Type gpt_Driver_Cfg[];
210 extern boolean Timer0Flag;
211
212 #endif /*GPT_H*/
213
214 //*****
215 * END OF FILE: Gpt.h
216 *****/
217
```

8. Buzzer Module APIs:

The screenshot shows the Visual Studio Code interface with the file `Buzzer.h` open. The code defines several functions for initializing and controlling a buzzer:

```
ECU2 > src > ONBOARD > Inc > C_Buzzer.h ...  
1 #include <Std_Types.h>  
2 #include <DIO.h>  
3 /*\Syntax : uint8 Init_Buzzer(Pin_Num pin , Port_Num port);  
 * \Description : Initialize a Buzzer with port and pin numbers  
 * \Parameters (in) : pin Buzzer pin number  
 * \Parameters (out) : None  
 * \Return value : uint32 buzzer id  
 */  
14 uint32 Init_Buzzer(Pin_Num pin , Port_Num port);  
15 /*\Syntax : void Buzzer_HIGH(uint32 buzzerID)  
 * \Description : set the value on the buzzer to high  
 * \Parameters (in) : buzzerID the buzzer number  
 * \Parameters (out) : None  
 * \Return value : None  
 */  
25 void Buzzer_HIGH(uint32 buzzerID);  
26 /*\Syntax : void Buzzer_LOW(uint32 buzzerID)  
 * \Description : set the value on the buzzer to low  
 * \Parameters (in) : buzzerID the buzzer number  
 * \Parameters (out) : None  
 * \Return value : None  
 */  
35 void Buzzer_LOW(uint32 buzzerID);  
37
```

The code uses Doxygen-style comments to describe each function's parameters and return value.

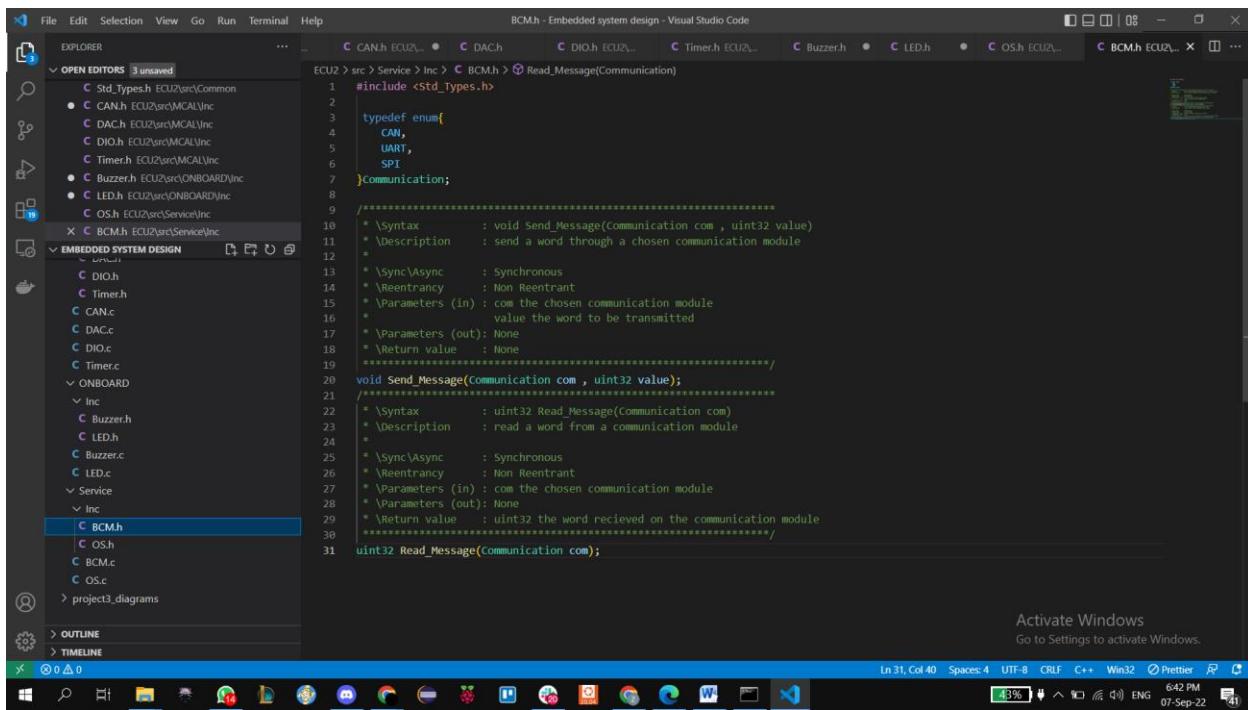
9. LED Module APIs:

The screenshot shows the Visual Studio Code interface with the file `LED.h` open. The code defines functions for initializing and controlling an LED:

```
ECU2 > src > ONBOARD > Inc > C_LED.h ...  
1 #include <Std_Types.h>  
2 #include <DIO.h>  
3 /*\Syntax : uint8 Init_LED(Pin_Num pin , Port_Num port);  
 * \Description : initialize a LED with port and pin numbers  
 * \Parameters (in) : pin LED pin number  
 * \Parameters (out) : None  
 * \Return value : uint32 LED id  
 */  
14 uint32 Init_LED(Pin_Num pin , Port_Num port);  
15 /*\Syntax : void LED_HIGH(uint32 LED_ID)  
 * \Description : set the value on the LED to high  
 * \Parameters (in) : LED_ID the LED number  
 * \Parameters (out) : None  
 * \Return value : None  
 */  
25 void LED_HIGH(uint32 LED_ID);  
26 /*\Syntax : void LED_LOW(uint32 LED_ID)  
 * \Description : set the value on the LED to low  
 * \Parameters (in) : LED_ID the LED number  
 * \Parameters (out) : None  
 * \Return value : None  
 */  
36 void LED_LOW(uint32 LED_ID);
```

The code uses Doxygen-style comments to describe each function's parameters and return value.

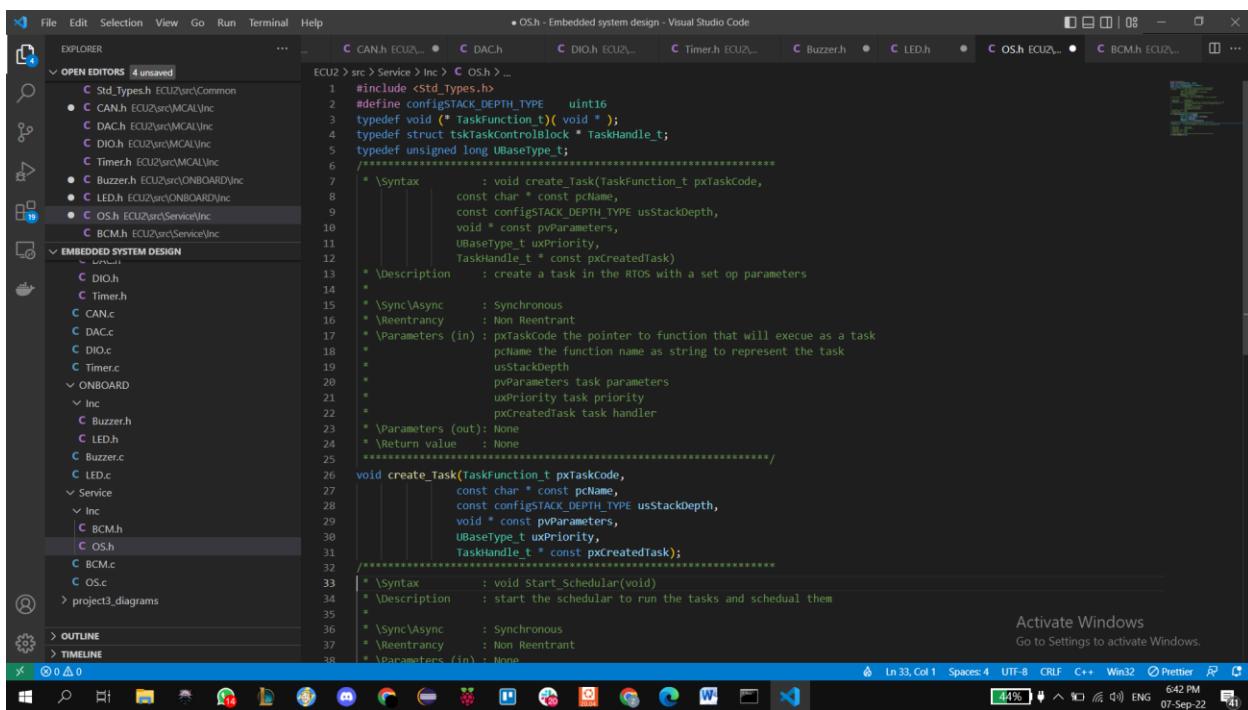
10. BCM Module APIs:



The screenshot shows the Visual Studio Code interface with the file `BCM.h` open. The code defines a communication module with methods for sending and reading messages.

```
BCM.h - Embedded system design - Visual Studio Code
File Edi Selection View Go Run Terminal Help
OPEN EDITORS 3 unsaved
EXPLORER
ECU2 > src > Service > Inc > BCM.h > Read_Message(Communication)
1 #include <Std_types.h>
2
3 typedef enum{
4     CAN,
5     UART,
6     SPI
7 }Communication;
8
9 /**
10  * \Syntax      : void Send_Message(Communication com , uint32 value)
11 * \Description : send a word through a chosen communication module
12 *
13 * \Sync\Async  : Synchronous
14 * \Reentrancy : Non Reentrant
15 * \Parameters (in) : com the chosen communication module
16 *                   value the word to be transmitted
17 * \Parameters (out): None
18 * \Return value : None
19 */
20 void Send_Message(Communication com , uint32 value);
21
22 /**
23  * \Syntax      : uint32 Read_Message(Communication com)
24  * \Description : read a word from a communication module
25 *
26 * \Sync\Async  : Synchronous
27 * \Reentrancy : Non Reentrant
28 * \Parameters (in) : com the chosen communication module
29 * \Parameters (out): None
30 * \Return value : uint32 the word received on the communication module
31 */
32 uint32 Read_Message(Communication com);
```

11. OS Module APIs:



The screenshot shows the Visual Studio Code interface with the file `OS.h` open. The code defines a task creation API.

```
OS.h - Embedded system design - Visual Studio Code
File Edit Selection View Go Run Terminal Help
OPEN EDITORS 4 unsaved
EXPLORER
ECU2 > src > Service > Inc > OS.h ...
1 #include <Std_types.h>
2 #define configSTACK_DEPTH_TYPE uint16
3 typedef void (* TaskFunction_t)( void * );
4 typedef struct tsksTaskControlBlock * TaskHandle_t;
5 typedef unsigned long UBaseType_t;
6
7 /**
8  * \Syntax      : void create_Task(TaskFunction_t pxTaskCode,
9  *                      const char * const pcName,
10 *                      const configSTACK_DEPTH_TYPE usStackDepth,
11 *                      void * const pvParameters,
12 *                      UBaseType_t uxPriority,
13 *                      TaskHandle_t * const pxCreatedTask)
14  * \Description : create a task in the RTOS with a set of parameters
15 *
16 * \Sync\Async  : Synchronous
17 * \Reentrancy : Non Reentrant
18 * \Parameters (in) : pxTaskCode the pointer to function that will execute as a task
19 *                   pcName the function name as string to represent the task
20 *                   usStackDepth
21 *                   pvParameters task parameters
22 *                   uxPriority task priority
23 *                   pxCreatedTask task handler
24  * \Parameters (out): None
25 * \Return value : None
26 */
27 void create_Task(TaskFunction_t pxTaskCode,
28                  const char * const pcName,
29                  const configSTACK_DEPTH_TYPE usStackDepth,
30                  void * const pvParameters,
31                  UBaseType_t uxPriority,
32                  TaskHandle_t * const pxCreatedTask);
33
34 /**
35  * \Syntax      : void Start_Scheduler(void)
36  * \Description : start the scheduler to run the tasks and schedule them
37 *
38 * \Sync\Async  : Synchronous
39 * \Reentrancy : Non Reentrant
40 * \Parameters (in) : None
```

The screenshot shows a Visual Studio Code window titled "OS.h - Embedded system design - Visual Studio Code". The interface includes a top navigation bar with File, Edit, Selection, View, Go, Run, Terminal, and Help. Below the navigation bar is the Explorer sidebar, which lists several source files under "OPEN EDITORS" and "EMBEDDED SYSTEM DESIGN". The main editor area displays a C++ code snippet for task creation:

```
ECU2 > src > Service > Inc > OS.h > ...
18 * \Name      : void create_Task(TaskFunction_t pxtaskCode,
19 *                                const char * const pcName,
20 *                                const configSTACK_DEPTH_TYPE usStackDepth,
21 *                                pvParameters taskParameters
22 *                                uxPriority taskPriority
23 *                                pxCreatedTask taskHandler
24 * \Parameters (out): None
25 * \Return value   : None
26 ****
27 void create_Task(TaskFunction_t pxtaskCode,
28                   const char * const pcName,
29                   const configSTACK_DEPTH_TYPE usStackDepth,
30                   void * const pvParameters,
31                   uBaseType_t uxPriority,
32                   TaskHandle_t * const pxCreatedTask);
33 ****
34 * \Syntax       : void Start_Schedular(void)
35 * \Description  : start the scheduler to run the tasks and schedual them
36 * \Sync\Async   : Synchronous
37 * \Reentrancy   : Non Reentrant
38 * \Parameters (in): None
39 * \Parameters (out): None
40 * \Return value : None
41 ****
42 void Start_Schedular(void);
```

The status bar at the bottom of the window shows "Activate Windows" and "Go to Settings to activate Windows." It also displays "Ln 7, Col 3" and other system information like battery level (44%), date (07-Sep-22), and time (6:43 PM).