

GOS2022

1.1

Generated by Doxygen 1.8.20

1 GOS2022 OS documentation	1
1.1 Revision history	1
2 Module Index	3
2.1 Modules	3
3 Data Structure Index	5
3.1 Data Structures	5
4 File Index	7
4.1 File List	7
5 Module Documentation	9
5.1 Global definitions	9
5.1.1 Detailed Description	9
5.1.2 Macro Definition Documentation	9
5.1.2.1 GLOBAL_STACK	10
5.1.2.2 GOS_ASM	10
5.1.2.3 GOS_CONST	10
5.1.2.4 GOS_DEFAULT_TASK_ID	10
5.1.2.5 GOS_EXTERN	10
5.1.2.6 GOS_INLINE	11
5.1.2.7 GOS_INVALID_TASK_ID	11
5.1.2.8 GOS_NAKED	11
5.1.2.9 GOS_NOP	11
5.1.2.10 GOS_STATIC	11
5.1.2.11 GOS_STATIC_INLINE	12
5.1.2.12 GOS_TASK_IDLE_PRIO	12
5.1.2.13 GOS_TASK_MAX_BLOCK_TIME_MS	12
5.1.2.14 GOS_TASK_MAX_PRIO_LEVELS	12
5.1.2.15 GOS_UNUSED	12
5.1.2.16 MAIN_STACK	13
5.1.2.17 NULL	13
5.1.2.18 RAM_SIZE	13
5.1.2.19 RAM_START	13
5.2 Control macros for scheduling and ISR state setting	14
5.2.1 Detailed Description	14
5.2.2 Macro Definition Documentation	14
5.2.2.1 GOS_ATOMIC_ENTER	14
5.2.2.2 GOS_ATOMIC_EXIT	14
5.2.2.3 GOS_DISABLE_SCHED	15
5.2.2.4 GOS_ENABLE_SCHED	15
5.2.2.5 GOS_ISR_ENTER	15
5.2.2.6 GOS_ISR_EXIT	15

5.3 Privilege bit definitions	16
5.3.1 Detailed Description	16
5.3.2 Macro Definition Documentation	16
5.3.2.1 GOS_PRIV_RESERVED_3	16
5.3.2.2 GOS_PRIV_RESERVED_4	16
5.3.2.3 GOS_PRIV_RESERVED_5	16
5.3.2.4 GOS_PRIV_SIGNALING	17
5.3.2.5 GOS_PRIV_TASK_MANIPULATE	17
5.3.2.6 GOS_PRIV_TASK_PRIO_CHANGE	17
5.3.2.7 GOS_PRIV_TRACE	17
5.4 Basic data type definitions to be used in a GOS system	18
5.4.1 Detailed Description	18
5.5 Task-related type definitions	19
5.5.1 Detailed Description	19
5.6 Hook function type definitions	20
5.6.1 Detailed Description	20
5.7 Time-related definitions	21
5.7.1 Detailed Description	21
5.8 Kernel functions	22
5.8.1 Detailed Description	23
5.8.2 Function Documentation	23
5.8.2.1 gos_kernelCalculateTaskCpuUsages()	23
5.8.2.2 gos_kernelDelayMs()	24
5.8.2.3 gos_kernelDelayUs()	25
5.8.2.4 gos_kernelDump()	25
5.8.2.5 gos_kernelGetCpuUsage()	26
5.8.2.6 gos_kernelGetMaxCpuLoad()	27
5.8.2.7 gos_kernelGetSysTicks()	28
5.8.2.8 gos_kernellInit()	28
5.8.2.9 gos_kernellsCallerlsr()	29
5.8.2.10 gos_kernelPrivilegedModeSetRequired()	30
5.8.2.11 gos_kernelRegisterIdleHook()	30
5.8.2.12 gos_kernelRegisterPreResetHook()	31
5.8.2.13 gos_kernelRegisterPrivilegedHook()	31
5.8.2.14 gos_kernelRegisterSwapHook()	32
5.8.2.15 gos_kernelRegisterSysTickHook()	32
5.8.2.16 gos_kernelReschedule()	33
5.8.2.17 gos_kernelReset()	34
5.8.2.18 gos_kernelSetMaxCpuLoad()	36
5.8.2.19 gos_kernelStart()	36
5.8.2.20 gos_kernelSubscribeDumpReadySignal()	37
5.8.2.21 gos_taskSubscribeDeleteSignal()	38

5.9 Task-related functions	39
5.9.1 Detailed Description	40
5.9.2 Function Documentation	40
5.9.2.1 gos_taskAddPrivilege()	40
5.9.2.2 gos_taskBlock()	41
5.9.2.3 gos_taskDelete()	42
5.9.2.4 gos_taskGetCurrentId()	43
5.9.2.5 gos_taskGetData()	44
5.9.2.6 gos_taskGetDataByIndex()	45
5.9.2.7 gos_taskGetId()	46
5.9.2.8 gos_taskGetName()	47
5.9.2.9 gos_taskGetNumber()	47
5.9.2.10 gos_taskGetOriginalPriority()	48
5.9.2.11 gos_taskGetPriority()	48
5.9.2.12 gos_taskGetPrivileges()	49
5.9.2.13 gos_taskRegister()	50
5.9.2.14 gos_taskRegisterTasks()	52
5.9.2.15 gos_taskRemovePrivilege()	52
5.9.2.16 gos_taskResume()	53
5.9.2.17 gos_taskSetOriginalPriority()	54
5.9.2.18 gos_taskSetPriority()	55
5.9.2.19 gos_taskSetPrivileges()	57
5.9.2.20 gos_taskSleep()	57
5.9.2.21 gos_taskSuspend()	59
5.9.2.22 gos_taskUnblock()	60
5.9.2.23 gos_taskWakeup()	61
5.9.2.24 gos_taskYield()	62
5.10 Platform initializer weak functions	63
5.10.1 Detailed Description	63
5.10.2 Function Documentation	63
5.10.2.1 gos_platformDriverInit()	63
5.10.2.2 gos_userApplicationInit()	64
5.11 Trace formatter macros	65
5.11.1 Detailed Description	65
5.12 System monitoring message structures	66
5.12.1 Detailed Description	66
6 Data Structure Documentation	67
6.1 gos_driver_functions_t Struct Reference	67
6.1.1 Detailed Description	67
6.2 gos_gcpChannelFunctions_t Struct Reference	68
6.2.1 Detailed Description	68

6.3 gos_gcpHeaderFrame_t Struct Reference	68
6.3.1 Detailed Description	68
6.4 gos_initStruct_t Struct Reference	69
6.4.1 Detailed Description	69
6.5 gos_message_t Struct Reference	69
6.5.1 Detailed Description	69
6.6 gos_messageWaiterDesc_t Struct Reference	70
6.6.1 Detailed Description	70
6.7 gos_mutex_t Struct Reference	70
6.7.1 Detailed Description	71
6.8 gos_queue_t Struct Reference	71
6.8.1 Detailed Description	72
6.9 gos_queueDescriptor_t Struct Reference	72
6.9.1 Detailed Description	72
6.10 gos_queueElement_t Struct Reference	72
6.10.1 Detailed Description	72
6.11 gos_runtime_t Struct Reference	73
6.11.1 Detailed Description	73
6.12 gos_shellCommand_t Struct Reference	73
6.12.1 Detailed Description	74
6.13 gos_signalDescriptor_t Struct Reference	74
6.13.1 Detailed Description	74
6.14 gos_signallInvokeDescriptor Struct Reference	74
6.14.1 Detailed Description	75
6.15 gos_sysmonCpuUsageMessage_t Struct Reference	75
6.15.1 Detailed Description	75
6.16 gos_sysmonLut_t Struct Reference	75
6.16.1 Detailed Description	76
6.17 gos_sysmonPingMessage_t Struct Reference	76
6.17.1 Detailed Description	76
6.18 gos_sysmonSysruntimeGetResultMessage_t Struct Reference	76
6.18.1 Detailed Description	77
6.19 gos_sysmonSystimeSetMessage_t Struct Reference	77
6.19.1 Detailed Description	77
6.20 gos_sysmonSystimeSetResultMessage_t Struct Reference	78
6.20.1 Detailed Description	78
6.21 gos_sysmonTaskData_t Struct Reference	78
6.21.1 Detailed Description	79
6.22 gos_sysmonTaskDataGetMessage_t Struct Reference	79
6.22.1 Detailed Description	79
6.23 gos_sysmonTaskDataMessage_t Struct Reference	80
6.23.1 Detailed Description	80

6.24 gos_sysmonTaskModifyMessage_t Struct Reference	80
6.24.1 Detailed Description	81
6.25 gos_sysmonTaskModifyResultMessage_t Struct Reference	81
6.25.1 Detailed Description	81
6.26 gos_sysmonTaskVariableData Struct Reference	81
6.26.1 Detailed Description	82
6.27 gos_sysmonTaskVariableDataMessage_t Struct Reference	82
6.27.1 Detailed Description	83
6.28 gos_sysmonUserMessageDescriptor_t Struct Reference	83
6.28.1 Detailed Description	83
6.29 gos_taskDescriptor_t Struct Reference	84
6.29.1 Detailed Description	85
6.30 gos_time_t Struct Reference	85
6.30.1 Detailed Description	86
6.31 gos_trigger_t Struct Reference	86
6.31.1 Detailed Description	86
7 File Documentation	87
7.1 GOS2022/OS/driver/inc/gos_crc_driver.h File Reference	87
7.1.1 Detailed Description	88
7.1.2 Function Documentation	88
7.1.2.1 gos_crcDriverGetCrc()	88
7.2 GOS2022/OS/driver/inc/gos_driver.h File Reference	89
7.2.1 Detailed Description	90
7.2.2 Function Documentation	90
7.2.2.1 gos_driverInit()	90
7.3 GOS2022/OS/driver/inc/gos_shell_driver.h File Reference	91
7.3.1 Detailed Description	92
7.3.2 Typedef Documentation	92
7.3.2.1 gos_shellDriverReceiveChar_t	93
7.3.2.2 gos_shellDriverTransmitString_t	93
7.3.3 Function Documentation	93
7.3.3.1 gos_shellDriverReceiveChar()	93
7.3.3.2 gos_shellDriverTransmitString()	94
7.4 GOS2022/OS/driver/inc/gos_sysmon_driver.h File Reference	95
7.4.1 Detailed Description	96
7.4.2 Typedef Documentation	96
7.4.2.1 gos_sysmonDriverReceive_t	97
7.4.2.2 gos_sysmonDriverTransmit_t	97
7.4.3 Function Documentation	97
7.4.3.1 gos_sysmonDriverReceive()	97
7.4.3.2 gos_sysmonDriverTransmit()	98

7.5 GOS2022/OS/driver/inc/gos_timer_driver.h File Reference	99
7.5.1 Detailed Description	100
7.5.2 Typedef Documentation	100
7.5.2.1 gos_timerDriverSysTimerGetVal_t	100
7.5.3 Function Documentation	100
7.5.3.1 gos_timerDriverSysTimerGet()	100
7.6 GOS2022/OS/driver/inc/gos_trace_driver.h File Reference	101
7.6.1 Detailed Description	102
7.6.2 Typedef Documentation	102
7.6.2.1 gos_traceDriverTransmitString_t	103
7.6.2.2 gos_traceDriverTransmitString_Unsafe_t	103
7.6.3 Function Documentation	103
7.6.3.1 gos_traceDriverTransmitString()	103
7.6.3.2 gos_traceDriverTransmitString_Unsafe()	104
7.7 GOS2022/OS/driver/src/gos_crc_driver.c File Reference	105
7.7.1 Detailed Description	106
7.7.2 Macro Definition Documentation	107
7.7.2.1 CRC_INITIAL_VALUE	107
7.7.2.2 CRC_POLYNOMIAL_VALUE	107
7.7.3 Function Documentation	107
7.7.3.1 gos_crcDriverGetCrc()	107
7.8 GOS2022/OS/driver/src/gos_driver.c File Reference	108
7.8.1 Detailed Description	109
7.8.2 Function Documentation	109
7.8.2.1 gos_driverInit()	109
7.9 GOS2022/OS/driver/src/gos_shell_driver.c File Reference	110
7.9.1 Detailed Description	110
7.9.2 Function Documentation	111
7.9.2.1 gos_shellDriverReceiveChar()	111
7.9.2.2 gos_shellDriverTransmitString()	111
7.9.3 Variable Documentation	112
7.9.3.1 formattedBuffer	112
7.10 GOS2022/OS/driver/src/gos_sysmon_driver.c File Reference	113
7.10.1 Detailed Description	113
7.10.2 Function Documentation	114
7.10.2.1 gos_sysmonDriverReceive()	114
7.10.2.2 gos_sysmonDriverTransmit()	114
7.11 GOS2022/OS/driver/src/gos_timer_driver.c File Reference	116
7.11.1 Detailed Description	117
7.11.2 Function Documentation	117
7.11.2.1 gos_timerDriverSysTimerGet()	117
7.12 GOS2022/OS/driver/src/gos_trace_driver.c File Reference	118

7.12.1 Detailed Description	119
7.12.2 Function Documentation	119
7.12.2.1 gos_traceDriverTransmitString()	119
7.12.2.2 gos_traceDriverTransmitString_Unsafe()	120
7.13 GOS2022/OS/kernel/inc/gos_bootloader_config.h File Reference	121
7.13.1 Detailed Description	123
7.13.2 Macro Definition Documentation	123
7.13.2.1 ARM_CORTEX_M4	123
7.13.2.2 CFG_GCP_CHANNELS_MAX_NUMBER	123
7.13.2.3 CFG_IDLE_TASK_STACK_SIZE	123
7.13.2.4 CFG_MESSAGE_MAX_LENGTH	124
7.13.2.5 CFG_MESSAGE_MAX_NUMBER	124
7.13.2.6 CFG_MESSAGE_MAX_WAITER_IDS	124
7.13.2.7 CFG_MESSAGE_MAX_WAITERS	124
7.13.2.8 CFG_QUEUE_MAX_ELEMENTS	124
7.13.2.9 CFG_QUEUE_MAX_LENGTH	125
7.13.2.10 CFG_QUEUE_MAX_NAME_LENGTH	125
7.13.2.11 CFG_QUEUE_MAX_NUMBER	125
7.13.2.12 CFG_QUEUE_USE_NAME	125
7.13.2.13 CFG_RESET_ON_ERROR	125
7.13.2.14 CFG_RESET_ON_ERROR_DELAY_MS	126
7.13.2.15 CFG_SCHED_COOPERATIVE	126
7.13.2.16 CFG_SHELL_COMMAND_BUFFER_SIZE	126
7.13.2.17 CFG_SHELL_MAX_COMMAND_LENGTH	126
7.13.2.18 CFG_SHELL_MAX_COMMAND_NUMBER	126
7.13.2.19 CFG_SHELL_MAX_PARAMS_LENGTH	127
7.13.2.20 CFG_SHELL_STARTUP_DELAY_MS	127
7.13.2.21 CFG_SHELL_USE_SERVICE	127
7.13.2.22 CFG_SIGNAL_MAX_NUMBER	127
7.13.2.23 CFG_SIGNAL_MAX_SUBSCRIBERS	127
7.13.2.24 CFG_SYSMON_GCP_CHANNEL_NUM	128
7.13.2.25 CFG_SYSMON_MAX_USER_MESSAGES	128
7.13.2.26 CFG_SYSMON_USE_SERVICE	128
7.13.2.27 CFG_SYSTEM_TASK_STACK_SIZE	128
7.13.2.28 CFG_TARGET_CPU	128
7.13.2.29 CFG_TASK_MAX_NAME_LENGTH	129
7.13.2.30 CFG_TASK_MAX_NUMBER	129
7.13.2.31 CFG_TASK_MAX_STACK_SIZE	129
7.13.2.32 CFG_TASK_MESSAGE_DAEMON_PRIO	129
7.13.2.33 CFG_TASK_MESSAGE_DAEMON_STACK	129
7.13.2.34 CFG_TASK_MIN_STACK_SIZE	130
7.13.2.35 CFG_TASK_SHELL_DAEMON_PRIO	130

7.13.2.36 CFG_TASK_SHELL_DAEMON_STACK	130
7.13.2.37 CFG_TASK_SIGNAL_DAEMON_PRIO	130
7.13.2.38 CFG_TASK_SIGNAL_DAEMON_STACK	130
7.13.2.39 CFG_TASK_SYS_PRIO	131
7.13.2.40 CFG_TASK_SYSMON_DAEMON_PRIO	131
7.13.2.41 CFG_TASK_SYSMON_DAEMON_STACK	131
7.13.2.42 CFG_TASK_TIME_DAEMON_PRIO	131
7.13.2.43 CFG_TASK_TIME_DAEMON_STACK	131
7.13.2.44 CFG_TASK_TRACE_DAEMON_PRIO	132
7.13.2.45 CFG_TASK_TRACE_DAEMON_STACK	132
7.13.2.46 CFG_TRACE_MAX_LENGTH	132
7.13.2.47 CFG_USE_PRIO_INHERITANCE	132
7.14 GOS2022/OS/kernel/inc/gos_config.h File Reference	133
7.14.1 Detailed Description	134
7.14.2 Macro Definition Documentation	134
7.14.2.1 ARM_CORTEX_M4	135
7.14.2.2 CFG_GCP_CHANNELS_MAX_NUMBER	135
7.14.2.3 CFG_IDLE_TASK_STACK_SIZE	135
7.14.2.4 CFG_MESSAGE_MAX_ADDRESSEES	135
7.14.2.5 CFG_MESSAGE_MAX_LENGTH	135
7.14.2.6 CFG_MESSAGE_MAX_NUMBER	136
7.14.2.7 CFG_MESSAGE_MAX_WAITER_IDS	136
7.14.2.8 CFG_MESSAGE_MAX_WAITERS	136
7.14.2.9 CFG_QUEUE_MAX_ELEMENTS	136
7.14.2.10 CFG_QUEUE_MAX_LENGTH	136
7.14.2.11 CFG_QUEUE_MAX_NAME_LENGTH	137
7.14.2.12 CFG_QUEUE_MAX_NUMBER	137
7.14.2.13 CFG_QUEUE_USE_NAME	137
7.14.2.14 CFG_RESET_ON_ERROR	137
7.14.2.15 CFG_RESET_ON_ERROR_DELAY_MS	137
7.14.2.16 CFG_SCHED_COOPERATIVE	138
7.14.2.17 CFG_SHELL_COMMAND_BUFFER_SIZE	138
7.14.2.18 CFG_SHELL_MAX_COMMAND_LENGTH	138
7.14.2.19 CFG_SHELL_MAX_COMMAND_NUMBER	138
7.14.2.20 CFG_SHELL_MAX_PARAMS_LENGTH	138
7.14.2.21 CFG_SHELL_STARTUP_DELAY_MS	139
7.14.2.22 CFG_SHELL_USE_SERVICE	139
7.14.2.23 CFG_SIGNAL_MAX_NUMBER	139
7.14.2.24 CFG_SIGNAL_MAX_SUBSCRIBERS	139
7.14.2.25 CFG_SYSMON_GCP_CHANNEL_NUM	139
7.14.2.26 CFG_SYSMON_MAX_USER_MESSAGES	140
7.14.2.27 CFG_SYSMON_USE_SERVICE	140

7.14.2.28 CFG_SYSTEM_TASK_STACK_SIZE	140
7.14.2.29 CFG_TARGET_CPU	140
7.14.2.30 CFG_TASK_MAX_NAME_LENGTH	140
7.14.2.31 CFG_TASK_MAX_NUMBER	141
7.14.2.32 CFG_TASK_MAX_STACK_SIZE	141
7.14.2.33 CFG_TASK_MESSAGE_DAEMON_PRIO	141
7.14.2.34 CFG_TASK_MESSAGE_DAEMON_STACK	141
7.14.2.35 CFG_TASK_MIN_STACK_SIZE	141
7.14.2.36 CFG_TASK_SHELL_DAEMON_PRIO	142
7.14.2.37 CFG_TASK_SHELL_DAEMON_STACK	142
7.14.2.38 CFG_TASK_SIGNAL_DAEMON_PRIO	142
7.14.2.39 CFG_TASK_SIGNAL_DAEMON_STACK	142
7.14.2.40 CFG_TASK_SYS_PRIO	142
7.14.2.41 CFG_TASK_SYSMON_DAEMON_PRIO	143
7.14.2.42 CFG_TASK_SYSMON_DAEMON_STACK	143
7.14.2.43 CFG_TASK_TIME_DAEMON_PRIO	143
7.14.2.44 CFG_TASK_TIME_DAEMON_STACK	143
7.14.2.45 CFG_TASK_TRACE_DAEMON_PRIO	143
7.14.2.46 CFG_TASK_TRACE_DAEMON_STACK	144
7.14.2.47 CFG_TRACE_MAX_LENGTH	144
7.14.2.48 CFG_USE_PRIO_INHERITANCE	144
7.15 GOS2022/OS/kernel/inc/gos_kernel.h File Reference	144
7.15.1 Detailed Description	150
7.15.2 Macro Definition Documentation	150
7.15.2.1 GOS_CONCAT_RESULT	150
7.15.3 Enumeration Type Documentation	150
7.15.3.1 gos_boolValue_t	150
7.15.3.2 gos_kernel_privilege_t	151
7.15.3.3 gos_result_t	151
7.15.3.4 gos_taskPrivilegeLevel_t	152
7.15.3.5 gos_taskState_t	152
7.16 GOS2022/OS/kernel/inc/gos_port.h File Reference	153
7.16.1 Detailed Description	154
7.16.2 Macro Definition Documentation	154
7.16.2.1 gos_ported_doContextSwitch	154
7.16.2.2 gos_ported_enableFaultHandlers	155
7.16.2.3 gos_ported_handleSVC	155
7.16.2.4 gos_ported_handleSVCMain	156
7.16.2.5 gos_ported_kernelStartInit	156
7.16.2.6 gos_ported_pendSVHandler	156
7.16.2.7 gos_ported_procReset	157
7.16.2.8 gos_ported_reschedule	157

7.16.2.9 gos_ported_svcHandler	157
7.16.2.10 gos_ported_svcHandlerMain	158
7.16.2.11 gos_ported_sysTickInterrupt	158
7.17 GOS2022/OS/kernel/src/gos_kernel.c File Reference	158
7.17.1 Detailed Description	161
7.17.2 Macro Definition Documentation	161
7.17.2.1 BINARY_PATTERN	161
7.17.2.2 CONFIG_DUMP_SEPARATOR	161
7.17.2.3 ICSR	161
7.17.2.4 MAX_CPU_DUMP_SEPARATOR	162
7.17.2.5 SHCSR	162
7.17.2.6 STACK_STATS_SEPARATOR	162
7.17.2.7 TASK_DUMP_SEPARATOR	162
7.17.2.8 TO_BINARY	162
7.17.3 Function Documentation	163
7.17.3.1 gos_idleTask()	163
7.17.3.2 gos_kernelCheckTaskStack()	164
7.17.3.3 gos_kernelGetCurrentPsp()	164
7.17.3.4 gos_kernelGetTaskStateString()	165
7.17.3.5 gos_kernelProcessorReset()	165
7.17.3.6 gos_kernelSaveCurrentPsp()	165
7.17.3.7 gos_kernelSelectNextTask()	166
7.17.4 Variable Documentation	166
7.17.4.1 atomicCntr	166
7.17.4.2 cpuUseLimit	167
7.17.4.3 currentTaskIndex	167
7.17.4.4 inlsr	167
7.17.4.5 isKernelRunning	167
7.17.4.6 kernelDumpReadySignal	167
7.17.4.7 kernelDumpSignal	168
7.17.4.8 kernelPreResetHookFunction	168
7.17.4.9 kernelPrivilegedHookFunction	168
7.17.4.10 kernelSwapHookFunction	168
7.17.4.11 kernelSysTickHookFunction	168
7.17.4.12 monitoringTime	169
7.17.4.13 previousTick	169
7.17.4.14 primask	169
7.17.4.15 privilegedModeSetRequired	169
7.17.4.16 resetRequired	169
7.17.4.17 schedDisableCntr	170
7.17.4.18 sysTicks	170
7.17.4.19 sysTimerValue	170

7.18 GOS2022/OS/kernel/src/gos_task.c File Reference	170
7.18.1 Detailed Description	173
7.18.2 Function Documentation	173
7.18.2.1 gos_idleTask()	173
7.18.2.2 gos_taskCheckDescriptor()	174
7.18.3 Variable Documentation	175
7.18.3.1 kernelIdleHookFunction	175
7.18.3.2 kernelTaskDeleteSignal	176
7.18.3.3 taskDescriptors	176
7.19 GOS2022/OS/services/inc/gos.h File Reference	176
7.19.1 Detailed Description	177
7.19.2 Macro Definition Documentation	177
7.19.2.1 GOS_VERSION_MAJOR	178
7.19.2.2 GOS_VERSION_MINOR	178
7.19.3 Function Documentation	178
7.19.3.1 gos_Dump()	178
7.20 GOS2022/OS/services/inc/gos_error.h File Reference	179
7.20.1 Detailed Description	180
7.20.2 Enumeration Type Documentation	180
7.20.2.1 gos_errorLevel_t	180
7.20.3 Function Documentation	181
7.20.3.1 gos_errorHandler()	181
7.20.3.2 gos_errorTraceInit()	182
7.20.3.3 gos_printStartupLogo()	183
7.21 GOS2022/OS/services/inc/gos_gcp.h File Reference	184
7.21.1 Detailed Description	185
7.21.2 Typedef Documentation	185
7.21.2.1 gos_gcpReceiveFunction_t	185
7.21.2.2 gos_gcpTransmitFunction_t	186
7.21.3 Function Documentation	186
7.21.3.1 gos_gcplInit()	186
7.21.3.2 gos_gcpReceiveMessage()	186
7.21.3.3 gos_gcpRegisterPhysicalDriver()	188
7.21.3.4 gos_gcpTransmitMessage()	188
7.22 GOS2022/OS/services/inc/gos_message.h File Reference	190
7.22.1 Detailed Description	191
7.22.2 Macro Definition Documentation	192
7.22.2.1 GOS_MESSAGE_ENDLESS_TMO	192
7.22.2.2 GOS_MESSAGE_INVALID_ID	192
7.22.3 Typedef Documentation	192
7.22.3.1 gos_messageId_t	192
7.22.3.2 gos_messageTimeout_t	192

7.22.4 Function Documentation	192
7.22.4.1 gos_messageInit()	192
7.22.4.2 gos_messageRx()	193
7.22.4.3 gos_messageTx()	194
7.23 GOS2022/OS/services/inc/gos_mutex.h File Reference	195
7.23.1 Detailed Description	196
7.23.2 Macro Definition Documentation	196
7.23.2.1 GOS_MUTEX_ENDLESS_TMO	197
7.23.2.2 GOS_MUTEX_NO_TMO	197
7.23.3 Enumeration Type Documentation	197
7.23.3.1 gos_mutexState_t	197
7.23.4 Function Documentation	197
7.23.4.1 gos_mutexInit()	197
7.23.4.2 gos_mutexLock()	198
7.23.4.3 gos_mutexUnlock()	200
7.24 GOS2022/OS/services/inc/gos_queue.h File Reference	201
7.24.1 Detailed Description	203
7.24.2 Macro Definition Documentation	204
7.24.2.1 GOS_DEFAULT_QUEUE_ID	204
7.24.2.2 GOS_INVALID_QUEUE_ID	204
7.24.3 Typedef Documentation	204
7.24.3.1 gos_queueByte_t	204
7.24.3.2 gos_queueEmptyHook	205
7.24.3.3 gos_queueFullHook	205
7.24.4 Function Documentation	205
7.24.4.1 gos_queueCreate()	205
7.24.4.2 gos_queueDump()	206
7.24.4.3 gos_queueGet()	207
7.24.4.4 gos_queueGetElementNumber()	208
7.24.4.5 gos_queueGetName()	208
7.24.4.6 gos_queueInit()	209
7.24.4.7 gos_queuePeek()	209
7.24.4.8 gos_queuePut()	210
7.24.4.9 gos_queueRegisterEmptyHook()	211
7.24.4.10 gos_queueRegisterFullHook()	212
7.24.4.11 gos_queueReset()	212
7.25 GOS2022/OS/services/inc/gos_shell.h File Reference	214
7.25.1 Detailed Description	216
7.25.2 Typedef Documentation	216
7.25.2.1 gos_shellFunction	217
7.25.3 Function Documentation	217
7.25.3.1 gos_shellEchoOff()	217

7.25.3.2 gos_shellEchoOn()	217
7.25.3.3 gos_shellInit()	218
7.25.3.4 gos_shellRegisterCommand()	218
7.25.3.5 gos_shellRegisterCommands()	219
7.25.3.6 gos_shellResume()	220
7.25.3.7 gos_shellSuspend()	221
7.26 GOS2022/OS/services/inc/gos_signal.h File Reference	222
7.26.1 Detailed Description	223
7.26.2 Function Documentation	223
7.26.2.1 gos_signalCreate()	224
7.26.2.2 gos_signalInit()	225
7.26.2.3 gos_signalInvoke()	226
7.26.2.4 gos_signalSubscribe()	227
7.27 GOS2022/OS/services/inc/gos_sysmon.h File Reference	228
7.27.1 Detailed Description	229
7.27.2 Typedef Documentation	229
7.27.2.1 gos_sysmonMessageReceivedCallback	229
7.27.3 Function Documentation	230
7.27.3.1 gos_sysmonInit()	230
7.27.3.2 gos_sysmonRegisterUserMessage()	230
7.28 GOS2022/OS/services/inc/gos_time.h File Reference	231
7.28.1 Detailed Description	233
7.28.2 Enumeration Type Documentation	233
7.28.2.1 gos_timeCompareResult_t	233
7.28.2.2 gos_timeElapsedSenderId_t	233
7.28.2.3 gos_timeMonthEnum_t	234
7.28.3 Function Documentation	234
7.28.3.1 gos_runTimeAddMicroseconds()	234
7.28.3.2 gos_runTimeAddMilliseconds()	235
7.28.3.3 gos_runTimeAddSeconds()	236
7.28.3.4 gos_runTimeGet()	237
7.28.3.5 gos_timeAddMilliseconds()	238
7.28.3.6 gos_timeAddSeconds()	238
7.28.3.7 gos_timeCompare()	239
7.28.3.8 gos_timeGet()	240
7.28.3.9 gos_timeIncreaseSystemTime()	240
7.28.3.10 gos_timeInit()	241
7.28.3.11 gos_timeSet()	242
7.29 GOS2022/OS/services/inc/gos_trace.h File Reference	243
7.29.1 Detailed Description	245
7.29.2 Function Documentation	245
7.29.2.1 gos_tracelnit()	245

7.29.2.2 gos_traceTrace()	246
7.29.2.3 gos_traceTraceFormatted()	247
7.29.2.4 gos_traceTraceFormattedUnsafe()	248
7.30 GOS2022/OS/services/inc/gos_trigger.h File Reference	249
7.30.1 Detailed Description	251
7.30.2 Macro Definition Documentation	251
7.30.2.1 GOS_TRIGGER_ENDLESS_TMO	251
7.30.2.2 GOS_TRIGGER_NO_TMO	252
7.30.3 Function Documentation	252
7.30.3.1 gos_triggerDecrement()	252
7.30.3.2 gos_triggerIncrement()	253
7.30.3.3 gos_triggerInit()	254
7.30.3.4 gos_triggerReset()	254
7.30.3.5 gos_triggerWait()	255
7.31 GOS2022/OS/services/src/gos.c File Reference	256
7.31.1 Detailed Description	257
7.31.2 Macro Definition Documentation	257
7.31.2.1 GOS_SYS_TASK_SLEEP_TIME	258
7.31.3 Typedef Documentation	258
7.31.3.1 gos_initFunc_t	258
7.31.4 Function Documentation	258
7.31.4.1 gos_Dump()	258
7.31.4.2 gos_Start()	259
7.31.4.3 gos_systemTask()	260
7.31.5 Variable Documentation	260
7.31.5.1 dumpRequired	260
7.31.5.2 initError	261
7.31.5.3 initializers	261
7.31.5.4 systemTaskDesc	261
7.31.5.5 systemTaskId	261
7.32 GOS2022/OS/services/src/gos_error.c File Reference	262
7.32.1 Detailed Description	263
7.32.2 Macro Definition Documentation	263
7.32.2.1 ERROR_BUFFER_SIZE	263
7.32.2.2 RESULT_STRING_ERROR	264
7.32.2.3 RESULT_STRING_SUCCESS	264
7.32.2.4 RESULT_STRING_UNKNOWN	264
7.32.2.5 SEPARATOR_LINE	264
7.32.3 Function Documentation	264
7.32.3.1 gos_errorHandler()	264
7.32.3.2 gos_errorTraceInit()	266
7.32.3.3 gos_printStartupLogo()	267

7.32.3.4 gos_traceResultToString()	268
7.32.4 Variable Documentation	268
7.32.4.1 errorBuffer	269
7.33 GOS2022/OS/services/src/gos_gcp.c File Reference	269
7.33.1 Detailed Description	270
7.33.2 Macro Definition Documentation	271
7.33.2.1 GCP_PROTOCOL_VERSION_MAJOR	271
7.33.2.2 GCP_PROTOCOL_VERSION_MINOR	271
7.33.3 Enumeration Type Documentation	271
7.33.3.1 gos_gcpAck_t	271
7.33.4 Function Documentation	271
7.33.4.1 gos_gcpInit()	272
7.33.4.2 gos_gcpReceiveMessage()	272
7.33.4.3 gos_gcpReceiveMessageInternal()	273
7.33.4.4 gos_gcpRegisterPhysicalDriver()	274
7.33.4.5 gos_gcpTransmitMessage()	275
7.33.4.6 gos_gcpTransmitMessageInternal()	276
7.33.4.7 gos_gcpValidateData()	278
7.33.4.8 gos_gcpValidateHeader()	279
7.33.5 Variable Documentation	280
7.33.5.1 channelFunctions	280
7.33.5.2 gcpRxMutexes	280
7.33.5.3 gcpTxMutexes	280
7.34 GOS2022/OS/services/src/gos_message.c File Reference	280
7.34.1 Detailed Description	282
7.34.2 Macro Definition Documentation	282
7.34.2.1 GOS_MESSAGE_DAEMON_POLL_TIME_MS	282
7.34.3 Function Documentation	282
7.34.3.1 gos_messageDaemonTask()	283
7.34.3.2 gos_messageInit()	283
7.34.3.3 gos_messageRx()	284
7.34.3.4 gos_messageTx()	285
7.34.4 Variable Documentation	286
7.34.4.1 messageArray	286
7.34.4.2 messageDaemonTaskDesc	286
7.34.4.3 messageDaemonTaskId	286
7.34.4.4 messageMutex	286
7.34.4.5 messageWaiterArray	287
7.34.4.6 nextMessageIndex	287
7.34.4.7 nextWaiterIndex	287
7.35 GOS2022/OS/services/src/gos_mutex.c File Reference	287
7.35.1 Detailed Description	288

7.35.2 Macro Definition Documentation	289
7.35.2.1 MUTEX_LOCK_SLEEP_MS	289
7.35.3 Function Documentation	289
7.35.3.1 gos_mutexInit()	289
7.35.3.2 gos_mutexLock()	290
7.35.3.3 gos_mutexUnlock()	291
7.36 GOS2022/OS/services/src/gos_queue.c File Reference	292
7.36.1 Detailed Description	294
7.36.2 Macro Definition Documentation	294
7.36.2.1 DUMP_SEPARATOR	295
7.36.3 Function Documentation	295
7.36.3.1 gos_queueCreate()	295
7.36.3.2 gos_queueDump()	296
7.36.3.3 gos_queueGet()	296
7.36.3.4 gos_queueGetElementNumber()	297
7.36.3.5 gos_queueGetName()	298
7.36.3.6 gos_queueInit()	299
7.36.3.7 gos_queuePeek()	299
7.36.3.8 gos_queuePut()	300
7.36.3.9 gos_queueRegisterEmptyHook()	301
7.36.3.10 gos_queueRegisterFullHook()	302
7.36.3.11 gos_queueReset()	302
7.36.4 Variable Documentation	303
7.36.4.1 queueEmptyHook	303
7.36.4.2 queueFullHook	303
7.36.4.3 queueMutex	303
7.36.4.4 queues	304
7.36.4.5 readCounters	304
7.36.4.6 writeCounters	304
7.37 GOS2022/OS/services/src/gos_shell.c File Reference	304
7.37.1 Detailed Description	305
7.37.2 Macro Definition Documentation	306
7.37.2.1 GOS_SHELL_DAEMON_POLL_TIME_MS	306
7.37.2.2 GOS_SHELL_DISPLAY_TEXT	306
7.37.3 Function Documentation	306
7.37.3.1 gos_shellCommandHandler()	306
7.37.3.2 gos_shellDaemonTask()	307
7.37.3.3 gos_shellEchoOff()	308
7.37.3.4 gos_shellEchoOn()	308
7.37.3.5 gos_shellInit()	309
7.37.3.6 gos_shellRegisterCommand()	309
7.37.3.7 gos_shellRegisterCommands()	310

7.37.3.8 gos_shellResume()	311
7.37.3.9 gos_shellSuspend()	312
7.37.4 Variable Documentation	313
7.37.4.1 actualCommand	313
7.37.4.2 commandBuffer	313
7.37.4.3 commandBufferIndex	313
7.37.4.4 commandParams	313
7.37.4.5 shellCommand	314
7.37.4.6 shellCommands	314
7.37.4.7 shellDaemonTaskDesc	314
7.37.4.8 shellDaemonTaskId	314
7.37.4.9 useEcho	315
7.38 GOS2022/OS/services/src/gos_signal.c File Reference	315
7.38.1 Detailed Description	316
7.38.2 Macro Definition Documentation	316
7.38.2.1 GOS_SIGNAL_DAEMON_POLL_TIME_MS	317
7.38.3 Function Documentation	317
7.38.3.1 gos_signalCreate()	317
7.38.3.2 gos_signalDaemonTask()	318
7.38.3.3 gos_signalInit()	318
7.38.3.4 gos_signalInvoke()	319
7.38.3.5 gos_signalSubscribe()	320
7.38.4 Variable Documentation	320
7.38.4.1 callerTaskDesc	320
7.38.4.2 signalArray	320
7.38.4.3 signalDaemonTaskDescriptor	321
7.38.4.4 signalInvokeTrigger	321
7.39 GOS2022/OS/services/src/gos_sysmon.c File Reference	321
7.39.1 Detailed Description	324
7.39.2 Macro Definition Documentation	324
7.39.2.1 RECEIVE_BUFFER_SIZE	324
7.39.3 Typedef Documentation	325
7.39.3.1 gos_sysmonMessageHandler_t	325
7.39.4 Enumeration Type Documentation	325
7.39.4.1 gos_sysmonMessageEnum_t	325
7.39.4.2 gos_sysmonMessageId_t	326
7.39.4.3 gos_sysmonMessagePv_t	326
7.39.4.4 gos_sysmonMessageResult_t	327
7.39.4.5 gos_sysmonTaskModifyType_t	327
7.39.5 Function Documentation	328
7.39.5.1 gos_sysmonCheckMessage()	328
7.39.5.2 gos_sysmonDaemonTask()	329

7.39.5.3 gos_sysmonGetLutIndex()	330
7.39.5.4 gos_sysmonHandleCpuUsageGet()	330
7.39.5.5 gos_sysmonHandlePingRequest()	331
7.39.5.6 gos_sysmonHandleResetRequest()	331
7.39.5.7 gos_sysmonHandleSysRuntimeGet()	332
7.39.5.8 gos_sysmonHandleSystimeSet()	333
7.39.5.9 gos_sysmonHandleTaskDataGet()	333
7.39.5.10 gos_sysmonHandleTaskModification()	334
7.39.5.11 gos_sysmonHandleTaskVariableDataGet()	335
7.39.5.12 gos_sysmonInit()	335
7.39.5.13 gos_sysmonRegisterUserMessage()	336
7.39.5.14 gos_sysmonSendResponse()	336
7.39.6 Variable Documentation	337
7.39.6.1 cpuMessage	338
7.39.6.2 pingMessage	338
7.39.6.3 receiveBuffer	338
7.39.6.4 sysmonDaemonTaskDesc	338
7.39.6.5 sysmonLut	339
7.39.6.6 sysRuntimeGetResultMessage	339
7.39.6.7 sysTimeSetMessage	339
7.39.6.8 sysTimeSetResultMessage	339
7.39.6.9 taskDataGetMsg	339
7.39.6.10 taskDataMsg	340
7.39.6.11 taskDesc	340
7.39.6.12 taskModifyMessage	340
7.39.6.13 taskModifyResultMessage	340
7.39.6.14 taskVariableDataMsg	340
7.39.6.15 userMessages	341
7.40 GOS2022/OS/services/src/gos_time.c File Reference	341
7.40.1 Detailed Description	342
7.40.2 Macro Definition Documentation	343
7.40.2.1 TIME_DEFAULT_DAY	343
7.40.2.2 TIME_DEFAULT_MONTH	343
7.40.2.3 TIME_DEFAULT_YEAR	343
7.40.2.4 TIME_SLEEP_TIME_MS	343
7.40.3 Function Documentation	343
7.40.3.1 gos_runTimeAddMicroseconds()	343
7.40.3.2 gos_runTimeAddMilliseconds()	344
7.40.3.3 gos_runTimeAddSeconds()	345
7.40.3.4 gos_runTimeGet()	346
7.40.3.5 gos_timeAddMilliseconds()	346
7.40.3.6 gos_timeAddSeconds()	347

7.40.3.7 gos_timeCompare()	348
7.40.3.8 gos_timeDaemonTask()	348
7.40.3.9 gos_timeGet()	349
7.40.3.10 gos_timeIncreaseSystemTime()	350
7.40.3.11 gos_timeInit()	351
7.40.3.12 gos_timeSet()	352
7.40.4 Variable Documentation	352
7.40.4.1 dayLookupTable	352
7.40.4.2 systemRunTime	353
7.40.4.3 systemTime	353
7.40.4.4 timeDaemonTaskDesc	353
7.40.4.5 timeDaemonTaskId	354
7.40.4.6 timeSignalId	354
7.41 GOS2022/OS/services/src/gos_trace.c File Reference	354
7.41.1 Detailed Description	355
7.41.2 Macro Definition Documentation	355
7.41.2.1 GOS_TRACE_MUTEX_TMO_MS	356
7.41.2.2 GOS_TRACE_QUEUE_TMO_MS	356
7.41.2.3 GOS_TRACE_TIMESTAMP_FORMAT	356
7.41.2.4 GOS_TRACE_TIMESTAMP_LENGTH	356
7.41.3 Function Documentation	356
7.41.3.1 gos_traceDaemonTask()	357
7.41.3.2 gos_traceInit()	357
7.41.3.3 gos_traceTrace()	358
7.41.3.4 gos_traceTraceFormatted()	359
7.41.3.5 gos_traceTraceFormattedUnsafe()	360
7.41.4 Variable Documentation	361
7.41.4.1 formattedBuffer	361
7.41.4.2 timeStampBuffer	362
7.41.4.3 traceDaemonTaskDesc	362
7.41.4.4 traceLine	362
7.41.4.5 traceMutex	362
7.41.4.6 traceQueue	363
7.42 GOS2022/OS/services/src/gos_trigger.c File Reference	363
7.42.1 Detailed Description	364
7.42.2 Function Documentation	364
7.42.2.1 gos_triggerDecrement()	364
7.42.2.2 gos_triggerIncrement()	365
7.42.2.3 gos_triggerInit()	366
7.42.2.4 gos_triggerReset()	367
7.42.2.5 gos_triggerWait()	368

Chapter 1

GOS2022 OS documentation

Author: Ahmed Gazar

Date: 2025-04-16

Current Version: 1.0

1.1 Revision history

Version	Date	Author	Change
1.0	2025-04-16	Ahmed Gazar	Initial version

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Global definitions	9
Control macros for scheduling and ISR state setting	14
Privilege bit definitions	16
Basic data type definitions to be used in a GOS system	18
Task-related type definitions	19
Hook function type definitions	20
Time-related definitions	21
Kernel functions	22
Task-related functions	39
Platform initializer weak functions	63
Trace formatter macros	65
System monitoring message structures	66

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

gos_driver_functions_t	67
gos_gcpChannelFunctions_t	68
gos_gcpHeaderFrame_t	68
gos_initStruct_t	69
gos_message_t	69
gos_messageWaiterDesc_t	70
gos_mutex_t	70
gos_queue_t	71
gos_queueDescriptor_t	72
gos_queueElement_t	72
gos_runtime_t	73
gos_shellCommand_t	73
gos_signalDescriptor_t	74
gos_signallInvokeDescriptor	74
gos_sysmonCpuUsageMessage_t	75
gos_sysmonLut_t	75
gos_sysmonPingMessage_t	76
gos_sysmonSysruntimeGetResultMessage_t	76
gos_sysmonSystimeSetMessage_t	77
gos_sysmonSystimeSetResultMessage_t	78
gos_sysmonTaskData_t	78
gos_sysmonTaskDataGetMessage_t	79
gos_sysmonTaskDataMessage_t	80
gos_sysmonTaskModifyMessage_t	80
gos_sysmonTaskModifyResultMessage_t	81
gos_sysmonTaskVariableData	81
gos_sysmonTaskVariableDataMessage_t	82
gos_sysmonUserMessageDescriptor_t	83
gos_taskDescriptor_t	84
gos_time_t	85
gos_trigger_t	86

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

GOS2022/OS driver/inc/ gos_crc_driver.h	
GOS Cyclic Redundancy Check driver header	87
GOS2022/OS driver/inc/ gos_driver.h	
GOS driver header	89
GOS2022/OS driver/inc/ gos_shell_driver.h	
GOS SHELL driver header	91
GOS2022/OS driver/inc/ gos_sysmon_driver.h	
GOS SYSMON driver header	95
GOS2022/OS driver/inc/ gos_timer_driver.h	
GOS timer driver header	99
GOS2022/OS driver/inc/ gos_trace_driver.h	
GOS trace driver header	101
GOS2022/OS driver/src/ gos_crc_driver.c	
GOS Cyclic Redundancy Check driver source	105
GOS2022/OS driver/src/ gos_driver.c	
GOS driver source	108
GOS2022/OS driver/src/ gos_shell_driver.c	
GOS SHELL driver source	110
GOS2022/OS driver/src/ gos_sysmon_driver.c	
GOS SYSMON driver source	113
GOS2022/OS driver/src/ gos_timer_driver.c	
GOS timer driver source	116
GOS2022/OS driver/src/ gos_trace_driver.c	
GOS trace driver source	118
GOS2022/OS/kernel/inc/ gos_bootloader_config.h	
GOS bootloader configuration header	121
GOS2022/OS/kernel/inc/ gos_config.h	
GOS configuration header	133
GOS2022/OS/kernel/inc/ gos_kernel.h	
GOS kernel header	144
GOS2022/OS/kernel/inc/ gos_port.h	
GOS port header	153
GOS2022/OS/kernel/src/ gos_kernel.c	
GOS kernel source	158
GOS2022/OS/kernel/src/ gos_task.c	
GOS task source	170

GOS2022/OS/services/inc/ gos.h	
GOS header	176
GOS2022/OS/services/inc/ gos_error.h	
GOS error handler header	179
GOS2022/OS/services/inc/ gos_gcp.h	
GOS General Communication Protocol header	184
GOS2022/OS/services/inc/ gos_message.h	
GOS message service header	190
GOS2022/OS/services/inc/ gos_mutex.h	
GOS mutex service header	195
GOS2022/OS/services/inc/ gos_queue.h	
GOS queue service header	201
GOS2022/OS/services/inc/ gos_shell.h	
GOS shell service header	214
GOS2022/OS/services/inc/ gos_signal.h	
GOS signal service header	222
GOS2022/OS/services/inc/ gos_sysmon.h	
GOS system monitoring service header	228
GOS2022/OS/services/inc/ gos_time.h	
GOS time service header	231
GOS2022/OS/services/inc/ gos_trace.h	
GOS trace service header	243
GOS2022/OS/services/inc/ gos_trigger.h	
GOS trigger service header	249
GOS2022/OS/services/src/ gos.c	
GOS source	256
GOS2022/OS/services/src/ gos_error.c	
GOS error handler service source	262
GOS2022/OS/services/src/ gos_gcp.c	
GOS General Communication Protocol handler service source	269
GOS2022/OS/services/src/ gos_message.c	
GOS message service source	280
GOS2022/OS/services/src/ gos_mutex.c	
GOS mutex service source	287
GOS2022/OS/services/src/ gos_queue.c	
GOS queue service source	292
GOS2022/OS/services/src/ gos_shell.c	
GOS shell service source	304
GOS2022/OS/services/src/ gos_signal.c	
GOS signal service source	315
GOS2022/OS/services/src/ gos_sysmon.c	
GOS system monitoring service source	321
GOS2022/OS/services/src/ gos_time.c	
GOS time service source	341
GOS2022/OS/services/src/ gos_trace.c	
GOS trace service source	354
GOS2022/OS/services/src/ gos_trigger.c	
GOS trigger service source	363

Chapter 5

Module Documentation

5.1 Global definitions

Macros

- #define **NULL** ((void *) 0)
- #define **RAM_START** (0x20000000u)
- #define **RAM_SIZE** (128 * 1024)
- #define **MAIN_STACK** (**RAM_START** + **RAM_SIZE**)
- #define **GLOBAL_STACK** (0x1200)
- #define **GOS_DEFAULT_TASK_ID** ((gos_tid_t)0x8000)
- #define **GOS_INVALID_TASK_ID** ((gos_tid_t)0x0100)
- #define **GOS_TASK_MAX_PRIO_LEVELS** (**UINT8_MAX**)
- #define **GOS_TASK_IDLE_PRIO** (**GOS_TASK_MAX_PRIO_LEVELS**)
- #define **GOS_TASK_MAX_BLOCK_TIME_MS** (0xFFFFFFFFu)
- #define **GOS_STATIC** static
- #define **GOS_CONST** const
- #define **GOS_INLINE** inline __attribute__((always_inline))
- #define **GOS_STATIC_INLINE** **GOS_STATIC** **GOS_INLINE**
- #define **GOS_EXTERN** extern
- #define **GOS_NAKED** __attribute__((naked))
- #define **GOS_UNUSED** __attribute__((unused))
- #define **GOS_NOP** __asm volatile ("NOP")
- #define **GOS_ASM** __asm volatile

5.1.1 Detailed Description

5.1.2 Macro Definition Documentation

5.1.2.1 GLOBAL_STACK

```
#define GLOBAL_STACK ( 0x1200 )
```

Global stack.

Definition at line 148 of file gos_kernel.h.

5.1.2.2 GOS_ASM

```
#define GOS_ASM __asm volatile
```

ASM.

Definition at line 218 of file gos_kernel.h.

5.1.2.3 GOS_CONST

```
#define GOS_CONST const
```

Constant macro.

Definition at line 183 of file gos_kernel.h.

5.1.2.4 GOS_DEFAULT_TASK_ID

```
#define GOS_DEFAULT_TASK_ID ( (gos_tid_t)0x8000 )
```

Default task ID.

Definition at line 153 of file gos_kernel.h.

5.1.2.5 GOS_EXTERN

```
#define GOS_EXTERN extern
```

Extern macro.

Definition at line 198 of file gos_kernel.h.

5.1.2.6 GOS_INLINE

```
#define GOS_INLINE inline __attribute__((always_inline))
```

Inline macro.

Definition at line 188 of file gos_kernel.h.

5.1.2.7 GOS_INVALID_TASK_ID

```
#define GOS_INVALID_TASK_ID ( (gos_tid_t)0x0100 )
```

Invalid task ID.

Definition at line 158 of file gos_kernel.h.

5.1.2.8 GOS_NAKED

```
#define GOS_NAKED __attribute__ ((naked))
```

Naked.

Definition at line 203 of file gos_kernel.h.

5.1.2.9 GOS_NOP

```
#define GOS_NOP __asm volatile ("NOP")
```

NOP.

Definition at line 213 of file gos_kernel.h.

5.1.2.10 GOS_STATIC

```
#define GOS_STATIC static
```

Static macro.

Definition at line 178 of file gos_kernel.h.

5.1.2.11 GOS_STATIC_INLINE

```
#define GOS_STATIC_INLINE GOS_STATIC GOS_INLINE
```

Static in-line macro.

Definition at line 193 of file gos_kernel.h.

5.1.2.12 GOS_TASK_IDLE_PRIO

```
#define GOS_TASK_IDLE_PRIO ( GOS_TASK_MAX_PRIO_LEVELS )
```

Idle task priority.

Definition at line 168 of file gos_kernel.h.

5.1.2.13 GOS_TASK_MAX_BLOCK_TIME_MS

```
#define GOS_TASK_MAX_BLOCK_TIME_MS ( 0xFFFFFFFFu )
```

Task maximum block time.

Definition at line 173 of file gos_kernel.h.

5.1.2.14 GOS_TASK_MAX_PRIO_LEVELS

```
#define GOS_TASK_MAX_PRIO_LEVELS ( UINT8_MAX )
```

Maximum task priority levels.

Definition at line 163 of file gos_kernel.h.

5.1.2.15 GOS_UNUSED

```
#define GOS_UNUSED __attribute__ ((unused))
```

Unused.

Definition at line 208 of file gos_kernel.h.

5.1.2.16 MAIN_STACK

```
#define MAIN_STACK ( RAM_START + RAM_SIZE )
```

Main stack.

Definition at line 143 of file gos_kernel.h.

5.1.2.17 NULL

```
#define NULL ( (void *) 0 )
```

NULL pointer.

Definition at line 127 of file gos_kernel.h.

5.1.2.18 RAM_SIZE

```
#define RAM_SIZE ( 128 * 1024 )
```

RAM size (128kB).

Definition at line 138 of file gos_kernel.h.

5.1.2.19 RAM_START

```
#define RAM_START ( 0x20000000u )
```

RAM start address.

Definition at line 133 of file gos_kernel.h.

5.2 Control macros for scheduling and ISR state setting

Macros

- #define GOS_DISABLE_SCHED
- #define GOS_ENABLE_SCHED
- #define GOS_ISR_ENTER
- #define GOS_ISR_EXIT
- #define GOS_ATOMIC_ENTER
- #define GOS_ATOMIC_EXIT

5.2.1 Detailed Description

5.2.2 Macro Definition Documentation

5.2.2.1 GOS_ATOMIC_ENTER

```
#define GOS_ATOMIC_ENTER
```

Value:

```
\n
{
    GOS_EXTERN u8_t atomicCntr;
    GOS_EXTERN u32_t primask;
    if (atomicCntr == 0)
    {
        GOS_ASM( " cpsid i " ::: "memory" );
        GOS_ASM( "MRS %0, primask" : "=r" (primask) :: "memory" );
        GOS_ASM( "dsb" ::: "memory" );
        GOS_ASM( "isb" );
    }
    atomicCntr++;
    GOS_DISABLE_SCHED
}
```

Atomic operation enter - disable interrupts and kernel rescheduling.

Definition at line 264 of file gos_kernel.h.

5.2.2.2 GOS_ATOMIC_EXIT

```
#define GOS_ATOMIC_EXIT
```

Value:

```
\n
{
    GOS_EXTERN u8_t atomicCntr;
    GOS_EXTERN u32_t primask;
    if (atomicCntr > 0)
    {
        atomicCntr--;
    }
    if (atomicCntr == 0)
    {
        GOS_ASM( "MSR primask, %0" : : "r" (primask) : "memory" );
        GOS_ASM( " cpsie i " ::: "memory" );
        GOS_ASM( "dsb" ::: "memory" );
        GOS_ASM( "isb" );
    }
    GOS_ENABLE_SCHED
}
```

Atomic operation exit - enable interrupts kernel rescheduling.

Definition at line 281 of file gos_kernel.h.

5.2.2.3 GOS_DISABLE_SCHED

```
#define GOS_DISABLE_SCHED
```

Value:

```
{
    GOS_EXTERN u8_t schedDisableCntr;
    schedDisableCntr++;
}
```

Disable scheduling.

Definition at line 231 of file gos_kernel.h.

5.2.2.4 GOS_ENABLE_SCHED

```
#define GOS_ENABLE_SCHED
```

Value:

```
{
    GOS_EXTERN u8_t schedDisableCntr;
    if (schedDisableCntr > 0)
        schedDisableCntr--;
}
```

Enable scheduling.

Definition at line 238 of file gos_kernel.h.

5.2.2.5 GOS_ISR_ENTER

```
#define GOS_ISR_ENTER
```

Value:

```
{
    GOS_EXTERN u8_t inIsr;
    if (inIsr == 0) { GOS_DISABLE_SCHED }
    inIsr++;
}
```

Interrupt Service Routine enter.

Definition at line 247 of file gos_kernel.h.

5.2.2.6 GOS_ISR_EXIT

```
#define GOS_ISR_EXIT
```

Value:

```
{
    GOS_EXTERN u8_t inIsr;
    if (inIsr > 0) { inIsr--; }
    if (inIsr == 0) { GOS_ENABLE_SCHED }
}
```

Interrupt service routine exit.

Definition at line 255 of file gos_kernel.h.

5.3 Privilege bit definitions

Macros

- #define GOS_PRIV_TASK_MANIPULATE (1 << 15)
- #define GOS_PRIV_TASK_PRIO_CHANGE (1 << 14)
- #define GOS_PRIV_TRACE (1 << 13)
- #define GOS_PRIV_SIGNALING (1 << 11)
- #define GOS_PRIV_RESERVED_3 (1 << 10)
- #define GOS_PRIV_RESERVED_4 (1 << 9)
- #define GOS_PRIV_RESERVED_5 (1 << 8)

5.3.1 Detailed Description

5.3.2 Macro Definition Documentation

5.3.2.1 GOS_PRIV_RESERVED_3

```
#define GOS_PRIV_RESERVED_3 ( 1 << 10 )
```

Kernel reserved.

Definition at line 328 of file gos_kernel.h.

5.3.2.2 GOS_PRIV_RESERVED_4

```
#define GOS_PRIV_RESERVED_4 ( 1 << 9 )
```

Kernel reserved.

Definition at line 333 of file gos_kernel.h.

5.3.2.3 GOS_PRIV_RESERVED_5

```
#define GOS_PRIV_RESERVED_5 ( 1 << 8 )
```

Kernel reserved.

Definition at line 338 of file gos_kernel.h.

5.3.2.4 GOS_PRIV_SIGNALING

```
#define GOS_PRIV_SIGNALING ( 1 << 11 )
```

Task signal invoking privilege flag.

Definition at line 323 of file gos_kernel.h.

5.3.2.5 GOS_PRIV_TASK_MANIPULATE

```
#define GOS_PRIV_TASK_MANIPULATE ( 1 << 15 )
```

Task manipulation privilege flag.

Definition at line 308 of file gos_kernel.h.

5.3.2.6 GOS_PRIV_TASK_PRIO_CHANGE

```
#define GOS_PRIV_TASK_PRIO_CHANGE ( 1 << 14 )
```

Task priority change privilege flag.

Definition at line 313 of file gos_kernel.h.

5.3.2.7 GOS_PRIV_TRACE

```
#define GOS_PRIV_TRACE ( 1 << 13 )
```

Tracing privilege flag.

Definition at line 318 of file gos_kernel.h.

5.4 Basic data type definitions to be used in a GOS system

Typedefs

- `typedef uint8_t bool_t`
Boolean logic type.
- `typedef uint8_t u8_t`
8-bit unsigned type.
- `typedef uint16_t u16_t`
16-bit unsigned type.
- `typedef uint32_t u32_t`
32-bit unsigned type.
- `typedef uint64_t u64_t`
64-bit unsigned type.
- `typedef int8_t s8_t`
8-bit signed type.
- `typedef int16_t s16_t`
16-bit signed type.
- `typedef int32_t s32_t`
32-bit signed type.
- `typedef int64_t s64_t`
64-bit signed type.
- `typedef char char_t`
8-bit character type.
- `typedef float float_t`
Single precision float type.
- `typedef double double_t`
Double precision float type.
- `typedef void void_t`
Void type.

5.4.1 Detailed Description

5.5 Task-related type definitions

Typedefs

- `typedef u16_t gos_tid_t`
Task ID type.
- `typedef char_t gos_taskName_t[CFG_TASK_MAX_NAME_LENGTH]`
Task name type.
- `typedef void_t(* gos_task_t) (void_t)`
Task function type.
- `typedef u8_t gos_taskPrio_t`
Task priority type.
- `typedef u32_t gos_taskSleepTick_t`
Sleep tick type.
- `typedef u32_t gos_blockMaxTick_t`
Block max. tick type.
- `typedef u32_t gos_taskAddress_t`
Memory address type.
- `typedef u32_t gos_taskRunCounter_t`
Run counter type.
- `typedef u64_t gos_taskRunTime_t`
Run-time type.
- `typedef u32_t gos_taskCSCounter_t`
Context-switch counter type.
- `typedef u16_t gos_taskStackSize_t`
Task stack size type.

5.5.1 Detailed Description

5.6 Hook function type definitions

Typedefs

- `typedef void_t(* gos_taskSwapHook_t) (gos_tid_t, gos_tid_t)`
Task swap hook type.
- `typedef void_t(* gos_taskIdleHook_t) (void_t)`
Task idle hook type.
- `typedef void_t(* gos_taskSleepHook_t) (gos_tid_t)`
Task sleep hook type.
- `typedef void_t(* gos_taskWakeupHook_t) (gos_tid_t)`
Task wake-up hook type.
- `typedef void_t(* gos_taskSuspendHook_t) (gos_tid_t)`
Task suspend hook type.
- `typedef void_t(* gos_taskResumeHook_t) (gos_tid_t)`
Task resume hook type.
- `typedef void_t(* gos_taskBlockHook_t) (gos_tid_t)`
Task block hook type.
- `typedef void_t(* gos_taskUnblockHook_t) (gos_tid_t)`
Task unblock hook type.
- `typedef void_t(* gos_taskDeleteHook_t) (gos_tid_t)`
Task delete hook type.
- `typedef void_t(* gos_sysTickHook_t) (void_t)`
System tick hook type.
- `typedef void_t(* gos_privilegedHook_t) (void_t)`
Privileged mode hook type.
- `typedef void_t(* gos_preResetHook_t) (void_t)`
Kernel pre-reset hook type.

5.6.1 Detailed Description

5.7 Time-related definitions

Data Structures

- struct [gos_runtime_t](#)

Typedefs

- [typedef u16_t gos_microsecond_t](#)
Microsecond type.
- [typedef u16_t gos_millisecond_t](#)
Millisecond type.
- [typedef u8_t gos_second_t](#)
Second type.
- [typedef u8_t gos_minute_t](#)
Minute type.
- [typedef u8_t gos_hour_t](#)
Hour type.
- [typedef u16_t gos_day_t](#)
Day type.
- [typedef u8_t gos_month_t](#)
Month type.
- [typedef u16_t gos_year_t](#)
Year type.

5.7.1 Detailed Description

5.8 Kernel functions

Functions

- [`gos_result_t gos_kernellInit \(void_t\)`](#)
This function initializes the kernel.
- [`gos_result_t gos_kernelRegisterSwapHook \(gos_taskSwapHook_t swapHookFunction\)`](#)
Registers a task swap hook function.
- [`gos_result_t gos_kernelRegisterIdleHook \(gos_taskIdleHook_t idleHookFunction\)`](#)
Registers an idle hook function.
- [`gos_result_t gos_kernelRegisterSysTickHook \(gos_sysTickHook_t sysTickHookFunction\)`](#)
Registers a system tick hook function.
- [`gos_result_t gos_kernelRegisterPrivilegedHook \(gos_privilegedHook_t privilegedHookFunction\)`](#)
Registers a privileged hook function.
- [`gos_result_t gos_kernelRegisterPreResetHook \(gos_preResetHook_t preResetHookFunction\)`](#)
Registers a pre-reset hook function.
- [`gos_result_t gos_taskSubscribeDeleteSignal \(void_t\(*deleteSignalHandler\)\(u16_t\)\)`](#)
Subscribes the given handler to the task delete signal.
- [`gos_result_t gos_kernelSubscribeDumpReadySignal \(void_t\(*dumpReadySignalHandler\)\(u16_t\)\)`](#)
Subscribes the given handler to the dump ready signal.
- [`u32_t gos_kernelGetSysTicks \(void_t\)`](#)
Returns the system ticks.
- [`u16_t gos_kernelGetCpuUsage \(void_t\)`](#)
Returns the CPU usage.
- [`gos_result_t gos_kernelStart \(void_t\)`](#)
Starts the kernel.
- [`void_t gos_kernelReset \(void_t\)`](#)
Resets the microcontroller.
- [`void_t gos_kernelPrivilegedModeSetRequired \(void_t\)`](#)
Requests a privileged mode setting.
- [`void_t gos_kernelDelayUs \(u16_t microseconds\)`](#)
Blocking delay in microsecond range.
- [`void_t gos_kernelDelayMs \(u16_t milliseconds\)`](#)
Blocking delay in millisecond range.
- [`void_t gos_kernelCalculateTaskCpuUsages \(bool_t isResetRequired\)`](#)
Calculates the CPU usage for the tasks.
- [`void_t gos_kernelDump \(void_t\)`](#)
Kernel dump.
- [`gos_result_t gos_kernelSetMaxCpuLoad \(u16_t maxCpuLoad\)`](#)
Sets the maximum (global) CPU load.
- [`gos_result_t gos_kernelGetMaxCpuLoad \(u16_t *pMaxCpuLoad\)`](#)
Gets the maximum (global) CPU load.
- [`bool_t gos_kernellsCallerIsr \(void_t\)`](#)
Returns if the current task is ISR.
- [`void_t gos_kernelReschedule \(gos_kernel_privilege_t privilege\)`](#)
Reschedules the kernel.

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 gos_kernelCalculateTaskCpuUsages()

```
void_t gos_kernelCalculateTaskCpuUsages (
    bool_t isResetRequired )
```

Calculates the CPU usage for the tasks.

Based on the total system time range, it refreshes the CPU-usage statistics of tasks.

Parameters

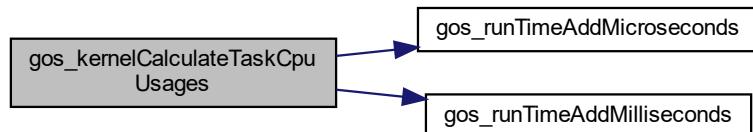
in	<i>isResetRequired</i>	Flag to indicate whether the measurement should be reset.
----	------------------------	---

Returns

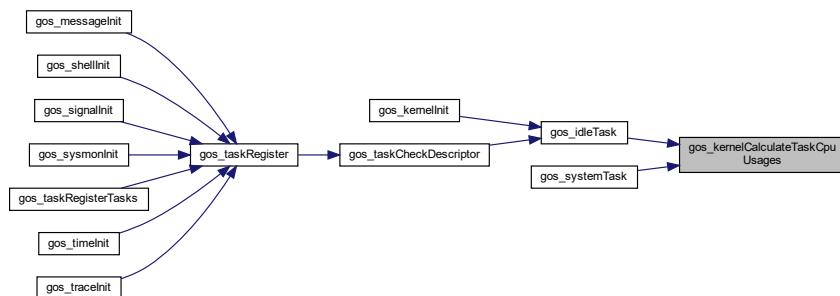
-

Definition at line 658 of file gos_kernel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.2 gos_kernelDelayMs()

```
void_t gos_kernelDelayMs (
    u16_t milliseconds )
```

Blocking delay in millisecond range.

This function waits in a while loop until the given number of system ticks have elapsed.

Parameters

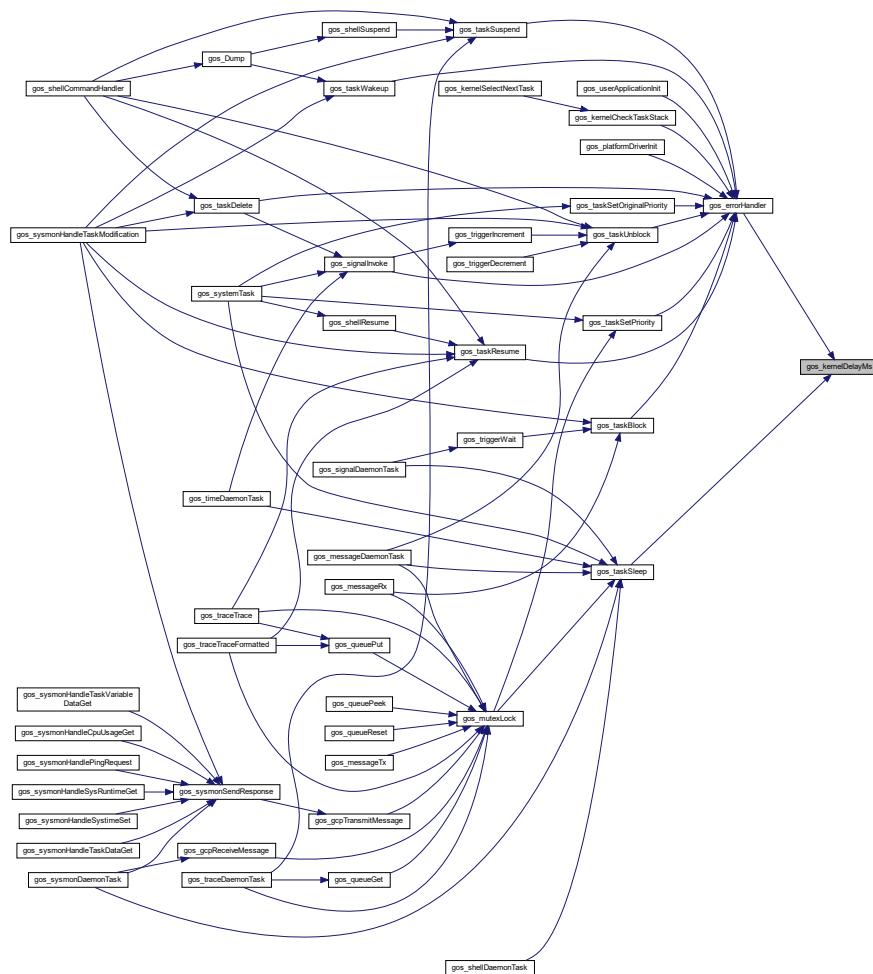
in	<i>milliseconds</i>	Milliseconds to wait.
----	---------------------	-----------------------

Returns

-

Definition at line 642 of file gos_kernel.c.

Here is the caller graph for this function:



5.8.2.3 gos_kernelDelayUs()

```
void_t gos_kernelDelayUs (
    u16_t microseconds )
```

Blocking delay in microsecond range.

This function waits in a while loop until the given number of milliseconds have elapsed based on the system timer.

Parameters

in	<i>microseconds</i>	Microseconds to wait.
----	---------------------	-----------------------

Returns

Definition at line 620 of file gos_kernel.c.

Here is the call graph for this function:



5.8.2.4 gos_kernelDump()

```
void_t gos_kernelDump (
    void_t )
```

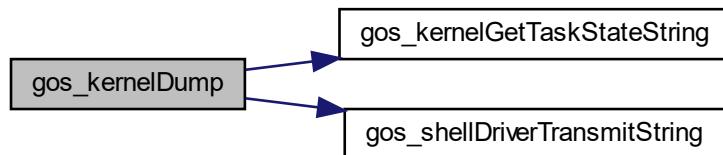
Kernel dump.

This function prints the kernel configuration and task data to the trace output.

Returns

-
Definition at line 760 of file gos_kernel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.5 gos_kernelGetCpuUsage()

```
u16_t gos_kernelGetCpuUsage (   
    void_t )
```

Returns the CPU usage.

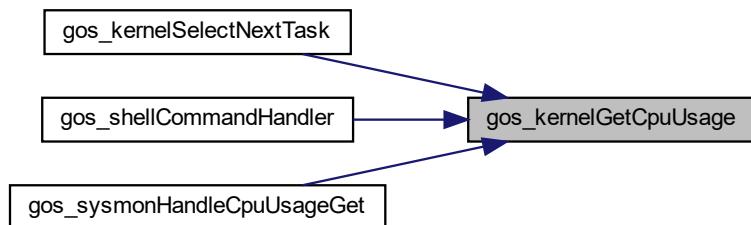
Refreshes the CPU statistics and returns the current CPU usage in [CPU%] * 100 format that results in a range of 0...10000 where the last two digits represent two decimals. The usage is 100% - idle task%.

Returns

Overall CPU usage.

Definition at line 576 of file gos_kernel.c.

Here is the caller graph for this function:

**5.8.2.6 gos_kernelGetMaxCpuLoad()**

```
gos_result_t gos_kernelGetMaxCpuLoad (
    u16_t * pMaxCpuLoad )
```

Gets the maximum (global) CPU load.

Returns the value of the maximum CPU load (limit).

Parameters

<code>out</code>	<code>pMaxCpuLoad</code>	Target variable to store the maximum CPU load.
------------------	--------------------------	--

Returns

Result of maximum CPU load getting.

Return values

<code>GOS_SUCCESS</code>	Maximum CPU load getting successful.
<code>GOS_ERROR</code>	Target variable is NULL.

Definition at line 898 of file gos_kernel.c.

5.8.2.7 gos_kernelGetSysTicks()

```
u32_t gos_kernelGetSysTicks (
    void_t
)
```

Returns the system ticks.

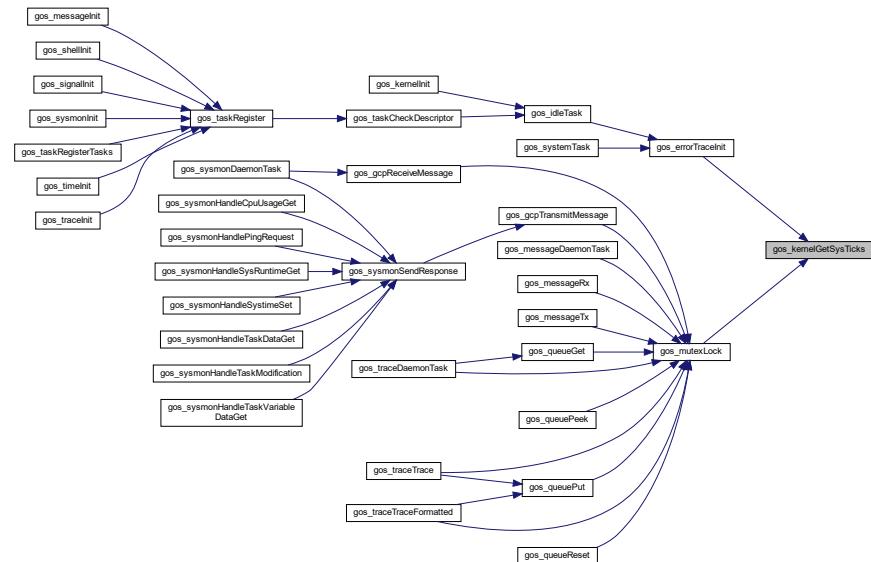
Returns the internal system tick counter value.

Returns

System ticks.

Definition at line 565 of file gos_kernel.c.

Here is the caller graph for this function:



5.8.2.8 gos_kernelnInit()

```
gos_result_t gos_kernelnInit (
    void_t
)
```

This function initializes the kernel.

Initializes the internal task array, fills out the PSP for the idle task, and registers the kernel dump task and suspends it.

Returns

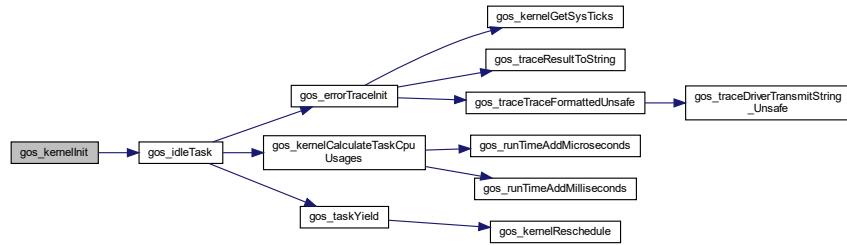
Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Kernel initialization successful.
<i>GOS_ERROR</i>	Kernel task suspension unsuccessful.

Definition at line 293 of file gos_kernel.c.

Here is the call graph for this function:

**5.8.2.9 gos_kernelIsCallerIsr()**

```
bool_t gos_kernelIsCallerIsr (
    void_t )
```

Returns if the current task is ISR.

Returns if the inlsr flag is greater than zero.

Returns

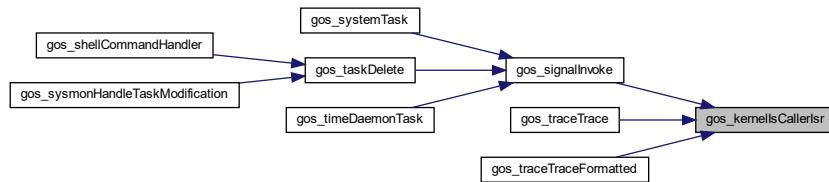
Whether the caller is ISR (Interrupt Service Routine).

Return values

<i>GOS_TRUE</i>	Caller is ISR.
<i>GOS_FALSE</i>	Caller is not ISR.

Definition at line 924 of file gos_kernel.c.

Here is the caller graph for this function:



5.8.2.10 gos_kernelPrivilegedModeSetRequired()

```
void_t gos_kernelPrivilegedModeSetRequired (
    void_t )
```

Requests a privileged mode setting.

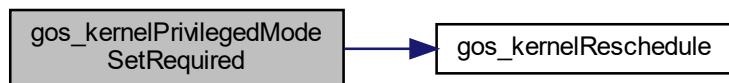
Initiates an SVC call while setting the corresponding flag to set the execution mode to privileged (permanently). After this, it calls the corresponding hook function, so the caller can access special registers immediately.

Returns

-

Definition at line 607 of file gos_kernel.c.

Here is the call graph for this function:



5.8.2.11 gos_kernelRegisterIdleHook()

```
gos_result_t gos_kernelRegisterIdleHook (
    gos_taskIdleHook_t idleHookFunction )
```

Registers an idle hook function.

Checks whether the param is NULL pointer and a hook function is already registered, and both conditions are false, it registers the hook function.

Parameters

in	<i>idleHookFunction</i>	Idle hook function.
----	-------------------------	---------------------

Returns

Result of registration.

Return values

<i>GOS_SUCCESS</i>	Registration successful.
<i>GOS_ERROR</i>	Registration failed (hook function already exists or parameter is NULL).

5.8.2.12 gos_kernelRegisterPreResetHook()

```
gos_result_t gos_kernelRegisterPreResetHook (
    gos_preResetHook_t preResetHookFunction )
```

Registers a pre-reset hook function.

Checks if a hook function has already been registered, and, if not, it registers the new hook function.

Parameters

in	<i>preResetHookFunction</i>	Pre-reset hook function.
----	-----------------------------	--------------------------

Returns

Result of registration.

Return values

<i>GOS_SUCCESS</i>	Registration successful.
<i>GOS_ERROR</i>	Registration failed (hook function already exists or parameter is NULL).

Definition at line 470 of file gos_kernel.c.

5.8.2.13 gos_kernelRegisterPrivilegedHook()

```
gos_result_t gos_kernelRegisterPrivilegedHook (
    gos_privilegedHook_t privilegedHookFunction )
```

Registers a privileged hook function.

Checks if a hook function has already been registered, and, if not, it registers the new hook function.

Parameters

in	<i>privilegedHookFunction</i>	Privileged hook function.
----	-------------------------------	---------------------------

Returns

Result of registration.

Return values

<i>GOS_SUCCESS</i>	Registration successful.
<i>GOS_ERROR</i>	Registration failed (hook function already exists or parameter is NULL).

Definition at line 444 of file gos_kernel.c.

5.8.2.14 gos_kernelRegisterSwapHook()

```
gos_result_t gos_kernelRegisterSwapHook (
    gos_taskSwapHook_t swapHookFunction )
```

Registers a task swap hook function.

Checks whether the param is NULL pointer and a hook function is already registered, and both conditions are false, it registers the hook function.

Parameters

in	<i>swapHookFunction</i>	Swap hook function.
----	-------------------------	---------------------

Returns

Result of registration.

Return values

<i>GOS_SUCCESS</i>	Registration successful.
<i>GOS_ERROR</i>	Registration failed (hook function already exists or parameter is NULL).

Definition at line 392 of file gos_kernel.c.

5.8.2.15 gos_kernelRegisterSysTickHook()

```
gos_result_t gos_kernelRegisterSysTickHook (
    gos_sysTickHook_t sysTickHookFunction )
```

Registers a system tick hook function.

Checks whether the param is NULL pointer and a hook function is already registered, and both conditions are false, it registers the hook function.

Parameters

in	<i>sysTickHookFunction</i>	System tick hook function.
----	----------------------------	----------------------------

Returns

Result of registration.

Return values

<i>GOS_SUCCESS</i>	Registration successful.
<i>GOS_ERROR</i>	Registration failed (hook function already exists or parameter is NULL).

Definition at line 418 of file gos_kernel.c.

5.8.2.16 gos_kernelReschedule()

```
void_t gos_kernelReschedule (
    gos_kernel_privilege_t privilege )
```

Reschedules the kernel.

Based on the privilege, it invokes a kernel reschedule event.

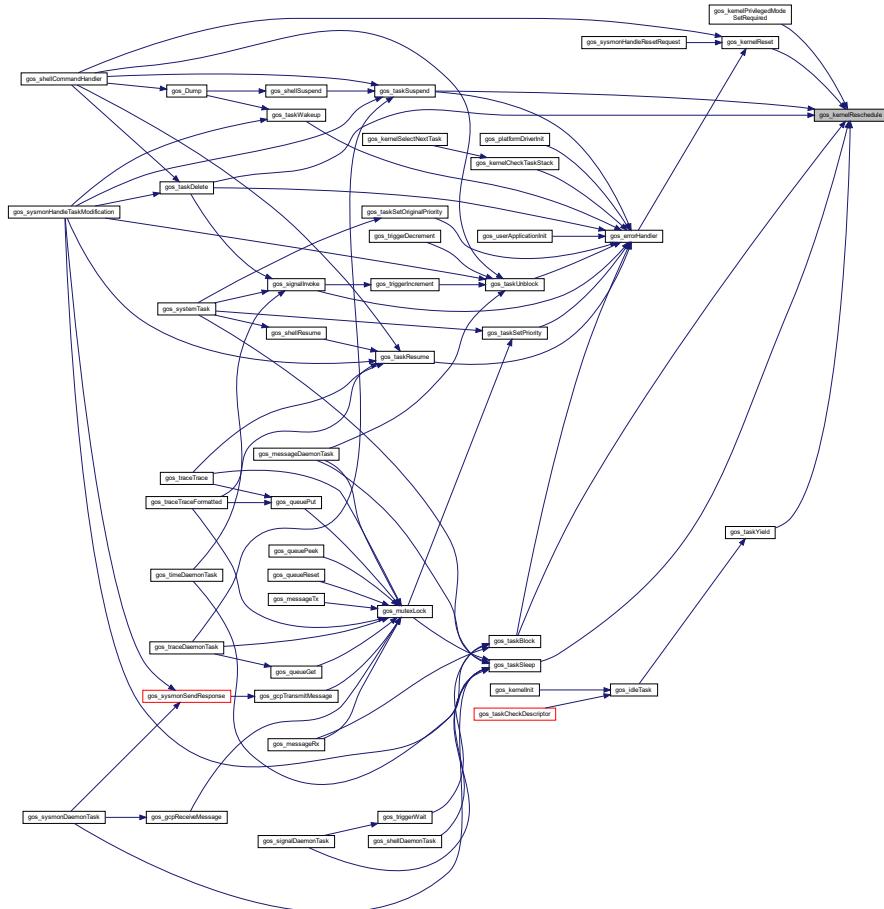
Parameters

in	<i>privilege</i>	Privilege level.
----	------------------	------------------

Returns

Definition at line 983 of file gos_kernel.c.

Here is the caller graph for this function:



5.8.2.17 gos_kernelReset()

```
void_t gos_kernelReset ( void_t )
```

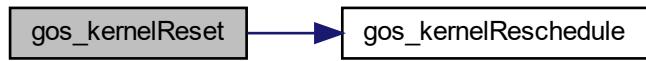
Resets the microcontroller.

Sets the reset required flag and gets privileged access to reset the controller.

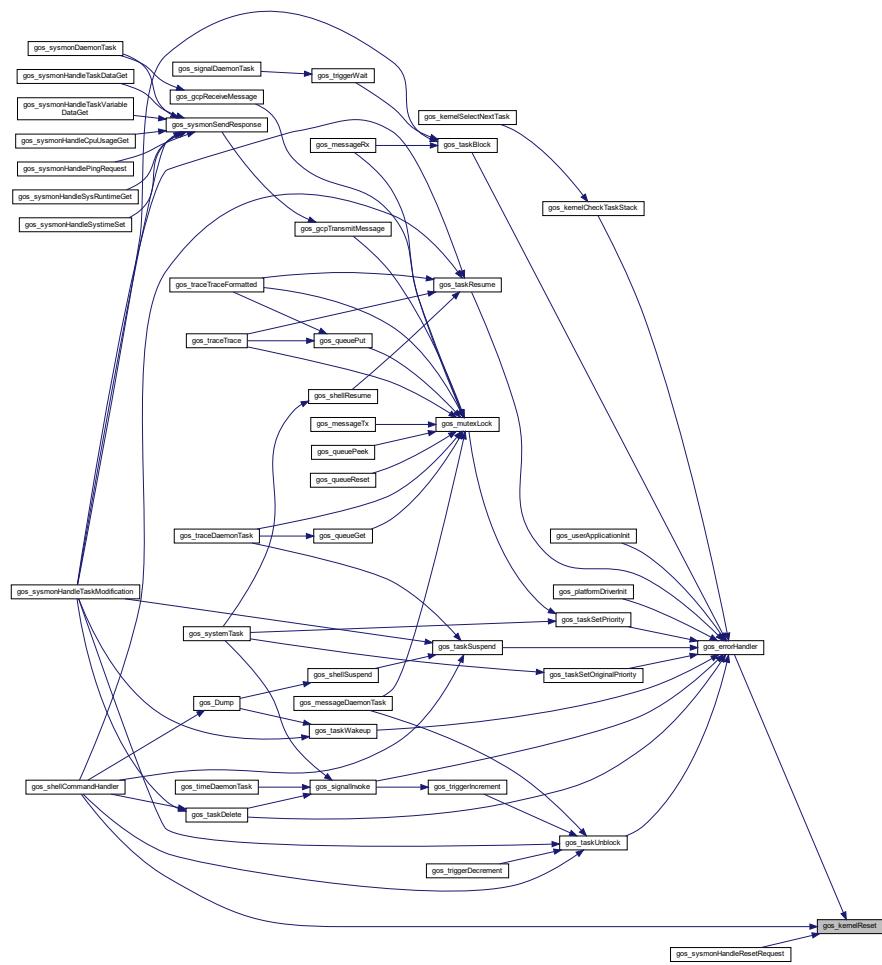
Returns

Definition at line 587 of file gos_kernel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.18 gos_kernelSetMaxCpuLoad()

```
gos_result_t gos_kernelSetMaxCpuLoad (
    u16_t maxCpuLoad )
```

Sets the maximum (global) CPU load.

Sets the value of the global CPU load above which the scheduler will start running the idle task until the CPU load falls below the limit value.

Parameters

in	<i>maxCpuLoad</i>	Desired maximum CPU load (0...10000 where 100 % = 10000).
----	-------------------	---

Returns

Result of maximum CPU load setting.

Return values

<i>GOS_SUCCESS</i>	Maximum CPU load setting successful.
<i>GOS_ERROR</i>	Desired value is out of range.

Definition at line 872 of file gos_kernel.c.

5.8.2.19 gos_kernelStart()

```
gos_result_t gos_kernelStart (
    void_t )
```

Starts the kernel.

Prepares the PSP for the first task, changes to unprivileged level, initializes the system timer value, and starts executing the first task.

Returns

Result of kernel start.

Return values

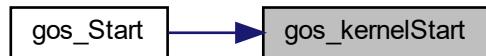
<i>GOS_ERROR</i>	First task terminated.
------------------	------------------------

Definition at line 354 of file gos_kernel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.20 gos_kernelSubscribeDumpReadySignal()

```
gos_result_t gos_kernelSubscribeDumpReadySignal (
    void_t (*) (u16_t) dumpReadySignalHandler )
```

Subscribes the given handler to the dump ready signal.

Subscribes the given handler to the dump ready signal.

Parameters

in	dumpReadySignalHandler	Dump ready signal handler function.
----	------------------------	-------------------------------------

Returns

Result of subscription.

Return values

GOS_SUCCESS	Subscription successful.
GOS_ERROR	Subscription failed or signal handler is NULL.

5.8.2.21 gos_taskSubscribeDeleteSignal()

```
gos_result_t gos_taskSubscribeDeleteSignal (
    void_t (*) (u16_t) deleteSignalHandler )
```

Subscribes the given handler to the task delete signal.

Subscribes the given handler to the task delete signal.

Parameters

in	<i>deleteSignalHandler</i>	Delete signal handler function.
----	----------------------------	---------------------------------

Returns

Result of subscription.

Return values

<i>GOS_SUCCESS</i>	Subscription successful.
<i>GOS_ERROR</i>	Subscription failed or signal handler is NULL.

5.9 Task-related functions

Functions

- `gos_result_t gos_taskRegisterTasks (gos_taskDescriptor_t *taskDescriptors, u16_t arraySize)`
This function registers an array of tasks for scheduling.
- `gos_result_t gos_taskRegister (gos_taskDescriptor_t *taskDescriptor, gos_tid_t *taskId)`
This function registers a task for scheduling.
- `gos_result_t gos_taskSleep (gos_taskSleepTick_t sleepTicks)`
Sends the current task to sleeping state.
- `gos_result_t gos_taskWakeup (gos_tid_t taskId)`
Wakes up the given task.
- `gos_result_t gos_taskSuspend (gos_tid_t taskId)`
Sends the given task to suspended state.
- `gos_result_t gos_taskResume (gos_tid_t taskId)`
Resumes the given task.
- `gos_result_t gos_taskBlock (gos_tid_t taskId, gos_blockMaxTick_t blockTicks)`
Sends the given task to blocked state.
- `gos_result_t gos_taskUnblock (gos_tid_t taskId)`
Unblocks the given task.
- `gos_result_t gos_taskDelete (gos_tid_t taskId)`
Deletes the given task from the scheduling array.
- `gos_result_t gos_taskSetPriority (gos_tid_t taskId, gos_taskPrio_t taskPriority)`
Sets the current priority of the given task to the given value (for temporary change).
- `gos_result_t gos_taskSetOriginalPriority (gos_tid_t taskId, gos_taskPrio_t taskPriority)`
Sets the original priority of the given task to the given value (for permanent change).
- `gos_result_t gos_taskGetPriority (gos_tid_t taskId, gos_taskPrio_t *taskPriority)`
Gets the current priority of the given task.
- `gos_result_t gos_taskGetOriginalPriority (gos_tid_t taskId, gos_taskPrio_t *taskPriority)`
Gets the original priority of the given task.
- `gos_result_t gos_taskAddPrivilege (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)`
Adds the given privileges to the given task.
- `gos_result_t gos_taskRemovePrivilege (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)`
Removes the given privileges from the given task.
- `gos_result_t gos_taskSetPrivileges (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)`
Sets the given privileges for the given task.
- `gos_result_t gos_taskGetPrivileges (gos_tid_t taskId, gos_taskPrivilegeLevel_t *pPrivileges)`
Gets the privileges of the given task.
- `gos_result_t gos_taskGetName (gos_tid_t taskId, gos_taskName_t taskName)`
Gets the task name of the task with the given ID.
- `gos_result_t gos_taskGetId (gos_taskName_t taskName, gos_tid_t *pTaskId)`
Gets the task ID of the task with the given name.
- `gos_result_t gos_taskGetCurrentId (gos_tid_t *pTaskId)`
Returns the ID of the currently running task.
- `gos_result_t gos_taskGetData (gos_tid_t taskId, gos_taskDescriptor_t *pTaskData)`
Returns the task data of the given task.
- `gos_result_t gos_taskGetDataByIndex (u16_t taskIndex, gos_taskDescriptor_t *pTaskData)`
Returns the task data of the given task.
- `gos_result_t gos_taskGetNumber (u16_t *pTaskNum)`
Returns the number of registered tasks.
- `gos_result_t gos_taskYield (void_t)`
Yields the current task.

5.9.1 Detailed Description

5.9.2 Function Documentation

5.9.2.1 gos_taskAddPrivilege()

```
gos_result_t gos_taskAddPrivilege (
    gos_tid_t taskId,
    gos_taskPrivilegeLevel_t privileges )
```

Adds the given privileges to the given task.

Checks the caller task if it has the privilege to modify task privileges and if so, it adds the given privileges to the given task.

Parameters

in	<i>taskId</i>	ID of the task to give the privileges to.
in	<i>privileges</i>	Privileges to be added.

Returns

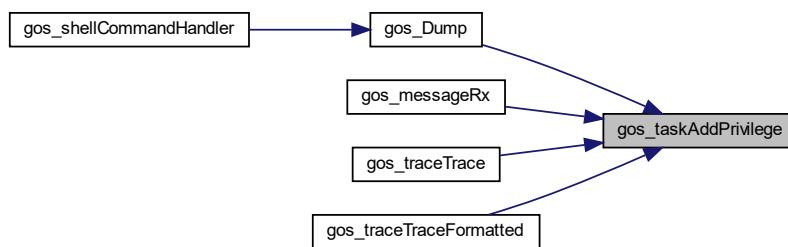
Result of privilege adding.

Return values

<i>GOS_SUCCESS</i>	Privileges added successfully.
<i>GOS_ERROR</i>	Invalid task ID or caller does not have the privilege to modify task privileges.

Definition at line 882 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.2 gos_taskBlock()

```
gos_result_t gos_taskBlock (
    gos_tid_t taskId,
    gos_blockMaxTick_t blockTicks )
```

Sends the given task to blocked state.

Checks the given task ID and its state, modified it to blocked, and if there is a block hook function registered, it calls it. If the blocked function is the currently running one, it invokes a rescheduling.

Parameters

in	<i>taskId</i>	ID of the task to be blocked.
in	<i>blockTicks</i>	Maximum number of ticks for blocking the task (timeout).

Returns

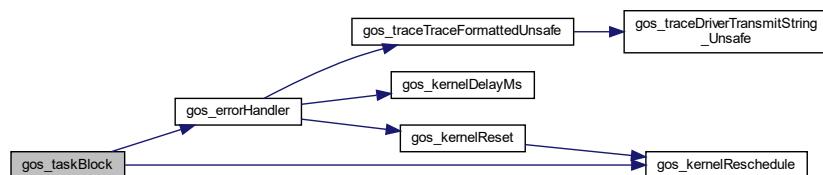
Result of task blocking.

Return values

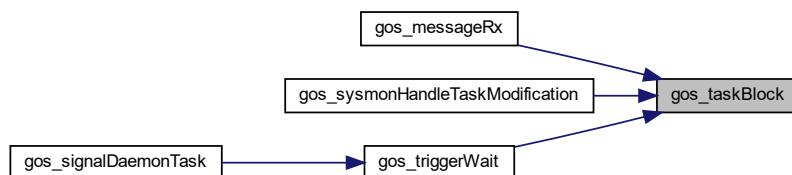
<i>GOS_SUCCESS</i>	Task blocked successfully.
<i>GOS_ERROR</i>	Task ID is invalid, or task state is not ready.

Definition at line 516 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.3 gos_taskDelete()

```
gos_result_t gos_taskDelete (
```

Deletes the given task from the scheduling array.

Checks the given task ID and its state, modifies it to zombie, and if there is a delete hook function registered, it calls it.

Parameters

in	<i>task</i> <i>Id</i>	ID of the task to be deleted.
----	--------------------------	-------------------------------

Returns

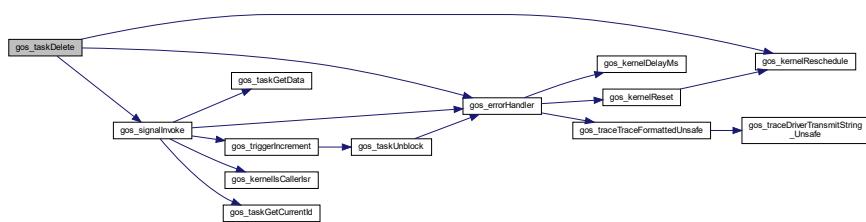
Result of deletion.

Return values

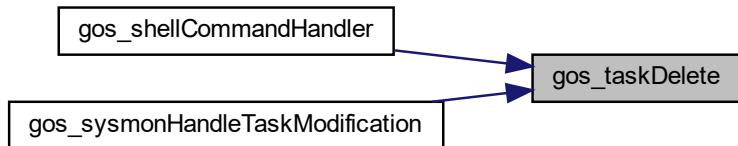
<i>GOS_SUCCESS</i>	Task deleted successfully.
<i>GOS_ERROR</i>	Task is already a zombie.

Definition at line 648 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.4 gos_taskGetCurrentId()

```
gos_result_t gos_taskGetCurrentId (
    gos_tid_t * pTaskId )
```

Returns the ID of the currently running task.

Returns the ID of the currently running task.

Parameters

out	$p \leftarrow$ <i>TaskId</i>	Pointer to a task ID variable to store the current task ID.
-----	---------------------------------	---

Returns

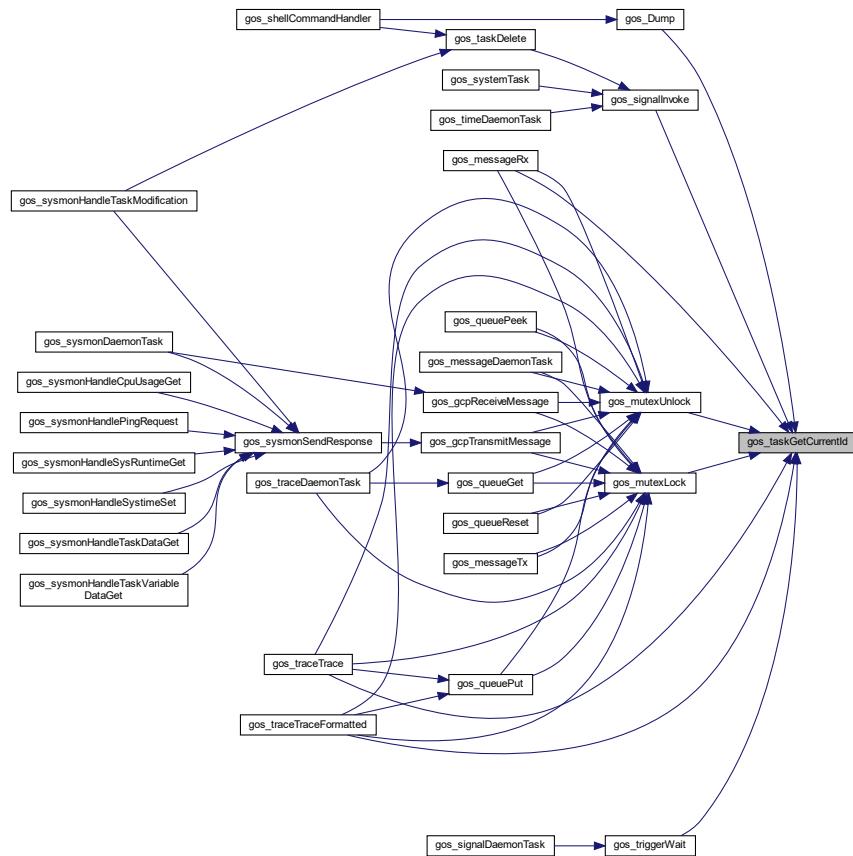
Result of current task ID get.

Return values

<i>GOS_SUCCESS</i>	Current task ID returned successfully.
<i>GOS_ERROR</i>	Task ID pointer is NULL.

Definition at line 1083 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.5 gos_taskGetData()

```
gos_result_t gos_taskGetData (
```

Returns the task data of the given task.

Based on the task ID, it copies the content of the internal task descriptor array element to the given task descriptor.

Parameters

in	<i>taskId</i>	ID of the task to get the data of.
out	<i>pTaskData</i>	Pointer to the task descriptor to save the task data in.

Returns

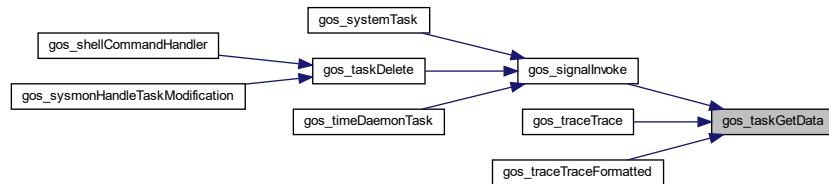
Result of task data get.

Return values

GOS_SUCCESS	Task data copied successfully.
GOS_ERROR	Invalid task ID or task data pointer is NULL.

Definition at line 1111 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.6 gos_taskGetDataByIndex()

```
gos_result_t gos_taskGetDataByIndex (
    u16_t taskIndex,
    gos_taskDescriptor_t * pTaskData )
```

Returns the task data of the given task.

Based on the task index, it copies the content of the internal task descriptor array element to the given task descriptor.

Parameters

in	<i>taskIndex</i>	Index of the task to get the data of.
out	<i>pTaskData</i>	Pointer to the task descriptor to save the task data in.

Returns

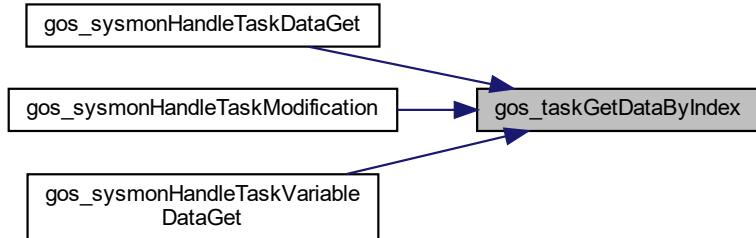
Result of task data get.

Return values

GOS_SUCCESS	Task data copied successfully.
GOS_ERROR	Invalid task index or task data pointer is NULL.

Definition at line 1144 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.7 gos_taskGetId()

```
gos_result_t gos_taskGetId (
    gos_taskName_t taskName,
    gos_tid_t * pTaskId )
```

Gets the task ID of the task with the given name.

This function loops through the internal task array and tries to find the given task name to get the corresponding task ID.

Parameters

in	<i>taskName</i>	Name of the task (string).
out	<i>pTaskId</i>	Pointer to a task ID variable to store the returned ID.

Returns

Success of task ID get.

Return values

<i>GOS_SUCCESS</i>	Task ID found successfully.
<i>GOS_ERROR</i>	Task name not found.

Definition at line 1052 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.8 gos_taskGetName()

```
gos_result_t gos_taskGetName (
    gos_tid_t taskId,
    gos_taskName_t taskName )
```

Gets the task name of the task with the given ID.

Copies the task name corresponding with the given task ID to the task name variable.

Parameters

in	<i>taskId</i>	Pointer to a task ID variable to store the returned ID.
out	<i>taskName</i>	Task name pointer to store the returned task name.

Returns

Success of task name get.

Return values

<i>GOS_SUCCESS</i>	Task name found successfully.
<i>GOS_ERROR</i>	Invalid task ID or task name variable is NULL.

Definition at line 1021 of file gos_task.c.

5.9.2.9 gos_taskGetNumber()

```
gos_result_t gos_taskGetNumber (
    u16_t * pTaskNum )
```

Returns the number of registered tasks.

Loops through the internal task array and counts the entries where a task function is registered.

Parameters

<code>out</code>	<code>pTaskNum</code>	Variable to store the number of tasks.
------------------	-----------------------	--

Returns

Success of task number counting.

Return values

<code>GOS_SUCCESS</code>	Task number counted successfully.
<code>GOS_ERROR</code>	Target variable is NULL pointer.

Definition at line 1175 of file gos_task.c.

5.9.2.10 gos_taskGetOriginalPriority()

```
gos_result_t gos_taskGetOriginalPriority (
    gos_tid_t taskId,
    gos_taskPrio_t * taskPriority )
```

Gets the original priority of the given task.

Checks the given parameters and saves the original priority in the given variable.

Parameters

<code>in</code>	<code>taskId</code>	ID of the task to get the priority of.
<code>out</code>	<code>taskPriority</code>	Pointer to a priority variable to store the priority in.

Returns

Result of priority getting.

Return values

<code>GOS_SUCCESS</code>	Original priority getting successfully.
<code>GOS_ERROR</code>	Invalid task ID or priority variable is NULL.

Definition at line 851 of file gos_task.c.

5.9.2.11 gos_taskGetPriority()

```
gos_result_t gos_taskGetPriority (
```

```
gos_tid_t taskId,
gos_taskPrio_t * taskPriority )
```

Gets the current priority of the given task.

Checks the given parameters and saves the current priority in the given variable.

Parameters

in	<i>taskId</i>	ID of the task to get the priority of.
out	<i>taskPriority</i>	Pointer to a priority variable to store the priority in.

Returns

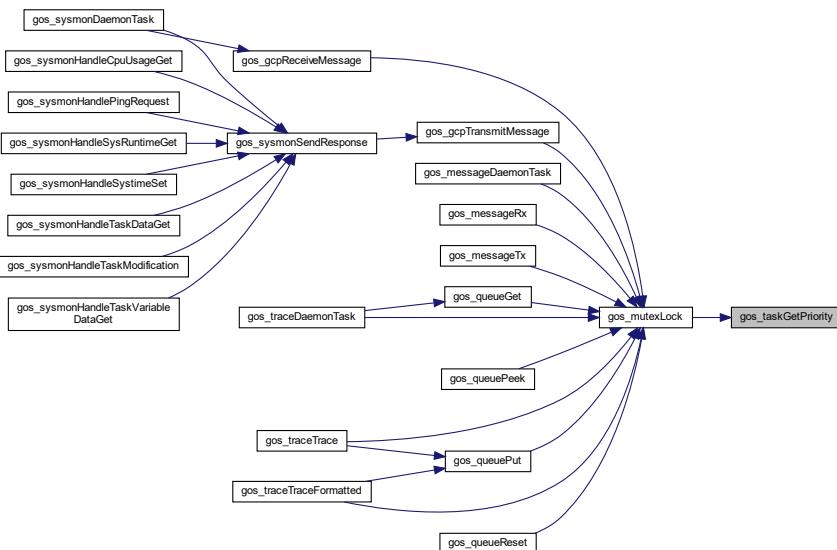
Result of priority getting.

Return values

<i>GOS_SUCCESS</i>	Current priority getting successfully.
<i>GOS_ERROR</i>	Invalid task ID or priority variable is NULL.

Definition at line 820 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.12 `gos_taskGetPrivileges()`

```
gos_result_t gos_taskGetPrivileges (
    gos_tid_t taskId,
    gos_taskPrivilegeLevel_t * pPrivileges )
```

Gets the privileges of the given task.

Returns the privilege flags of the given task.

Parameters

in	<i>taskId</i>	ID of the task to get the privileges of.
out	<i>pPrivileges</i>	Variable to store the privilege flags.

Returns

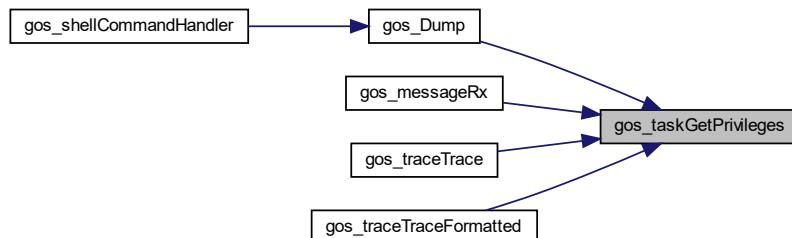
Result of privilege getting.

Return values

<i>GOS_SUCCESS</i>	Privileges get successful.
<i>GOS_ERROR</i>	Invalid task ID or privilege variable is NULL pointer.

Definition at line 972 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.13 gos_taskRegister()

```
gos_result_t gos_taskRegister (
    gos_taskDescriptor_t * taskDescriptor,
    gos_tid_t * taskId )
```

This function registers a task for scheduling.

Checks the task descriptor parameters and then tries to find the next empty slot in the internal task array. When it is found, it registers the task in that slot.

Parameters

in	<i>taskDescriptor</i>	Pointer to a task descriptor structure.
in	<i>taskId</i>	Pointer to a variable to hold to assigned task ID value.

Returns

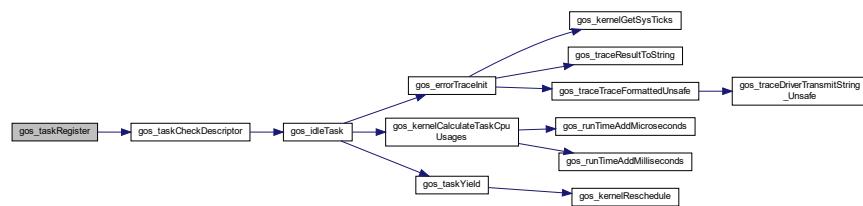
Result of task registration.

Return values

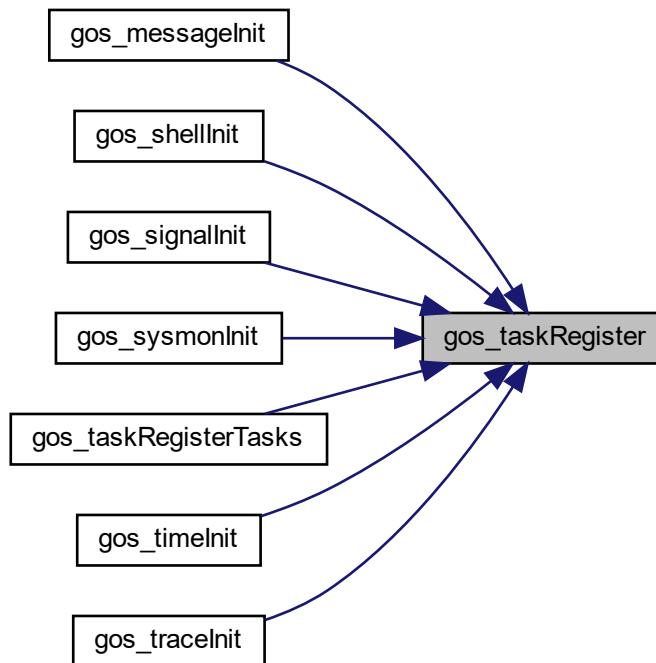
GOS_SUCCESS	Task registered successfully.
GOS_ERROR	Invalid task descriptor (NULL function pointer, invalid priority level, invalid stack size, idle task registration, or stack size is not 4-byte-aligned) or task array is full.

Definition at line 161 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.14 gos_taskRegisterTasks()

```
gos_result_t gos_taskRegisterTasks (
    gos_taskDescriptor_t * taskDescriptors,
    u16_t arraySize )
```

This function registers an array of tasks for scheduling.

Checks the task descriptor array pointer and registers the tasks one by one.

Parameters

in	<i>taskDescriptors</i>	Pointer to a task descriptor structure array.
in	<i>arraySize</i>	Size of the array in bytes.

Returns

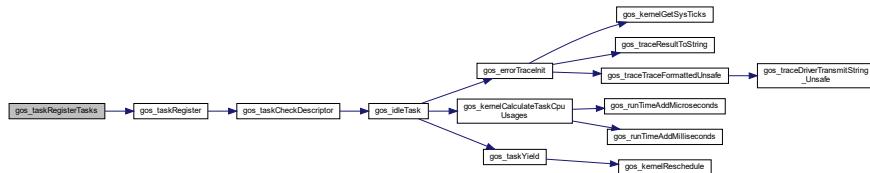
Result of task registration.

Return values

<i>GOS_SUCCESS</i>	Tasks registered successfully.
<i>GOS_ERROR</i>	Invalid task descriptor (NULL function pointer, invalid priority level, invalid stack size, idle task registration, or stack size is not 4-byte-aligned) in one of the array elements or task array is full.

Definition at line 113 of file gos_task.c.

Here is the call graph for this function:



5.9.2.15 gos_taskRemovePrivilege()

```
gos_result_t gos_taskRemovePrivilege (
    gos_tid_t taskId,
    gos_taskPrivilegeLevel_t privileges )
```

Removes the given privileges from the given task.

Checks the caller task if it has the privilege to modify task privileges and if so, it removes the given privileges from the given task.

Parameters

in	<i>taskId</i>	ID of the task to remove the privileges from.
in	<i>privileges</i>	Privileges to be removed.

Returns

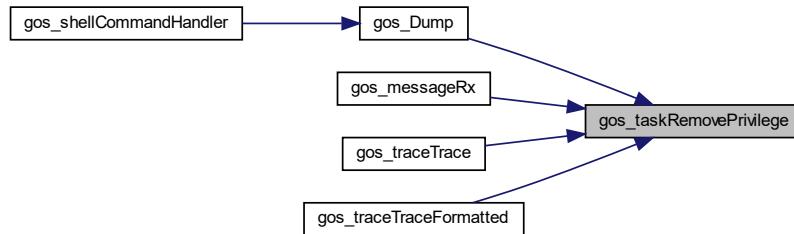
Result of privilege removing.

Return values

<i>GOS_SUCCESS</i>	Privileges removed successfully.
<i>GOS_ERROR</i>	Invalid task ID or caller does not have the privilege to modify task privileges.

Definition at line 912 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.16 gos_taskResume()

```
gos_result_t gos_taskResume (
    gos_tid_t taskId )
```

Resumes the given task.

Checks the given task ID and its state, modified it to ready, and if there is a resume hook function registered, it calls it.

Parameters

in	<i>task←Id</i>	ID of the task to be resumed.
----	----------------	-------------------------------

Returns

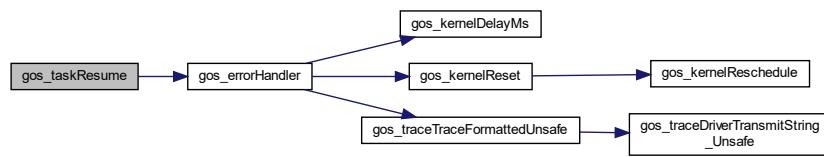
Result of task resumption.

Return values

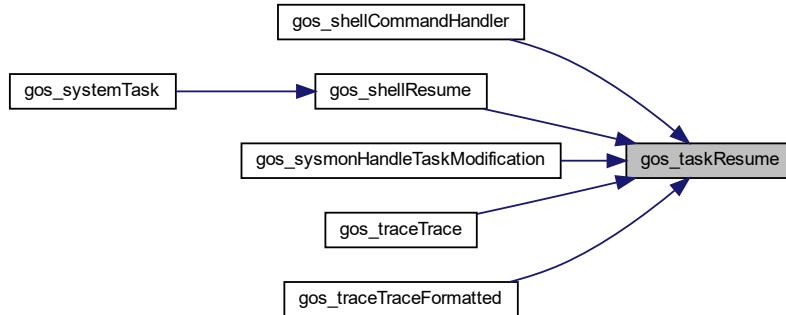
GOS_SUCCESS	Task resumed successfully.
GOS_ERROR	Task ID is invalid, or task is not suspended.

Definition at line 465 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:

**5.9.2.17 gos_taskSetOriginalPriority()**

```
gos_result_t gos_taskSetOriginalPriority (
    gos_tid_t taskId,
    gos_taskPrio_t taskPriority )
```

Sets the original priority of the given task to the given value (for permanent change).

Checks the given parameters and sets the original priority of the given task.

Parameters

in	<i>taskId</i>	ID of the task to change the priority of.
in	<i>taskPriority</i>	The desired task priority.

Returns

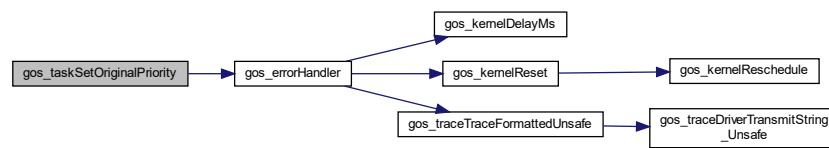
Result of priority change.

Return values

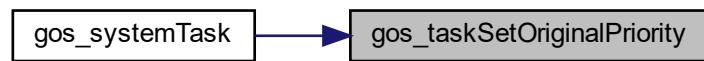
<i>GOS_SUCCESS</i>	Original priority changed successfully.
<i>GOS_ERROR</i>	Invalid task ID or priority.

Definition at line 774 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.18 gos_taskSetPriority()

```

gos_result_t gos_taskSetPriority (
    gos_tid_t taskId,
    gos_taskPrio_t taskPriority )
  
```

Sets the current priority of the given task to the given value (for temporary change).

Checks the given parameters and sets the current priority of the given task.

Parameters

in	<i>taskId</i>	ID of the task to change the priority of.
in	<i>taskPriority</i>	The desired task priority.

Returns

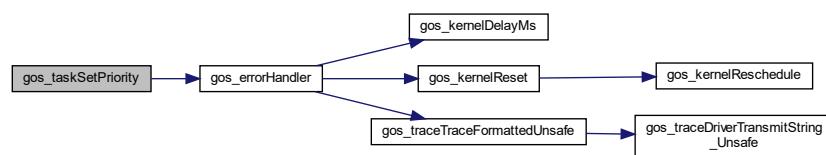
Result of priority change.

Return values

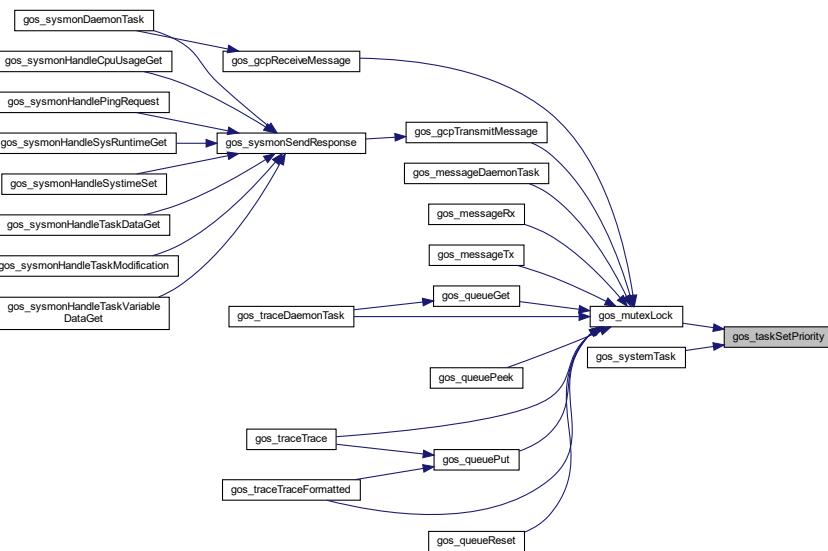
<i>GOS_SUCCESS</i>	Current priority changed successfully.
<i>GOS_ERROR</i>	Invalid task ID or priority.

Definition at line 728 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.19 gos_taskSetPrivileges()

```
gos_result_t gos_taskSetPrivileges (
    gos_tid_t taskId,
    gos_taskPrivilegeLevel_t privileges )
```

Sets the given privileges for the given task.

Checks the caller task if it has the privilege to modify task privileges and if so, it sets the given privileges for the given task.

Parameters

in	<i>taskId</i>	ID of the task to set the privileges for.
in	<i>privileges</i>	Privileges to be set.

Returns

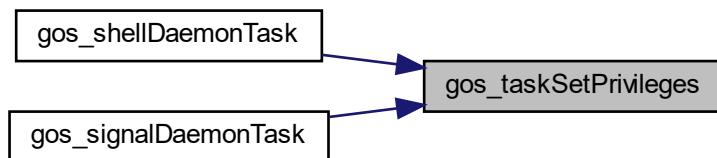
Result of privilege setting.

Return values

<i>GOS_SUCCESS</i>	Privileges set successfully.
<i>GOS_ERROR</i>	Invalid task ID or caller does not have the privilege to modify task privileges.

Definition at line 942 of file gos_task.c.

Here is the caller graph for this function:



5.9.2.20 gos_taskSleep()

```
gos_result_t gos_taskSleep (
    gos_taskSleepTick_t sleepTicks )
```

Sends the current task to sleeping state.

Checks the current task and its state, modifies it to sleeping, and if there is a sleep hook function registered, it calls it. Then, it invokes a rescheduling.

Parameters

in	<i>sleepTicks</i>	Minimum number of ticks until the task should remain in sleeping state.
----	-------------------	---

Returns

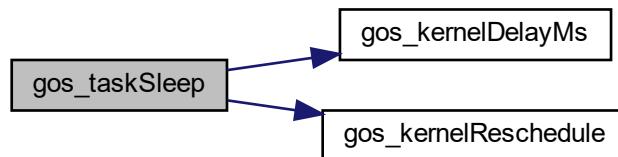
Result of task sleeping.

Return values

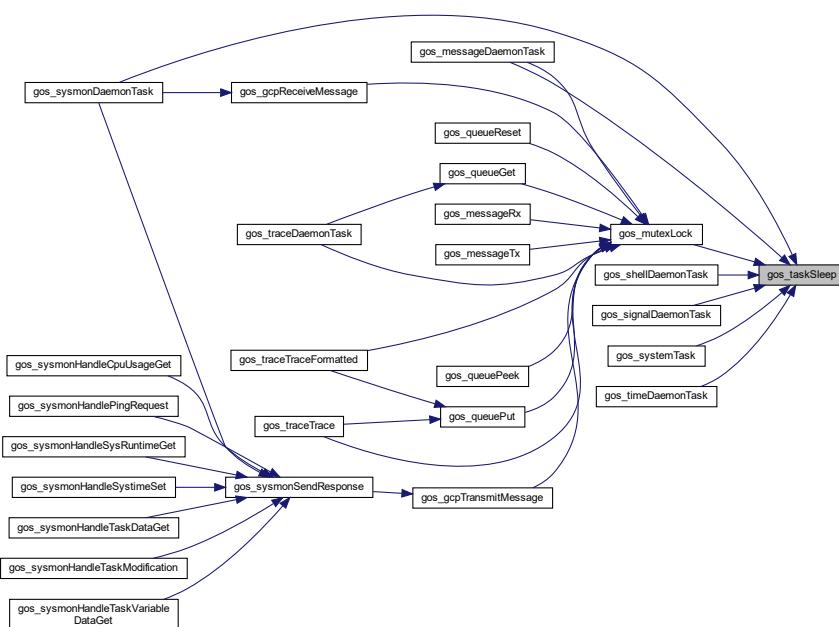
<i>GOS_SUCCESS</i>	Task successfully sent to sleeping state.
<i>GOS_ERROR</i>	Function called from idle task or task state is not ready.

Definition at line 284 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.21 gos_taskSuspend()

```
gos_result_t gos_taskSuspend (
    gos_tid_t taskId )
```

Sends the given task to suspended state.

Checks the given task ID and its state, modified it to suspended, and if there is a suspend hook function registered, it calls it. If the suspended function is the currently running one, it invokes a rescheduling.

Parameters

in	<i>task</i> → <i>Id</i>	ID of the task to be suspended.
----	----------------------------	---------------------------------

Returns

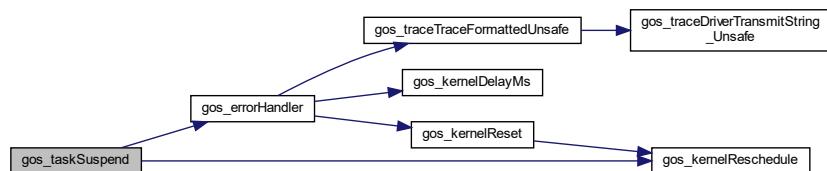
Result of task suspension.

Return values

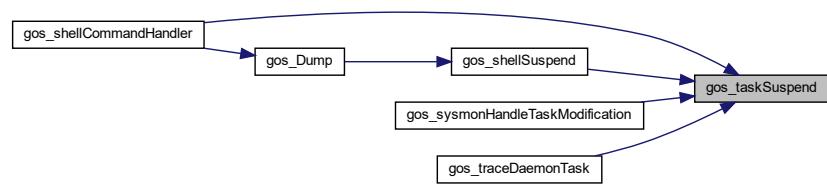
<i>GOS_SUCCESS</i>	Task suspended successfully.
<i>GOS_ERROR</i>	Task ID is invalid, or task state is not ready or sleeping.

Definition at line 391 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.22 gos_taskUnblock()

```
gos_result_t gos_taskUnblock (
    gos_tid_t taskId )
```

Unblocks the given task.

Checks the given task ID and its state, modified it to ready, and if there is an unblock hook function registered, it calls it.

Parameters

in	<i>task</i> ↪ <i>Id</i>	ID of the task to be unblocked.
----	----------------------------	---------------------------------

Returns

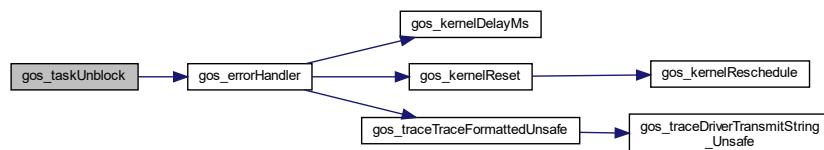
Result of task unblocking.

Return values

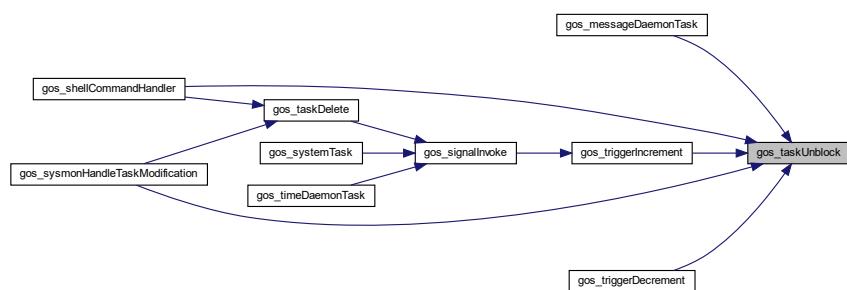
GOS_SUCCESS	Task unblocked successfully.
GOS_ERROR	Task ID is invalid, or task is not blocked.

Definition at line 591 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.23 gos_taskWakeup()

```
gos_result_t gos_taskWakeup (
    gos_tid_t taskId )
```

Wakes up the given task.

Checks the current task and its state, modifies it to ready, and if there is a wake-up hook function registered, it calls it.

Parameters

in	<i>task</i> <i>Id</i>	ID of the task to be waken up.
----	--------------------------	--------------------------------

Returns

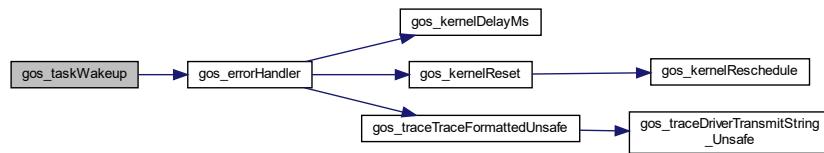
Result of task wake-up.

Return values

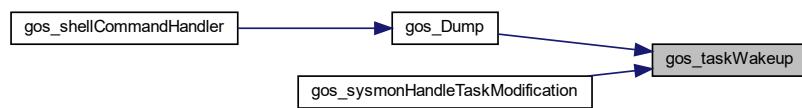
<i>GOS_SUCCESS</i>	Task waken up successfully.
<i>GOS_ERROR</i>	Task ID is invalid, or task is not sleeping.

Definition at line 340 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.24 gos_taskYield()

```
gos_result_t gos_taskYield (
    void_t    )
```

Yields the current task.

Invokes rescheduling.

Returns

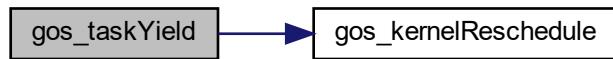
Result of task yield.

Return values

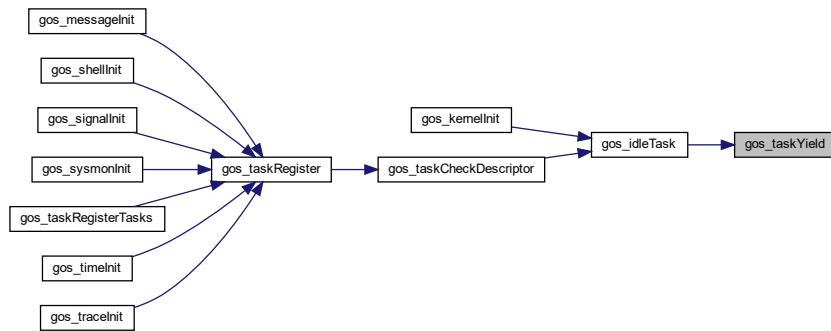
GOS_SUCCESS	Yield successful.
-----------------------------	-------------------

Definition at line 1004 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10 Platform initializer weak functions

Functions

- `gos_result_t gos_platformDriverInit (void_t)`
Platform driver initializer. Used for the platform-specific driver initializations.
- `gos_result_t gos_userApplicationInit (void_t)`
User application initializer. Used for the application-related initializations.

5.10.1 Detailed Description

5.10.2 Function Documentation

5.10.2.1 `gos_platformDriverInit()`

```
gos_result_t gos_platformDriverInit (
    void_t )
```

Platform driver initializer. Used for the platform-specific driver initializations.

This function is weak and therefore should be over-defined by the user. It prints a warning message to the log output in case it is not over-defined.

Returns

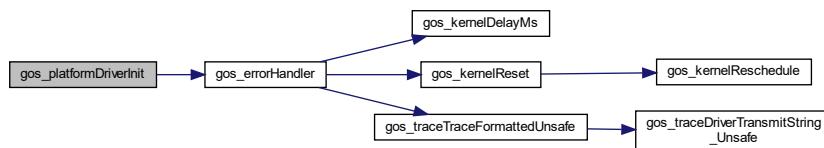
-

Return values

<code>GOS_ERROR</code>	-
------------------------	---

Definition at line 240 of file gos.c.

Here is the call graph for this function:



5.10.2.2 gos_userApplicationInit()

```
gos_result_t gos_userApplicationInit (
    void_t     )
```

User application initializer. Used for the application-related initializations.

This function is weak and therefore should be over-defined by the user. It prints a warning message to the log output in case it is not over-defined.

Returns

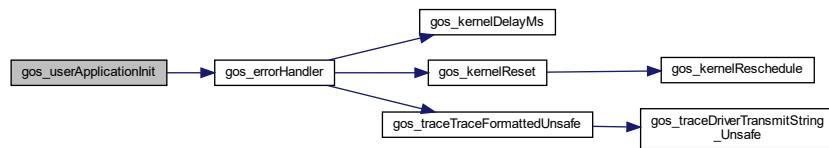
-

Return values

GOS_ERROR	-
-----------	---

Definition at line 252 of file gos.c.

Here is the call graph for this function:



5.11 Trace formatter macros

Macros

- `#define TRACE_FORMAT_RESET "\x1B[0m"`
Reset formatting.
- `#define TRACE_FG_RED_START "\x1B[31m"`
Red foreground start.
- `#define TRACE_FG_GREEN_START "\x1B[32m"`
Green foreground start.
- `#define TRACE_FG_YELLOW_START "\x1B[33m"`
Yellow foreground start.
- `#define TRACE_FG_BLUE_START "\x1B[34m"`
Blue foreground start.
- `#define TRACE_FG_MAGENTA_START "\x1B[35m"`
Magenta foreground start.
- `#define TRACE_FG_CYAN_START "\x1B[36m"`
Cyan foreground start.
- `#define TRACE_FG_WHITE_START "\x1B[37m"`
White foreground start.
- `#define TRACE_BG_RED_START "\x1B[41m"`
Red background start.
- `#define TRACE_BG_GREEN_START "\x1B[42m"`
Green background start.
- `#define TRACE_BG_YELLOW_START "\x1B[43m"`
Yellow background start.
- `#define TRACE_BG_BLUE_START "\x1B[44m"`
Blue background start.
- `#define TRACE_BG_MAGENTA_START "\x1B[45m"`
Magenta background start.
- `#define TRACE_BG_CYAN_START "\x1B[46m"`
Cyan background start.
- `#define TRACE_BG_WHITE_START "\x1B[47m"`
White background start.
- `#define TRACE_BOLD_START "\x1B[1m"`
Bold start.
- `#define TRACE_ITALIC_START "\x1B[3m"`
Italic start.
- `#define TRACE_UNDERLINE_START "\x1B[4m"`
Underline start.
- `#define TRACE_STRIKETHROUGH_START "\x1B[9m"`
Strikethrough start.

5.11.1 Detailed Description

5.12 System monitoring message structures

Data Structures

- struct `gos_sysmonPingMessage_t`
- struct `gos_sysmonCpuUsageMessage_t`
- struct `gos_sysmonTaskDataGetMessage_t`
- struct `gos_sysmonTaskDataMessage_t`
- struct `gos_sysmonTaskVariableDataMessage_t`
- struct `gos_sysmonTaskModifyMessage_t`
- struct `gos_sysmonTaskModifyResultMessage_t`
- struct `gos_sysmonSysruntimeGetResultMessage_t`
- struct `gos_sysmonSystimeSetMessage_t`
- struct `gos_sysmonSystimeSetResultMessage_t`

5.12.1 Detailed Description

Chapter 6

Data Structure Documentation

6.1 gos_driver_functions_t Struct Reference

```
#include <GOS2022/OS/driver/inc/gos_driver.h>
```

Data Fields

- [gos_shellDriverReceiveChar_t shellDriverReceiveChar](#)
Shell character receive function.
- [gos_shellDriverTransmitString_t shellDriverTransmitString](#)
Shell string transmit function.
- [gos_traceDriverTransmitString_t traceDriverTransmitString](#)
Log string transmit function.
- [gos_traceDriverTransmitString_Unsafe_t traceDriverTransmitStringUnsafe](#)
Log unsafe string transmit function.
- [gos_timerDriverSysTimerGetVal_t timerDriverSysTimerGetValue](#)
System timer get function.
- [gos_sysmonDriverTransmit_t sysmonDriverTransmit](#)
Sysmon transmit function.
- [gos_sysmonDriverReceive_t sysmonDriverReceive](#)
Sysmon receive function.

6.1.1 Detailed Description

Driver functions type.

Definition at line 67 of file gos_driver.h.

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

- GOS2022/OS/driver/inc/[gos_driver.h](#)

6.2 gos_gcpChannelFunctions_t Struct Reference

Data Fields

- `gos_gcpTransmitFunction_t gcpTransmitFunction`
GCP transmit function.
- `gos_gcpReceiveFunction_t gcpReceiveFunction`
GCP receive function.

6.2.1 Detailed Description

GCP channel functions type.

Definition at line 114 of file `gos_gcp.c`.

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

- GOS2022/OS/services/src/[gos_gcp.c](#)

6.3 gos_gcpHeaderFrame_t Struct Reference

Data Fields

- `u8_t protocolMajor`
Protocol version major.
- `u8_t protocolMinor`
Protocol version minor.
- `u8_t ackType`
Acknowledge type.
- `u8_t dummy`
Dummy byte (padding).
- `u16_t messageId`
Message ID.
- `u16_t dataSize`
Data size.
- `u32_t dataCrc`
Data CRC.
- `u32_t headerCrc`
Header CRC.

6.3.1 Detailed Description

GCP frame header type.

Definition at line 99 of file `gos_gcp.c`.

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

- GOS2022/OS/services/src/[gos_gcp.c](#)

6.4 gos_initStruct_t Struct Reference

Data Fields

- [char_t initDesc](#) [32]
Initialization descriptor text.
- [gos_initFunc_t initFunc](#)
Initializer function.

6.4.1 Detailed Description

Initializer structure type.

Definition at line 89 of file gos.c.

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

- GOS2022/OS/services/src/[gos.c](#)

6.5 gos_message_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_message.h>
```

Data Fields

- [gos_messageId_t messageId](#)
Message ID.
- [gos_messageSize_t messageSize](#)
Message size.
- [u8_t messageBytes](#) [[CFG_MESSAGE_MAX_LENGTH](#)]
Message bytes.

6.5.1 Detailed Description

Message type.

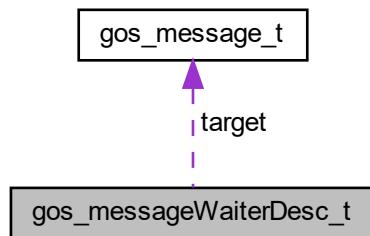
Definition at line 125 of file gos_message.h.

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

- GOS2022/OS/services/inc/[gos_message.h](#)

6.6 gos_messageWaiterDesc_t Struct Reference

Collaboration diagram for gos_messageWaiterDesc_t:



Data Fields

- `gos_tid_t waiterTaskId`
Waiter task ID.
- `gos_messageTimeout_t waitTmo`
Wait timeout value.
- `gos_messageTimeout_t waitTmoCounter`
Wait timeout counter.
- `gos_messageId_t messageIdArray [CFG_MESSAGE_MAX_WAITER_IDS]`
Message ID array.
- `gos_message_t * target`
Target buffer.
- `bool_t waiterServed`
Waiter served flag.

6.6.1 Detailed Description

Message waiter descriptor type.

Definition at line 87 of file gos_message.c.

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

- GOS2022/OS/services/src/gos_message.c

6.7 gos_mutex_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_mutex.h>
```

Data Fields

- [gos_mutexState_t mutexState](#)
Mutex state.
- [gos_tid_t owner](#)
Mutex owner task.

6.7.1 Detailed Description

Mutex type.

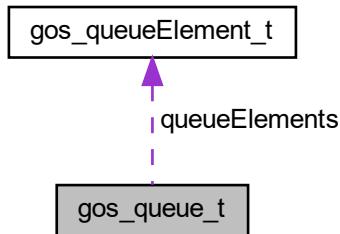
Definition at line 88 of file [gos_mutex.h](#).

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

- GOS2022/OS/services/inc/[gos_mutex.h](#)

6.8 gos_queue_t Struct Reference

Collaboration diagram for gos_queue_t:



Data Fields

- [gos_queueId_t queueId](#)
Queue ID.
- [gos_queueElement_t queueElements \[CFG_QUEUE_MAX_ELEMENTS\]](#)
Queue element array.
- [gos_queueIndex_t actualElementNumber](#)
Actual number of queue elements.

6.8.1 Detailed Description

Queue type.

Definition at line 101 of file gos_queue.c.

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

- GOS2022/OS/services/src/gos_queue.c

6.9 gos_queueDescriptor_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_queue.h>
```

Data Fields

- [gos_queueId_t queueId](#)

Queue ID.

6.9.1 Detailed Description

Public queue descriptor type.

Definition at line 121 of file gos_queue.h.

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

- GOS2022/OS/services/inc/gos_queue.h

6.10 gos_queueElement_t Struct Reference

Data Fields

- [gos_queueByte_t queueElementBytes \[CFG_QUEUE_MAX_LENGTH\]](#)
Queue element bytes.
- [gos_queueLength_t elementLength](#)
Queue element length.

6.10.1 Detailed Description

Queue element type.

Definition at line 92 of file gos_queue.c.

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

- GOS2022/OS/services/src/gos_queue.c

6.11 gos_runtime_t Struct Reference

```
#include <GOS2022/OS/kernel/inc/gos_kernel.h>
```

Data Fields

- `gos_microsecond_t microseconds`
Microseconds.
- `gos_millisecond_t milliseconds`
Milliseconds.
- `gos_second_t seconds`
Seconds.
- `gos_minute_t minutes`
Minutes.
- `gos_hour_t hours`
Hours.
- `gos_day_t days`
Days.

6.11.1 Detailed Description

Run-time type.

Definition at line 501 of file `gos_kernel.h`.

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

- GOS2022/OS/kernel/inc/`gos_kernel.h`

6.12 gos_shellCommand_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_shell.h>
```

Data Fields

- `char_t command [CFG_SHELL_MAX_COMMAND_LENGTH]`
Command name.
- `gos_shellFunction commandHandler`
Command handler function.
- `gos_taskPrivilegeLevel_t commandHandlerPrivileges`
Command handler privileges.

6.12.1 Detailed Description

Shell command type.

Definition at line 75 of file gos_shell.h.

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

- GOS2022/OS/services/inc/gos_shell.h

6.13 gos_signalDescriptor_t Struct Reference

Data Fields

- `bool_t inUse`
Flag to indicate whether the signal is in use.
- `gos_signalHandler_t handlers [CFG_SIGNAL_MAX_SUBSCRIBERS]`
Signal handler array.
- `gos_taskPrivilegeLevel_t handlerPrivileges [CFG_SIGNAL_MAX_SUBSCRIBERS]`
Signal handler privileges array.
- `bool_t invokeRequired`
Invoke required flag.
- `gos_signalSenderId_t senderId`
Sender ID.

6.13.1 Detailed Description

Signal descriptor type.

Definition at line 84 of file gos_signal.c.

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

- GOS2022/OS/services/src/gos_signal.c

6.14 gos_signallInvokeDescriptor Struct Reference

Data Fields

- `gos_signallId_t signallId`
Signal ID.
- `gos_signalSenderId_t senderId`
Sender ID.

6.14.1 Detailed Description

Signal invoke descriptor type.

Definition at line 96 of file gos_signal.c.

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

- GOS2022/OS/services/src/gos_signal.c

6.15 gos_sysmonCpuUsageMessage_t Struct Reference

Data Fields

- `gos_sysmonMessageResult_t messageResult`
Message result.
- `u16_t cpuUsage`
CPU usage x100[%].

6.15.1 Detailed Description

CPU usage message structure.

Definition at line 222 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/gos_sysmon.c

6.16 gos_sysmonLut_t Struct Reference

Data Fields

- `gos_sysmonMessageId_t messageId`
Message ID.
- `gos_sysmonMessagePv_t messagePv`
Message PV.
- `void_t * pMessagePayload`
Payload pointer.
- `u16_t payloadSize`
Payload size.
- `gos_sysmonMessageHandler_t pHandler`
Handler function pointer.

6.16.1 Detailed Description

Look-up table entry structure.

Definition at line 308 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.17 gos_sysmonPingMessage_t Struct Reference

Data Fields

- [gos_sysmonMessageResult_t messageResult](#)
Message result.

6.17.1 Detailed Description

Ping message structure.

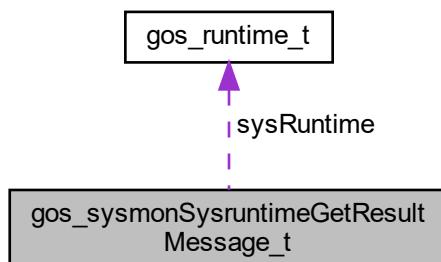
Definition at line 214 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.18 gos_sysmonSysruntimeGetResultMessage_t Struct Reference

Collaboration diagram for gos_sysmonSysruntimeGetResultMessage_t:



Data Fields

- [gos_sysmonMessageResult_t messageResult](#)
Message result.
- [gos_runtime_t sysRuntime](#)
System runtime.

6.18.1 Detailed Description

System runtime get message result structure.

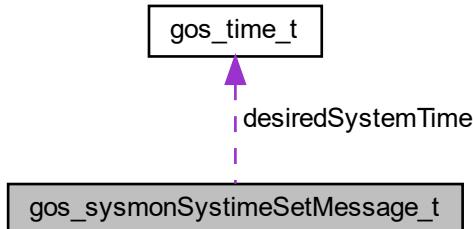
Definition at line 275 of file [gos_sysmon.c](#).

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.19 gos_sysmonSystimeSetMessage_t Struct Reference

Collaboration diagram for [gos_sysmonSystimeSetMessage_t](#):



Data Fields

- [gos_time_t desiredSystemTime](#)
Desired system time.

6.19.1 Detailed Description

System time set message structure.

Definition at line 284 of file [gos_sysmon.c](#).

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.20 gos_sysmonSystimeSetResultMessage_t Struct Reference

Data Fields

- [gos_sysmonMessageResult_t messageResult](#)
Message result.

6.20.1 Detailed Description

System time set message result structure.

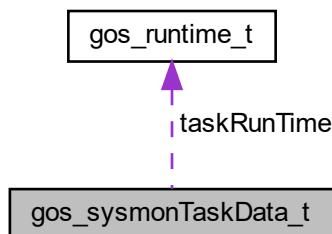
Definition at line 292 of file [gos_sysmon.c](#).

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.21 gos_sysmonTaskData_t Struct Reference

Collaboration diagram for gos_sysmonTaskData_t:



Data Fields

- [gos_taskState_t taskState](#)
Task state.
- [gos_taskPrio_t taskPriority](#)
Task priority.
- [gos_taskPrio_t taskOriginalPriority](#)
Task original priority.
- [gos_taskPrivilegeLevel_t taskPrivilegeLevel](#)
Task privilege level.
- [gos_taskName_t taskName](#)
Task name.

- `gos_tid_t taskId`
Task ID (internal).
- `gos_taskCSCCounter_t taskCsCounter`
Task context-switch counter.
- `gos_taskStackSize_t taskStackSize`
Task stack size.
- `gos_runtime_t taskRunTime`
Task run-time.
- `u16_t taskCpuUsageLimit`
Task CPU usage limit in [% x 100].
- `u16_t taskCpuUsageMax`
Task CPU usage max value in [% x 100].
- `u16_t taskCpuUsage`
Task processor usage in [% x 100].
- `gos_taskStackSize_t taskStackMaxUsage`
Task max. stack usage.

6.21.1 Detailed Description

Task data message structure.

Definition at line 176 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/gos_sysmon.c

6.22 gos_sysmonTaskDataGetMessage_t Struct Reference

Data Fields

- `u16_t taskIndex`
Task index.

6.22.1 Detailed Description

Task data get message structure.

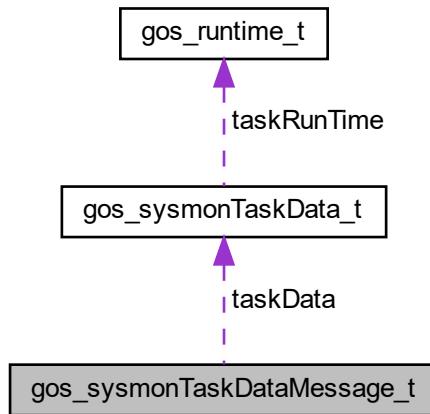
Definition at line 231 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/gos_sysmon.c

6.23 gos_sysmonTaskDataMessage_t Struct Reference

Collaboration diagram for gos_sysmonTaskDataMessage_t:



Data Fields

- [gos_sysmonMessageResult_t messageResult](#)
Message result.
- [gos_sysmonTaskData_t taskData](#)
Task data.

6.23.1 Detailed Description

Task data message structure.

Definition at line 239 of file [gos_sysmon.c](#).

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.24 gos_sysmonTaskModifyMessage_t Struct Reference

Data Fields

- [u16_t taskIndex](#)
Task index.
- [gos_sysmonTaskModifyType_t modificationType](#)
Task modification type.
- [u32_t param](#)
Parameter for functions where timeout is required.

6.24.1 Detailed Description

Task modify message structure.

Definition at line 257 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.25 gos_sysmonTaskModifyResultMessage_t Struct Reference

Data Fields

- [gos_sysmonMessageResult_t messageResult](#)

Message result.

6.25.1 Detailed Description

Task modify message result structure.

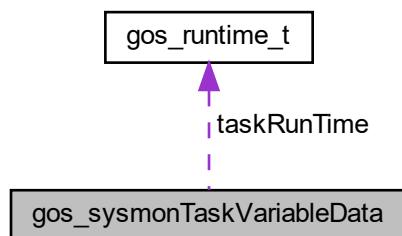
Definition at line 267 of file gos_sysmon.c.

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.26 gos_sysmonTaskVariableData Struct Reference

Collaboration diagram for gos_sysmonTaskVariableData:



Data Fields

- `gos_taskState_t taskState`
Task state.
- `gos_taskPrio_t taskPriority`
Task priority.
- `gos_taskCSCounter_t taskCsCounter`
Task context-switch counter.
- `gos_runtime_t taskRunTime`
Task run-time.
- `u16_t taskCpuUsageMax`
Task CPU usage max value in [% x 100].
- `u16_t taskCpuUsage`
Task processor usage in [% x 100].
- `gos_taskStackSize_t taskStackMaxUsage`
Task stack usage.

6.26.1 Detailed Description

Task variable data message structure.

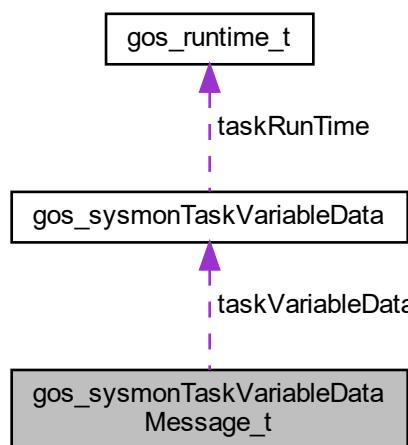
Definition at line 196 of file `gos_sysmon.c`.

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.27 `gos_sysmonTaskVariableDataMessage_t` Struct Reference

Collaboration diagram for `gos_sysmonTaskVariableDataMessage_t`:



Data Fields

- [gos_sysmonMessageResult_t messageResult](#)
Message result.
- [gos_sysmonTaskVariableData taskVariableData](#)
Task variable data.

6.27.1 Detailed Description

Task variable data message structure.

Definition at line 248 of file [gos_sysmon.c](#).

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

- GOS2022/OS/services/src/[gos_sysmon.c](#)

6.28 gos_sysmonUserMessageDescriptor_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_sysmon.h>
```

Data Fields

- [u16_t messageId](#)
Message ID.
- [u16_t protocolVersion](#)
Message PV.
- [void_t * payload](#)
Pointer to payload target.
- [u32_t payloadSize](#)
Size of payload.
- [gos_sysmonMessageReceivedCallback callback](#)
Callback function pointer.

6.28.1 Detailed Description

User sysmon message descriptor.

Definition at line 70 of file [gos_sysmon.h](#).

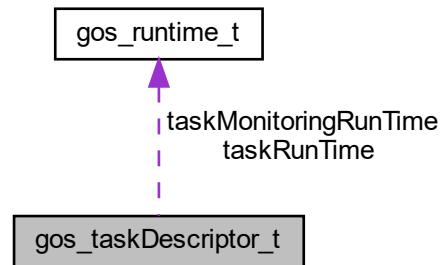
The documentation for this struct was generated from the following file:

- GOS2022/OS/services/inc/[gos_sysmon.h](#)

6.29 gos_taskDescriptor_t Struct Reference

```
#include <GOS2022/OS/kernel/inc/gos_kernel.h>
```

Collaboration diagram for gos_taskDescriptor_t:



Data Fields

- `gos_task_t taskFunction`
Task function.
- `gos_taskState_t taskState`
Task state.
- `gos_taskState_t taskPreviousState`
Task previous state (for restoration).
- `gos_taskPrio_t taskPriority`
Task priority.
- `gos_taskPrio_t taskOriginalPriority`
Task original priority.
- `gos_taskPrivilegeLevel_t taskPrivilegeLevel`
Task privilege level.
- `gos_taskName_t taskName`
Task name.
- `gos_tid_t taskId`
Task ID (internal).
- `gos_taskSleepTick_t taskSleepTicks`
Task sleep ticks.
- `gos_taskSleepTick_t taskSleepTickCounter`
Task sleep tick counter.
- `gos_blockMaxTick_t taskBlockTicks`
Task block ticks.
- `gos_blockMaxTick_t taskBlockTickCounter`
Task block tick counter.
- `gos_taskAddress_t taskPsp`
Task PSP.
- `gos_taskRunCounter_t taskRunCounter`

- `gos_taskCSCCounter_t taskCsCounter`
Task run counter.
- `gos_taskStackSize_t taskStackSize`
Task allocated stack size.
- `gos_taskStackSize_t taskStackSizeMaxUsage`
Task max. stack size usage.
- `gos_runtime_t taskRunTime`
Task run-time.
- `gos_runtime_t taskMonitoringRunTime`
Task monitoring run-time (not erased).
- `u16_t taskCpuUsageLimit`
Task CPU usage limit in [% x 100].
- `u16_t taskCpuUsageMax`
Task CPU usage max value in [% x 100].
- `u16_t taskCpuUsage`
Task processor usage in [% x 100].
- `u16_t taskCpuMonitoringUsage`
Task CPU usage monitoring value in [% x 100].
- `u32_t taskStackOverflowThreshold`
Task stack overflow threshold address.

6.29.1 Detailed Description

Task descriptor structure.

Definition at line 517 of file gos_kernel.h.

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

- GOS2022/OS/kernel/inc/gos_kernel.h

6.30 gos_time_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_time.h>
```

Data Fields

- `gos_millisecond_t milliseconds`
Milliseconds.
- `gos_second_t seconds`
Seconds.
- `gos_minute_t minutes`
Minutes.
- `gos_hour_t hours`
Hours.
- `gos_day_t days`
Days.
- `gos_month_t months`
Months.
- `gos_year_t years`
Years.

6.30.1 Detailed Description

Time type.

Definition at line 76 of file gos_time.h.

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

- GOS2022/OS/services/inc/gos_time.h

6.31 gos_trigger_t Struct Reference

```
#include <GOS2022/OS/services/inc/gos_trigger.h>
```

Data Fields

- `u32_t valueCounter`
Value counter.
- `u32_t desiredValue`
Desired value.
- `gos_tid_t waiterTaskId`
Owner task ID.

6.31.1 Detailed Description

Trigger descriptor type.

Definition at line 85 of file gos_trigger.h.

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

- GOS2022/OS/services/inc/gos_trigger.h

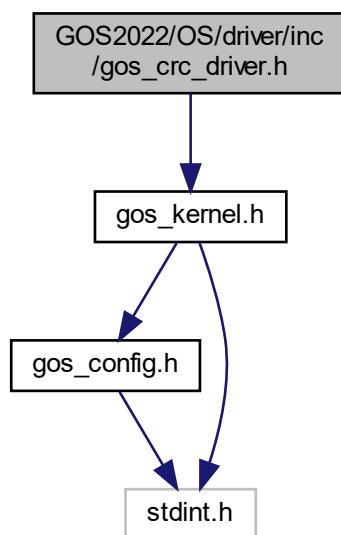
Chapter 7

File Documentation

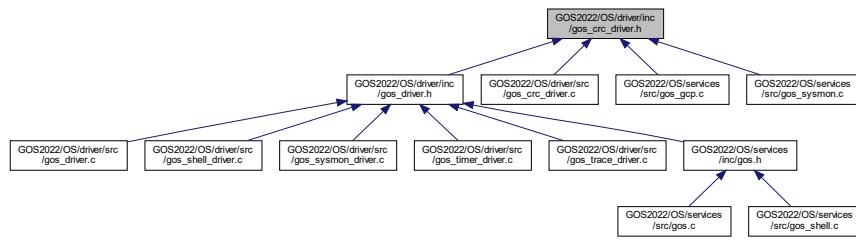
7.1 GOS2022/OS/driver/inc/gos_crc_driver.h File Reference

GOS Cyclic Redundancy Check driver header.

```
#include "gos_kernel.h"  
Include dependency graph for gos_crc_driver.h:
```



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



Functions

- `u32_t gos_crcDriverGetCrc (u8_t *pData, u32_t dataSize)`
Calculates the 32-bit CRC value of the given data buffer.

7.1.1 Detailed Description

GOS Cyclic Redundancy Check driver header.

Author

Ahmed Gazar

Date

2022-12-10

Version

1.0

This driver provides a simple 32-bit CRC calculator algorithm.

7.1.2 Function Documentation

7.1.2.1 gos_crcDriverGetCrc()

```
u32_t gos_crcDriverGetCrc (
    u8_t * pData,
    u32_t dataSize )
```

Calculates the 32-bit CRC value of the given data buffer.

Calculates the 32-bit CRC value of the given data buffer.

Parameters

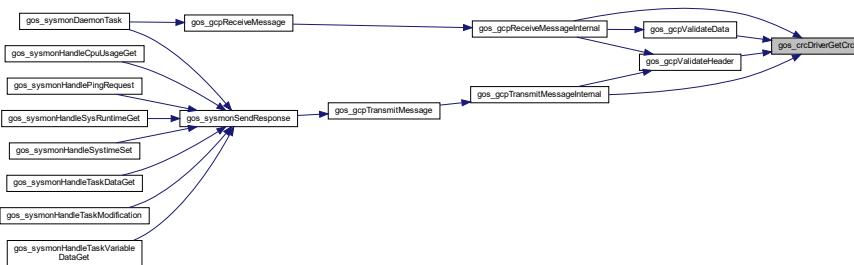
in	<i>pData</i>	Pointer to the data buffer.
in	<i>dataSize</i>	Size of the data buffer in bytes.

Returns

32-bit CRC value.

Definition at line 70 of file gos_crc_driver.c.

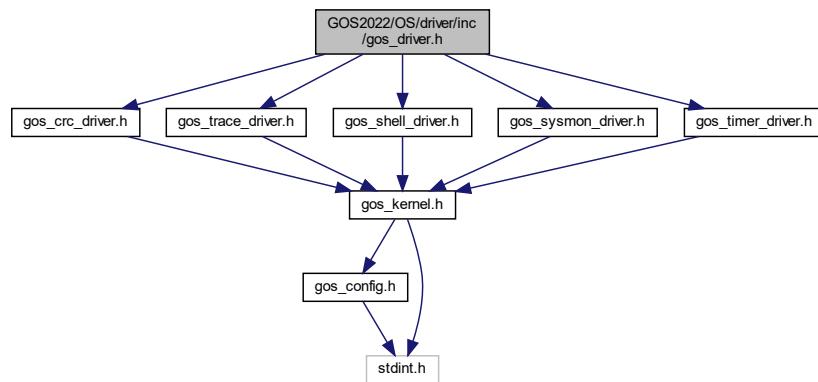
Here is the caller graph for this function:



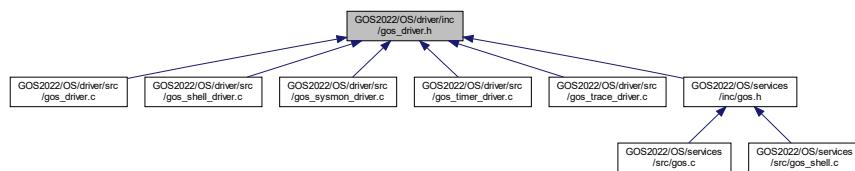
7.2 GOS2022/OS/driver/inc/gos_driver.h File Reference

GOS driver header.

```
#include <gos_crc_driver.h>
#include <gos_trace_driver.h>
#include <gos_shell_driver.h>
#include <gos_sysmon_driver.h>
#include <gos_timer_driver.h>
Include dependency graph for gos_driver.h:
```



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



Data Structures

- struct [gos_driver_functions_t](#)

Functions

- [`gos_result_t gos_driverInit \(gos_driver_functions_t *pDriverFunctions\)`](#)
Initializes the kernel drivers.

7.2.1 Detailed Description

GOS driver header.

Author

Ahmed Gazar

Date

2023-07-25

Version

1.2

This header is used for the inclusion of all driver skeletons.

7.2.2 Function Documentation

7.2.2.1 `gos_driverInit()`

```

gos_result_t gos_driverInit (
    gos_driver_functions_t * pDriverFunctions )
  
```

Initializes the kernel drivers.

Copies the function pointers into the internal structure.

Parameters

in	<i>pDriverFunctions</i>	Pointer to the function pointer structure.
----	-------------------------	--

Returns

Result of initialization.

Return values

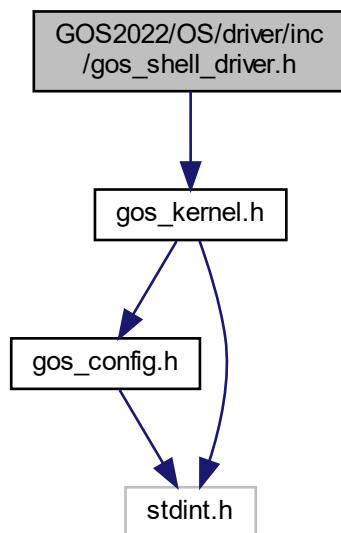
<i>GOS_SUCCESS</i>	Initialization successful.
<i>GOS_ERROR</i>	NULL pointer parameter.

Definition at line 63 of file gos_driver.c.

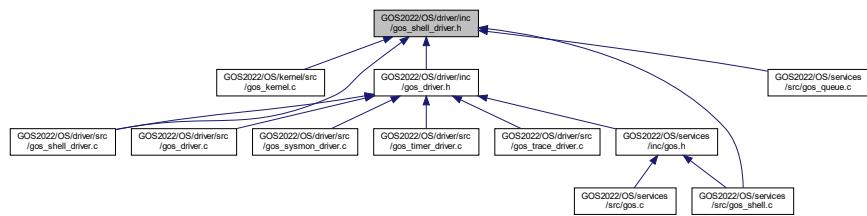
7.3 GOS2022/OS/driver/inc/gos_shell_driver.h File Reference

GOS SHELL driver header.

```
#include "gos_kernel.h"  
Include dependency graph for gos_shell_driver.h:
```



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



Typedefs

- `typedef gos_result_t(* gos_shellDriverReceiveChar_t) (char_t *)`
- `typedef gos_result_t(* gos_shellDriverTransmitString_t) (char_t *)`

Functions

- `gos_result_t gos_shellDriverReceiveChar (char_t *pChar)`
Receives a character.
- `gos_result_t gos_shellDriverTransmitString (char_t *pString,...)`
Transmits a string.

7.3.1 Detailed Description

GOS SHELL driver header.

Author

Ahmed Gazar

Date

2023-06-17

Version

1.1

This driver provides a skeleton for the driver for the shell service.

7.3.2 Typedef Documentation

7.3.2.1 gos_shellDriverReceiveChar_t

```
typedef gos_result_t(* gos_shellDriverReceiveChar_t) (char_t *)
```

Shell driver receive character function type.

Definition at line 63 of file gos_shell_driver.h.

7.3.2.2 gos_shellDriverTransmitString_t

```
typedef gos_result_t(* gos_shellDriverTransmitString_t) (char_t *)
```

Shell driver transmit string function type.

Definition at line 68 of file gos_shell_driver.h.

7.3.3 Function Documentation

7.3.3.1 gos_shellDriverReceiveChar()

```
gos_result_t gos_shellDriverReceiveChar (
    char_t * pChar )
```

Receives a character.

If registered, it calls the custom character receiver function.

Parameters

out	<i>pChar</i>	Pointer to a character variable to store the received character in.
-----	--------------	---

Returns

Result of character reception.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 75 of file gos_shell_driver.c.

Here is the caller graph for this function:



7.3.3.2 gos_shellDriverTransmitString()

```
gos_result_t gos_shellDriverTransmitString (
    char_t * pString,
    ...
)
```

Transmits a string.

If registered, it calls the custom string transmitter function.

Parameters

in	<i>pString</i>	Pointer to the string to be transmitted.
in	...	Variable parameters for formatted strings.

Returns

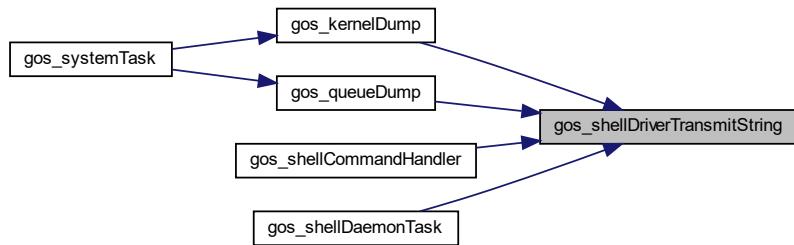
Result of string transmission.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 100 of file gos_shell_driver.c.

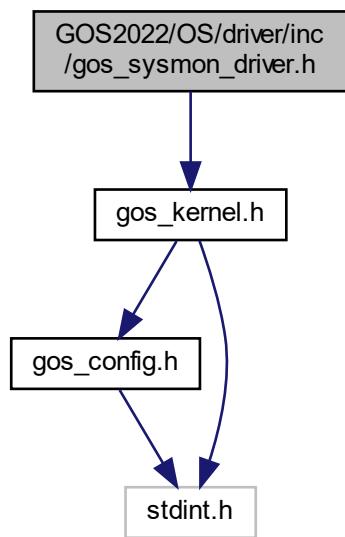
Here is the caller graph for this function:



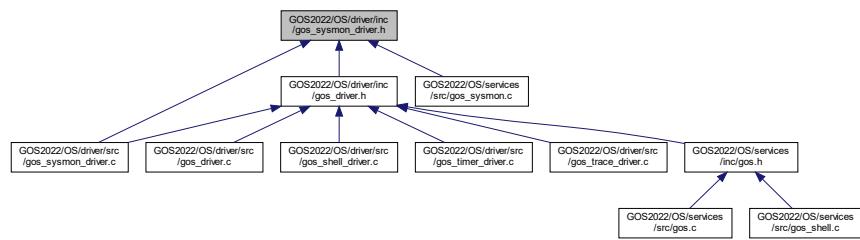
7.4 GOS2022/OS/driver/inc/gos_sysmon_driver.h File Reference

GOS SYSMON driver header.

```
#include "gos_kernel.h"  
Include dependency graph for gos_sysmon_driver.h:
```



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



Typedefs

- `typedef gos_result_t(* gos_sysmonDriverTransmit_t) (u8_t *, u16_t)`
- `typedef gos_result_t(* gos_sysmonDriverReceive_t) (u8_t *, u16_t)`

Functions

- `gos_result_t gos_sysmonDriverReceive (u8_t *pBuffer, u16_t bufferSize)`
It receives to the given buffer.
- `gos_result_t gos_sysmonDriverTransmit (u8_t *pBuffer, u16_t bufferSize)`
It transmits the given buffer.

7.4.1 Detailed Description

GOS SYSMON driver header.

Author

Ahmed Gazar

Date

2023-07-12

Version

1.0

This driver provides a skeleton for the driver for the sysmon service.

7.4.2 Typedef Documentation

7.4.2.1 gos_sysmonDriverReceive_t

```
typedef gos_result_t(* gos_sysmonDriverReceive_t) (u8_t *, u16_t)
```

Shell driver transmit string function type.

Definition at line 66 of file gos_sysmon_driver.h.

7.4.2.2 gos_sysmonDriverTransmit_t

```
typedef gos_result_t(* gos_sysmonDriverTransmit_t) (u8_t *, u16_t)
```

Shell driver receive character function type.

Definition at line 61 of file gos_sysmon_driver.h.

7.4.3 Function Documentation

7.4.3.1 gos_sysmonDriverReceive()

```
gos_result_t gos_sysmonDriverReceive (
    u8_t * pBuffer,
    u16_t bufferSize )
```

It receives to the given buffer.

If registered, it calls the receiver function.

Parameters

out	<i>pBuffer</i>	Target buffer to receive to.
in	<i>bufferSize</i>	Size of the target buffer.

Returns

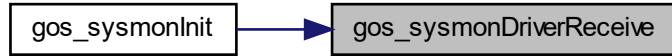
Result of reception.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 63 of file gos_sysmon_driver.c.

Here is the caller graph for this function:



7.4.3.2 gos_sysmonDriverTransmit()

```
gos_result_t gos_sysmonDriverTransmit (
    u8_t * pBuffer,
    u16_t bufferSize )
```

It transmits the given buffer.

If registered, it calls the transmitter function.

Parameters

in	<i>pBuffer</i>	Buffer to transmit.
in	<i>bufferSize</i>	Size of the buffer.

Returns

Result of string transmission.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 88 of file gos_sysmon_driver.c.

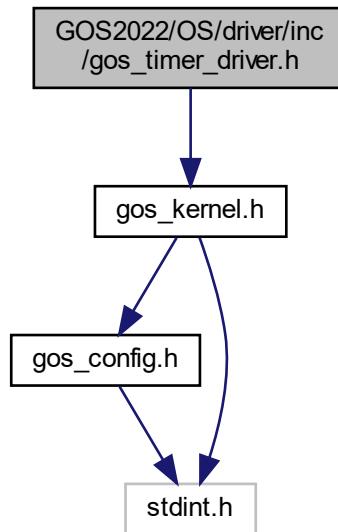
Here is the caller graph for this function:



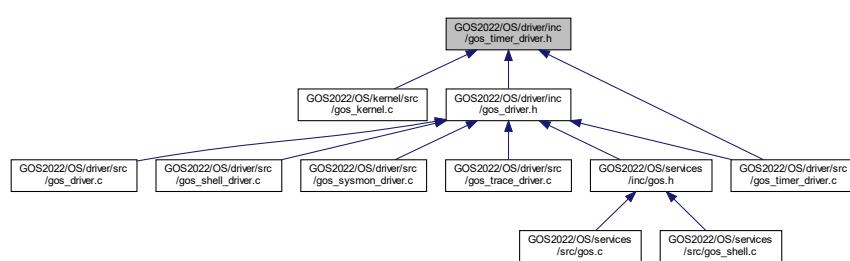
7.5 GOS2022/OS/driver/inc/gos_timer_driver.h File Reference

GOS timer driver header.

```
#include "gos_kernel.h"
Include dependency graph for gos_timer_driver.h:
```



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



Typedefs

- `typedef gos_result_t(* gos_timerDriverSysTimerGetVal_t) (u16_t *)`

Functions

- `gos_result_t gos_timerDriverSysTimerGet (u16_t *pValue)`
System timer value getter skeleton.

7.5.1 Detailed Description

GOS timer driver header.

Author

Ahmed Gazar

Date

2022-12-09

Version

1.0

This is the timer driver skeleton. It contains the required interface functions for the OS. These functions shall be implemented by the user in order to be able to use microsecond delay and task run-time monitoring.

7.5.2 Typedef Documentation

7.5.2.1 gos_timerDriverSysTimerGetVal_t

```
typedef gos_result_t(* gos_timerDriverSysTimerGetVal_t) (u16_t *)
```

System timer value get function type.

Definition at line 63 of file gos_timer_driver.h.

7.5.3 Function Documentation

7.5.3.1 gos_timerDriverSysTimerGet()

```
gos_result_t gos_timerDriverSysTimerGet (
    u16_t * pValue )
```

System timer value getter skeleton.

If registered, it calls the custom system timer getter function.

Parameters

out	pValue	Pointer to the variable to store the timer value in.
-----	--------	--

Returns

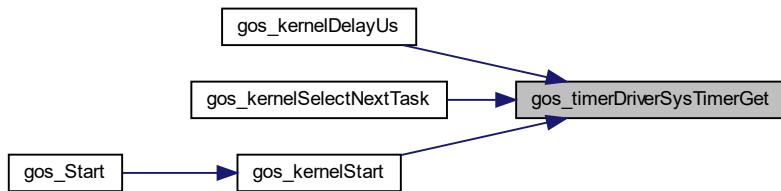
Result of system timer value getting.

Return values

GOS_SUCCESS	According to user implementation.
GOS_ERROR	According to user implementation / function not registered.

Definition at line 64 of file gos_timer_driver.c.

Here is the caller graph for this function:

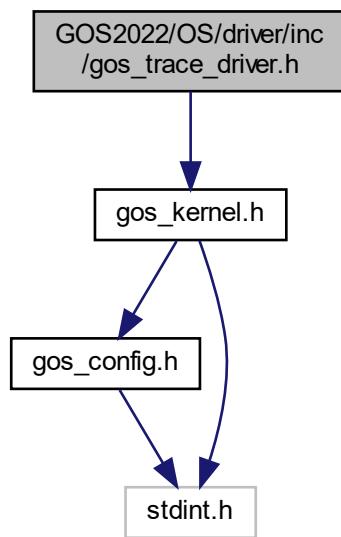


7.6 GOS2022/OS/driver/inc/gos_trace_driver.h File Reference

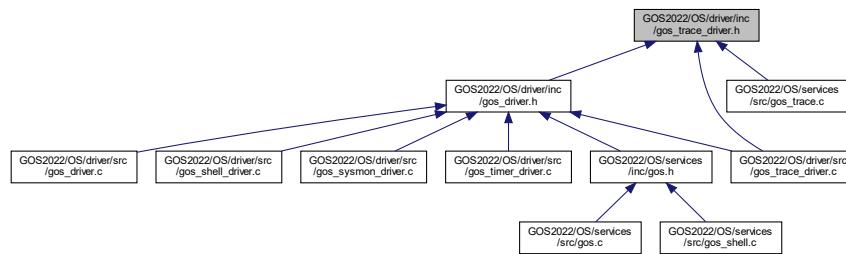
GOS trace driver header.

```
#include "gos_kernel.h"
```

Include dependency graph for gos_trace_driver.h:



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



Typedefs

- `typedef gos_result_t(* gos_traceDriverTransmitString_t) (char_t *)`
- `typedef gos_result_t(* gos_traceDriverTransmitString_Unsafe_t) (char_t *)`

Functions

- `gos_result_t gos_traceDriverTransmitString (char_t *pString)`
Trace driver transmit string function skeleton.
- `gos_result_t gos_traceDriverTransmitString_Unsafe (char_t *pString)`
Trace driver unsafe transmit string function skeleton.

7.6.1 Detailed Description

GOS trace driver header.

Author

Ahmed Gazar

Date

2023-01-13

Version

1.2

This driver provides a skeleton for the driver for the trace service.

7.6.2 Typedef Documentation

7.6.2.1 gos_traceDriverTransmitString_t

```
typedef gos_result_t(* gos_traceDriverTransmitString_t) (char_t *)
```

Trace driver transmit string function type.

Definition at line 63 of file gos_trace_driver.h.

7.6.2.2 gos_traceDriverTransmitString_Unsafe_t

```
typedef gos_result_t(* gos_traceDriverTransmitString_Unsafe_t) (char_t *)
```

Trace driver unsafe transmit string function type.

Definition at line 68 of file gos_trace_driver.h.

7.6.3 Function Documentation

7.6.3.1 gos_traceDriverTransmitString()

```
gos_result_t gos_traceDriverTransmitString (
    char_t * pString )
```

Trace driver transmit string function skeleton.

If registered, it calls the custom transmit function.

Parameters

in	pString	String to trace.
----	---------	------------------

Returns

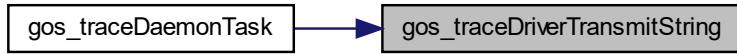
Result of string transmission.

Return values

GOS_SUCCESS	According to user implementation.
GOS_ERROR	According to user implementation / function not registered.

Definition at line 65 of file gos_trace_driver.c.

Here is the caller graph for this function:



7.6.3.2 gos_traceDriverTransmitString_Unsafe()

```
gos_result_t gos_traceDriverTransmitString_Unsafe (
    char_t * pString )
```

Trace driver unsafe transmit string function skeleton.

If registered, it calls the custom unsafe transmit function.

Parameters

in	<i>pString</i>	String to trace.
----	----------------	------------------

Returns

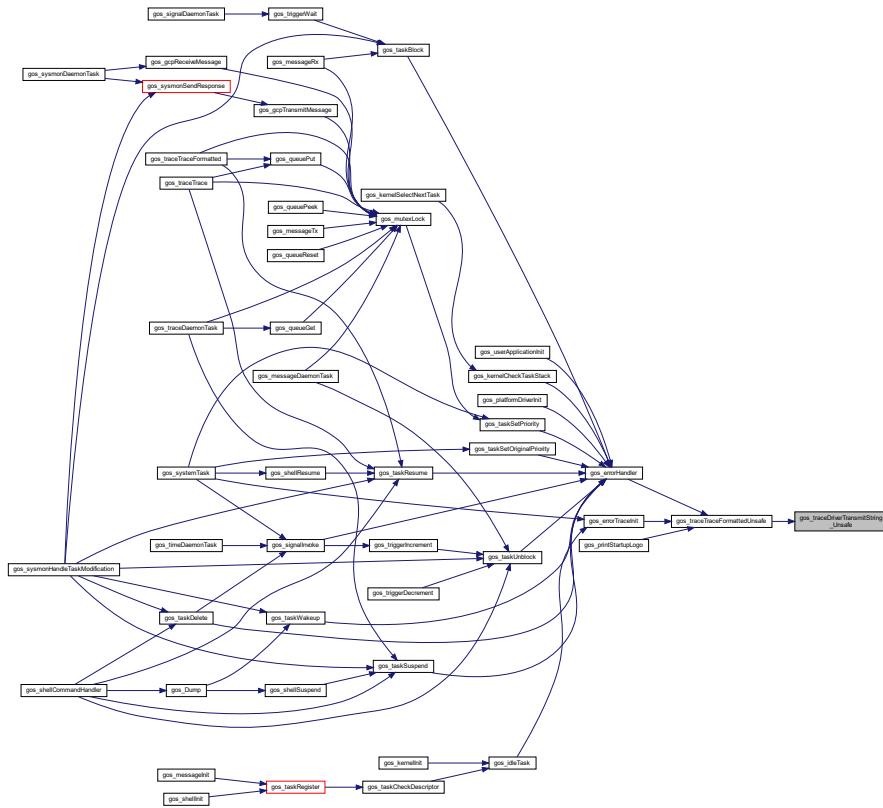
Result of string transmission.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 90 of file gos_trace_driver.c.

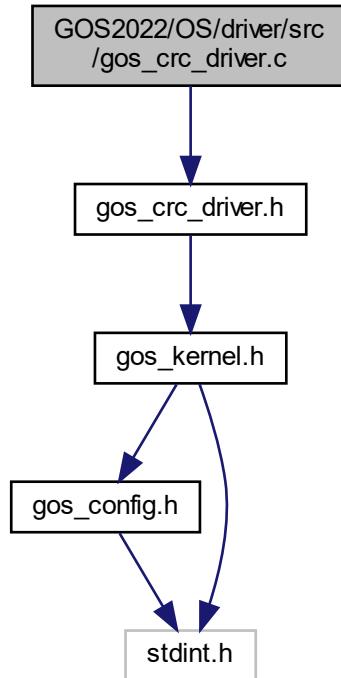
Here is the caller graph for this function:



7.7 GOS2022/OS/driver/src/gos_crc_driver.c File Reference

GOS Cyclic Redundancy Check driver source.

```
#include "gos_crc_driver.h"
Include dependency graph for gos_crc_driver.c:
```



Macros

- #define **CRC_INITIAL_VALUE** (0xFFFFFFFF)
- #define **CRC_POLYNOMIAL_VALUE** (0xEDB88320)

Functions

- **u32_t gos_crcDriverGetCrc (u8_t *pData, u32_t dataSize)**
Calculates the 32-bit CRC value of the given data buffer.

7.7.1 Detailed Description

GOS Cyclic Redundancy Check driver source.

Author

Ahmed Gazar

Date

2022-12-10

Version

1.0

For a more detailed description of this driver, please refer to [gos_crc_driver.h](#)

7.7.2 Macro Definition Documentation

7.7.2.1 CRC_INITIAL_VALUE

```
#define CRC_INITIAL_VALUE ( 0xFFFFFFFF )
```

CRC initializer value.

Definition at line 60 of file gos_crc_driver.c.

7.7.2.2 CRC_POLYNOMIAL_VALUE

```
#define CRC_POLYNOMIAL_VALUE ( 0xEDB88320 )
```

CRC polynomial value.

Definition at line 65 of file gos_crc_driver.c.

7.7.3 Function Documentation

7.7.3.1 gos_crcDriverGetCrc()

```
u32_t gos_crcDriverGetCrc (
    u8_t * pData,
    u32_t dataSize )
```

Calculates the 32-bit CRC value of the given data buffer.

Calculates the 32-bit CRC value of the given data buffer.

Parameters

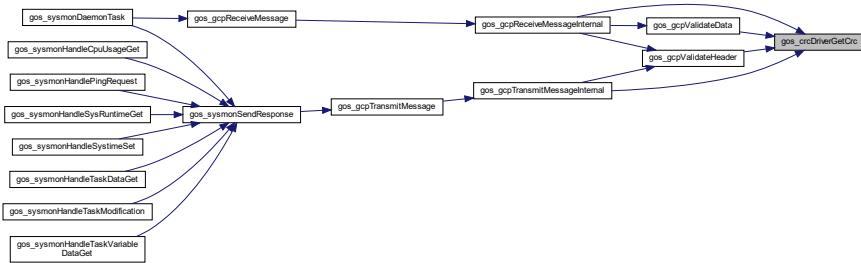
in	<i>pData</i>	Pointer to the data buffer.
in	<i>dataSize</i>	Size of the data buffer in bytes.

Returns

32-bit CRC value.

Definition at line 70 of file gos_crc_driver.c.

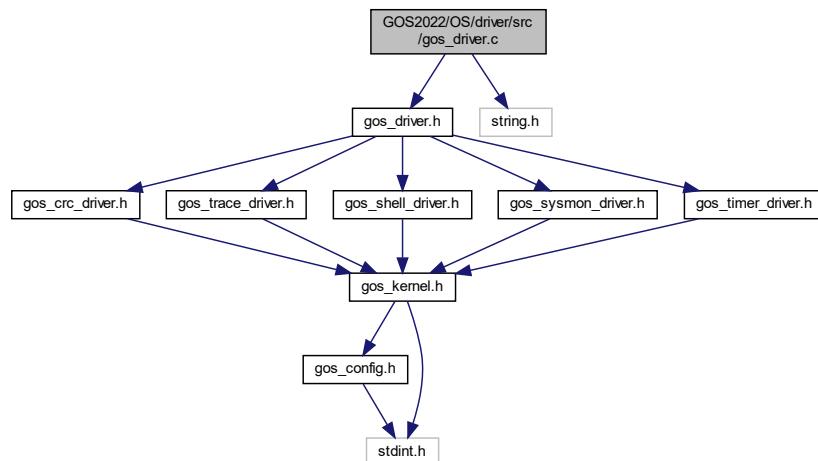
Here is the caller graph for this function:



7.8 GOS2022/OS/driver/src/gos_driver.c File Reference

GOS driver source.

```
#include "gos_driver.h"
#include <string.h>
Include dependency graph for gos_driver.c:
```



Functions

- [gos_result_t gos_driverInit \(gos_driver_functions_t *pDriverFunctions\)](#)

Initializes the kernel drivers.

Variables

- [gos_driver_functions_t driverFunctions = { NULL, NULL, NULL, NULL, NULL, NULL, NULL }](#)

7.8.1 Detailed Description

GOS driver source.

Author

Ahmed Gazar

Date

2022-12-11

Version

1.0

For a more detailed description of this driver, please refer to [gos_driver.h](#)

7.8.2 Function Documentation

7.8.2.1 gos_driverInit()

```
gos_result_t gos_driverInit (
    gos_driver_functions_t * pDriverFunctions )
```

Initializes the kernel drivers.

Copies the function pointers into the internal structure.

Parameters

in	<i>pDriverFunctions</i>	Pointer to the function pointer structure.
----	-------------------------	--

Returns

Result of initialization.

Return values

GOS_SUCCESS	Initialization successful.
GOS_ERROR	NULL pointer parameter.

Definition at line 63 of file gos_driver.c.

7.9 GOS2022/OS/driver/src/gos_shell_driver.c File Reference

GOS SHELL driver source.

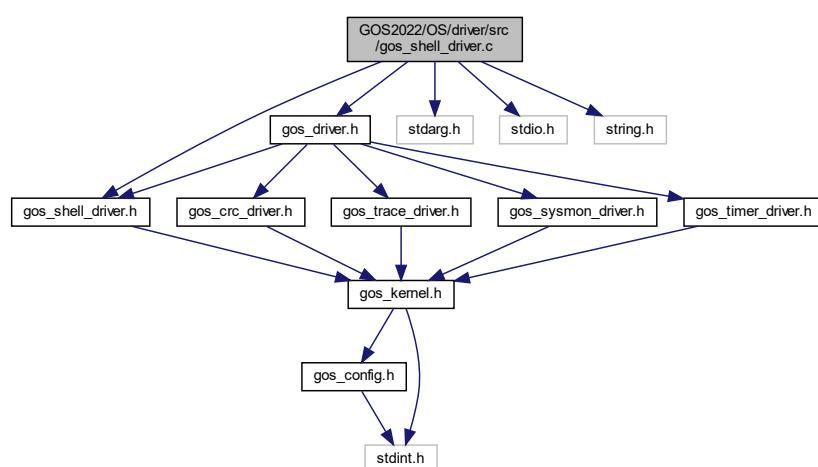
```
#include "gos_shell_driver.h"
#include "gos_driver.h"
```

```
#include <stdarg.h>
```

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

```
#include <string.h>
```

Include dependency graph for gos_shell_driver.c:



Functions

- [gos_result_t gos_shellDriverReceiveChar \(char_t *pChar\)](#)
Receives a character.
- [gos_result_t gos_shellDriverTransmitString \(char_t *pString,...\)](#)
Transmits a string.

Variables

- [GOS_STATIC char_t formattedBuffer \[CFG_SHELL_COMMAND_BUFFER_SIZE\]](#)
- [GOS_EXTERN gos_driver_functions_t driverFunctions](#)

7.9.1 Detailed Description

GOS SHELL driver source.

Author

Ahmed Gazar

Date

2023-06-17

Version

1.1

For a more detailed description of this driver, please refer to [gos_shell_driver.h](#)

7.9.2 Function Documentation

7.9.2.1 gos_shellDriverReceiveChar()

```
gos_result_t gos_shellDriverReceiveChar (
    char_t * pChar )
```

Receives a character.

If registered, it calls the custom character receiver function.

Parameters

out	<i>pChar</i>	Pointer to a character variable to store the received character in.
-----	--------------	---

Returns

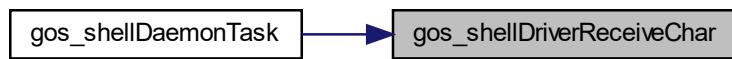
Result of character reception.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 75 of file gos_shell_driver.c.

Here is the caller graph for this function:



7.9.2.2 gos_shellDriverTransmitString()

```
gos_result_t gos_shellDriverTransmitString (
    char_t * pString,
    ... )
```

Transmits a string.

If registered, it calls the custom string transmitter function.

Parameters

in	<i>pString</i>	Pointer to the string to be transmitted.
in	...	Variable parameters for formatted strings.

Returns

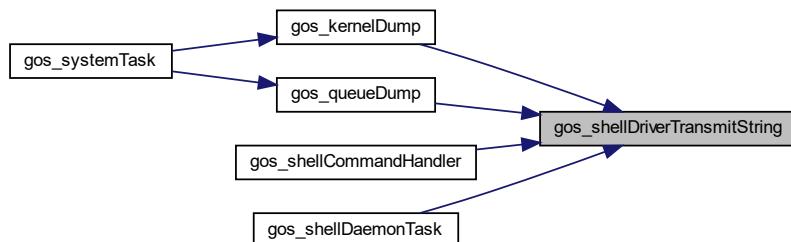
Result of string transmission.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 100 of file gos_shell_driver.c.

Here is the caller graph for this function:



7.9.3 Variable Documentation

7.9.3.1 **formattedBuffer**

```
GOS_STATIC char_t formattedBuffer[CFG_SHELL_COMMAND_BUFFER_SIZE]
```

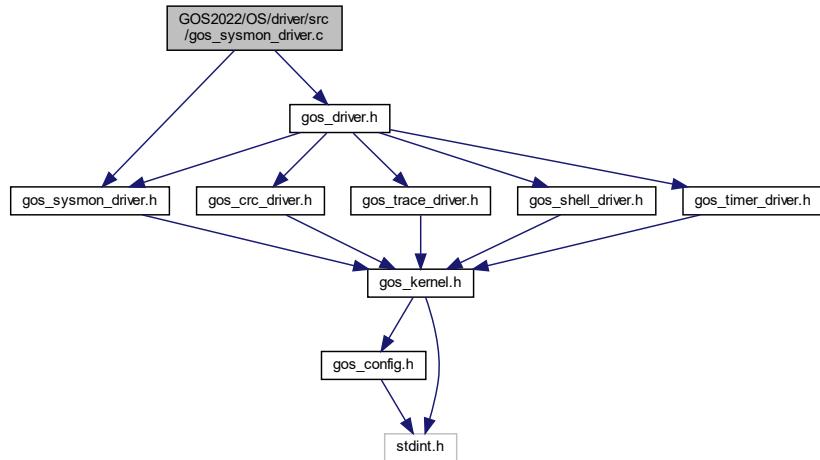
Buffer for formatted strings.

Definition at line 65 of file gos_shell_driver.c.

7.10 GOS2022/OS/driver/src/gos_sysmon_driver.c File Reference

GOS SYSMON driver source.

```
#include "gos_sysmon_driver.h"
#include "gos_driver.h"
Include dependency graph for gos_sysmon_driver.c:
```



Functions

- `gos_result_t gos_sysmonDriverReceive (u8_t *pBuffer, u16_t bufferSize)`
It receives to the given buffer.
- `gos_result_t gos_sysmonDriverTransmit (u8_t *pBuffer, u16_t bufferSize)`
It transmits the given buffer.

Variables

- `GOS_EXTERN gos_driver_functions_t driverFunctions`

7.10.1 Detailed Description

GOS SYSMON driver source.

Author

Ahmed Gazar

Date

2023-07-12

Version

1.0

For a more detailed description of this driver, please refer to [gos_sysmon_driver.h](#)

7.10.2 Function Documentation

7.10.2.1 gos_sysmonDriverReceive()

```
gos_result_t gos_sysmonDriverReceive (
    u8_t * pBuffer,
    u16_t bufferSize )
```

It receives to the given buffer.

If registered, it calls the receiver function.

Parameters

out	<i>pBuffer</i>	Target buffer to receive to.
in	<i>bufferSize</i>	Size of the target buffer.

Returns

Result of reception.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 63 of file gos_sysmon_driver.c.

Here is the caller graph for this function:



7.10.2.2 gos_sysmonDriverTransmit()

```
gos_result_t gos_sysmonDriverTransmit (
    u8_t * pBuffer,
    u16_t bufferSize )
```

It transmits the given buffer.

If registered, it calls the transmitter function.

Parameters

in	<i>pBuffer</i>	Buffer to transmit.
in	<i>bufferSize</i>	Size of the buffer.

Returns

Result of string transmission.

Return values

<i>GOS_SUCCESS</i>	According to user implementation.
<i>GOS_ERROR</i>	According to user implementation / function not registered.

Definition at line 88 of file gos_sysmon_driver.c.

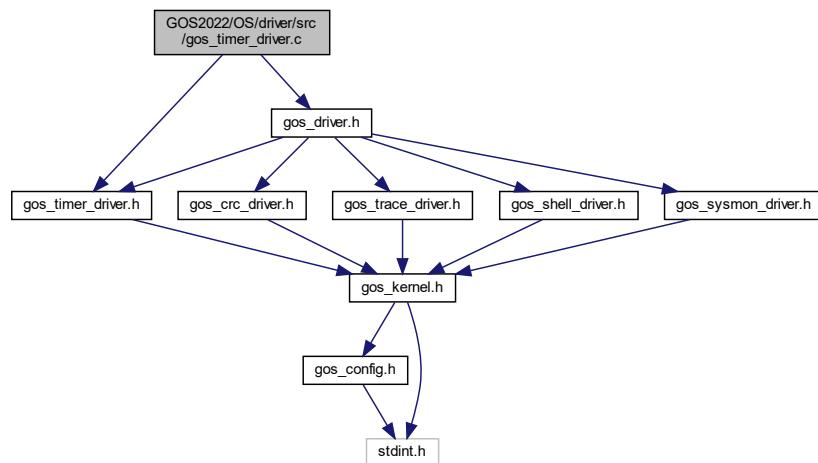
Here is the caller graph for this function:



7.11 GOS2022/OS/driver/src/gos_timer_driver.c File Reference

GOS timer driver source.

```
#include "gos_timer_driver.h"
#include "gos_driver.h"
Include dependency graph for gos_timer_driver.c:
```



Functions

- `gos_result_t gos_timerDriverSysTimerGet (u16_t *pValue)`
System timer value getter skeleton.

Variables

- `GOS_EXTERN gos_driver_functions_t driverFunctions`

7.11.1 Detailed Description

GOS timer driver source.

Author

Ahmed Gazar

Date

2022-11-15

Version

1.1

For a more detailed description of this driver, please refer to [gos_timer_driver.h](#)

7.11.2 Function Documentation

7.11.2.1 gos_timerDriverSysTimerGet()

```
gos_result_t gos_timerDriverSysTimerGet (
    u16_t * pValue )
```

System timer value getter skeleton.

If registered, it calls the custom system timer getter function.

Parameters

out	<code>pValue</code>	Pointer to the variable to store the timer value in.
-----	---------------------	--

Returns

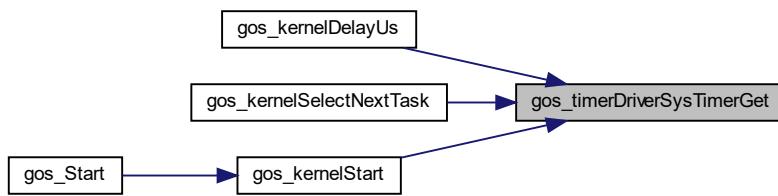
Result of system timer value getting.

Return values

GOS_SUCCESS	According to user implementation.
GOS_ERROR	According to user implementation / function not registered.

Definition at line 64 of file gos_timer_driver.c.

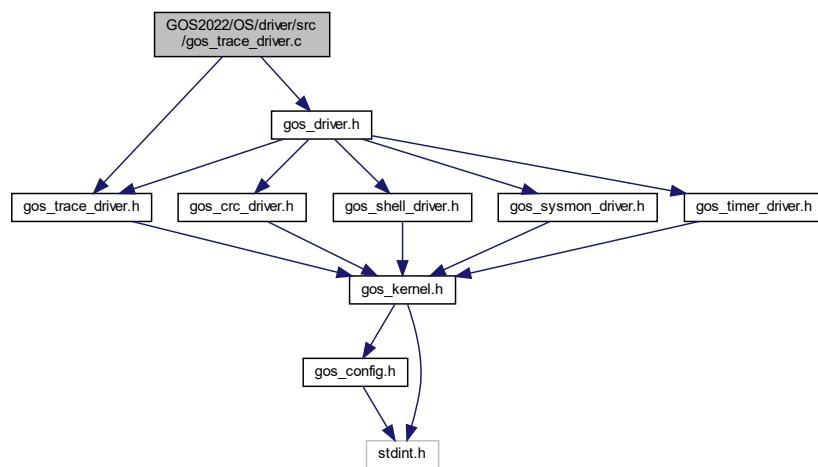
Here is the caller graph for this function:



7.12 GOS2022/OS/driver/src/gos_trace_driver.c File Reference

GOS trace driver source.

```
#include <gos_trace_driver.h>
#include <gos_driver.h>
Include dependency graph for gos_trace_driver.c:
```



Functions

- `gos_result_t gos_traceDriverTransmitString (char_t *pString)`
Trace driver transmit string function skeleton.
- `gos_result_t gos_traceDriverTransmitString_Unsafe (char_t *pString)`
Trace driver unsafe transmit string function skeleton.

Variables

- `GOS_EXTERN gos_driver_functions_t driverFunctions`

7.12.1 Detailed Description

GOS trace driver source.

Author

Ahmed Gazar

Date

2023-01-13

Version

1.2

For a more detailed description of this driver, please refer to [gos_trace_driver.h](#)

7.12.2 Function Documentation

7.12.2.1 gos_traceDriverTransmitString()

```
gos_result_t gos_traceDriverTransmitString (
    char_t * pString )
```

Trace driver transmit string function skeleton.

If registered, it calls the custom transmit function.

Parameters

in	<code>pString</code>	String to trace.
----	----------------------	------------------

Returns

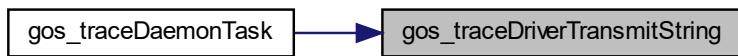
Result of string transmission.

Return values

GOS_SUCCESS	According to user implementation.
GOS_ERROR	According to user implementation / function not registered.

Definition at line 65 of file gos_trace_driver.c.

Here is the caller graph for this function:

**7.12.2.2 gos_traceDriverTransmitString_Unsafe()**

```
gos_result_t gos_traceDriverTransmitString_Unsafe (
    char_t * pString )
```

Trace driver unsafe transmit string function skeleton.

If registered, it calls the custom unsafe transmit function.

Parameters

in	<i>pString</i>	String to trace.
----	----------------	------------------

Returns

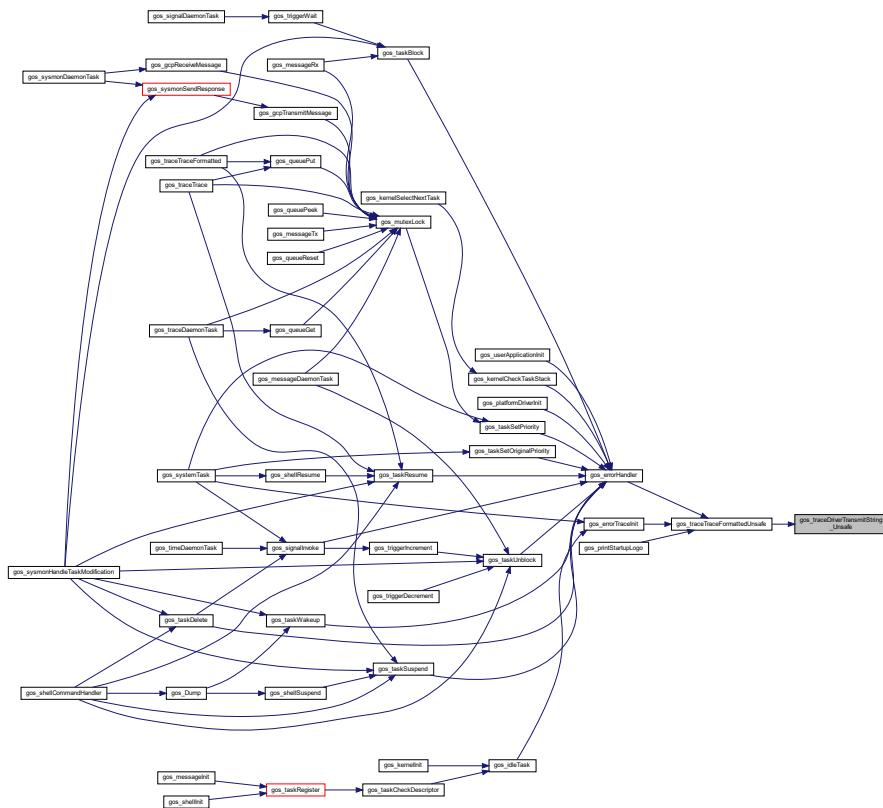
Result of string transmission.

Return values

GOS_SUCCESS	According to user implementation.
GOS_ERROR	According to user implementation / function not registered.

Definition at line 90 of file gos_trace_driver.c.

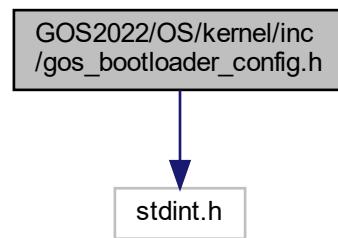
Here is the caller graph for this function:



7.13 GOS2022/OS/kernel/inc/gos_bootloader_config.h File Reference

GOS bootloader configuration header.

```
#include <stdint.h>
Include dependency graph for gos_bootloader_config.h:
```



Macros

- #define ARM_CORTEX_M4 (1)
- #define CFG_TARGET_CPU (ARM_CORTEX_M4)
- #define CFG_SCHED_COOPERATIVE (0)
- #define CFG_USE_PRIO_INHERITANCE (1)
- #define CFG_TASK_MAX_NAME_LENGTH (32)
- #define CFG_TASK_MAX_NUMBER (16)
- #define CFG_TASK_MIN_STACK_SIZE (0x200)
- #define CFG_TASK_MAX_STACK_SIZE (0x4000)
- #define CFG_IDLE_TASK_STACK_SIZE (0x400)
- #define CFG_SYSTEM_TASK_STACK_SIZE (0x800)
- #define CFG_TASK_SIGNAL_DAEMON_STACK (0x400)
- #define CFG_TASK_TIME_DAEMON_STACK (0x300)
- #define CFG_TASK_MESSAGE_DAEMON_STACK (0x300)
- #define CFG_TASK_SHELL_DAEMON_STACK (0x300)
- #define CFG_TASK_TRACE_DAEMON_STACK (0x400)
- #define CFG_TASK_SYSMON_DAEMON_STACK (0x800)
- #define CFG_TASK_TRACE_DAEMON_PRIO (193)
- #define CFG_TASK_MESSAGE_DAEMON_PRIO (198)
- #define CFG_TASK_SIGNAL_DAEMON_PRIO (197)
- #define CFG_TASK_SHELL_DAEMON_PRIO (192)
- #define CFG_TASK_TIME_DAEMON_PRIO (196)
- #define CFG_TASK_SYS_PRIO (195)
- #define CFG_TASK_SYSMON_DAEMON_PRIO (191)
- #define CFG_QUEUE_MAX_NUMBER (1)
- #define CFG_QUEUE_MAX_ELEMENTS (30)
- #define CFG_QUEUE_MAX_LENGTH (200)
- #define CFG_QUEUE_USE_NAME (1)
- #define CFG_QUEUE_MAX_NAME_LENGTH (24)
- #define CFG_SIGNAL_MAX_NUMBER (3)
- #define CFG_SIGNAL_MAX_SUBSCRIBERS (6)
- #define CFG_MESSAGE_MAX_NUMBER (16)
- #define CFG_MESSAGE_MAX_LENGTH (80)
- #define CFG_MESSAGE_MAX_WAITERS (16)
- #define CFG_MESSAGE_MAX_WAITER_IDS (8)
- #define CFG_SHELL_USE_SERVICE (0)
- #define CFG_SHELL_MAX_COMMAND_NUMBER (8)
- #define CFG_SHELL_MAX_COMMAND_LENGTH (20)
- #define CFG_SHELL_MAX_PARAMS_LENGTH (64)
- #define CFG_SHELL_COMMAND_BUFFER_SIZE (200)
- #define CFG_SHELL_STARTUP_DELAY_MS (500)
- #define CFG_GCP_CHANNELS_MAX_NUMBER (1)
- #define CFG_TRACE_MAX_LENGTH (200)
- #define CFG_SYSMON_USE_SERVICE (1)
- #define CFG_SYSMON_GCP_CHANNEL_NUM (0)
- #define CFG_SYSMON_MAX_USER_MESSAGES (24)
- #define CFG_RESET_ON_ERROR (1)
- #define CFG_RESET_ON_ERROR_DELAY_MS (2000)

7.13.1 Detailed Description

GOS bootloader configuration header.

Author

Ahmed Gazar

Date

2023-09-26

Version

1.0

This header contains the kernel and service configurations of the operating system.

7.13.2 Macro Definition Documentation

7.13.2.1 ARM_CORTEX_M4

```
#define ARM_CORTEX_M4 ( 1 )
```

ARM Cortex-M4.

Definition at line 45 of file gos_bootloader_config.h.

7.13.2.2 CFG_GCP_CHANNELS_MAX_NUMBER

```
#define CFG_GCP_CHANNELS_MAX_NUMBER ( 1 )
```

GCP maximum number of channels.

Definition at line 243 of file gos_bootloader_config.h.

7.13.2.3 CFG_IDLE_TASK_STACK_SIZE

```
#define CFG_IDLE_TASK_STACK_SIZE ( 0x400 )
```

Idle task stack size.

Definition at line 90 of file gos_bootloader_config.h.

7.13.2.4 CFG_MESSAGE_MAX_LENGTH

```
#define CFG_MESSAGE_MAX_LENGTH ( 80 )
```

Maximum length of a message in bytes.

Definition at line 199 of file gos_bootloader_config.h.

7.13.2.5 CFG_MESSAGE_MAX_NUMBER

```
#define CFG_MESSAGE_MAX_NUMBER ( 16 )
```

Maximum number of messages handled at once.

Definition at line 195 of file gos_bootloader_config.h.

7.13.2.6 CFG_MESSAGE_MAX_WAITER_IDS

```
#define CFG_MESSAGE_MAX_WAITER_IDS ( 8 )
```

Maximum number of message IDs a task can wait for (includes the terminating 0).

Definition at line 207 of file gos_bootloader_config.h.

7.13.2.7 CFG_MESSAGE_MAX_WAITERS

```
#define CFG_MESSAGE_MAX_WAITERS ( 16 )
```

Maximum number of message waiters.

Definition at line 203 of file gos_bootloader_config.h.

7.13.2.8 CFG_QUEUE_MAX_ELEMENTS

```
#define CFG_QUEUE_MAX_ELEMENTS ( 30 )
```

Maximum number of queue elements.

Definition at line 162 of file gos_bootloader_config.h.

7.13.2.9 CFG_QUEUE_MAX_LENGTH

```
#define CFG_QUEUE_MAX_LENGTH ( 200 )
```

Maximum queue length.

Definition at line 166 of file gos_bootloader_config.h.

7.13.2.10 CFG_QUEUE_MAX_NAME_LENGTH

```
#define CFG_QUEUE_MAX_NAME_LENGTH ( 24 )
```

Maximum queue name length.

Definition at line 174 of file gos_bootloader_config.h.

7.13.2.11 CFG_QUEUE_MAX_NUMBER

```
#define CFG_QUEUE_MAX_NUMBER ( 1 )
```

Maximum number of queues.

Definition at line 158 of file gos_bootloader_config.h.

7.13.2.12 CFG_QUEUE_USE_NAME

```
#define CFG_QUEUE_USE_NAME ( 1 )
```

Queue use name flag.

Definition at line 170 of file gos_bootloader_config.h.

7.13.2.13 CFG_RESET_ON_ERROR

```
#define CFG_RESET_ON_ERROR ( 1 )
```

Flag to indicate if the system should reset on error.

Definition at line 277 of file gos_bootloader_config.h.

7.13.2.14 CFG_RESET_ON_ERROR_DELAY_MS

```
#define CFG_RESET_ON_ERROR_DELAY_MS ( 2000 )
```

Delay time before system reset.

Definition at line 281 of file gos_bootloader_config.h.

7.13.2.15 CFG_SCHED_COOPERATIVE

```
#define CFG_SCHED_COOPERATIVE ( 0 )
```

Cooperative scheduling flag.

Definition at line 58 of file gos_bootloader_config.h.

7.13.2.16 CFG_SHELL_COMMAND_BUFFER_SIZE

```
#define CFG_SHELL_COMMAND_BUFFER_SIZE ( 200 )
```

Command buffer size.

Definition at line 231 of file gos_bootloader_config.h.

7.13.2.17 CFG_SHELL_MAX_COMMAND_LENGTH

```
#define CFG_SHELL_MAX_COMMAND_LENGTH ( 20 )
```

Maximum command length.

Definition at line 223 of file gos_bootloader_config.h.

7.13.2.18 CFG_SHELL_MAX_COMMAND_NUMBER

```
#define CFG_SHELL_MAX_COMMAND_NUMBER ( 8 )
```

Maximum number of shell commands.

Definition at line 219 of file gos_bootloader_config.h.

7.13.2.19 CFG_SHELL_MAX_PARAMS_LENGTH

```
#define CFG_SHELL_MAX_PARAMS_LENGTH ( 64 )
```

Maximum parameters length.

Definition at line 227 of file gos_bootloader_config.h.

7.13.2.20 CFG_SHELL_STARTUP_DELAY_MS

```
#define CFG_SHELL_STARTUP_DELAY_MS ( 500 )
```

Shell daemon startup delay time [ms].

Definition at line 235 of file gos_bootloader_config.h.

7.13.2.21 CFG_SHELL_USE_SERVICE

```
#define CFG_SHELL_USE_SERVICE ( 0 )
```

Shell service use flag.

Definition at line 215 of file gos_bootloader_config.h.

7.13.2.22 CFG_SIGNAL_MAX_NUMBER

```
#define CFG_SIGNAL_MAX_NUMBER ( 3 )
```

Maximum number of signals.

Definition at line 183 of file gos_bootloader_config.h.

7.13.2.23 CFG_SIGNAL_MAX_SUBSCRIBERS

```
#define CFG_SIGNAL_MAX_SUBSCRIBERS ( 6 )
```

Maximum number of signal subscribers.

Definition at line 187 of file gos_bootloader_config.h.

7.13.2.24 CFG_SYSMON_GCP_CHANNEL_NUM

```
#define CFG_SYSMON_GCP_CHANNEL_NUM ( 0 )
```

Define sysmon GCP channel number.

Definition at line 264 of file gos_bootloader_config.h.

7.13.2.25 CFG_SYSMON_MAX_USER_MESSAGES

```
#define CFG_SYSMON_MAX_USER_MESSAGES ( 24 )
```

Maximum number of user messages.

Definition at line 269 of file gos_bootloader_config.h.

7.13.2.26 CFG_SYSMON_USE_SERVICE

```
#define CFG_SYSMON_USE_SERVICE ( 1 )
```

Sysmon use service flag.

Definition at line 259 of file gos_bootloader_config.h.

7.13.2.27 CFG_SYSTEM_TASK_STACK_SIZE

```
#define CFG_SYSTEM_TASK_STACK_SIZE ( 0x800 )
```

System task stack size.

Definition at line 94 of file gos_bootloader_config.h.

7.13.2.28 CFG_TARGET_CPU

```
#define CFG_TARGET_CPU ( ARM_CORTEX_M4 )
```

Target CPU.

Definition at line 50 of file gos_bootloader_config.h.

7.13.2.29 CFG_TASK_MAX_NAME_LENGTH

```
#define CFG_TASK_MAX_NAME_LENGTH ( 32 )
```

Maximum task name length.

Definition at line 70 of file gos_bootloader_config.h.

7.13.2.30 CFG_TASK_MAX_NUMBER

```
#define CFG_TASK_MAX_NUMBER ( 16 )
```

Maximum number of tasks.

Definition at line 74 of file gos_bootloader_config.h.

7.13.2.31 CFG_TASK_MAX_STACK_SIZE

```
#define CFG_TASK_MAX_STACK_SIZE ( 0x4000 )
```

Maximum task stack size.

Definition at line 86 of file gos_bootloader_config.h.

7.13.2.32 CFG_TASK_MESSAGE_DAEMON_PRIO

```
#define CFG_TASK_MESSAGE_DAEMON_PRIO ( 198 )
```

Message daemon task priority.

Definition at line 130 of file gos_bootloader_config.h.

7.13.2.33 CFG_TASK_MESSAGE_DAEMON_STACK

```
#define CFG_TASK_MESSAGE_DAEMON_STACK ( 0x300 )
```

Message daemon task stack size.

Definition at line 106 of file gos_bootloader_config.h.

7.13.2.34 CFG_TASK_MIN_STACK_SIZE

```
#define CFG_TASK_MIN_STACK_SIZE ( 0x200 )
```

Minimum task stack size.

Definition at line 82 of file gos_bootloader_config.h.

7.13.2.35 CFG_TASK_SHELL_DAEMON_PRIO

```
#define CFG_TASK_SHELL_DAEMON_PRIO ( 192 )
```

Shell daemon task priority.

Definition at line 138 of file gos_bootloader_config.h.

7.13.2.36 CFG_TASK_SHELL_DAEMON_STACK

```
#define CFG_TASK_SHELL_DAEMON_STACK ( 0x300 )
```

Shell daemon task stack size.

Definition at line 110 of file gos_bootloader_config.h.

7.13.2.37 CFG_TASK_SIGNAL_DAEMON_PRIO

```
#define CFG_TASK_SIGNAL_DAEMON_PRIO ( 197 )
```

Signal daemon task priority.

Definition at line 134 of file gos_bootloader_config.h.

7.13.2.38 CFG_TASK_SIGNAL_DAEMON_STACK

```
#define CFG_TASK_SIGNAL_DAEMON_STACK ( 0x400 )
```

Signal daemon task stack size.

Definition at line 98 of file gos_bootloader_config.h.

7.13.2.39 CFG_TASK_SYS_PRIO

```
#define CFG_TASK_SYS_PRIO ( 195 )
```

System task priority.

Definition at line 146 of file gos_bootloader_config.h.

7.13.2.40 CFG_TASK_SYSMON_DAEMON_PRIO

```
#define CFG_TASK_SYSMON_DAEMON_PRIO ( 191 )
```

Sysmon daemon task priority.

Definition at line 150 of file gos_bootloader_config.h.

7.13.2.41 CFG_TASK_SYSMON_DAEMON_STACK

```
#define CFG_TASK_SYSMON_DAEMON_STACK ( 0x800 )
```

Sysmon daemon task stack size.

Definition at line 118 of file gos_bootloader_config.h.

7.13.2.42 CFG_TASK_TIME_DAEMON_PRIO

```
#define CFG_TASK_TIME_DAEMON_PRIO ( 196 )
```

Time daemon task priority.

Definition at line 142 of file gos_bootloader_config.h.

7.13.2.43 CFG_TASK_TIME_DAEMON_STACK

```
#define CFG_TASK_TIME_DAEMON_STACK ( 0x300 )
```

Time daemon task stack size.

Definition at line 102 of file gos_bootloader_config.h.

7.13.2.44 CFG_TASK_TRACE_DAEMON_PRIO

```
#define CFG_TASK_TRACE_DAEMON_PRIO ( 193 )
```

Trace daemon task priority.

Definition at line 126 of file gos_bootloader_config.h.

7.13.2.45 CFG_TASK_TRACE_DAEMON_STACK

```
#define CFG_TASK_TRACE_DAEMON_STACK ( 0x400 )
```

Log daemon task stack size.

Definition at line 114 of file gos_bootloader_config.h.

7.13.2.46 CFG_TRACE_MAX_LENGTH

```
#define CFG_TRACE_MAX_LENGTH ( 200 )
```

Trace maximum (line) length.

Definition at line 251 of file gos_bootloader_config.h.

7.13.2.47 CFG_USE_PRIO_INHERITANCE

```
#define CFG_USE_PRIO_INHERITANCE ( 1 )
```

Priority inheritance flag for lock.

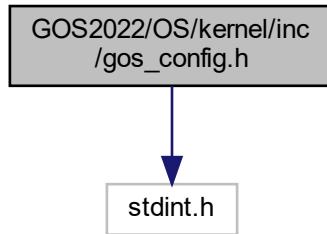
Definition at line 62 of file gos_bootloader_config.h.

7.14 GOS2022/OS/kernel/inc/gos_config.h File Reference

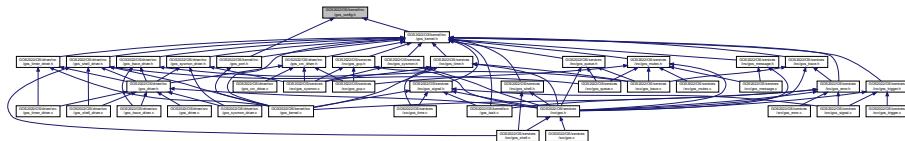
GOS configuration header.

```
#include <stdint.h>
```

Include dependency graph for gos_config.h:



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



Macros

- #define ARM_CORTEX_M4 (1)
- #define CFG_TARGET_CPU (ARM_CORTEX_M4)
- #define CFG_SCHED_COOPERATIVE (0)
- #define CFG_USE_PRIO_INHERITANCE (1)
- #define CFG_TASK_MAX_NAME_LENGTH (32)
- #define CFG_TASK_MAX_NUMBER (48)
- #define CFG_TASK_MIN_STACK_SIZE (0x200)
- #define CFG_TASK_MAX_STACK_SIZE (0x4000)
- #define CFG_IDLE_TASK_STACK_SIZE (0x400)
- #define CFG_SYSTEM_TASK_STACK_SIZE (0x800)
- #define CFG_TASK_SIGNAL_DAEMON_STACK (0x400)
- #define CFG_TASK_TIME_DAEMON_STACK (0x300)
- #define CFG_TASK_MESSAGE_DAEMON_STACK (0x600)
- #define CFG_TASK_SHELL_DAEMON_STACK (0x300)
- #define CFG_TASK_TRACE_DAEMON_STACK (0x400)
- #define CFG_TASK_SYSMON_DAEMON_STACK (0x800)
- #define CFG_TASK_TRACE_DAEMON_PRIO (193)
- #define CFG_TASK_MESSAGE_DAEMON_PRIO (198)
- #define CFG_TASK_SIGNAL_DAEMON_PRIO (197)
- #define CFG_TASK_SHELL_DAEMON_PRIO (192)

- #define CFG_TASK_TIME_DAEMON_PRIO (196)
- #define CFG_TASK_SYS_PRIO (195)
- #define CFG_TASK_SYSMON_DAEMON_PRIO (191)
- #define CFG_QUEUE_MAX_NUMBER (4)
- #define CFG_QUEUE_MAX_ELEMENTS (40)
- #define CFG_QUEUE_MAX_LENGTH (200)
- #define CFG_QUEUE_USE_NAME (1)
- #define CFG_QUEUE_MAX_NAME_LENGTH (24)
- #define CFG_SIGNAL_MAX_NUMBER (6)
- #define CFG_SIGNAL_MAX_SUBSCRIBERS (6)
- #define CFG_MESSAGE_MAX_NUMBER (8)
- #define CFG_MESSAGE_MAX_LENGTH (80)
- #define CFG_MESSAGE_MAX_WAITERS (10)
- #define CFG_MESSAGE_MAX_WAITER_IDS (8)
- #define CFG_MESSAGE_MAX_ADDRESSEES (8)
- #define CFG_SHELL_USE_SERVICE (1)
- #define CFG_SHELL_MAX_COMMAND_NUMBER (16)
- #define CFG_SHELL_MAX_COMMAND_LENGTH (20)
- #define CFG_SHELL_MAX_PARAMS_LENGTH (128)
- #define CFG_SHELL_COMMAND_BUFFER_SIZE (200)
- #define CFG_SHELL_STARTUP_DELAY_MS (500)
- #define CFG_GCP_CHANNELS_MAX_NUMBER (4)
- #define CFG_TRACE_MAX_LENGTH (200)
- #define CFG_SYSMON_USE_SERVICE (1)
- #define CFG_SYSMON_GCP_CHANNEL_NUM (0)
- #define CFG_SYSMON_MAX_USER_MESSAGES (24)
- #define CFG_RESET_ON_ERROR (1)
- #define CFG_RESET_ON_ERROR_DELAY_MS (3000)

7.14.1 Detailed Description

GOS configuration header.

Author

Ahmed Gazar

Date

2024-04-24

Version

1.10

This header contains the kernel and service configurations of the operating system.

7.14.2 Macro Definition Documentation

7.14.2.1 ARM_CORTEX_M4

```
#define ARM_CORTEX_M4 ( 1 )
```

ARM Cortex-M4

Definition at line 69 of file gos_config.h.

7.14.2.2 CFG_GCP_CHANNELS_MAX_NUMBER

```
#define CFG_GCP_CHANNELS_MAX_NUMBER ( 4 )
```

GCP maximum number of channels.

Definition at line 271 of file gos_config.h.

7.14.2.3 CFG_IDLE_TASK_STACK_SIZE

```
#define CFG_IDLE_TASK_STACK_SIZE ( 0x400 )
```

Idle task stack size.

Definition at line 114 of file gos_config.h.

7.14.2.4 CFG_MESSAGE_MAX_ADDRESSEES

```
#define CFG_MESSAGE_MAX_ADDRESSEES ( 8 )
```

Maximum number of message addressees.

Definition at line 235 of file gos_config.h.

7.14.2.5 CFG_MESSAGE_MAX_LENGTH

```
#define CFG_MESSAGE_MAX_LENGTH ( 80 )
```

Maximum length of a message in bytes.

Definition at line 223 of file gos_config.h.

7.14.2.6 CFG_MESSAGE_MAX_NUMBER

```
#define CFG_MESSAGE_MAX_NUMBER ( 8 )
```

Maximum number of messages handled at once.

Definition at line 219 of file gos_config.h.

7.14.2.7 CFG_MESSAGE_MAX_WAITER_IDS

```
#define CFG_MESSAGE_MAX_WAITER_IDS ( 8 )
```

Maximum number of message IDs a task can wait for (includes the terminating 0).

Definition at line 231 of file gos_config.h.

7.14.2.8 CFG_MESSAGE_MAX_WAITERS

```
#define CFG_MESSAGE_MAX_WAITERS ( 10 )
```

Maximum number of message waiters.

Definition at line 227 of file gos_config.h.

7.14.2.9 CFG_QUEUE_MAX_ELEMENTS

```
#define CFG_QUEUE_MAX_ELEMENTS ( 40 )
```

Maximum number of queue elements.

Definition at line 186 of file gos_config.h.

7.14.2.10 CFG_QUEUE_MAX_LENGTH

```
#define CFG_QUEUE_MAX_LENGTH ( 200 )
```

Maximum queue length.

Definition at line 190 of file gos_config.h.

7.14.2.11 CFG_QUEUE_MAX_NAME_LENGTH

```
#define CFG_QUEUE_MAX_NAME_LENGTH ( 24 )
```

Maximum queue name length.

Definition at line 198 of file gos_config.h.

7.14.2.12 CFG_QUEUE_MAX_NUMBER

```
#define CFG_QUEUE_MAX_NUMBER ( 4 )
```

Maximum number of queues.

Definition at line 182 of file gos_config.h.

7.14.2.13 CFG_QUEUE_USE_NAME

```
#define CFG_QUEUE_USE_NAME ( 1 )
```

Queue use name flag.

Definition at line 194 of file gos_config.h.

7.14.2.14 CFG_RESET_ON_ERROR

```
#define CFG_RESET_ON_ERROR ( 1 )
```

Flag to indicate if the system should reset on error.

Definition at line 305 of file gos_config.h.

7.14.2.15 CFG_RESET_ON_ERROR_DELAY_MS

```
#define CFG_RESET_ON_ERROR_DELAY_MS ( 3000 )
```

Delay time before system reset.

Definition at line 309 of file gos_config.h.

7.14.2.16 CFG_SCHED_COOPERATIVE

```
#define CFG_SCHED_COOPERATIVE ( 0 )
```

Cooperative scheduling flag.

Definition at line 82 of file gos_config.h.

7.14.2.17 CFG_SHELL_COMMAND_BUFFER_SIZE

```
#define CFG_SHELL_COMMAND_BUFFER_SIZE ( 200 )
```

Command buffer size.

Definition at line 259 of file gos_config.h.

7.14.2.18 CFG_SHELL_MAX_COMMAND_LENGTH

```
#define CFG_SHELL_MAX_COMMAND_LENGTH ( 20 )
```

Maximum command length.

Definition at line 251 of file gos_config.h.

7.14.2.19 CFG_SHELL_MAX_COMMAND_NUMBER

```
#define CFG_SHELL_MAX_COMMAND_NUMBER ( 16 )
```

Maximum number of shell commands.

Definition at line 247 of file gos_config.h.

7.14.2.20 CFG_SHELL_MAX_PARAMS_LENGTH

```
#define CFG_SHELL_MAX_PARAMS_LENGTH ( 128 )
```

Maximum parameters length.

Definition at line 255 of file gos_config.h.

7.14.2.21 CFG_SHELL_STARTUP_DELAY_MS

```
#define CFG_SHELL_STARTUP_DELAY_MS ( 500 )
```

Shell daemon startup delay time [ms].

Definition at line 263 of file gos_config.h.

7.14.2.22 CFG_SHELL_USE_SERVICE

```
#define CFG_SHELL_USE_SERVICE ( 1 )
```

Shell service use flag.

Definition at line 243 of file gos_config.h.

7.14.2.23 CFG_SIGNAL_MAX_NUMBER

```
#define CFG_SIGNAL_MAX_NUMBER ( 6 )
```

Maximum number of signals.

Definition at line 207 of file gos_config.h.

7.14.2.24 CFG_SIGNAL_MAX_SUBSCRIBERS

```
#define CFG_SIGNAL_MAX_SUBSCRIBERS ( 6 )
```

Maximum number of signal subscribers.

Definition at line 211 of file gos_config.h.

7.14.2.25 CFG_SYSMON_GCP_CHANNEL_NUM

```
#define CFG_SYSMON_GCP_CHANNEL_NUM ( 0 )
```

Define sysmon GCP channel number.

Definition at line 292 of file gos_config.h.

7.14.2.26 CFG_SYSMON_MAX_USER_MESSAGES

```
#define CFG_SYSMON_MAX_USER_MESSAGES ( 24 )
```

Maximum number of user messages.

Definition at line 297 of file gos_config.h.

7.14.2.27 CFG_SYSMON_USE_SERVICE

```
#define CFG_SYSMON_USE_SERVICE ( 1 )
```

Sysmon use service flag.

Definition at line 287 of file gos_config.h.

7.14.2.28 CFG_SYSTEM_TASK_STACK_SIZE

```
#define CFG_SYSTEM_TASK_STACK_SIZE ( 0x800 )
```

System task stack size.

Definition at line 118 of file gos_config.h.

7.14.2.29 CFG_TARGET_CPU

```
#define CFG_TARGET_CPU ( ARM_CORTEX_M4 )
```

Target CPU.

Definition at line 74 of file gos_config.h.

7.14.2.30 CFG_TASK_MAX_NAME_LENGTH

```
#define CFG_TASK_MAX_NAME_LENGTH ( 32 )
```

Maximum task name length.

Definition at line 94 of file gos_config.h.

7.14.2.31 CFG_TASK_MAX_NUMBER

```
#define CFG_TASK_MAX_NUMBER ( 48 )
```

Maximum number of tasks.

Definition at line 98 of file gos_config.h.

7.14.2.32 CFG_TASK_MAX_STACK_SIZE

```
#define CFG_TASK_MAX_STACK_SIZE ( 0x4000 )
```

Maximum task stack size.

Definition at line 110 of file gos_config.h.

7.14.2.33 CFG_TASK_MESSAGE_DAEMON_PRIO

```
#define CFG_TASK_MESSAGE_DAEMON_PRIO ( 198 )
```

Message daemon task priority.

Definition at line 154 of file gos_config.h.

7.14.2.34 CFG_TASK_MESSAGE_DAEMON_STACK

```
#define CFG_TASK_MESSAGE_DAEMON_STACK ( 0x600 )
```

Message daemon task stack size.

Definition at line 130 of file gos_config.h.

7.14.2.35 CFG_TASK_MIN_STACK_SIZE

```
#define CFG_TASK_MIN_STACK_SIZE ( 0x200 )
```

Minimum task stack size.

Definition at line 106 of file gos_config.h.

7.14.2.36 CFG_TASK_SHELL_DAEMON_PRIO

```
#define CFG_TASK_SHELL_DAEMON_PRIO ( 192 )
```

Shell daemon task priority.

Definition at line 162 of file gos_config.h.

7.14.2.37 CFG_TASK_SHELL_DAEMON_STACK

```
#define CFG_TASK_SHELL_DAEMON_STACK ( 0x300 )
```

Shell daemon task stack size.

Definition at line 134 of file gos_config.h.

7.14.2.38 CFG_TASK_SIGNAL_DAEMON_PRIO

```
#define CFG_TASK_SIGNAL_DAEMON_PRIO ( 197 )
```

Signal daemon task priority.

Definition at line 158 of file gos_config.h.

7.14.2.39 CFG_TASK_SIGNAL_DAEMON_STACK

```
#define CFG_TASK_SIGNAL_DAEMON_STACK ( 0x400 )
```

Signal daemon task stack size.

Definition at line 122 of file gos_config.h.

7.14.2.40 CFG_TASK_SYS_PRIO

```
#define CFG_TASK_SYS_PRIO ( 195 )
```

System task priority.

Definition at line 170 of file gos_config.h.

7.14.2.41 CFG_TASK_SYSMON_DAEMON_PRIO

```
#define CFG_TASK_SYSMON_DAEMON_PRIO ( 191 )
```

Sysmon daemon task priority.

Definition at line 174 of file gos_config.h.

7.14.2.42 CFG_TASK_SYSMON_DAEMON_STACK

```
#define CFG_TASK_SYSMON_DAEMON_STACK ( 0x800 )
```

Sysmon daemon task stack size.

Definition at line 142 of file gos_config.h.

7.14.2.43 CFG_TASK_TIME_DAEMON_PRIO

```
#define CFG_TASK_TIME_DAEMON_PRIO ( 196 )
```

Time daemon task priority.

Definition at line 166 of file gos_config.h.

7.14.2.44 CFG_TASK_TIME_DAEMON_STACK

```
#define CFG_TASK_TIME_DAEMON_STACK ( 0x300 )
```

Time daemon task stack size.

Definition at line 126 of file gos_config.h.

7.14.2.45 CFG_TASK_TRACE_DAEMON_PRIO

```
#define CFG_TASK_TRACE_DAEMON_PRIO ( 193 )
```

Trace daemon task priority.

Definition at line 150 of file gos_config.h.

7.14.2.46 CFG_TASK_TRACE_DAEMON_STACK

```
#define CFG_TASK_TRACE_DAEMON_STACK ( 0x400 )
```

Log daemon task stack size.

Definition at line 138 of file gos_config.h.

7.14.2.47 CFG_TRACE_MAX_LENGTH

```
#define CFG_TRACE_MAX_LENGTH ( 200 )
```

Trace maximum (line) length.

Definition at line 279 of file gos_config.h.

7.14.2.48 CFG_USE_PRIO_INHERITANCE

```
#define CFG_USE_PRIO_INHERITANCE ( 1 )
```

Priority inheritance flag for lock.

Definition at line 86 of file gos_config.h.

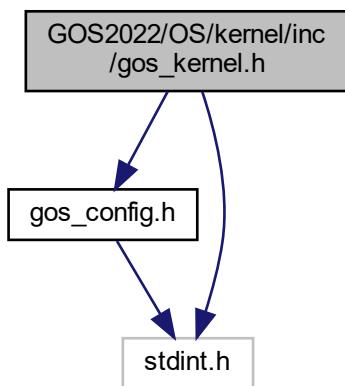
7.15 GOS2022/OS/kernel/inc/gos_kernel.h File Reference

GOS kernel header.

```
#include <gos_config.h>
```

```
#include <stdint.h>
```

Include dependency graph for gos_kernel.h:



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



Data Structures

- struct `gos_runtime_t`
- struct `gos_taskDescriptor_t`

Macros

- `#define NULL (void *) 0`
- `#define RAM_START (0x20000000u)`
- `#define RAM_SIZE (128 * 1024)`
- `#define MAIN_STACK (RAM_START + RAM_SIZE)`
- `#define GLOBAL_STACK (0x1200)`
- `#define GOS_DEFAULT_TASK_ID ((gos_tid_t)0x8000)`
- `#define GOS_INVALID_TASK_ID ((gos_tid_t)0x0100)`
- `#define GOS_TASK_MAX_PRIO_LEVELS (UINT8_MAX)`
- `#define GOS_TASK_IDLE_PRIO (GOS_TASK_MAX_PRIO_LEVELS)`
- `#define GOS_TASK_MAX_BLOCK_TIME_MS (0xFFFFFFFFu)`
- `#define GOS_STATIC static`
- `#define GOS_CONST const`
- `#define GOS_INLINE inline __attribute__((always_inline))`
- `#define GOS_STATIC_INLINE GOS_STATIC GOS_INLINE`
- `#define GOS_EXTERN extern`
- `#define GOS_NAKED __attribute__((naked))`
- `#define GOS_UNUSED __attribute__((unused))`
- `#define GOS_NOP __asm volatile ("NOP")`
- `#define GOS_ASM __asm volatile`
- `#define GOS_DISABLE_SCHED`
- `#define GOS_ENABLE_SCHED`
- `#define GOS_ISR_ENTER`
- `#define GOS_ISR_EXIT`
- `#define GOS_ATOMIC_ENTER`
- `#define GOS_ATOMIC_EXIT`
- `#define GOS_PRIV_TASK_MANIPULATE (1 << 15)`
- `#define GOS_PRIV_TASK_PRIO_CHANGE (1 << 14)`
- `#define GOS_PRIV_TRACE (1 << 13)`
- `#define GOS_PRIV_SIGNALING (1 << 11)`
- `#define GOS_PRIV_RESERVED_3 (1 << 10)`
- `#define GOS_PRIV_RESERVED_4 (1 << 9)`
- `#define GOS_PRIV_RESERVED_5 (1 << 8)`
- `#define GOS_CONCAT_RESULT(finalResult, currentResult)`

Typedefs

- `typedef uint8_t bool_t`
Boolean logic type.
- `typedef uint8_t u8_t`
8-bit unsigned type.
- `typedef uint16_t u16_t`
16-bit unsigned type.
- `typedef uint32_t u32_t`
32-bit unsigned type.
- `typedef uint64_t u64_t`
64-bit unsigned type.
- `typedef int8_t s8_t`
8-bit signed type.
- `typedef int16_t s16_t`
16-bit signed type.
- `typedef int32_t s32_t`
32-bit signed type.
- `typedef int64_t s64_t`
64-bit signed type.
- `typedef char char_t`
8-bit character type.
- `typedef float float_t`
Single precision float type.
- `typedef double double_t`
Double precision float type.
- `typedef void void_t`
Void type.
- `typedef u16_t gos_tid_t`
Task ID type.
- `typedef char_t gos_taskName_t[CFG_TASK_MAX_NAME_LENGTH]`
Task name type.
- `typedef void_t(* gos_task_t) (void_t)`
Task function type.
- `typedef u8_t gos_taskPrio_t`
Task priority type.
- `typedef u32_t gos_taskSleepTick_t`
Sleep tick type.
- `typedef u32_t gos_blockMaxTick_t`
Block max. tick type.
- `typedef u32_t gos_taskAddress_t`
Memory address type.
- `typedef u32_t gos_taskRunCounter_t`
Run counter type.
- `typedef u64_t gos_taskRunTime_t`
Run-time type.
- `typedef u32_t gos_taskCSCCounter_t`
Context-switch counter type.
- `typedef u16_t gos_taskStackSize_t`
Task stack size type.
- `typedef void_t(* gos_taskSwapHook_t) (gos_tid_t, gos_tid_t)`

- *Task swap hook type.*
- `typedef void_t(* gos_taskIdleHook_t) (void_t)`
- Task idle hook type.*
- `typedef void_t(* gos_taskSleepHook_t) (gos_tid_t)`
- Task sleep hook type.*
- `typedef void_t(* gos_taskWakeupHook_t) (gos_tid_t)`
- Task wake-up hook type.*
- `typedef void_t(* gos_taskSuspendHook_t) (gos_tid_t)`
- Task suspend hook type.*
- `typedef void_t(* gos_taskResumeHook_t) (gos_tid_t)`
- Task resume hook type.*
- `typedef void_t(* gos_taskBlockHook_t) (gos_tid_t)`
- Task block hook type.*
- `typedef void_t(* gos_taskUnblockHook_t) (gos_tid_t)`
- Task unblock hook type.*
- `typedef void_t(* gos_taskDeleteHook_t) (gos_tid_t)`
- Task delete hook type.*
- `typedef void_t(* gos_sysTickHook_t) (void_t)`
- System tick hook type.*
- `typedef void_t(* gos_privilegedHook_t) (void_t)`
- Privileged mode hook type.*
- `typedef void_t(* gos_preResetHook_t) (void_t)`
- Kernel pre-reset hook type.*
- `typedef u16_t gos_microsecond_t`
- Microsecond type.*
- `typedef u16_t gos_millisecond_t`
- Millisecond type.*
- `typedef u8_t gos_second_t`
- Second type.*
- `typedef u8_t gos_minute_t`
- Minute type.*
- `typedef u8_t gos_hour_t`
- Hour type.*
- `typedef u16_t gos_day_t`
- Day type.*
- `typedef u8_t gos_month_t`
- Month type.*
- `typedef u16_t gos_year_t`
- Year type.*

Enumerations

- `enum gos_taskState_t {
 GOS_TASK_READY = 0b01010, GOS_TASK_SLEEPING = 0b10110, GOS_TASK_BLOCKED = 0b11001,
 GOS_TASK_SUSPENDED = 0b00101,
 GOS_TASK_ZOMBIE = 0b01101 }`
- `enum gos_taskPrivilegeLevel_t { GOS_TASK_PRIVILEGE_SUPERVISOR = 0xFFFF, GOS_TASK_PRIVILEGE_KERNEL
= 0xFF00, GOS_TASK_PRIVILEGE_USER = 0x00FF, GOS_TASK_PRIVILEGED_USER = 0x20FF }`
- `enum gos_result_t { GOS_SUCCESS = 0b01010101, GOS_ERROR = 0b10101110, GOS_BUSY =
0b10110001 }`
- `enum gos_boolValue_t { GOS_TRUE = 0b00110110, GOS_FALSE = 0b01001001 }`
- `enum gos_kernel_privilege_t { GOS_PRIVILEGED = 0b10110, GOS_UNPRIVILEGED = 0b01001 }`

Functions

- `gos_result_t gos_kernellInit (void_t)`
This function initializes the kernel.
- `gos_result_t gos_kernelRegisterSwapHook (gos_taskSwapHook_t swapHookFunction)`
Registers a task swap hook function.
- `gos_result_t gos_kernelRegisterIdleHook (gos_taskIdleHook_t idleHookFunction)`
Registers an idle hook function.
- `gos_result_t gos_kernelRegisterSysTickHook (gos_sysTickHook_t sysTickHookFunction)`
Registers a system tick hook function.
- `gos_result_t gos_kernelRegisterPrivilegedHook (gos_privilegedHook_t privilegedHookFunction)`
Registers a privileged hook function.
- `gos_result_t gos_kernelRegisterPreResetHook (gos_preResetHook_t preResetHookFunction)`
Registers a pre-reset hook function.
- `gos_result_t gos_taskSubscribeDeleteSignal (void_t(*deleteSignalHandler)(u16_t))`
Subscribes the given handler to the task delete signal.
- `gos_result_t gos_kernelSubscribeDumpReadySignal (void_t(*dumpReadySignalHandler)(u16_t))`
Subscribes the given handler to the dump ready signal.
- `u32_t gos_kernelGetSysTicks (void_t)`
Returns the system ticks.
- `u16_t gos_kernelGetCpuUsage (void_t)`
Returns the CPU usage.
- `gos_result_t gos_kernelStart (void_t)`
Starts the kernel.
- `void_t gos_kernelReset (void_t)`
Resets the microcontroller.
- `void_t gos_kernelPrivilegedModeSetRequired (void_t)`
Requests a privileged mode setting.
- `void_t gos_kernelDelayUs (u16_t microseconds)`
Blocking delay in microsecond range.
- `void_t gos_kernelDelayMs (u16_t milliseconds)`
Blocking delay in millisecond range.
- `void_t gos_kernelCalculateTaskCpuUsages (bool_t isResetRequired)`
Calculates the CPU usage for the tasks.
- `void_t gos_kernelDump (void_t)`
Kernel dump.
- `gos_result_t gos_kernelSetMaxCpuLoad (u16_t maxCpuLoad)`
Sets the maximum (global) CPU load.
- `gos_result_t gos_kernelGetMaxCpuLoad (u16_t *pMaxCpuLoad)`
Gets the maximum (global) CPU load.
- `bool_t gos_kernelsCallerIsr (void_t)`
Returns if the current task is ISR.
- `void_t gos_kernelReschedule (gos_kernel_privilege_t privilege)`
Reschedules the kernel.
- `gos_result_t gos_taskRegisterTasks (gos_taskDescriptor_t *taskDescriptors, u16_t arraySize)`
This function registers an array of tasks for scheduling.
- `gos_result_t gos_taskRegister (gos_taskDescriptor_t *taskDescriptor, gos_tid_t *taskId)`
This function registers a task for scheduling.
- `gos_result_t gos_taskSleep (gos_taskSleepTick_t sleepTicks)`
Sends the current task to sleeping state.
- `gos_result_t gos_taskWakeup (gos_tid_t taskId)`

- Wakes up the given task.*
- `gos_result_t gos_taskSuspend (gos_tid_t taskId)`
Sends the given task to suspended state.
 - `gos_result_t gos_taskResume (gos_tid_t taskId)`
Resumes the given task.
 - `gos_result_t gos_taskBlock (gos_tid_t taskId, gos_blockMaxTick_t blockTicks)`
Sends the given task to blocked state.
 - `gos_result_t gos_taskUnblock (gos_tid_t taskId)`
Unblocks the given task.
 - `gos_result_t gos_taskDelete (gos_tid_t taskId)`
Deletes the given task from the scheduling array.
 - `gos_result_t gos_taskSetPriority (gos_tid_t taskId, gos_taskPrio_t taskPriority)`
Sets the current priority of the given task to the given value (for temporary change).
 - `gos_result_t gos_taskSetOriginalPriority (gos_tid_t taskId, gos_taskPrio_t taskPriority)`
Sets the original priority of the given task to the given value (for permanent change).
 - `gos_result_t gos_taskGetPriority (gos_tid_t taskId, gos_taskPrio_t *taskPriority)`
Gets the current priority of the given task.
 - `gos_result_t gos_taskGetOriginalPriority (gos_tid_t taskId, gos_taskPrio_t *taskPriority)`
Gets the original priority of the given task.
 - `gos_result_t gos_taskAddPrivilege (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)`
Adds the given privileges to the given task.
 - `gos_result_t gos_taskRemovePrivilege (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)`
Removes the given privileges from the given task.
 - `gos_result_t gos_taskSetPrivileges (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)`
Sets the given privileges for the given task.
 - `gos_result_t gos_taskGetPrivileges (gos_tid_t taskId, gos_taskPrivilegeLevel_t *pPrivileges)`
Gets the privileges of the given task.
 - `gos_result_t gos_taskGetName (gos_tid_t taskId, gos_taskName_t taskName)`
Gets the task name of the task with the given ID.
 - `gos_result_t gos_taskGetId (gos_taskName_t taskName, gos_tid_t *pTaskId)`
Gets the task ID of the task with the given name.
 - `gos_result_t gos_taskGetCurrentId (gos_tid_t *pTaskId)`
Returns the ID of the currently running task.
 - `gos_result_t gos_taskGetData (gos_tid_t taskId, gos_taskDescriptor_t *pTaskData)`
Returns the task data of the given task.
 - `gos_result_t gos_taskGetDataByIndex (u16_t taskIndex, gos_taskDescriptor_t *pTaskData)`
Returns the task data of the given task.
 - `gos_result_t gos_taskGetNumber (u16_t *pTaskNum)`
Returns the number of registered tasks.
 - `gos_result_t gos_taskYield (void_t)`
Yields the current task.
 - `gos_result_t gos_platformDriverInit (void_t)`
Platform driver initializer. Used for the platform-specific driver initializations.
 - `gos_result_t gos_userApplicationInit (void_t)`
User application initializer. Used for the application-related initializations.

7.15.1 Detailed Description

GOS kernel header.

Author

Ahmed Gazar

Date

2025-03-22

Version

1.22

The GOS kernel is the core of the GOS system. It contains the basic type definitions of the system (these are used across the OS in other services), and it handles the system tick, tasks, and scheduling. The kernel module also provides an interface for registering tasks, changing their states, registering hook functions, disabling the scheduler. It also provides a service to dump the kernel configuration and task information on the log output.

7.15.2 Macro Definition Documentation

7.15.2.1 GOS_CONCAT_RESULT

```
#define GOS_CONCAT_RESULT( finalResult, currentResult )
```

Value:

```
{
    if (finalResult == GOS_SUCCESS)
    {
        finalResult = currentResult;
    }
    else
    {
        finalResult = GOS_ERROR;
    }
}
```

Result concatenating macro.

Definition at line 345 of file gos_kernel.h.

7.15.3 Enumeration Type Documentation

7.15.3.1 gos_boolValue_t

```
enum gos_boolValue_t
```

Boolean values.

Note

Hamming distance of true and false value: 7.

Enumerator

GOS_TRUE	True value.
GOS_FALSE	False value.

Definition at line 470 of file gos_kernel.h.

7.15.3.2 gos_kernel_privilege_t

```
enum gos_kernel_privilege_t
```

Kernel privilege levels.

Enumerator

GOS_PRIVILEGED	GOS_PRIVILEGED.
GOS_UNPRIVILEGED	GOS_UNPRIVILEGED.

Definition at line 479 of file gos_kernel.h.

7.15.3.3 gos_result_t

```
enum gos_result_t
```

Result type enumerator.

Note

Hamming distance of

- success and error: 7
- success and busy: 4
- error and busy: 5

Enumerator

GOS_SUCCESS	Success.
GOS_ERROR	Error.
GOS_BUSY	Busy.

Definition at line 459 of file gos_kernel.h.

7.15.3.4 gos_taskPrivilegeLevel_t

enum `gos_taskPrivilegeLevel_t`

Task privilege level enumerator.

Enumerator

<code>GOS_TASK_PRIVILEGE_SUPERVISOR</code>	Task supervisor privilege level.
<code>GOS_TASK_PRIVILEGE_KERNEL</code>	Task kernel privilege level.
<code>GOS_TASK_PRIVILEGE_USER</code>	Task user privilege level.
<code>GOS_TASK_PRIVILEGED_USER</code>	User with logging right.

Definition at line 424 of file gos_kernel.h.

7.15.3.5 gos_taskState_t

enum `gos_taskState_t`

Task state enumerator.

Note

Hamming distance of

- ready and sleeping: 3
- ready and blocked: 3
- ready and suspended: 4
- ready and zombie: 3
- sleeping and blocked: 4
- sleeping and suspended: 3
- sleeping and zombie: 4
- blocked and suspended: 3
- blocked and zombie: 2
- suspended and zombie: 1

Enumerator

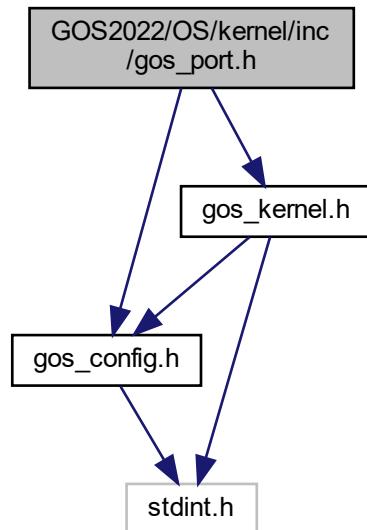
<code>GOS_TASK_READY</code>	Task is ready to be scheduled.
<code>GOS_TASK_SLEEPING</code>	Task is sleeping (waiting for wake-up ticks).
<code>GOS_TASK_BLOCKED</code>	Task is blocked (waiting for resource).
<code>GOS_TASK_SUSPENDED</code>	Task is suspended (not to be scheduled), but can be resumed.
<code>GOS_TASK_ZOMBIE</code>	Task deleted (physically existing in memory, but cannot be resumed).

Definition at line 412 of file gos_kernel.h.

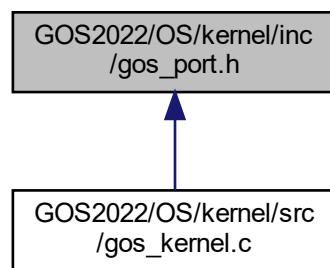
7.16 GOS2022/OS/kernel/inc/gos_port.h File Reference

GOS port header.

```
#include "gos_kernel.h"
#include "gos_config.h"
Include dependency graph for gos_port.h:
```



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



Macros

- `#define gos_ported_procReset()`

- #define gos_ported_reschedule(privilege)
- #define gos_ported_pendSVHandler GOS_NAKED PendSV_Handler
- #define gos_ported_doContextSwitch()
- #define gos_ported_svcHandler GOS_NAKED SVC_Handler
- #define gos_ported_handleSVC()
- #define gos_ported_svcHandlerMain gos_kernelSVC_HandlerMain
- #define gos_ported_handleSVCMain(sp)
- #define gos_ported_sysTickInterrupt SysTick_Handler
- #define gos_ported_kernelStartInit()
- #define gos_ported_enableFaultHandlers()

7.16.1 Detailed Description

GOS port header.

Author

Ahmed Gazar

Date

2023-07-12

Version

1.0

This header contains the platform-specific ported definitions of the OS.

7.16.2 Macro Definition Documentation

7.16.2.1 gos_ported_doContextSwitch

```
#define gos_ported_doContextSwitch( )
```

Value:

```
{
    \
    /* Save LR back to main, must do this firstly. */
    GOS_ASM("PUSH {LR}");

    /* Save the context of current task. */
    /* Get current PSP. */
    GOS_ASM("MRS R0, PSP");
    /* Save R4 to R11 to PSP Frame Stack. */
    GOS_ASM("STMDB R0!, {R4-R11}"); /* R0 is updated after decrement */
    /* Save current value of PSP. */
    GOS_ASM("BL gos_kernelSaveCurrentPsp"); /* R0 is first argument */

    /* Do scheduling. */
    /* Select next task. */
    GOS_ASM("BL gos_kernelSelectNextTask");

    /* Retrieve the context of next task. */
}
```

```

/* Get its past PSP value. */
GOS_ASM("BL gos_kernelGetCurrentPsp"); /* return PSP is in R0 */ \
/* Retrieve R4-R11 from PSP Fram Stack. */
GOS_ASM("LDMIA R0!, {R4-R11}"); /* R0 is updated after increment */ \
/* Update PSP. */
GOS_ASM("MSR PSP, R0"); \
/* Exit. */
GOS_ASM("POP {LR}"); \
GOS_ASM("BX LR");
}
)

```

Context-switch function.

Definition at line 82 of file gos_port.h.

7.16.2.2 gos_ported_enableFaultHandlers

```
#define gos_ported_enableFaultHandlers( )
```

Value:

```

( \
{
    SHCSR |= (1 << 16); /* Memory Fault */ \
    SHCSR |= (1 << 17); /* Bus Fault */ \
    SHCSR |= (1 << 18); /* Usage Fault */ \
}
)
```

Fault handler enable function.

Definition at line 204 of file gos_port.h.

7.16.2.3 gos_ported_handleSVC

```
#define gos_ported_handleSVC( )
```

Value:

```

( \
{
    /* Check LR to know which stack is used. */
    GOS_ASM("TST LR, #4");
    /* 2 next instructions are conditional. */
    GOS_ASM("ITE EQ");
    /* Save MSP if bit 2 is 0. */
    GOS_ASM("MRSEQ R0, MSP");
    /* Save PSP if bit 2 is 1. */
    GOS_ASM("MRSNE R0, PSP");

    /* Check if reset is required. */
    if (resetRequired == GOS_TRUE)
    {
        resetRequired = GOS_FALSE;
        gos_kernelProcessorReset();
    }

    /* Pass R0 as the argument. */
    GOS_ASM("B gos_kernelSVC_HandlerMain");
}
)
```

SVC handler function.

Definition at line 121 of file gos_port.h.

7.16.2.4 gos_ported_handleSVCMain

```
#define gos_ported_handleSVCMain(
    sp )
```

Value:

```
{
    /* Get the address of the instruction saved in PC. */
    u8_t* pInstruction = (u8_t*)(sp[6]);
```

```
    /* Go back 2 bytes (16-bit opcode). */
    pInstruction -= 2;
```

```
    /* Get the opcode, in little-endian. */
    u8_t svcNum = *pInstruction;
```

```
    switch (svcNum)
    {
        case 0xFF:
            /* Trigger PendSV. */
            ICSR |= (1 << 28);
            break;
        default: break;
    }
}
```

SVC handler main function.

Definition at line 152 of file gos_port.h.

7.16.2.5 gos_ported_kernelStartInit

```
#define gos_ported_kernelStartInit( )
```

Value:

```
{
    /* Prepare PSP of the first task. */
    GOS_ASM("BL gos_kernelGetCurrentPsp"); /* return PSP in R0 */
    GOS_ASM("MSR PSP, R0"); /* set PSP */
```

```
    /* Change to use PSP. */
    GOS_ASM("MRS R0, CONTROL");
    GOS_ASM("ORR R0, R0, #2"); /* set bit[1] SPSEL */
    GOS_ASM("MSR CONTROL, R0");
```

```
    /* Move to unprivileged level. */
    GOS_ASM("MRS R0, CONTROL");
    GOS_ASM("ORR R0, R0, #1"); /* Set bit[0] nPRIV */
    GOS_ASM("MSR CONTROL, R0");
    /* Right after here, access is limited. */
}
```

Kernel start initialization function.

Definition at line 182 of file gos_port.h.

7.16.2.6 gos_ported_pendSVHandler

```
#define gos_ported_pendSVHandler GOS_NAKED PendSV_Handler
```

Pend SV handler function name.

Definition at line 77 of file gos_port.h.

7.16.2.7 gos_ported_procReset

```
#define gos_ported_procReset( )
```

Value:

```
(  
{  
    \\\n    GOS_ASM ("dsb 0xF:::memory");  
    \\\n    * (u32_t*) (0xE000ED0CUL) = (u32_t) ((0x5FAUL < 16U) | (* (u32_t*) (0xE000ED0CUL) & (7UL < 8U)) | (1UL <  
        2U)); \\\n    GOS_ASM ("dsb 0xF:::memory");  
    \\\n}  
)  
\\
```

Processor reset function.

Definition at line 48 of file gos_port.h.

7.16.2.8 gos_ported_reschedule

```
#define gos_ported_reschedule( privilege )
```

Value:

```
( \
{ \
    if (privilege == GOS_PRIVILEGED) \
    { \
        /* Trigger PendSV directly. */ \
        ICSR |= (1 << 28); \
    } \
    else \
    { \
        /* Call Supervisor exception to get Privileged access. */ \
        GOS_ASM("SVC #255"); \
    } \
} \
}
```

Reschedule function.

Definition at line 59 of file gos_port.h.

7.16.2.9 gos_ported_svcHandler

```
#define gos_ported_svcHandler GOS_NAKED SVC_Handler
```

SVC handler function name.

Definition at line 116 of file gos_port.h.

7.16.2.10 gos_ported_svcHandlerMain

```
#define gos_ported_svcHandlerMain gos_kernelSVC_HandlerMain
```

SVC handler main function name.

Definition at line 147 of file gos_port.h.

7.16.2.11 gos_ported_sysTickInterrupt

```
#define gos_ported_sysTickInterrupt SysTick_Handler
```

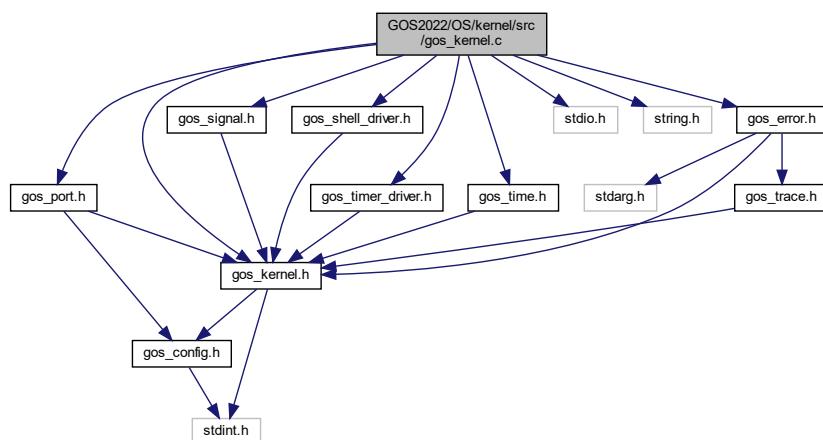
System tick handler function name.

Definition at line 177 of file gos_port.h.

7.17 GOS2022/OS/kernel/src/gos_kernel.c File Reference

GOS kernel source.

```
#include <gos_error.h>
#include <gos_kernel.h>
#include <gos_port.h>
#include <gos_signal.h>
#include <gos_shell_driver.h>
#include <gos_timer_driver.h>
#include <stdio.h>
#include <string.h>
#include <gos_time.h>
Include dependency graph for gos_kernel.c:
```



Macros

- `#define SHCSR *((volatile u32_t*) 0xE000ED24u)`
- `#define ICSR *((volatile u32_t*) 0xE000ED04u)`
- `#define TASK_DUMP_SEPARATOR "+-----+-----+-----+-----+\r\n"`
- `#define MAX_CPU_DUMP_SEPARATOR "+-----+-----+-----+-----+\r\n"`
- `#define STACK_STATS_SEPARATOR "+-----+-----+-----+-----+\r\n"`
- `#define CONFIG_DUMP_SEPARATOR "+-----+-----+-----+\r\n"`
- `#define BINARY_PATTERN "%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s"TRACE_FORMAT_RESET`
- `#define TO_BINARY(word)`

Functions

- `GOS_STATIC void_t gos_kernelCheckTaskStack (void_t)`
Checks the stack use of the current task.
- `GOS_STATIC u32_t gos_kernelGetCurrentPsp (void_t)`
Returns the current PSP.
- `GOS_STATIC void_t gos_kernelSaveCurrentPsp (u32_t psp)`
Saves the current PSP.
- `GOS_STATIC void_t gos_kernelSelectNextTask (void_t)`
Selects the next task for execution.
- `GOS_STATIC char_t * gos_kernelGetTaskStateString (gos_taskState_t taskState)`
Translates the task state to a string.
- `GOS_STATIC void_t gos_kernelProcessorReset (void_t)`
Kernel processor reset.
- `GOS_EXTERN void_t gos_idleTask (void_t)`
Kernel idle task.
- `gos_result_t gos_kernellInit (void_t)`
This function initializes the kernel.
- `gos_result_t gos_kernelStart (void_t)`
Starts the kernel.
- `gos_result_t gos_kernelRegisterSwapHook (gos_taskSwapHook_t swapHookFunction)`
Registers a task swap hook function.
- `gos_result_t gos_kernelRegisterSysTickHook (gos_sysTickHook_t sysTickHookFunction)`
Registers a system tick hook function.
- `gos_result_t gos_kernelRegisterPrivilegedHook (gos_privilegedHook_t privilegedHookFunction)`
Registers a privileged hook function.
- `gos_result_t gos_kernelRegisterPreResetHook (gos_preResetHook_t preResetHookFunction)`
Registers a pre-reset hook function.
- `gos_result_t gos_kernelSubscribeDumpReadySignal (gos_signalHandler_t dumpReadySignalHandler)`
- `void_t gos_ported_sysTickInterrupt (void_t)`
- `u32_t gos_kernelGetSysTicks (void_t)`
Returns the system ticks.
- `u16_t gos_kernelGetCpuUsage (void_t)`
Returns the CPU usage.
- `void_t gos_kernelReset (void_t)`
Resets the microcontroller.
- `void_t gos_kernelPrivilegedModeSetRequired (void_t)`
Requests a privileged mode setting.
- `GOS_INLINE void_t gos_kernelDelayUs (u16_t microseconds)`

- **GOS_INLINE void_t gos_kernelDelayMs (u16_t milliseconds)**
Blocking delay in millisecond range.
- **GOS_INLINE void_t gos_kernelCalculateTaskCpuUsages (bool_t isResetRequired)**
Calculates the CPU usage for the tasks.
- **void_t gos_kernelDump (void_t)**
Kernel dump.
- **gos_result_t gos_kernelSetMaxCpuLoad (u16_t maxCpuLoad)**
Sets the maximum (global) CPU load.
- **gos_result_t gos_kernelGetMaxCpuLoad (u16_t *pMaxCpuLoad)**
Gets the maximum (global) CPU load.
- **bool_t gos_kernellsCallerIsr (void_t)**
Returns if the current task is ISR.
- **void_t gos_ported_svcHandler (void_t)**
- **void_t gos_ported_svcHandlerMain (u32_t *sp)**
- **void_t gos_ported_pendSVHandler (void_t)**
- **GOS_INLINE void_t gos_kernelReschedule (gos_kernel_privilege_t privilege)**
Reschedules the kernel.
- **void_t NMI_Handler (void_t)**
- **void_t HardFault_Handler (void_t)**
- **void_t MemManage_Handler (void_t)**
- **void_t BusFault_Handler (void_t)**
- **void_t UsageFault_Handler (void_t)**

Variables

- **gos_signallid_t kernelDumpSignal**
- **gos_signallid_t kernelDumpReadySignal**
- **u8_t schedDisableCntr = 0u**
- **u8_t inlsr = 0u**
- **u8_t atomicCntr = 0u**
- **u32_t primask = 0u**
- **u32_t currentTaskIndex = 0u**
- **u16_t cpuUseLimit = 10000**
- **bool_t isKernelRunning = GOS_FALSE**
- **GOS_STATIC u32_t sysTicks = 0u**
- **GOS_STATIC u16_t sysTimerValue = 0u**
- **GOS_STATIC gos_runtime_t monitoringTime = {0}**
- **GOS_STATIC gos_taskSwapHook_t kernelSwapHookFunction = NULL**
- **GOS_STATIC gos_sysTickHook_t kernelSysTickHookFunction = NULL**
- **GOS_STATIC gos_privilegedHook_t kernelPrivilegedHookFunction = NULL**
- **GOS_STATIC gos_preResetHook_t kernelPreResetHookFunction = NULL**
- **GOS_STATIC bool_t resetRequired = GOS_FALSE**
- **GOS_STATIC bool_t privilegedModeSetRequired = GOS_FALSE**
- **GOS_STATIC u32_t previousTick = 0u**
- **GOS_EXTERN gos_taskDescriptor_t taskDescriptors [CFG_TASK_MAX_NUMBER]**

7.17.1 Detailed Description

GOS kernel source.

Author

Ahmed Gazar

Date

2025-03-22

Version

1.21

For a more detailed description of this module, please refer to [gos_kernel.h](#)

7.17.2 Macro Definition Documentation

7.17.2.1 BINARY_PATTERN

```
#define BINARY_PATTERN "%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s"TRACE_FORMAT_RESET
```

Pattern for binary format printing.

Definition at line 146 of file gos_kernel.c.

7.17.2.2 CONFIG_DUMP_SEPARATOR

```
#define CONFIG_DUMP_SEPARATOR "+-----+\r\n"
```

Config dump separator line.

Definition at line 141 of file gos_kernel.c.

7.17.2.3 ICSR

```
#define ICSR *( volatile u32_t* ) 0xE000ED04u )
```

Interrupt Control and State Register.

Definition at line 121 of file gos_kernel.c.

7.17.2.4 MAX_CPU_DUMP_SEPARATOR

```
#define MAX_CPU_DUMP_SEPARATOR "+-----+-----+-----+\r\n"
```

Max CPU loads dump separator line.

Definition at line 131 of file gos_kernel.c.

7.17.2.5 SHCSR

```
#define SHCSR *( volatile u32_t*) 0xE000ED24u )
```

System Handler control and state register.

Definition at line 116 of file gos_kernel.c.

7.17.2.6 STACK_STATS_SEPARATOR

```
#define STACK_STATS_SEPARATOR "+-----+-----+-----+-----+-----+
```

Stack statistics separator line.

Definition at line 136 of file gos_kernel.c.

7.17.2.7 TASK_DUMP_SEPARATOR

```
#define TASK_DUMP_SEPARATOR "+-----+-----+-----+-----+-----+
```

Task dump separator line.

Definition at line 126 of file gos_kernel.c.

7.17.2.8 TO_BINARY

```
#define TO_BINARY( word )
```

Value:

```
(word & 0x8000 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x4000 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x2000 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x1000 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0800 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0400 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0200 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0100 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0080 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0040 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0020 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0010 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0008 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0004 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0002 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0"), \
(word & 0x0001 ? TRACE_FG_GREEN_START"1" : TRACE_FG_RED_START"0")
```

u16_t to binary converter macro.

Definition at line 151 of file gos_kernel.c.

7.17.3 Function Documentation

7.17.3.1 gos_idleTask()

```
void_t gos_idleTask (
    void_t )
```

Kernel idle task.

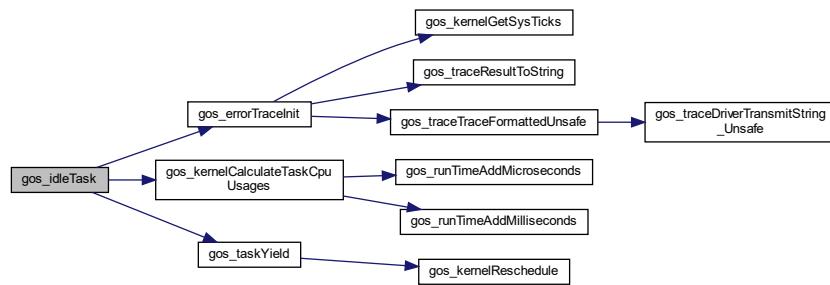
This task is executed when there is no other ready task in the system. When executed, this function refreshes the CPU-usage statistics of tasks.

Returns

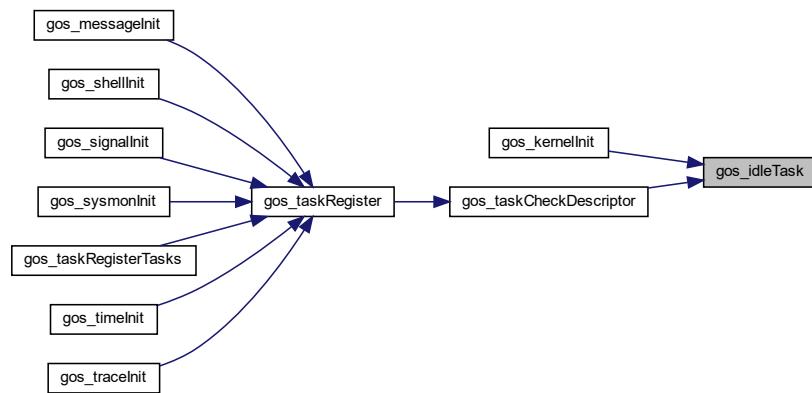
-

Definition at line 1323 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.3.2 gos_kernelCheckTaskStack()

```
GOS_STATIC void_t gos_kernelCheckTaskStack (
    void_t    )
```

Checks the stack use of the current task.

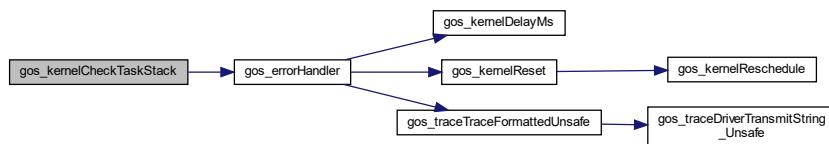
Gets the current stack pointer value and checks whether it is lower than the allowed threshold value defined for the current stack. After the overflow check, it calculates the stack usage and updates the maximum stack usage for the current task. In case of stack overflow, it goes to system error.

Returns

-

Definition at line 1001 of file gos_kernel.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.3.3 gos_kernelGetCurrentPsp()

```
GOS_UNUSED GOS_STATIC u32_t gos_kernelGetCurrentPsp (
    void_t    )
```

Returns the current PSP.

Returns the current PSP.

Returns

Current PSP value.

Definition at line 1047 of file gos_kernel.c.

7.17.3.4 gos_kernelGetTaskStateString()

```
GOS_STATIC char_t * gos_kernelGetTaskStateString (
    gos_taskState_t taskState )
```

Translates the task state to a string.

Based on the task state it returns a string with a printable form of the task state.

Parameters

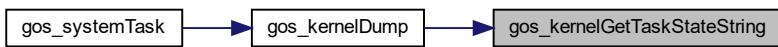
in	<i>taskState</i>	The task state variable to be translated.
----	------------------	---

Returns

String with the task state.

Definition at line 1207 of file gos_kernel.c.

Here is the caller graph for this function:



7.17.3.5 gos_kernelProcessorReset()

```
GOS_STATIC void_t gos_kernelProcessorReset (
    void_t )
```

Kernel processor reset.

Resets the processor.

Returns

-

Definition at line 1247 of file gos_kernel.c.

7.17.3.6 gos_kernelSaveCurrentPsp()

```
GOS_UNUSED GOS_STATIC void_t gos_kernelSaveCurrentPsp (
    u32_t psp )
```

Saves the current PSP.

Saves the current PSP.

Parameters

in	<i>psp</i>	Current PSP value.
----	------------	--------------------

Returns

Definition at line 1063 of file gos_kernel.c.

7.17.3.7 gos_kernelSelectNextTask()

```
GOS_UNUSED GOS_STATIC void_t gos_kernelSelectNextTask (
    void_t )
```

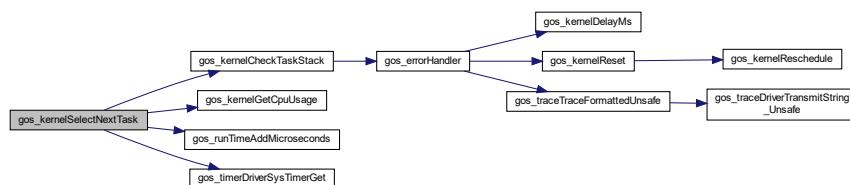
Selects the next task for execution.

Loops through the internal task descriptor array and first checks the sleeping tasks and wakes up the ones that passed their sleeping time. Then based on the priority of the ready tasks, it selects the one with the highest priority (lowest number in priority). If there is a swap-hook function registered, it calls it, and then it refreshes the task run-time statistics.

Returns

Definition at line 1082 of file gos_kernel.c.

Here is the call graph for this function:

**7.17.4 Variable Documentation****7.17.4.1 atomicCntr**

```
u8_t atomicCntr = 0u
```

Atomic counter (incremented when GOS_ATOMIC_ENTER is called).

Definition at line 195 of file gos_kernel.c.

7.17.4.2 cpuUseLimit

```
u16_t cpuUseLimit = 10000
```

CPU use limit.

Definition at line 210 of file gos_kernel.c.

7.17.4.3 currentTaskIndex

```
u32_t currentTaskIndex = 0u
```

Current task index - for priority scheduling.

Definition at line 205 of file gos_kernel.c.

7.17.4.4 inIsr

```
u8_t inIsr = 0u
```

In ISR counter.

Definition at line 190 of file gos_kernel.c.

7.17.4.5 isKernelRunning

```
bool_t isKernelRunning = GOS_FALSE
```

Flag to indicate whether kernel is running.

Definition at line 215 of file gos_kernel.c.

7.17.4.6 kernelDumpReadySignal

```
gos_signalId_t kernelDumpReadySignal
```

Kernel dump ready signal.

Definition at line 180 of file gos_kernel.c.

7.17.4.7 kernelDumpSignal

```
gos_signalId_t kernelDumpSignal
```

Kernel dump signal.

Definition at line 175 of file gos_kernel.c.

7.17.4.8 kernelPreResetHookFunction

```
GOS_STATIC gos_preResetHook_t kernelPreResetHookFunction = NULL
```

Kernel pre-reset hook function.

Definition at line 253 of file gos_kernel.c.

7.17.4.9 kernelPrivilegedHookFunction

```
GOS_STATIC gos_privilegedHook_t kernelPrivilegedHookFunction = NULL
```

Kernel privileged execution mode set hook function.

Definition at line 248 of file gos_kernel.c.

7.17.4.10 kernelSwapHookFunction

```
GOS_STATIC gos_taskSwapHook_t kernelSwapHookFunction = NULL
```

Kernel swap hook function.

Definition at line 238 of file gos_kernel.c.

7.17.4.11 kernelSysTickHookFunction

```
GOS_STATIC gos_sysTickHook_t kernelSysTickHookFunction = NULL
```

Kernel system tick hook function.

Definition at line 243 of file gos_kernel.c.

7.17.4.12 monitoringTime

```
GOS_STATIC gos_runtime_t monitoringTime = {0}
```

Monitoring system time since last statistics calculation.

Definition at line 233 of file gos_kernel.c.

7.17.4.13 previousTick

```
GOS_STATIC u32_t previousTick = 0u
```

Previous tick value for sleep and block tick calculation.

Definition at line 268 of file gos_kernel.c.

7.17.4.14 primask

```
u32_t primask = 0u
```

IRQ state for restoring after GOS_ATOMIC_EXIT.

Definition at line 200 of file gos_kernel.c.

7.17.4.15 privilegedModeSetRequired

```
GOS_STATIC bool_t privilegedModeSetRequired = GOS_FALSE
```

Flag to indicate whether privileged mode set is required.

Definition at line 263 of file gos_kernel.c.

7.17.4.16 resetRequired

```
GOS_STATIC bool_t resetRequired = GOS_FALSE
```

Reset required flag.

Definition at line 258 of file gos_kernel.c.

7.17.4.17 schedDisableCntr

```
u8_t schedDisableCntr = 0u
```

Scheduling disabled counter.

Definition at line 185 of file gos_kernel.c.

7.17.4.18 sysTicks

```
GOS_STATIC u32_t sysTicks = 0u
```

System tick value.

Definition at line 223 of file gos_kernel.c.

7.17.4.19 sysTimerValue

```
GOS_STATIC u16_t sysTimerValue = 0u
```

System timer value for run-time calculations.

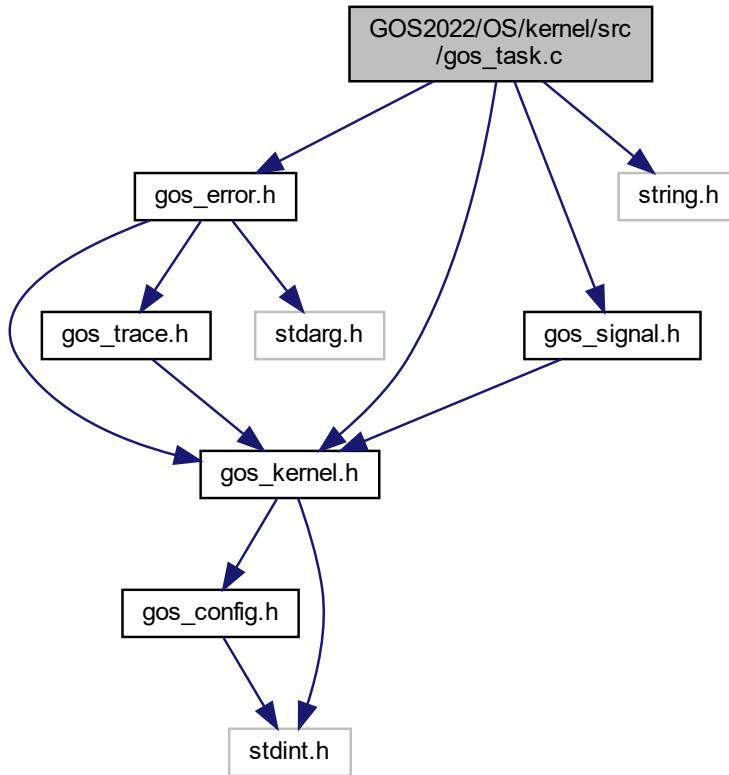
Definition at line 228 of file gos_kernel.c.

7.18 GOS2022/OS/kernel/src/gos_task.c File Reference

GOS task source.

```
#include <gos_error.h>
#include <gos_kernel.h>
#include <gos_signal.h>
```

```
#include <string.h>
Include dependency graph for gos_task.c:
```



Functions

- [GOS_STATIC gos_result_t gos_taskCheckDescriptor \(gos_taskDescriptor_t *taskDescriptor\)](#)
Checks the validity of the task descriptor.
- [void_t gos_idleTask \(void_t\)](#)
Kernel idle task.
- [gos_result_t gos_taskRegisterTasks \(gos_taskDescriptor_t *taskDescriptors, u16_t arraySize\)](#)
This function registers an array of tasks for scheduling.
- [gos_result_t gos_taskRegister \(gos_taskDescriptor_t *taskDescriptor, gos_tid_t *taskId\)](#)
This function registers a task for scheduling.
- [GOS_INLINE gos_result_t gos_taskSleep \(gos_taskSleepTick_t sleepTicks\)](#)
Sends the current task to sleeping state.
- [GOS_INLINE gos_result_t gos_taskWakeup \(gos_tid_t taskId\)](#)
Wakes up the given task.
- [GOS_INLINE gos_result_t gos_taskSuspend \(gos_tid_t taskId\)](#)
Sends the given task to suspended state.
- [GOS_INLINE gos_result_t gos_taskResume \(gos_tid_t taskId\)](#)
Resumes the given task.
- [GOS_INLINE gos_result_t gos_taskBlock \(gos_tid_t taskId, gos_blockMaxTick_t blockTicks\)](#)

- Sends the given task to blocked state.
- **GOS_INLINE gos_result_t gos_taskUnblock (gos_tid_t taskId)**
Unblocks the given task.
- **GOS_INLINE gos_result_t gos_taskDelete (gos_tid_t taskId)**
Deletes the given task from the scheduling array.
- **GOS_INLINE gos_result_t gos_taskSetPriority (gos_tid_t taskId, gos_taskPrio_t taskPriority)**
Sets the current priority of the given task to the given value (for temporary change).
- **GOS_INLINE gos_result_t gos_taskSetOriginalPriority (gos_tid_t taskId, gos_taskPrio_t taskPriority)**
Sets the original priority of the given task to the given value (for permanent change).
- **gos_result_t gos_taskGetPriority (gos_tid_t taskId, gos_taskPrio_t *taskPriority)**
Gets the current priority of the given task.
- **gos_result_t gos_taskGetOriginalPriority (gos_tid_t taskId, gos_taskPrio_t *taskPriority)**
Gets the original priority of the given task.
- **GOS_INLINE gos_result_t gos_taskAddPrivilege (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)**
Adds the given privileges to the given task.
- **GOS_INLINE gos_result_t gos_taskRemovePrivilege (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)**
Removes the given privileges from the given task.
- **GOS_INLINE gos_result_t gos_taskSetPrivileges (gos_tid_t taskId, gos_taskPrivilegeLevel_t privileges)**
Sets the given privileges for the given task.
- **GOS_INLINE gos_result_t gos_taskGetPrivileges (gos_tid_t taskId, gos_taskPrivilegeLevel_t *pPrivileges)**
Gets the privileges of the given task.
- **GOS_INLINE gos_result_t gos_taskYield (void_t)**
Yields the current task.
- **gos_result_t gos_taskGetName (gos_tid_t taskId, gos_taskName_t taskName)**
Gets the task name of the task with the given ID.
- **gos_result_t gos_taskGetId (gos_taskName_t taskName, gos_tid_t *pTaskId)**
Gets the task ID of the task with the given name.
- **GOS_INLINE gos_result_t gos_taskGetCurrentId (gos_tid_t *pTaskId)**
Returns the ID of the currently running task.
- **gos_result_t gos_taskGetData (gos_tid_t taskId, gos_taskDescriptor_t *pTaskData)**
Returns the task data of the given task.
- **gos_result_t gos_taskGetDataByIndex (u16_t taskIndex, gos_taskDescriptor_t *pTaskData)**
Returns the task data of the given task.
- **gos_result_t gos_taskGetNumber (u16_t *pTaskNum)**
Returns the number of registered tasks.
- **gos_result_t gos_taskRegisterIdleHook (gos_taskIdleHook_t idleHookFunction)**
- **gos_result_t gos_taskSubscribeDeleteSignal (gos_signalHandler_t deleteSignalHandler)**

Variables

- **gos_signalId_t kernelTaskDeleteSignal**
- **GOS_STATIC gos_taskIdleHook_t kernelIdleHookFunction = NULL**
- **GOS_EXTERN u8_t inLsr**
- **GOS_EXTERN u32_t currentTaskIndex**
- **GOS_EXTERN bool_t isKernelRunning**
- **gos_taskDescriptor_t taskDescriptors [CFG_TASK_MAX_NUMBER]**

7.18.1 Detailed Description

GOS task source.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.3

For a more detailed description of this module, please refer to [gos_kernel.h](#)

7.18.2 Function Documentation

7.18.2.1 gos_idleTask()

```
void_t gos_idleTask (
    void_t     )
```

Kernel idle task.

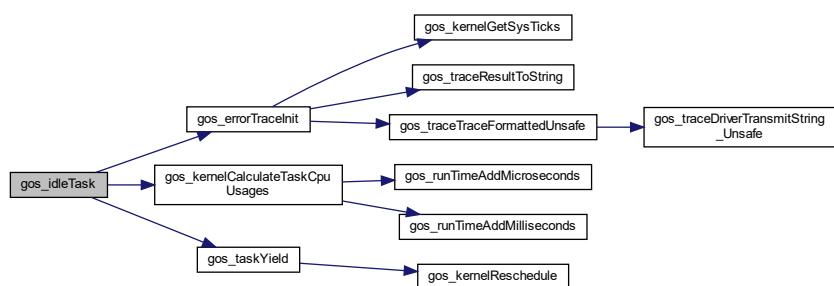
This task is executed when there is no other ready task in the system. When executed, this function refreshes the CPU-usage statistics of tasks.

Returns

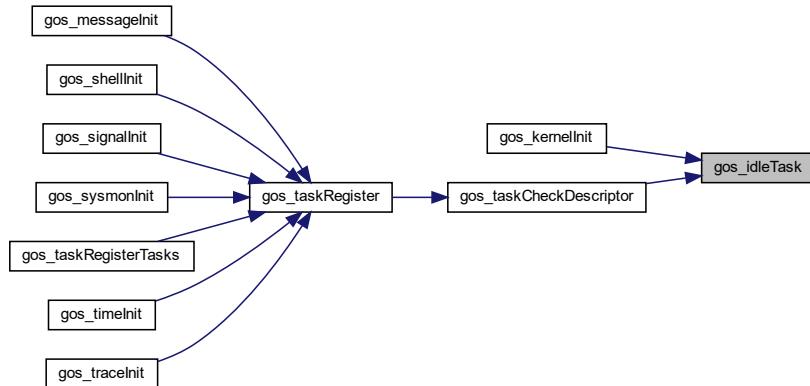
-

Definition at line 1323 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.2 `gos_taskCheckDescriptor()`

```
GOS_STATIC gos_result_t gos_taskCheckDescriptor (
    gos_taskDescriptor_t * taskDescriptor )
```

Checks the validity of the task descriptor.

Checks the task function pointer, task priority value, stack, and size parameters and return with error if the parameters are incorrect.

Parameters

in	<code>taskDescriptor</code>	Pointer to the task descriptor structure.
----	-----------------------------	---

Returns

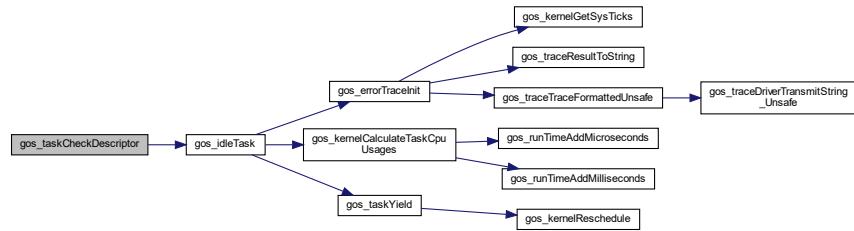
Result of task descriptor checking.

Return values

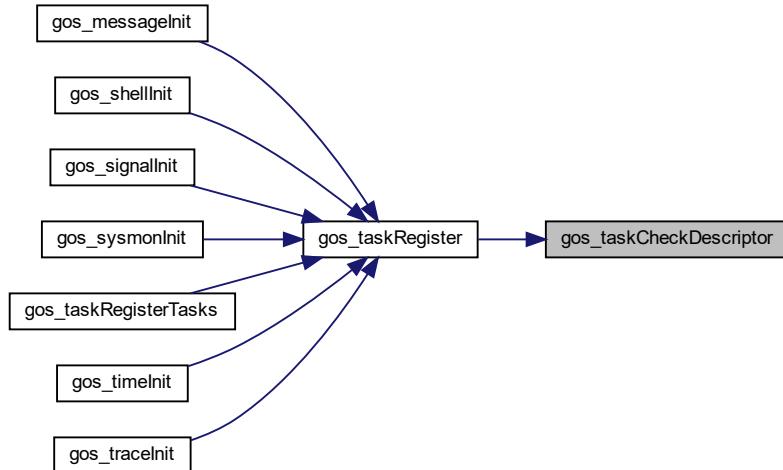
<code>GOS_SUCCESS</code>	Descriptor contains valid data.
<code>GOS_ERROR</code>	<p>One or more parameter is invalid:</p> <ul style="list-style-type: none"> • Task function is NULL pointer • Task function is the idle task • Priority exceeds the maximum priority level • Stack size is smaller than the minimum allowed • Stack size is greater than the maximum allowed • Stack size is not 4-byte aligned

Definition at line 1288 of file gos_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.3 Variable Documentation

7.18.3.1 kernelIdleHookFunction

```
GOS_STATIC gos_taskIdleHook_t kernelIdleHookFunction = NULL
```

Kernel idle hook function.

Definition at line 73 of file gos_task.c.

7.18.3.2 kernelTaskDeleteSignal

```
gos_signalId_t kernelTaskDeleteSignal
```

Kernel task delete signal.

Definition at line 65 of file gos_task.c.

7.18.3.3 taskDescriptors

```
gos_taskDescriptor_t taskDescriptors[CFG_TASK_MAX_NUMBER]
```

Initial value:

```
=
{
    [0] =
    {
        .taskFunction      = gos_idleTask,
        .taskId           = GOS_DEFAULT_TASK_ID,
        .taskPriority     = GOS_TASK_IDLE_PRIO,
        .taskName         = "gos_idle_task",
        .taskState        = GOS_TASK_READY,
        .taskStackSize    = CFG_IDLE_TASK_STACK_SIZE,
        .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL,
        .taskCpuUsageLimit = 10000
    }
}
```

Internal task array.

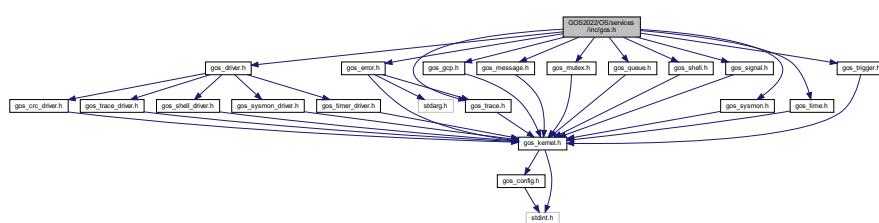
Definition at line 95 of file gos_task.c.

7.19 GOS2022/OS/services/inc/gos.h File Reference

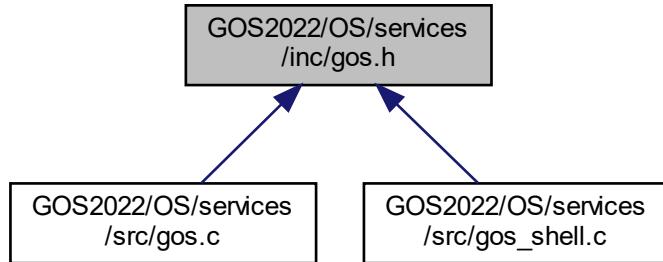
GOS header.

```
#include <gos_driver.h>
#include <gos_error.h>
#include <gos_gcp.h>
#include <gos_message.h>
#include <gos_mutex.h>
#include <gos_queue.h>
#include <gos_shell.h>
#include <gos_signal.h>
#include <gos_sysmon.h>
#include <gos_time.h>
#include <gos_trace.h>
#include <gos_trigger.h>
```

Include dependency graph for gos.h:



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



Macros

- `#define GOS_VERSION_MAJOR (1)`
- `#define GOS_VERSION_MINOR (1)`

Functions

- `void_t gos_Dump (void_t)`
GOS dump.

7.19.1 Detailed Description

GOS header.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.14

This header is a wrapper for the inclusion of all OS services and drivers for GOS2022 v1.1

7.19.2 Macro Definition Documentation

7.19.2.1 GOS_VERSION_MAJOR

```
#define GOS_VERSION_MAJOR ( 1 )
```

OS major version.

Definition at line 92 of file gos.h.

7.19.2.2 GOS_VERSION_MINOR

```
#define GOS_VERSION_MINOR ( 1 )
```

OS minor version.

Definition at line 97 of file gos.h.

7.19.3 Function Documentation

7.19.3.1 gos_Dump()

```
void_t gos_Dump (
    void_t   )
```

GOS dump.

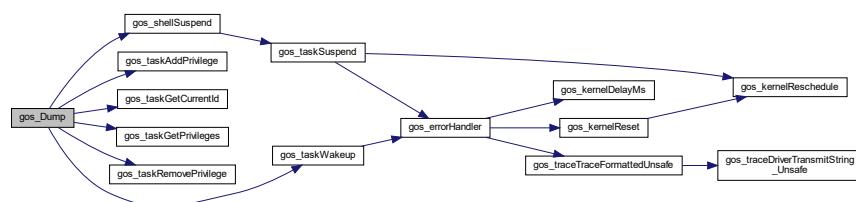
Sets the dump required flag and wakes up the system task that calls the individual dump functions.

Returns

-

Definition at line 207 of file gos.c.

Here is the call graph for this function:



Here is the caller graph for this function:

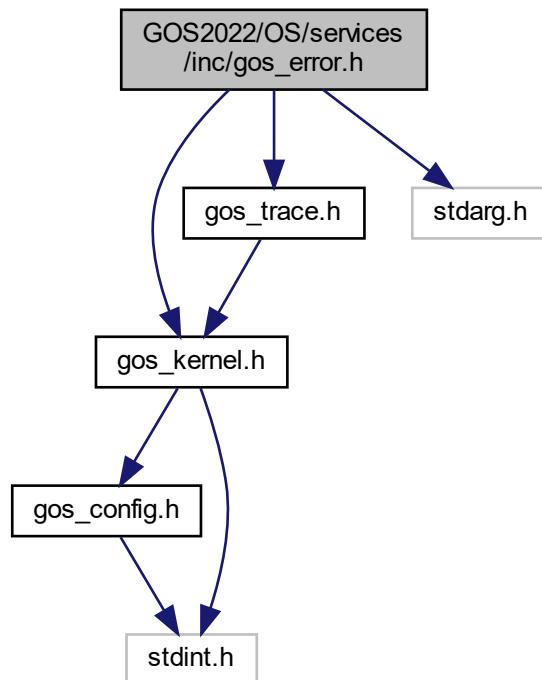


7.20 GOS2022/OS/services/inc/gos_error.h File Reference

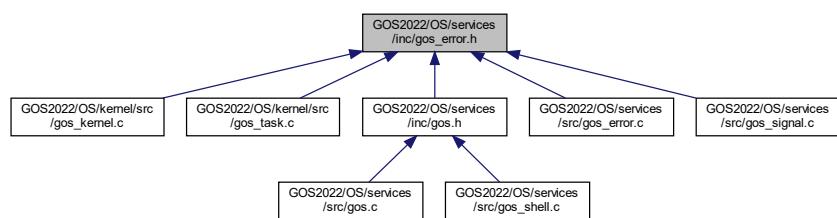
GOS error handler header.

```
#include <gos_kernel.h>
#include <gos_trace.h>
#include <stdarg.h>
```

Include dependency graph for gos_error.h:



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



Enumerations

- enum `gos_errorLevel_t` { `GOS_ERROR_LEVEL_OS_FATAL` = 0b110100, `GOS_ERROR_LEVEL_OS_WARNING` = 0b101110, `GOS_ERROR_LEVEL_USER_FATAL` = 0b011010, `GOS_ERROR_LEVEL_USER_WARNING` = 0b111010 }

Functions

- `void_t gos_printStartupLogo (void_t)`
Prints the startup logo on the trace output.
- `void_t gos_errorHandler (gos_errorLevel_t errorLevel, const char_t *function, u32_t line, const char_t *errorMessage,...)`
Handles the given error.
- `gos_result_t gos_errorTraceInit (const char_t *initDescription, gos_result_t initResult)`
Traces an initialization message.

7.20.1 Detailed Description

GOS error handler header.

Author

Ahmed Gazar

Date

2022-12-20

Version

2.0

This service is used for tracing error and initialization messages on the log output.

7.20.2 Enumeration Type Documentation

7.20.2.1 gos_errorLevel_t

enum `gos_errorLevel_t`

Error level enumerator.

Enumerator

<code>GOS_ERROR_LEVEL_OS_FATAL</code>	OS-level fatal error causing the system to stop.
<code>GOS_ERROR_LEVEL_OS_WARNING</code>	OS-level warning message will be logged, but system will not be stopped.
<code>GOS_ERROR_LEVEL_USER_FATAL</code>	User-level fatal error causing system stop.
<code>GOS_ERROR_LEVEL_USER_WARNING</code>	User-level warning.

Definition at line 66 of file gos_error.h.

7.20.3 Function Documentation

7.20.3.1 gos_errorHandler()

```
void_t gos_errorHandler (
    gos_errorLevel_t errorLevel,
    const char_t * function,
    u32_t line,
    const char_t * errorMessage,
    ...
)
```

Handles the given error.

Prints the formatted error message on the trace output, and based on the error level, it returns or stays in an infinite loop and disables scheduling.

Parameters

in	<i>errorLevel</i>	Level of error (OS/user, warning/fatal).
in	<i>function</i>	Function name.
in	<i>line</i>	Line number.
in	<i>errorMessage</i>	Error message.
in	...	Formatter variable arguments.

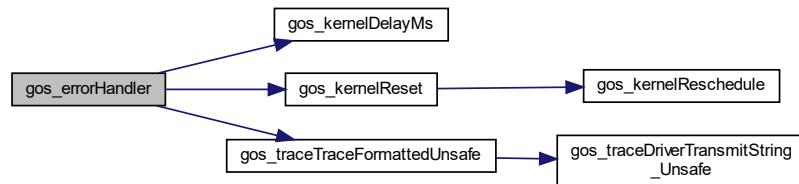
Returns

-

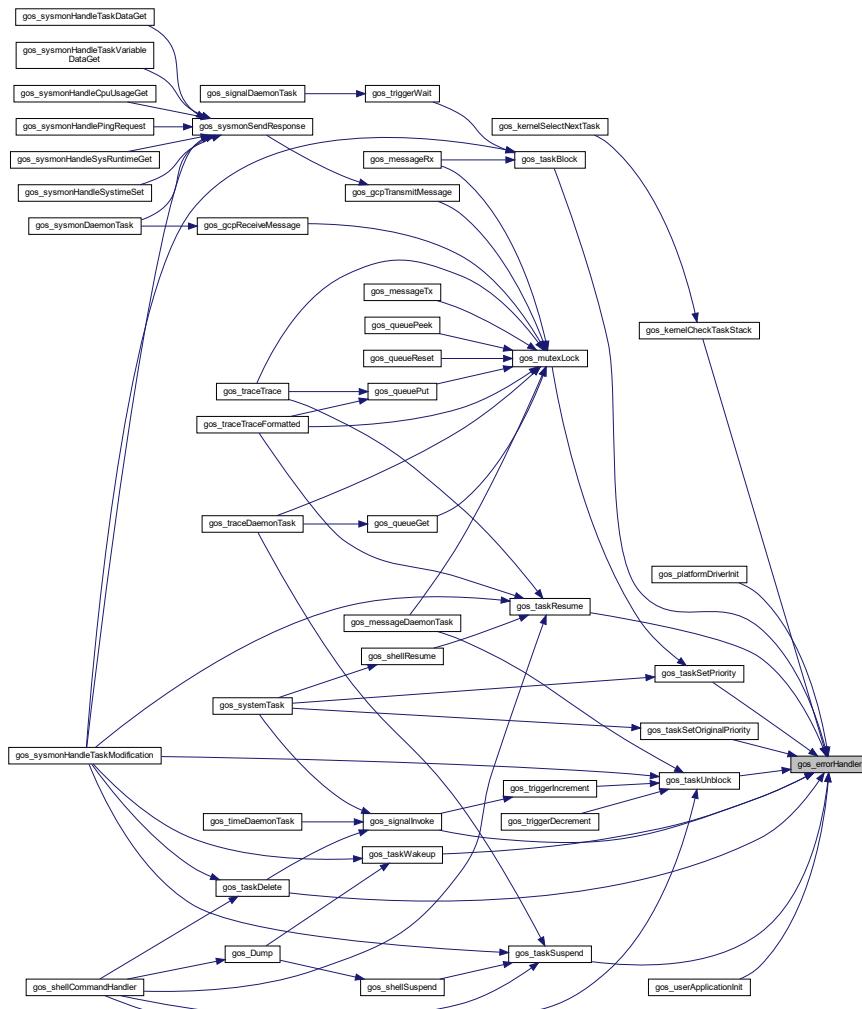
Definition at line 129 of file gos_error.c.

-

Here is the call graph for this function:



Here is the caller graph for this function:



7.20.3.2 `gos_errorTraceInit()`

```
gos_result_t gos_errorTraceInit (
    const char_t * initDescription,
    gos_result_t initResult )
```

Traces an initialization message.

Writes the formatted initialization message on the trace output.

Parameters

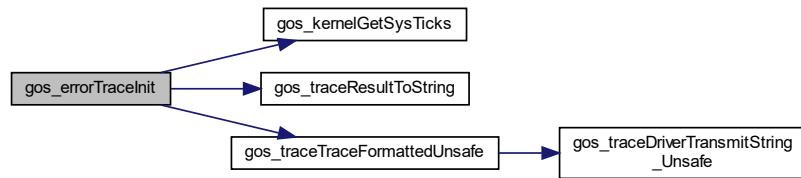
in	<i>initDescription</i>	Message to describe the initialization step.
in	<i>initResult</i>	Result of initialization.

Returns

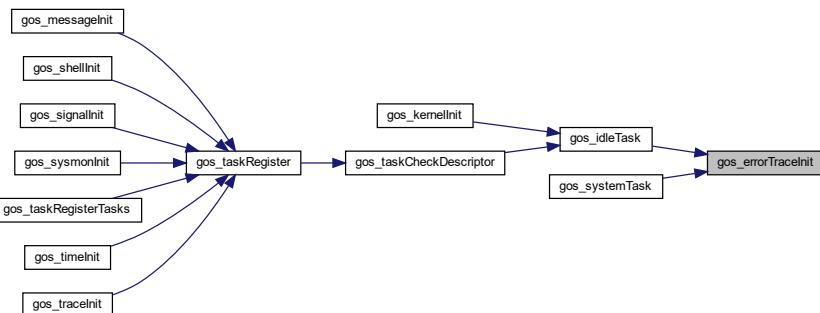
Result of initialization.

Definition at line 237 of file gos_error.c.

Here is the call graph for this function:



Here is the caller graph for this function:

**7.20.3.3 gos_printStartupLogo()**

```
void_t gos_printStartupLogo (
    void_t )
```

Prints the startup logo on the trace output.

Prints the GOS logo (similar to the ones used in the file headers) on the log output.

Returns

Definition at line 108 of file gos_error.c.

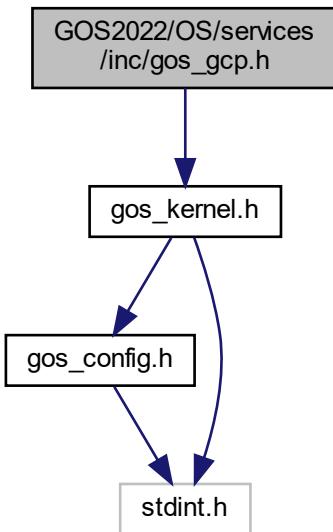
Here is the call graph for this function:



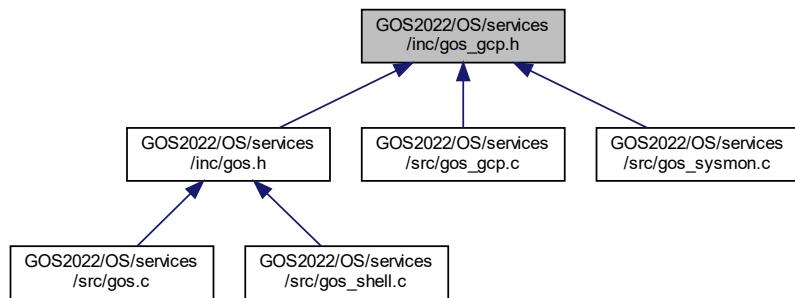
7.21 GOS2022/OS/services/inc/gos_gcp.h File Reference

GOS General Communication Protocol header.

```
#include <gos_kernel.h>
Include dependency graph for gos_gcp.h:
```



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



Typedefs

- `typedef u8_t gos_gcpChannelNumber_t`
GCP channel number.
- `typedef gos_result_t(* gos_gcpTransmitFunction_t) (u8_t *, u16_t)`
- `typedef gos_result_t(* gos_gcpReceiveFunction_t) (u8_t *, u16_t)`

Functions

- `gos_result_t gos_gcpInit (void_t)`
Initializes the GCP service.
- `gos_result_t gos_gcpRegisterPhysicalDriver (gos_gcpChannelNumber_t channel, gos_gcpTransmitFunction_t transmitFunction, gos_gcpReceiveFunction_t receiveFunction)`
Registers the physical-layer transmit and receive driver functions.
- `gos_result_t gos_gcpTransmitMessage (gos_gcpChannelNumber_t channel, u16_t messageId, void_t *pMessagePayload, u16_t payloadSize, u16_t maxChunkSize)`
Transmits the given message via the GCP protocol.
- `gos_result_t gos_gcpReceiveMessage (gos_gcpChannelNumber_t channel, u16_t *pMessageId, void_t *pPayloadTarget, u16_t targetSize, u16_t maxChunkSize)`
Receives the given message via the GCP protocol.

7.21.1 Detailed Description

GOS General Communication Protocol header.

Author

Ahmed Gazar

Date

2024-07-18

Version

3.0

This service implements the GCP frame and message layers.

7.21.2 Typedef Documentation

7.21.2.1 gos_gcpReceiveFunction_t

```
typedef gos_result_t (* gos_gcpReceiveFunction_t) (u8_t *, u16_t)
```

GCP physical layer receive function type.

Definition at line 75 of file gos_gcp.h.

7.21.2.2 gos_gcpTransmitFunction_t

```
typedef gos_result_t(* gos_gcpTransmitFunction_t) (u8_t *, u16_t)
```

GCP physical layer transmit function type.

Definition at line 70 of file gos_gcp.h.

7.21.3 Function Documentation

7.21.3.1 gos_gcplInit()

```
gos_result_t gos_gcplInit (
    void_t )
```

Initializes the GCP service.

Creates the GCP lock.

Returns

Result of initialization.

Return values

GOS_SUCCESS	GCP service initialized successfully.
GOS_ERROR	Lock creation error.

Definition at line 171 of file gos_gcp.c.

Here is the call graph for this function:



7.21.3.2 gos_gcpReceiveMessage()

```
gos_result_t gos_gcpReceiveMessage (
    gos_gcpChannelNumber_t channel,
```

```
u16_t * pMessageId,
void_t * pPayloadTarget,
u16_t targetSize,
u16_t maxChunkSize )
```

Receives the given message via the GCP protocol.

Calls the internal receiver function.

Parameters

in	<i>channel</i>	GCP channel.
out	<i>pMessageId</i>	Pointer to a variable to store the message ID.
out	<i>pPayloadTarget</i>	Pointer to the payload target buffer.
in	<i>targetSize</i>	Size of the target buffer (in bytes).
in	<i>maxChunkSize</i>	Maximum size of payload chunks.

Returns

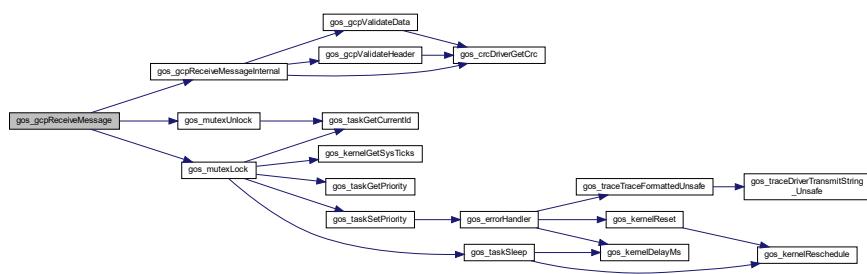
Result of message reception.

Return values

GOS_SUCCESS	Message received successfully.
GOS_ERROR	An error occurred during reception or validation.

Definition at line 258 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.3.3 gos_gcpRegisterPhysicalDriver()

```
gos_result_t gos_gcpRegisterPhysicalDriver (
    gos_gcpChannelNumber_t channel,
    gos_gcpTransmitFunction_t transmitFunction,
    gos_gcpReceiveFunction_t receiveFunction )
```

Registers the physical-layer transmit and receive driver functions.

Registers the physical-layer transmit and receive driver functions.

Parameters

in	<i>channel</i>	GCP channel.
in	<i>transmitFunction</i>	Transmit function to register.
in	<i>receiveFunction</i>	Receive function to register.

Returns

Result of physical driver registration.

Return values

<i>GOS_SUCCESS</i>	Physical driver registration successful.
<i>GOS_ERROR</i>	Transmit or receive function is NULL.

Definition at line 194 of file gos_gcp.c.

Here is the caller graph for this function:



7.21.3.4 gos_gcpTransmitMessage()

```
gos_result_t gos_gcpTransmitMessage (
    gos_gcpChannelNumber_t channel,
    u16_t messageId,
    void_t * pMessagePayload,
    u16_t payloadSize,
    u16_t maxChunkSize )
```

Transmits the given message via the GCP protocol.

Calls the internal message transmitter function.

Parameters

in	<i>channel</i>	GCP channel.
in	<i>messageId</i>	Message ID.
in	<i>pMessagePayload</i>	Pointer to the message payload.
in	<i>payloadSize</i>	Size of the payload (in bytes).
in	<i>maxChunkSize</i>	Maximum size of payload chunks.

Returns

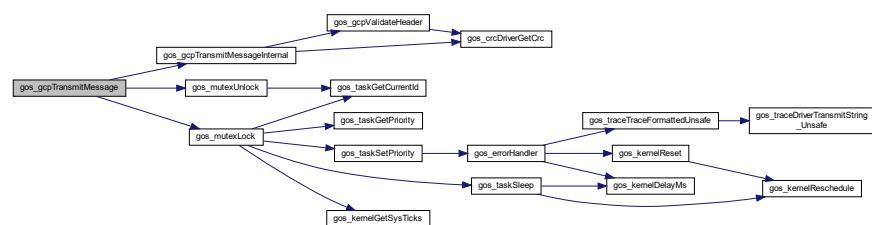
Result of message transmission.

Return values

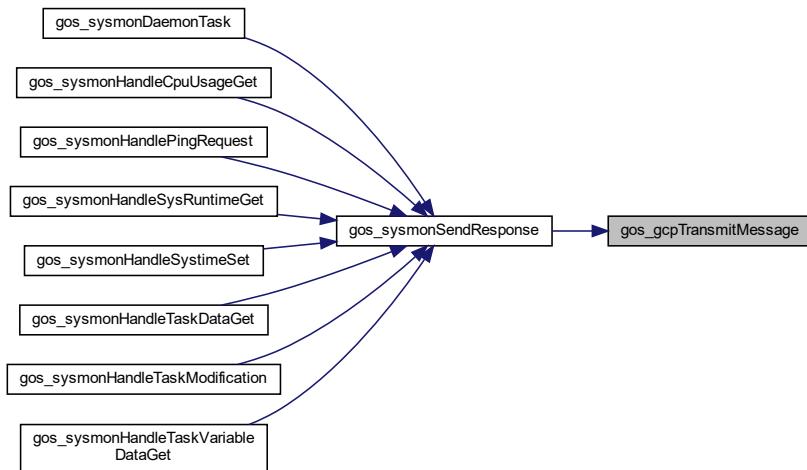
<i>GOS_SUCCESS</i>	Message transmitted successfully.
<i>GOS_ERROR</i>	An error occurred during transmission or validation.

Definition at line 225 of file gos_gcp.c.

Here is the call graph for this function:



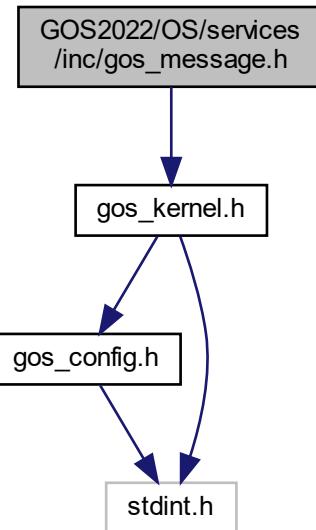
Here is the caller graph for this function:



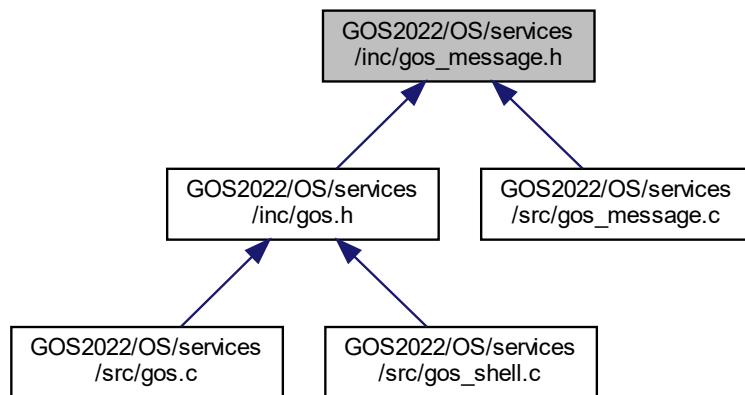
7.22 GOS2022/OS/services/inc/gos_message.h File Reference

GOS message service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_message.h:
```



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



Data Structures

- struct `gos_message_t`

Macros

- #define GOS_MESSAGE_ENDLESS_TMO (UINT16_MAX)
- #define GOS_MESSAGE_INVALID_ID (UINT16_MAX)

Typedefs

- typedef u16_t gos_messageId_t
 - Message size type.*
- typedef u16_t gos_messageTimeout_t
- typedef u8_t gos_messageSize_t
 - Message index type.*
- typedef u8_t gos_messageIndex_t
 - Message waiter index type.*
- typedef u8_t gos_messageWaiterIndex_t
 - Message ID index type.*
- typedef u8_t gos_messageIdIndex_t
 - Message ID index type.*

Functions

- gos_result_t gos_messageInit (void_t)
 - Initializes the message service.*
- gos_result_t gos_messageRx (gos_messageId_t *messageIdArray, gos_message_t *target, gos_messageTimeout_t tmo)
 - Receives the selected messages.*
- gos_result_t gos_messageTx (gos_message_t *message)
 - Transmits a message.*

7.22.1 Detailed Description

GOS message service header.

Author

Ahmed Gazar

Date

2022-11-15

Version

1.2

Message service is a way of inter-task communication provided by the operating system. With the use of messages, data can be passed between different tasks. The data sent through the message service is only limited in size which is a configuration parameter. Message passing is handled by the background daemon task thus ensuring that big data transfers do not block the system. Messages are first copied into an internal message array and whenever a waiter and a message is matched, it gets copied into the target buffer. If a message is not received, it will be stuck in the internal buffer and it causes the message service to halt if the internal circular buffer gets back to the position of the stuck message. Reception of messages can happen with a given timeout or with an endless timeout. Either way the caller task will go to blocked state until the message is received or the timeout elapses, so it will not be scheduled in the meantime. Each message contains a message ID. When receiving messages, the task can define a list of IDs for reception as a filter. This way one task can receive more than one message but a message can only be received by one task.

7.22.2 Macro Definition Documentation

7.22.2.1 GOS_MESSAGE_ENDLESS_TMO

```
#define GOS_MESSAGE_ENDLESS_TMO ( UINT16_MAX )
```

Endless timeout.

Definition at line 78 of file gos_message.h.

7.22.2.2 GOS_MESSAGE_INVALID_ID

```
#define GOS_MESSAGE_INVALID_ID ( UINT16_MAX )
```

Invalid message ID.

Definition at line 83 of file gos_message.h.

7.22.3 Typedef Documentation

7.22.3.1 gos_messageId_t

```
typedef u16_t gos_messageId_t
```

Message identifier type.

Definition at line 91 of file gos_message.h.

7.22.3.2 gos_messageTimeout_t

```
typedef u16_t gos_messageTimeout_t
```

Message timeout type.

Definition at line 96 of file gos_message.h.

7.22.4 Function Documentation

7.22.4.1 gos_messageInit()

```
gos_result_t gos_messageInit (
    void_t )
```

Initializes the message service.

Initializes the internal message and waiter arrays and registers the message daemon task.

Returns

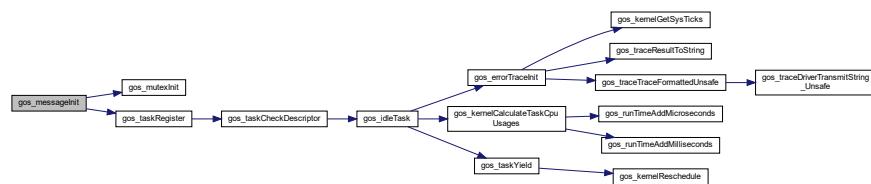
Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Initialization successful.
<i>GOS_ERROR</i>	Message daemon task registration failed.

Definition at line 150 of file gos_message.c.

Here is the call graph for this function:

7.22.4.2 `gos_messageRx()`

```

gos_result_t gos_messageRx (
    gos_messageId_t * messageIdArray,
    gos_message_t * target,
    gos_messageTimeout_t tmo )
  
```

Receives the selected messages.

Based on the selected message IDs, this function receives the first available message. Until the message is received, the caller task is put to blocked state. The caller will be unblocked if the message is received or the timeout elapsed.

Parameters

in	<i>messageIdArray</i>	Array of messages IDs the function should receive. Must be terminated by a 0 element!
out	<i>target</i>	Pointer to the target message structure. Received data will be placed here.
in	<i>tmo</i>	Timeout value in [ms]. The resolution of timeout is 10ms! Do not use endless timeout if it is not guaranteed that the message will be received!

Returns

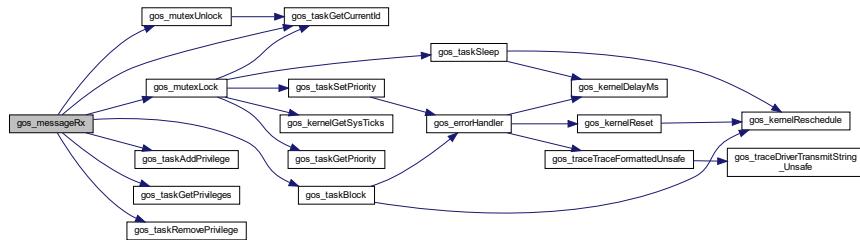
Result of message reception.

Return values

<i>GOS_SUCCESS</i>	Reception successful, data placed in the target structure.
<i>GOS_ERROR</i>	Reception failed because of invalid parameters or timeout.

Definition at line 193 of file gos_message.c.

Here is the call graph for this function:



7.22.4.3 gos_messageTx()

```
gos_result_t gos_messageTx (
    gos_message_t * message )
```

Transmits a message.

Copies the message in the internal message array for the message daemon to transfer it to the recipient task.

Parameters

in	<i>message</i>	Pointer to the message structure to be transmitted.
----	----------------	---

Returns

Result of message transmission.

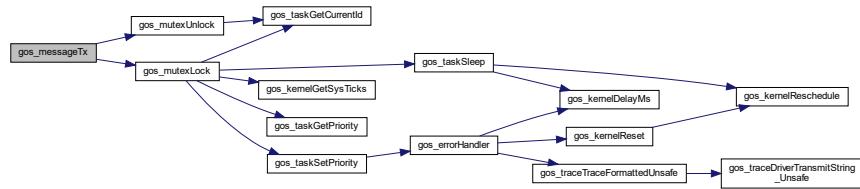
Return values

<i>GOS_SUCCESS</i>	Message transmission initiated successfully.
<i>GOS_ERROR</i>	Invalid message pointer or data or message array is full.

Function code.

Definition at line 306 of file gos_message.c.

Here is the call graph for this function:

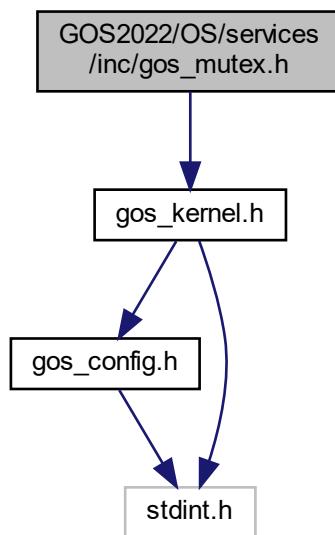


7.23 GOS2022/OS/services/inc/gos_mutex.h File Reference

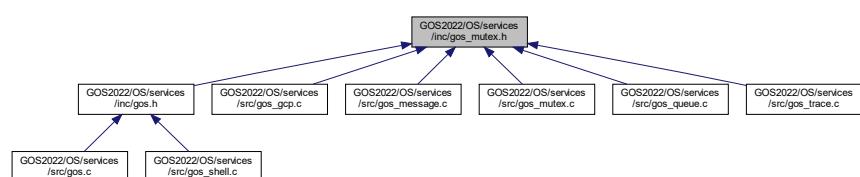
GOS mutex service header.

```
#include <gos_kernel.h>
```

Include dependency graph for gos_mutex.h:



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



Data Structures

- struct `gos_mutex_t`

Macros

- `#define GOS_MUTEX_ENDLESS_TMO (0xFFFFFFFFu)`
- `#define GOS_MUTEX_NO_TMO (0x00000000u)`

Enumerations

- enum `gos_mutexState_t` { `GOS_MUTEX_UNLOCKED` = 0b11010010, `GOS_MUTEX_LOCKED` = 0b01101011 }

Functions

- `gos_result_t gos_mutexInit (gos_mutex_t *pMutex)`
Initializes the mutex instance.
- `gos_result_t gos_mutexLock (gos_mutex_t *pMutex, u32_t timeout)`
Tries to lock the given mutex with the given timeout.
- `gos_result_t gos_mutexUnlock (gos_mutex_t *pMutex)`
Unlocks the mutex instance.

7.23.1 Detailed Description

GOS mutex service header.

Author

Ahmed Gazar

Date

2023-09-14

Version

1.1

Mutex (Mutual Exclusion) service is provided for protecting shared resources. A mutex has two states: locked or unlocked. When a task calls the lock function, the service checks the mutex state, and it locks the mutex if it is unlocked. If a mutex is locked by another task, it waits in a non-blocking way (yields) for the mutex to be unlocked.

7.23.2 Macro Definition Documentation

7.23.2.1 GOS_MUTEX_ENDLESS_TMO

```
#define GOS_MUTEX_ENDLESS_TMO ( 0xFFFFFFFFFu )
```

Mutex endless timeout.

Definition at line 66 of file gos_mutex.h.

7.23.2.2 GOS_MUTEX_NO_TMO

```
#define GOS_MUTEX_NO_TMO ( 0x00000000u )
```

Mutex no timeout.

Definition at line 71 of file gos_mutex.h.

7.23.3 Enumeration Type Documentation

7.23.3.1 gos_mutexState_t

```
enum gos_mutexState_t
```

Mutex state type.

Enumerator

GOS_MUTEX_UNLOCKED	Mutex unlocked.
GOS_MUTEX_LOCKED	Mutex locked.

Definition at line 79 of file gos_mutex.h.

7.23.4 Function Documentation

7.23.4.1 gos_mutexInit()

```
gos_result_t gos_mutexInit (
    gos_mutex_t * pMutex )
```

Initializes the mutex instance.

Sets the mutex state to unlocked.

Parameters

in, out	<i>pMutex</i>	Pointer to the mutex to be initialized.
---------	---------------	---

Returns

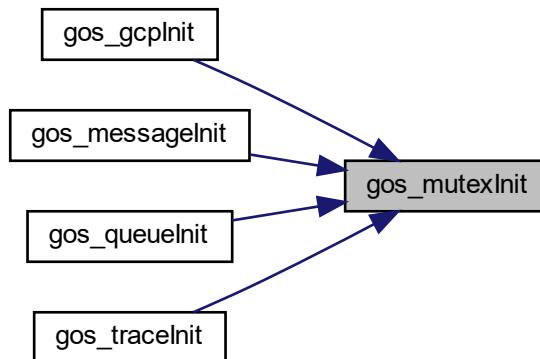
Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Mutex initialized successfully.
<i>GOS_ERROR</i>	Mutex pointer is NULL.

Definition at line 76 of file gos_mutex.c.

Here is the caller graph for this function:

**7.23.4.2 gos_mutexLock()**

```

gos_result_t gos_mutexLock (
    gos_mutex_t * pMutex,
    u32_t timeout )
  
```

Tries to lock the given mutex with the given timeout.

Waits in an unblocking way until the mutex is unlocked or the timeout value is reached. If the mutex becomes unlocked within the timeout value, it locks it.

Parameters

in, out	<i>pMutex</i>	Pointer to the mutex to be locked.
in	<i>timeout</i>	Timeout value.

Returns

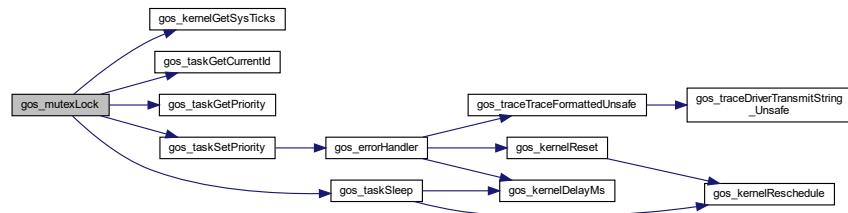
Result of mutex locking.

Return values

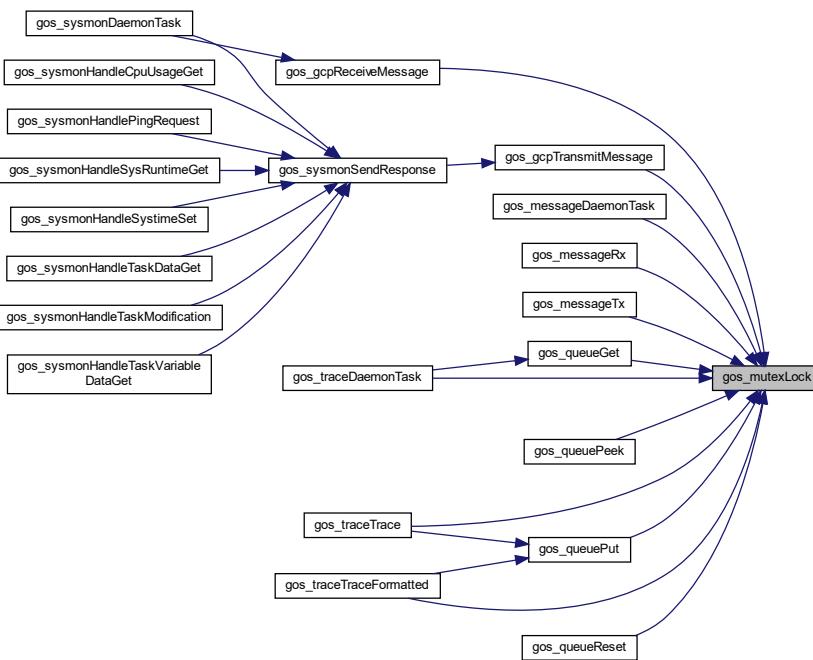
<i>GOS_SUCCESS</i>	Mutex locked successfully.
<i>GOS_ERROR</i>	Mutex could not be locked within the timeout value.

Definition at line 103 of file gos_mutex.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.23.4.3 gos_mutexUnlock()

```
gos_result_t gos_mutexUnlock (
    gos_mutex_t * pMutex )
```

Unlocks the mutex instance.

Sets the mutex state to unlocked.

Parameters

in, out	pMutex	Pointer to the mutex to be unlocked.
---------	--------	--------------------------------------

Returns

Result of mutex unlocking.

Return values

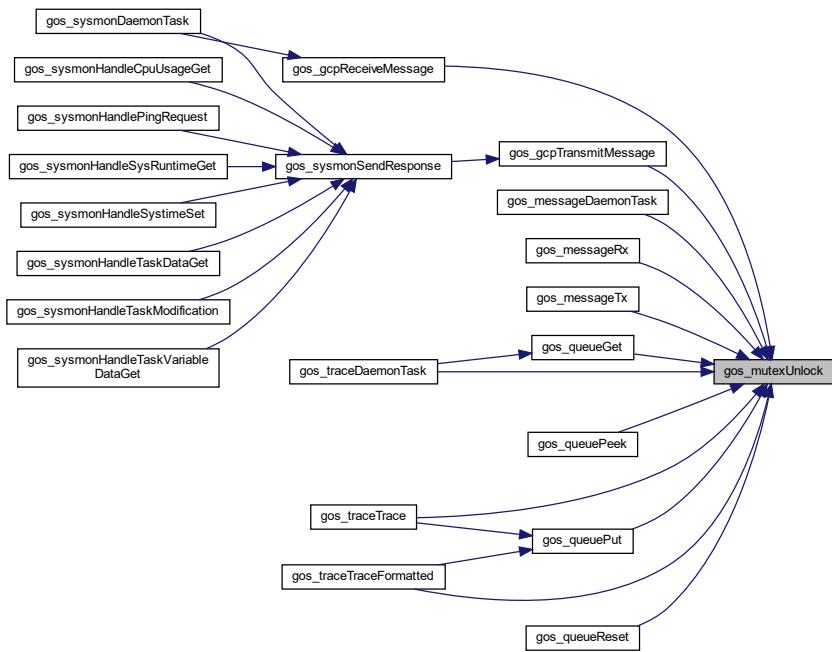
GOS_SUCCESS	Unlocking successful.
GOS_ERROR	Mutex is NULL or caller is not the owner of the mutex.

Definition at line 202 of file gos_mutex.c.

Here is the call graph for this function:



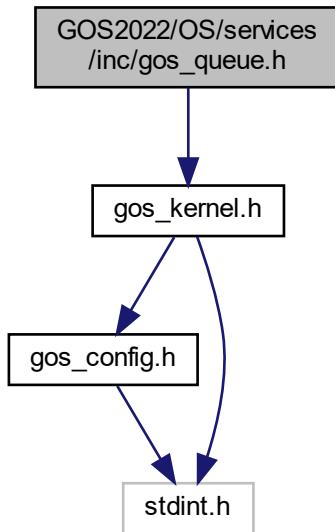
Here is the caller graph for this function:



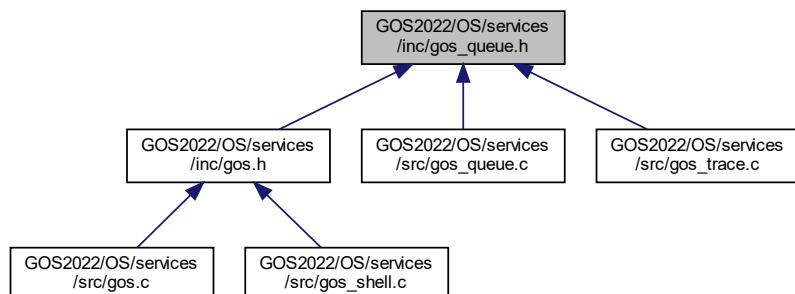
7.24 GOS2022/OS/services/inc/gos_queue.h File Reference

GOS queue service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_queue.h:
```



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



Data Structures

- struct [gos_queueDescriptor_t](#)

Macros

- `#define GOS_DEFAULT_QUEUE_ID ((gos_queueId_t) 0x3000)`
- `#define GOS_INVALID_QUEUE_ID ((gos_queueId_t) 0x0300)`

Typedefs

- `typedef u8_t gos_queueByte_t`
- `typedef u8_t gos_queueLength_t`
Queue length type.
- `typedef u8_t gos_queueIndex_t`
Queue index type.
- `typedef u16_t gos_queueId_t`
Queue ID type.
- `typedef void_t(* gos_queueFullHook) (gos_queueId_t)`
- `typedef void_t(* gos_queueEmptyHook) (gos_queueId_t)`

Functions

- `gos_result_t gos_queueInit (void_t)`
This function initializes the queue service.
- `gos_result_t gos_queueCreate (gos_queueDescriptor_t *pQueueDescriptor)`
This function creates a new queue.
- `gos_result_t gos_queuePut (gos_queueId_t queueId, void_t *element, gos_queueLength_t elementSize, u32_t timeout)`
This function puts an element in the given queue.
- `gos_result_t gos_queueGet (gos_queueId_t queueId, void_t *target, gos_queueLength_t targetSize, u32_t timeout)`
This function gets the next element from the given queue.
- `gos_result_t gos_queuePeek (gos_queueId_t queueId, void_t *target, gos_queueLength_t targetSize, u32_t timeout)`
This function gets the next element from the given queue without removing it.
- `gos_result_t gos_queueReset (gos_queueId_t queueId)`
Resets the given queue.
- `gos_result_t gos_queueRegisterFullHook (gos_queueFullHook fullHook)`
This function registers a queue full hook function.
- `gos_result_t gos_queueRegisterEmptyHook (gos_queueEmptyHook emptyHook)`
This function registers a queue empty hook function.
- `gos_result_t gos_queueGetName (gos_queueId_t queueId, gos_queueName_t queueName)`
This function gets the name of the given queue.
- `gos_result_t gos_queueGetElementNumber (gos_queueId_t queueId, gos_queueIndex_t *elementNumber)`
This function gets the number of elements in the given queue.
- `void_t gos_queueDump (void_t)`
Queue dump.

7.24.1 Detailed Description

GOS queue service header.

Author

Ahmed Gazar

Date

2024-04-02

Version

1.6

Queue service is one of the inter-task communication solutions offered by the OS. A queue can hold a given number of elements with a given element size. The queue is implemented as a FIFO. Queues are unsafe in a way that any task with the queue ID in their knowledge can put and get elements to/from the queue. So queues are most suitable for serializing data that are coming from multiple resources but processes by a single task. It is also possible to peek the queue, meaning that the next element of the queue can be requested without deleting it from the queue. This way multiple tasks can process data coming from a queue in a cooperative way, but they rely on each others data processing.

7.24.2 Macro Definition Documentation

7.24.2.1 GOS_DEFAULT_QUEUE_ID

```
#define GOS_DEFAULT_QUEUE_ID ( (gos_queueId_t) 0x3000 )
```

Default queue ID.

Definition at line 77 of file gos_queue.h.

7.24.2.2 GOS_INVALID_QUEUE_ID

```
#define GOS_INVALID_QUEUE_ID ( (gos_queueId_t) 0x0300 )
```

Invalid queue ID.

Definition at line 82 of file gos_queue.h.

7.24.3 Typedef Documentation

7.24.3.1 gos_queueByte_t

```
typedef u8_t gos_queueByte_t
```

One byte in a queue element.

Definition at line 90 of file gos_queue.h.

7.24.3.2 gos_queueEmptyHook

```
typedef void_t (* gos_queueEmptyHook) (gos_queueId_t)
```

Queue full hook type.

Definition at line 116 of file gos_queue.h.

7.24.3.3 gos_queueFullHook

```
typedef void_t (* gos_queueFullHook) (gos_queueId_t)
```

Queue full hook type.

Definition at line 111 of file gos_queue.h.

7.24.4 Function Documentation

7.24.4.1 gos_queueCreate()

```
gos_result_t gos_queueCreate (
    gos_queueDescriptor_t * pQueueDescriptor )
```

This function creates a new queue.

This function loops through the internal queue array and registers the new queue in the next free slot.

Parameters

in, out	<i>pQueueDescriptor</i>	Pointer to queue descriptor variable with queue data.
---------	-------------------------	---

Returns

Result of queue creation.

Return values

<i>GOS_SUCCESS</i>	Queue creation successful.
<i>GOS_ERROR</i>	Queue descriptor is NULL pointer or queue array is full.

Definition at line 181 of file gos_queue.c.

Here is the caller graph for this function:



7.24.4.2 gos_queueDump()

```
void_t gos_queueDump (
    void_t )
```

Queue dump.

Prints the queue data of all queues to the trace output.

Returns

-

Definition at line 598 of file gos_queue.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.24.4.3 gos_queueGet()

```
gos_result_t gos_queueGet (
    gos_queueId_t queueId,
    void_t * target,
    gos_queueLength_t targetSize,
    u32_t timeout )
```

This function gets the next element from the given queue.

This function checks the queue state and gets the next element from the queue.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>target</i>	Pointer to target variable.
in	<i>targetSize</i>	Size of target.
in	<i>timeout</i>	Timeout for locking queue mutex.

Returns

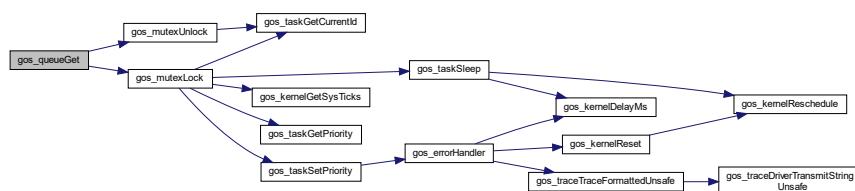
Result of element getting.

Return values

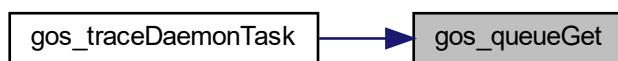
GOS_SUCCESS	Element successfully moved from queue to target.
GOS_ERROR	Invalid queue ID, invalid target size or queue is empty.

Definition at line 310 of file gos_queue.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.24.4.4 gos_queueGetElementNumber()

```
gos_result_t gos_queueGetElementNumber (
    gos_queueId_t queueId,
    gos_queueIndex_t * elementNumber )
```

This function gets the number of elements in the given queue.

This function copies the number of elements in the queue belonging to the given ID to the given variable.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>elementNumber</i>	Pointer to variable to store the element number in.

Returns

Result of queue element number getting.

Return values

<i>GOS_SUCCESS</i>	Element number getting successful.
<i>GOS_ERROR</i>	Invalid queue ID or element number variable is NULL.

Definition at line 564 of file gos_queue.c.

7.24.4.5 gos_queueGetName()

```
gos_result_t gos_queueGetName (
    gos_queueId_t queueId,
    gos_queueName_t queueName )
```

This function gets the name of the given queue.

This function copies the name of the queue belonging to the given ID to the given variable.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>queueName</i>	Queue name.

Returns

Result of queue name getting.

Return values

<i>GOS_SUCCESS</i>	Name getting successful.
<i>GOS_ERROR</i>	Invalid queue ID or queue name variable is NULL.

Definition at line 525 of file gos_queue.c.

7.24.4.6 gos_queueInit()

```
gos_result_t gos_queueInit (
    void_t )
```

This function initializes the queue service.

Initializes the internal queue array, creates the queue lock, and registers the queue dump task in the kernel.

Returns

Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Initialization successful.
<i>GOS_ERROR</i>	Lock creation, task registration or task suspension error.

Definition at line 147 of file gos_queue.c.

Here is the call graph for this function:



7.24.4.7 gos_queuePeek()

```
gos_result_t gos_queuePeek (
    gos_queueId_t queueId,
    void_t * target,
```

```
gos_queueLength_t targetSize,
u32_t timeout )
```

This function gets the next element from the given queue without removing it.

This function checks the queue state and returns the next element from the queue without modifying the queue counters.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>target</i>	Pointer to target variable.
in	<i>targetSize</i>	Size of target.
in	<i>timeout</i>	Timeout for locking queue mutex.

Returns

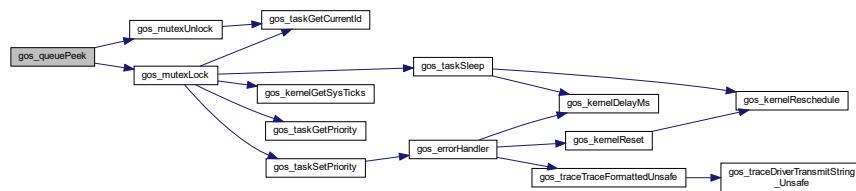
Result of element getting.

Return values

<i>GOS_SUCCESS</i>	Element successfully copied from queue to target.
<i>GOS_ERROR</i>	Invalid queue ID, invalid target size or queue is empty.

Definition at line 385 of file gos_queue.c.

Here is the call graph for this function:



7.24.4.8 gos_queuePut()

```
gos_result_t gos_queuePut (
    gos_queueId_t queueId,
    void_t * element,
    gos_queueLength_t elementSize,
    u32_t timeout )
```

This function puts an element in the given queue.

This function checks the queue state and places the given element in the next queue element.

Parameters

in	<i>queueId</i>	Queue ID.
in	<i>element</i>	Pointer to element.
in	<i>elementSize</i>	Size of element.
in	<i>timeout</i>	Timeout for locking queue mutex.

Returns

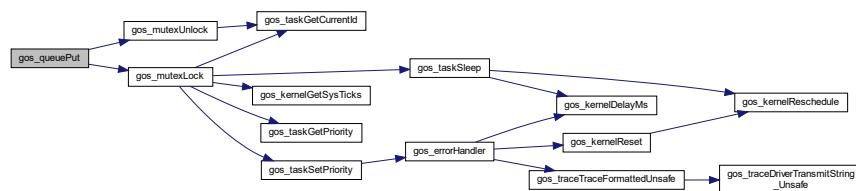
Result of element putting.

Return values

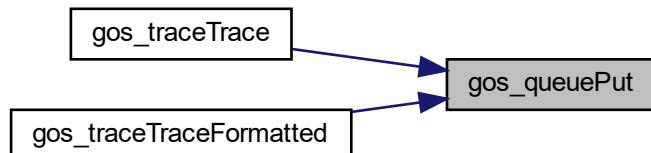
<i>GOS_SUCCESS</i>	Element successfully put in the queue.
<i>GOS_ERROR</i>	Invalid queue ID, invalid element size or queue is full.

Definition at line 231 of file gos_queue.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.24.4.9 gos_queueRegisterEmptyHook()

```
gos_result_t gos_queueRegisterEmptyHook (
    gos_queueEmptyHook emptyHook )
```

This function registers a queue empty hook function.

This function checks whether a hook has been already registered, and if not, it saves the given hook function.

Parameters

in	<i>emptyHook</i>	Hook function.
----	------------------	----------------

Returns

Result of hook registration.

Return values

<i>GOS_SUCCESS</i>	Hook registration successful.
<i>GOS_ERROR</i>	Hook already exists or parameter is NULL pointer.

Definition at line 499 of file gos_queue.c.

7.24.4.10 gos_queueRegisterFullHook()

```
gos_result_t gos_queueRegisterFullHook (
    gos_queueFullHook fullHook )
```

This function registers a queue full hook function.

This function checks whether a hook has been already registered, and if not, it saves the given hook function.

Parameters

in	<i>fullHook</i>	Hook function.
----	-----------------	----------------

Returns

Result of hook registration.

Return values

<i>GOS_SUCCESS</i>	Hook registration successful.
<i>GOS_ERROR</i>	Hook already exists or parameter is NULL pointer.

Definition at line 473 of file gos_queue.c.

7.24.4.11 gos_queueReset()

```
gos_result_t gos_queueReset (
    gos_queueId_t queueId )
```

Resets the given queue.

Sets the read and write counter to zero, making the queue empty.

Parameters

in	<i>queue</i> ↪ <i>Id</i>	Queue ID.
----	-----------------------------	-----------

Returns

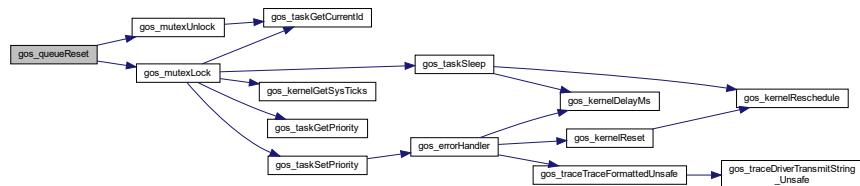
Result of queue resetting.

Return values

GOS_SUCCESS	Reset successful.
GOS_ERROR	Invalid queue ID.

Definition at line 435 of file gos_queue.c.

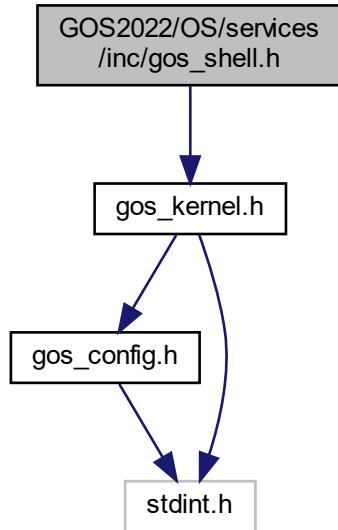
Here is the call graph for this function:



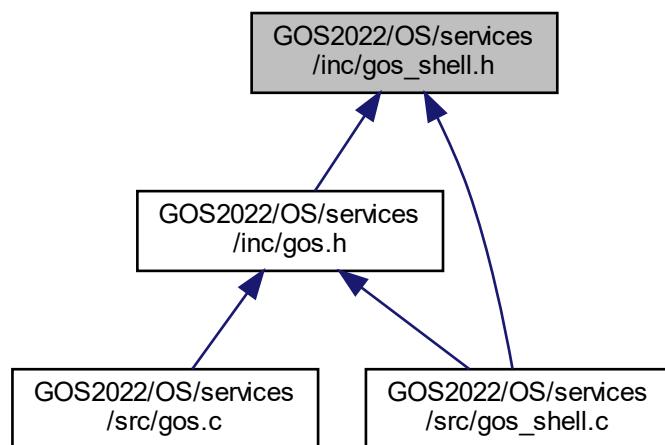
7.25 GOS2022/OS/services/inc/gos_shell.h File Reference

GOS shell service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_shell.h:
```



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



Data Structures

- struct [gos_shellCommand_t](#)

TypeDefs

- `typedef void_t(* gos_shellFunction) (char_t *params)`
- `typedef u8_t gos_shellCommandIndex_t`
Shell command index type.

Functions

- `gos_result_t gos_shellInit (void_t)`
This function initializes the shell service.
- `gos_result_t gos_shellRegisterCommands (gos_shellCommand_t *commands, u16_t arraySize)`
This function registers an array of commands in the shell.
- `gos_result_t gos_shellRegisterCommand (gos_shellCommand_t *command)`
This function registers a command in the shell.
- `gos_result_t gos_shellSuspend (void_t)`
Suspends the shell daemon task.
- `gos_result_t gos_shellResume (void_t)`
Resumes the shell daemon task.
- `gos_result_t gos_shellEchoOn (void_t)`
Turns the shell echoing on.
- `gos_result_t gos_shellEchoOff (void_t)`
Turns the shell echoing off.

7.25.1 Detailed Description

GOS shell service header.

Author

Ahmed Gazar

Date

2023-07-12

Version

1.3

The shell service provides an easy interface to receive and process commands in a terminal. Optionally the characters can be echoed back. The user can register different commands with callback functions. When the enter key is hit, the shell daemon processes the input. If it finds the input string in the command array, it calls the corresponding callback function and passes the parameter list as a string to the callback for further processing.

7.25.2 Typedef Documentation

7.25.2.1 gos_shellFunction

```
typedef void_t (* gos_shellFunction) (char_t *params)
```

Shell function type.

Definition at line 70 of file gos_shell.h.

7.25.3 Function Documentation

7.25.3.1 gos_shellEchoOff()

```
gos_result_t gos_shellEchoOff (
    void_t   )
```

Turns the shell echoing off.

Resets the internal echo flag.

Returns

Result of turning echoing off.

Return values

<code>GOS_SUCCESS</code>	Echoing turned off successfully.
--------------------------	----------------------------------

Definition at line 324 of file gos_shell.c.

7.25.3.2 gos_shellEchoOn()

```
gos_result_t gos_shellEchoOn (
    void_t   )
```

Turns the shell echoing on.

Sets the internal echo flag.

Returns

Result of turning echoing on.

Return values

GOS_SUCCESS	Echoing turned on successfully.
--------------------	---------------------------------

Definition at line 306 of file gos_shell.c.

7.25.3.3 gos_shellInit()

```
gos_result_t gos_shellInit (
    void_t )
```

This function initializes the shell service.

Initializes the internal command structure, and registers the shell daemon task in the kernel.

Returns

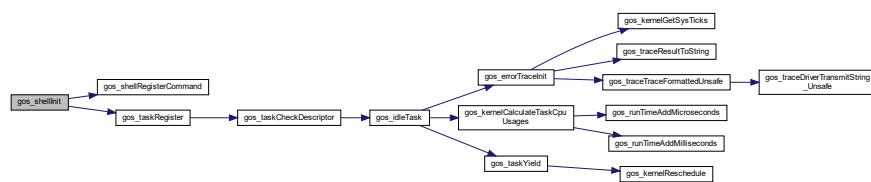
Result of initialization.

Return values

GOS_SUCCESS	Shell initialization successful.
GOS_ERROR	Shell daemon task registration failed.

Definition at line 153 of file gos_shell.c.

Here is the call graph for this function:



7.25.3.4 gos_shellRegisterCommand()

```
gos_result_t gos_shellRegisterCommand (
    gos_shellCommand_t * command )
```

This function registers a command in the shell.

Checks the command structure and registers the command in the next empty slot in the internal command array.

Parameters

in	<i>command</i>	Pointer to the command structure to register.
----	----------------	---

Returns

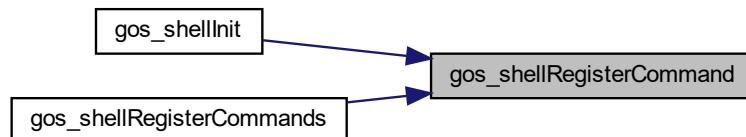
Result of shell command registration.

Return values

<i>GOS_SUCCESS</i>	Command registration successful.
<i>GOS_ERROR</i>	Command function or name is NULL or internal command array is full.

Definition at line 235 of file gos_shell.c.

Here is the caller graph for this function:

**7.25.3.5 gos_shellRegisterCommands()**

```
gos_result_t gos_shellRegisterCommands (
    gos_shellCommand_t * commands,
    u16_t arraySize )
```

This function registers an array of commands in the shell.

Checks the array pointer and registers the commands one by one.

Parameters

in	<i>commands</i>	Pointer to the command structure array to register.
in	<i>arraySize</i>	Size of the array in bytes.

Returns

Result of shell command registration.

Return values

GOS_SUCCESS	Command registration successful.
GOS_ERROR	Command function or name is NULL or internal command array is full.

Definition at line 187 of file gos_shell.c.

Here is the call graph for this function:



7.25.3.6 gos_shellResume()

```
gos_result_t gos_shellResume (
    void_t   )
```

Resumes the shell daemon task.

Resumes the shell daemon task.

Returns

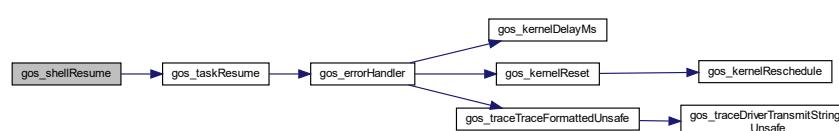
Result of shell resumption.

Return values

GOS_SUCCESS	Shell resumed successfully.
GOS_ERROR	Task resumption failed.

Definition at line 288 of file gos_shell.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.25.3.7 gos_shellSuspend()

```
gos_result_t gos_shellSuspend (
    void_t )
```

Suspends the shell daemon task.

Suspends the shell daemon task.

Returns

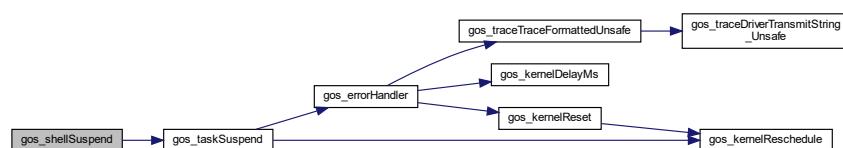
Result of shell suspension.

Return values

<i>GOS_SUCCESS</i>	Shell suspended successfully.
<i>GOS_ERROR</i>	Task suspension failed.

Definition at line 270 of file gos_shell.c.

Here is the call graph for this function:



Here is the caller graph for this function:

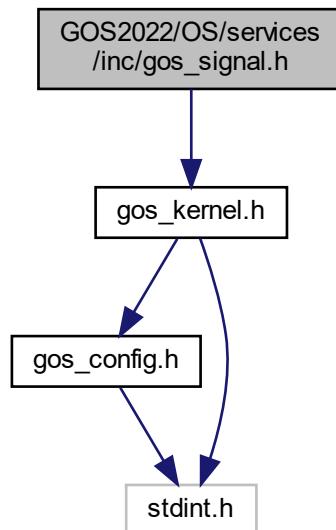


7.26 GOS2022/OS/services/inc/gos_signal.h File Reference

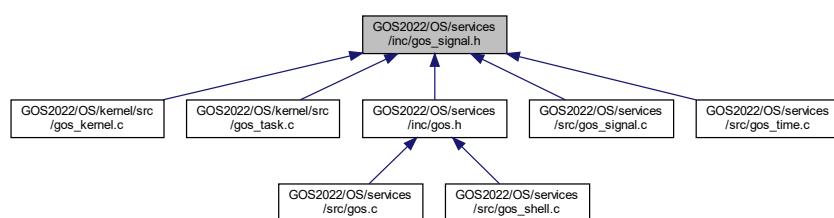
GOS signal service header.

```
#include <gos_kernel.h>
```

Include dependency graph for gos_signal.h:



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



Typedefs

- `typedef u8_t gos_signallId_t`
Signal ID.
- `typedef u16_t gos_signalSenderId_t`
Signal sender ID.
- `typedef u8_t gos_signallIndex_t`
Signal index for loops.
- `typedef u8_t gos_signalHandlerIndex_t`
Signal handler index type.
- `typedef void_t(* gos_signalHandler_t)(gos_signalSenderId_t)`
Signal handler function type.

Functions

- `gos_result_t gos_signallInit(void_t)`
Initializes the signal service.
- `gos_result_t gos_signalCreate(gos_signallId_t *pSignal)`
Creates a new signal.
- `gos_result_t gos_signalSubscribe(gos_signallId_t signallId, gos_signalHandler_t signalHandler, gos_taskPrivilegeLevel_t signalHandlerPrivileges)`
Subscribes to the given signal.
- `gos_result_t gos_signallInvoke(gos_signallId_t signallId, gos_signalSenderId_t senderId)`
Invokes the given signal.

7.26.1 Detailed Description

GOS signal service header.

Author

Ahmed Gazar

Date

2022-11-15

Version

1.1

Signal service is a way of inter-task or inter-process communication provided by the operating system. Signals can be created, subscribed to and invoked. When a signal is invoked, it is placed into an invoke queue and the signal daemon task will handle the invoke requests in the background. Thus signals are not instantly invoked. When a signal is invoked, all the subscribed functions get called. A signal can be invoked without any subscribers (in this case no function will be called).

7.26.2 Function Documentation

7.26.2.1 gos_signalCreate()

```
gos_result_t gos_signalCreate (
    gos_signalId_t * pSignal )
```

Creates a new signal.

Finds the next free slot in the signal array and registers the new signal there.

Parameters

out	<i>pSignal</i>	Pointer to a signal identifier.
-----	----------------	---------------------------------

Returns

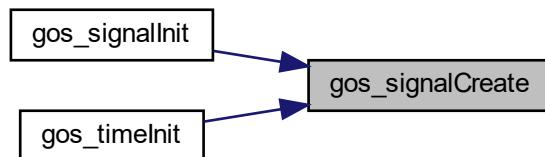
Success of signal creation.

Return values

<i>GOS_SUCCESS</i>	Creation successful.
<i>GOS_ERROR</i>	Signal array full.

Definition at line 183 of file gos_signal.c.

Here is the caller graph for this function:



7.26.2.2 gos_signallInit()

```
gos_result_t gos_signalInit (
    void_t )
```

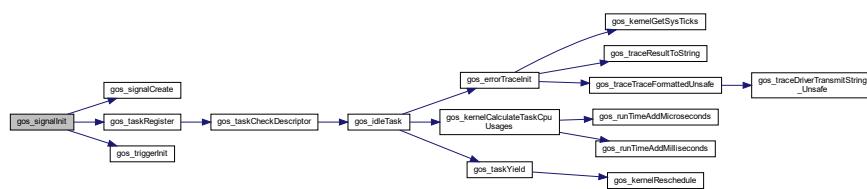
Initializes the signal service.

Initializes the internal signal array, creates a signal queue, registers the signal daemon task, creates the kernel dump signal, and subscribes the necessary components for the dump signal.

Returns

-
Definition at line 146 of file gos_signal.c.

Here is the call graph for this function:

**7.26.2.3 gos_signalInvoke()**

```
gos_result_t gos_signalInvoke (
    gos_signalId_t signalId,
    gos_signalSenderId_t senderId )
```

Invokes the given signal.

Places the given signal in the invoke queue (for the signal daemon to actually invoke the signal in the background).

Parameters

in	signalId	Signal identifier.
in	senderId	Sender identifier (or data to pass).

Returns

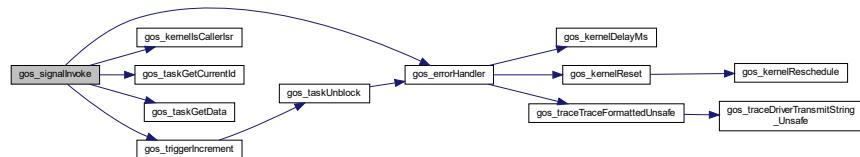
Success of signal invoking.

Return values

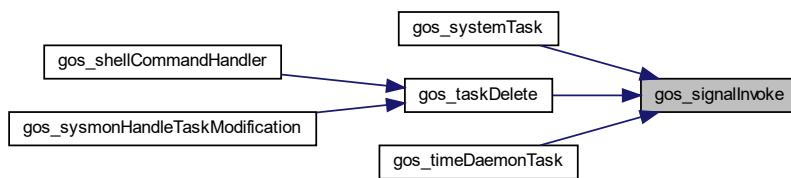
GOS_SUCCESS	Invoking successful.
GOS_ERROR	Invalid signal ID or signal unused.

Definition at line 260 of file gos_signal.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.26.2.4 gos_signalSubscribe()

```

gos_result_t gos_signalSubscribe (
    gos_signalId_t signalId,
    gos_signalHandler_t signalHandler,
    gos_taskPrivilegeLevel_t signalHandlerPrivileges )
  
```

Subscribes to the given signal.

Finds the next free slot in the signal handler array and registers the signal handler there.

Parameters

in	<i>signalId</i>	Signal identifier.
in	<i>signalHandler</i>	Signal handler function pointer.
in	<i>signalHandlerPrivileges</i>	Signal handler privilege level.

Returns

Success of signal subscription.

Return values

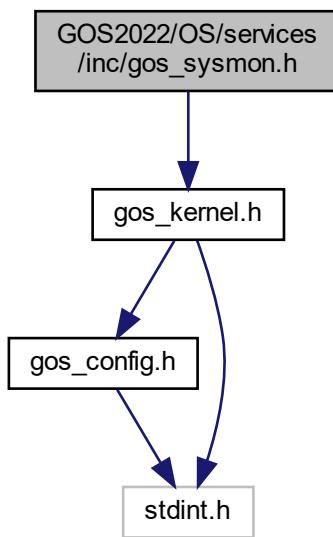
GOS_SUCCESS	Subscription successful.
GOS_ERROR	Invalid signal ID, signal handler NULL pointer, or handler array full.

Definition at line 217 of file gos_signal.c.

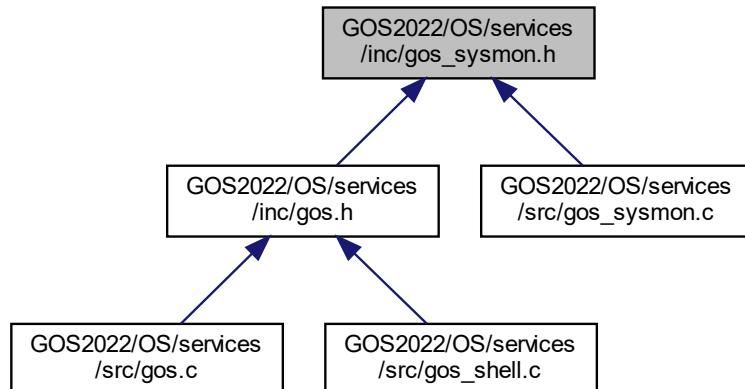
7.27 GOS2022/OS/services/inc/gos_sysmon.h File Reference

GOS system monitoring service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_sysmon.h:
```



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



Data Structures

- struct `gos_sysmonUserMessageDescriptor_t`

Typedefs

- typedef `void_t(* gos_sysmonMessageReceivedCallback) (void_t)`

Functions

- `gos_result_t gos_sysmonInit (void_t)`
This function initializes the system monitoring service.
- `gos_result_t gos_sysmonRegisterUserMessage (gos_sysmonUserMessageDescriptor_t *pDesc)`
This function registers a custom user sysmon message.

7.27.1 Detailed Description

GOS system monitoring service header.

Author

Ahmed Gazar

Date

2023-07-12

Version

1.1

This service is used to send and receive system information to an external client such as a PC tool.

7.27.2 Typedef Documentation

7.27.2.1 `gos_sysmonMessageReceivedCallback`

```
typedef void_t (* gos_sysmonMessageReceivedCallback) (void_t)
```

Sysmon message received callback function type.

Definition at line 65 of file gos_sysmon.h.

7.27.3 Function Documentation

7.27.3.1 gos_sysmonInit()

```
gos_result_t gos_sysmonInit (
    void_t
)
```

This function initializes the system monitoring service.

Registers the GCP physical driver and the sysmon daemon task in the kernel.

Returns

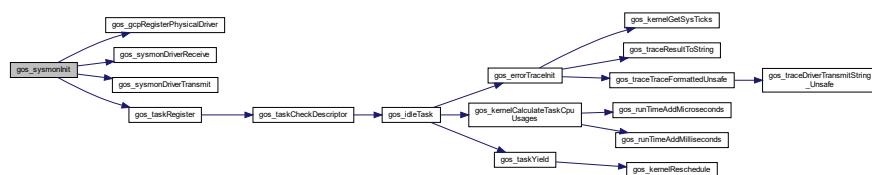
Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Sysmon initialization successful.
<i>GOS_ERROR</i>	Sysmon daemon task registration failed.

Definition at line 543 of file gos_sysmon.c.

Here is the call graph for this function:



7.27.3.2 gos_sysmonRegisterUserMessage()

```
gos_result_t gos_sysmonRegisterUserMessage (
    gos_sysmonUserMessageDescriptor_t * pDesc
)
```

This function registers a custom user sysmon message.

Registers a custom sysmon message given by its ID and PV. The message will only be processed if the required ID is not an existing sysmon ID. When a message is received with the given ID and PV, the message content will be copied into the buffer defined in the descriptor structure, and the registered callback function will be called.

Recommended ID range: 0x6000 ... 0x9999.

Parameters

in	<i>pDesc</i>	Pointer to a sysmon user message descriptor.
----	--------------	--

Returns

Result of registration.

Return values

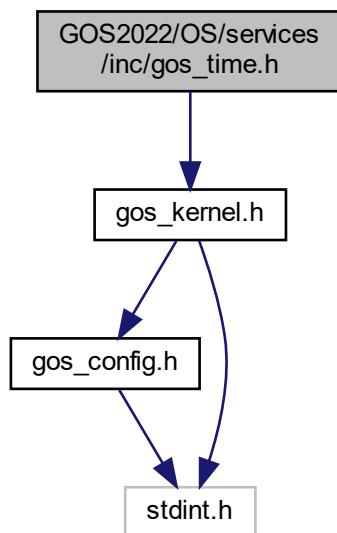
<i>GOS_SUCCESS</i>	User message registered successfully.
<i>GOS_ERROR</i>	Descriptor or callback is NULL or maximum number of user messages has been reached.

Definition at line 580 of file gos_sysmon.c.

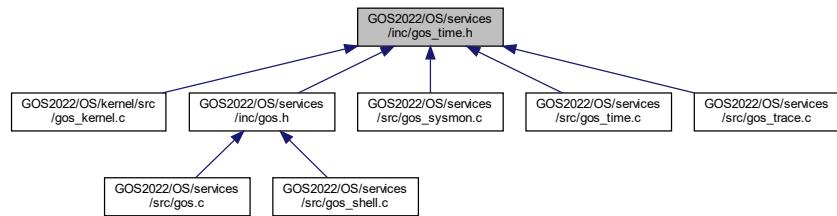
7.28 GOS2022/OS/services/inc/gos_time.h File Reference

GOS time service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_time.h:
```



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



Data Structures

- struct `gos_time_t`

Enumerations

- enum `gos_timeCompareResult_t` { `GOS_TIME_EARLIER`, `GOS_TIME_LATER`, `GOS_TIME_EQUAL` }
- enum `gos_timeMonthEnum_t` {
 `GOS_TIME_JANUARY` = 1, `GOS_TIME_FEBRUARY`, `GOS_TIME_MARCH`, `GOS_TIME_APRI`l,
 `GOS_TIME_MAY`, `GOS_TIME_JUNE`, `GOS_TIME_JULY`, `GOS_TIME_AUGUST`,
 `GOS_TIME_SEPTMBER`, `GOS_TIME_OCTOBER`, `GOS_TIME_NOVEMBER`, `GOS_TIME_DECEMBER`,
 `GOS_TIME_NUMBER_OF_MONTHS` = 12 }
- enum `gos_timeElapsedSenderId_t` {
 `GOS_TIME_SECOND_ELAPSED_SENDER_ID`, `GOS_TIME_MINUTE_ELAPSED_SENDER_ID`, `GOS_TIME_HOUR_ELAPS`
`GOS_TIME_DAY_ELAPSED_SENDER_ID`,
 `GOS_TIME_MONTH_ELAPSED_SENDER_ID`, `GOS_TIME_YEAR_ELAPSED_SENDER_ID` }

Functions

- `gos_result_t gos_timeInit (void_t)`

This function initializes the time service.
- `gos_result_t gos_timeGet (gos_time_t *pTime)`

This function gets the system time.
- `gos_result_t gos_timeSet (gos_time_t *pTime)`

This function sets the system time.
- `gos_result_t gos_timeCompare (gos_time_t *pTime1, gos_time_t *pTime2, gos_timeCompareResult_t *result)`

This function compares two time structures.
- `gos_result_t gos_timeIncreaseSystemTime (u16_t milliseconds)`

This function increases the system time and runtime with the given value of milliseconds.
- `gos_result_t gos_timeAddMilliseconds (gos_time_t *pTime, u16_t milliseconds)`

This function adds the given number of milliseconds to the given time structure.
- `gos_result_t gos_timeAddSeconds (gos_time_t *pTime, u16_t seconds)`

This function adds the given number of seconds to the given time variable.
- `gos_result_t gos_runTimeAddMicroseconds (gos_runtime_t *pRunTime1, gos_runtime_t *pRunTime2, u16_t microseconds)`

This function adds the given number of microseconds to the given time variables.
- `gos_result_t gos_runTimeAddMilliseconds (gos_runtime_t *pRunTime, u16_t milliseconds)`

This function adds the given number of milliseconds to the given time variables.

- `gos_result_t gos_runTimeAddSeconds (gos_runtime_t *pRunTime, u32_t seconds)`

This function adds the given number of seconds to the given run-time variable.

- `gos_result_t gos_runTimeGet (gos_runtime_t *pRunTime)`

This function gets the system run-time.

7.28.1 Detailed Description

GOS time service header.

Author

Ahmed Gazar

Date

2023-11-06

Version

1.6

Time service provides an easy interface to manipulate time structures, track the passage of time.

7.28.2 Enumeration Type Documentation

7.28.2.1 gos_timeComprareResult_t

```
enum gos_timeComprareResult_t
```

Time comparison result enumerator.

Enumerator

<code>GOS_TIME_EARLIER</code>	First time is earlier than second.
<code>GOS_TIME_LATER</code>	First time is later than second.
<code>GOS_TIME_EQUAL</code>	Times are equal.

Definition at line 90 of file gos_time.h.

7.28.2.2 gos_timeElapsedSenderId_t

```
enum gos_timeElapsedSenderId_t
```

Time elapsed sender IDs.

Enumerator

GOS_TIME_SECOND_ELAPSED_SENDER_ID	Second elapsed sender ID.
GOS_TIME_MINUTE_ELAPSED_SENDER_ID	Minute elapsed sender ID.
GOS_TIME_HOUR_ELAPSED_SENDER_ID	Hour elapsed sender ID.
GOS_TIME_DAY_ELAPSED_SENDER_ID	Day elapsed sender ID.
GOS_TIME_MONTH_ELAPSED_SENDER_ID	Month elapsed sender ID.
GOS_TIME_YEAR_ELAPSED_SENDER_ID	Year elapsed sender ID.

Definition at line 121 of file gos_time.h.

7.28.2.3 gos_timeMonthEnum_t

```
enum gos_timeMonthEnum_t
```

Month enumerator.

Enumerator

GOS_TIME_JANUARY	January.
GOS_TIME_FEBRUARY	February.
GOS_TIME_MARCH	March.
GOS_TIME_APRIIL	April.
GOS_TIME_MAY	May.
GOS_TIME_JUNE	June.
GOS_TIME_JULY	July.
GOS_TIME_AUGUST	August.
GOS_TIME_SEPTEMBER	September.
GOS_TIME_OCTOBER	October.
GOS_TIME_NOVEMBER	November.
GOS_TIME_DECEMBER	December.
GOS_TIME_NUMBER_OF_MONTHS	Number of months in a year.

Definition at line 100 of file gos_time.h.

7.28.3 Function Documentation

7.28.3.1 gos_runTimeAddMicroseconds()

```
gos_result_t gos_runTimeAddMicroseconds (
    gos_runtime_t * pRunTime1,
```

```
gos_runtime_t * pRunTime2,
u16_t microseconds )
```

This function adds the given number of microseconds to the given time variables.

This function adds the given number of microseconds to the given time variables.

Parameters

in, out	<i>pRunTime1</i>	Pointer to the time variable.
in, out	<i>pRunTime2</i>	Pointer to the time variable.
in	<i>microseconds</i>	Number of microseconds to add.

Returns

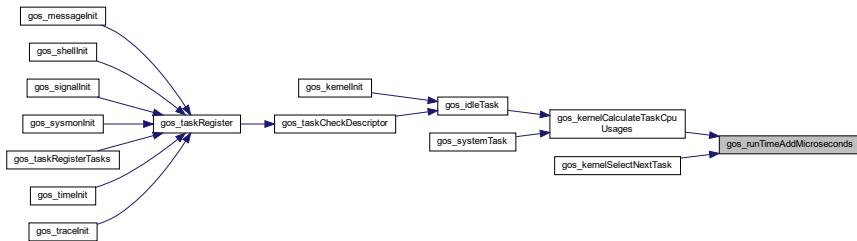
Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Microseconds added successfully.
<i>GOS_ERROR</i>	Time variable is NULL pointer.

Definition at line 498 of file gos_time.c.

Here is the caller graph for this function:



7.28.3.2 gos_runTimeAddMilliseconds()

```
gos_result_t gos_runTimeAddMilliseconds (
    gos_runtime_t * pRunTime,
    u16_t milliseconds )
```

This function adds the given number of milliseconds to the given time variables.

This function adds the given number of milliseconds to the given time variables.

Parameters

in, out	<i>pRunTime</i>	Pointer to the time variable.
in	<i>milliseconds</i>	Number of milliseconds to add.

Returns

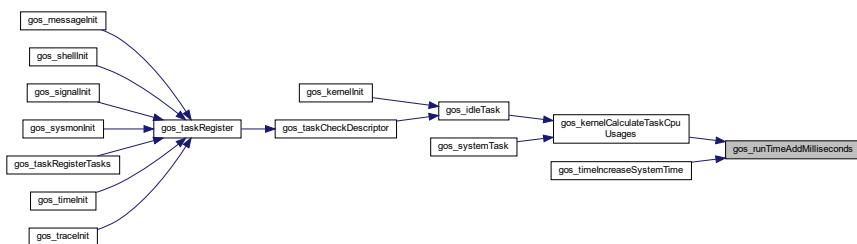
Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Milliseconds added successfully.
<i>GOS_ERROR</i>	Time variable is NULL pointer.

Definition at line 638 of file gos_time.c.

Here is the caller graph for this function:

**7.28.3.3 gos_runTimeAddSeconds()**

```
gos_result_t gos_runTimeAddSeconds (
    gos_runtime_t * pRunTime,
    u32_t seconds )
```

This function adds the given number of seconds to the given run-time variable.

This function adds the given number of seconds to the given run-time variable.

Parameters

in, out	<i>pRunTime</i>	Pointer to a run-time variable.
in	<i>seconds</i>	Number of seconds to add.

Returns

Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Seconds added successfully.
<i>GOS_ERROR</i>	Run-time variable is NULL pointer.

Definition at line 707 of file gos_time.c.

7.28.3.4 gos_runTimeGet()

```
gos_result_t gos_runTimeGet (
    gos_runtime_t * pRuntime )
```

This function gets the system run-time.

This function gets the system run-time.

Parameters

<i>out</i>	<i>pRuntime</i>	Pointer to a run-time variable to store the system run-time in.
------------	-----------------	---

Returns

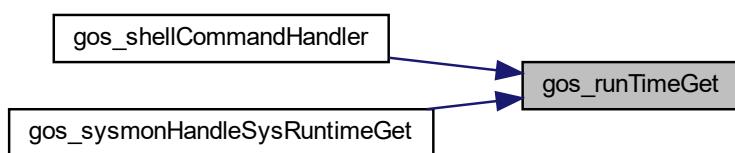
Result of run-time getting.

Return values

<i>GOS_SUCCESS</i>	Run-time getting is successful.
<i>GOS_ERROR</i>	Run-time variable is NULL pointer.

Definition at line 241 of file gos_time.c.

Here is the caller graph for this function:



7.28.3.5 gos_timeAddMilliseconds()

```
gos_result_t gos_timeAddMilliseconds (
    gos_time_t * pTime,
    u16_t milliseconds )
```

This function adds the given number of milliseconds to the given time structure.

This function adds the given number of milliseconds to the given time structure.

Parameters

in,out	<i>pTime</i>	Pointer to the time structure.
in	<i>milliseconds</i>	Number of milliseconds to add.

Returns

Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Increasing successful.
<i>GOS_ERROR</i>	Time structure is NULL pointer.

Definition at line 312 of file gos_time.c.

Here is the caller graph for this function:



7.28.3.6 gos_timeAddSeconds()

```
gos_result_t gos_timeAddSeconds (
    gos_time_t * pTime,
    u16_t seconds )
```

This function adds the given number of seconds to the given time variable.

This function adds the given number of seconds to the given time variable.

Parameters

in, out	<i>pTime</i>	Pointer to the time variable.
in	<i>seconds</i>	Number of seconds to add.

Returns

Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Seconds added successfully.
<i>GOS_ERROR</i>	Time variable is NULL pointer.

Definition at line 408 of file gos_time.c.

7.28.3.7 gos_timeCompare()

```
gos_result_t gos_timeCompare (
    gos_time_t * pTime1,
    gos_time_t * pTime2,
    gos_timeCompareResult_t * result )
```

This function compares two time structures.

This function compares two time structures.

Parameters

in	<i>pTime1</i>	Pointer to the first time variable.
in	<i>pTime2</i>	Pointer to the second time variable.
out	<i>result</i>	Pointer to the comparison result variable.

Returns

Result of time comparison.

Return values

<i>GOS_SUCCESS</i>	Time comparison successful.
<i>GOS_ERROR</i>	Either time structure and/or result variable is NULL pointer.

Definition at line 268 of file gos_time.c.

7.28.3.8 gos_timeGet()

```
gos_result_t gos_timeGet (
    gos_time_t * pTime )
```

This function gets the system time.

This function copies the system time value to the given time variable.

Parameters

out	<i>pTime</i>	Pointer to a time variable to store the system time value in.
-----	--------------	---

Returns

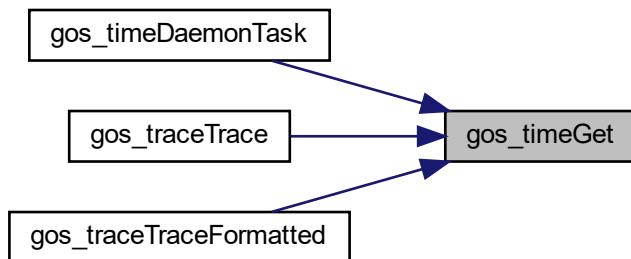
Result of time getting.

Return values

<i>GOS_SUCCESS</i>	Time getting successful.
<i>GOS_ERROR</i>	Time variable is NULL pointer.

Definition at line 187 of file gos_time.c.

Here is the caller graph for this function:



7.28.3.9 gos_timeIncreaseSystemTime()

```
gos_result_t gos_timeIncreaseSystemTime (
    u16_t milliseconds )
```

This function increases the system time and runtime with the given value of milliseconds.

This function increases the system time and runtime with the given value of milliseconds.

Parameters

in	<i>milliseconds</i>	Number of milliseconds to add to system time.
----	---------------------	---

Returns

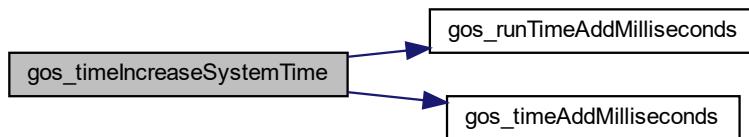
Result of system time increasing.

Return values

<i>GOS_SUCCESS</i>	Increasing successful.
<i>GOS_ERROR</i>	System time or runtime increasing failed.

Definition at line 768 of file gos_time.c.

Here is the call graph for this function:

**7.28.3.10 gos_timeInit()**

```
gos_result_t gos_timeInit (
    void_t )
```

This function initializes the time service.

Creates the time signal and registers the time daemon in the kernel.

Returns

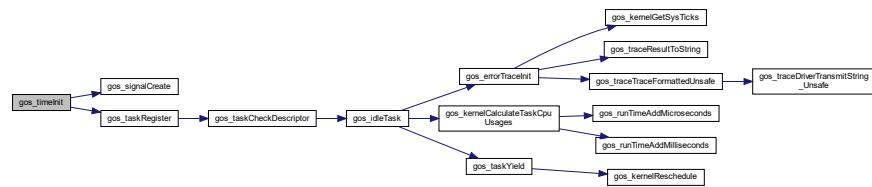
Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Initialization successful.
<i>GOS_ERROR</i>	Time signal registration error or time daemon registration error.

Definition at line 160 of file gos_time.c.

Here is the call graph for this function:



7.28.3.11 gos_timeSet()

```
gos_result_t gos_timeSet (
    gos_time_t * pTime )
```

This function sets the system time.

This function copies the time value from the given time variable to the system time variable.

Parameters

in	<i>pTime</i>	Pointer to a time variable holding the desired time value.
----	--------------	--

Returns

Result of time setting.

Return values

<i>GOS_SUCCESS</i>	Time setting successful.
<i>GOS_ERROR</i>	Time structure is NULL pointer.

Definition at line 214 of file gos_time.c.

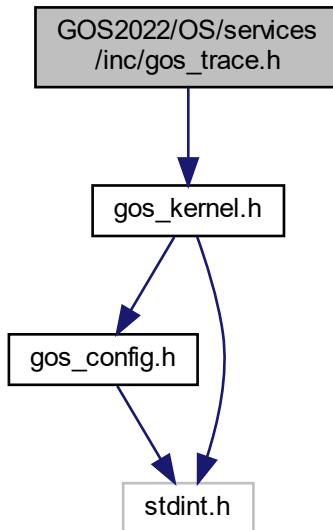
Here is the caller graph for this function:



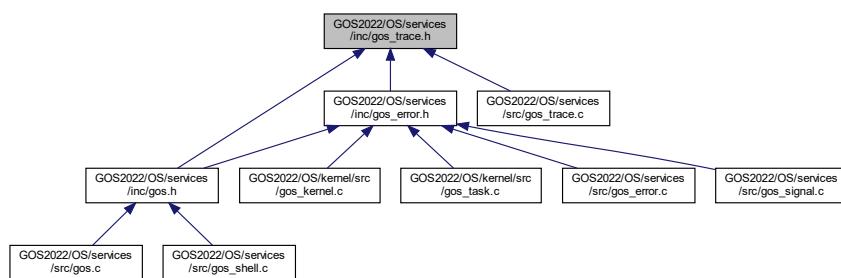
7.29 GOS2022/OS/services/inc/gos_trace.h File Reference

GOS trace service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_trace.h:
```



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



Macros

- `#define TRACE_FORMAT_RESET "\x1B[0m"`
Reset formatting.
- `#define TRACE_FG_RED_START "\x1B[31m"`
Red foreground start.
- `#define TRACE_FG_GREEN_START "\x1B[32m"`

- `#define TRACE_FG_YELLOW_START "\x1B[33m"`
Yellow foreground start.
- `#define TRACE_FG_BLUE_START "\x1B[34m"`
Blue foreground start.
- `#define TRACE_FG_MAGENTA_START "\x1B[35m"`
Magenta foreground start.
- `#define TRACE_FG_CYAN_START "\x1B[36m"`
Cyan foreground start.
- `#define TRACE_FG_WHITE_START "\x1B[37m"`
White foreground start.
- `#define TRACE_BG_RED_START "\x1B[41m"`
Red background start.
- `#define TRACE_BG_GREEN_START "\x1B[42m"`
Green background start.
- `#define TRACE_BG_YELLOW_START "\x1B[43m"`
Yellow background start.
- `#define TRACE_BG_BLUE_START "\x1B[44m"`
Blue background start.
- `#define TRACE_BG_MAGENTA_START "\x1B[45m"`
Magenta background start.
- `#define TRACE_BG_CYAN_START "\x1B[46m"`
Cyan background start.
- `#define TRACE_BG_WHITE_START "\x1B[47m"`
White background start.
- `#define TRACE_BOLD_START "\x1B[1m"`
Bold start.
- `#define TRACE_ITALIC_START "\x1B[3m"`
Italic start.
- `#define TRACE_UNDERLINE_START "\x1B[4m"`
Underline start.
- `#define TRACE_STRIKETHROUGH_START "\x1B[9m"`
Strikethrough start.

Functions

- `gos_result_t gos_traceInit (void_t)`
Initializes the trace service.
- `gos_result_t gos_traceTrace (bool_t addTimeStamp, char_t *traceMessage)`
Traces a given message.
- `gos_result_t gos_traceTraceFormatted (bool_t addTimeStamp, GOS_CONST char_t *traceFormat,...)`
Traces a given formatted message.
- `gos_result_t gos_traceTraceFormattedUnsafe (GOS_CONST char_t *traceFormat,...)`
Traces a given formatted message.

7.29.1 Detailed Description

GOS trace service header.

Author

Ahmed Gazar

Date

2023-01-13

Version

2.1

Trace service is a simple interface to send out strings via the configured trace periphery.

7.29.2 Function Documentation

7.29.2.1 gos_traceInit()

```
gos_result_t gos_traceInit (
    void_t    )
```

Initializes the trace service.

Creates a trace queue and registers the trace daemon in the kernel.

Returns

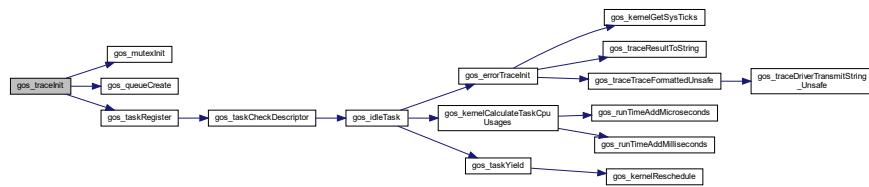
Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Initialization successful.
<i>GOS_ERROR</i>	Queue creation or task registration error.

Definition at line 152 of file gos_trace.c.

Here is the call graph for this function:



7.29.2.2 gos_traceTrace()

```
gos_result_t gos_traceTrace (
    bool_t addTimeStamp,
    char_t * traceMessage )
```

Traces a given message.

Places the given message to the trace queue (for the trace daemon to print it).

Parameters

in	<i>addTimeStamp</i>	Flag to indicate whether to add time-stamp or not.
in	<i>traceMessage</i>	String to trace.

Returns

Result of tracing.

Return values

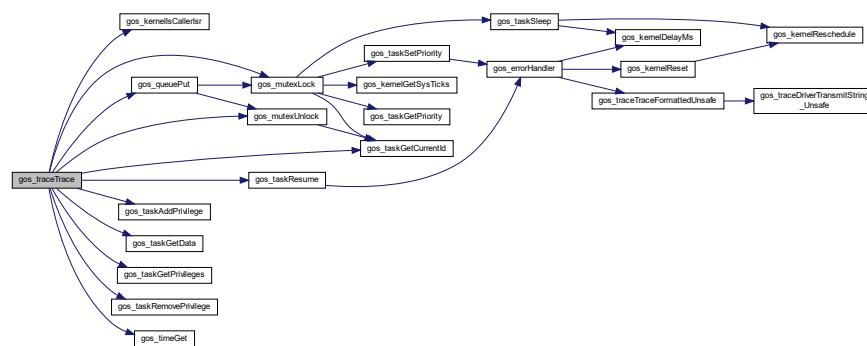
<i>GOS_SUCCESS</i>	Tracing successful.
<i>GOS_ERROR</i>	Queue put error.

Remarks

This function uses the queue service.

Definition at line 177 of file gos_trace.c.

Here is the call graph for this function:



7.29.2.3 gos_traceTraceFormatted()

```
gos_result_t gos_traceTraceFormatted (
    bool_t addTimeStamp,
    GOS_CONST char_t * traceFormat,
    ...
)
```

Traces a given formatted message.

Prints the formatted message into a local buffer and places it to the trace queue (for the trace daemon to print it).

Parameters

in	<i>addTimeStamp</i>	Flag to indicate whether to add time-stamp or not.
in	<i>traceFormat</i>	Formatter string.
in	...	Optional parameters.

Returns

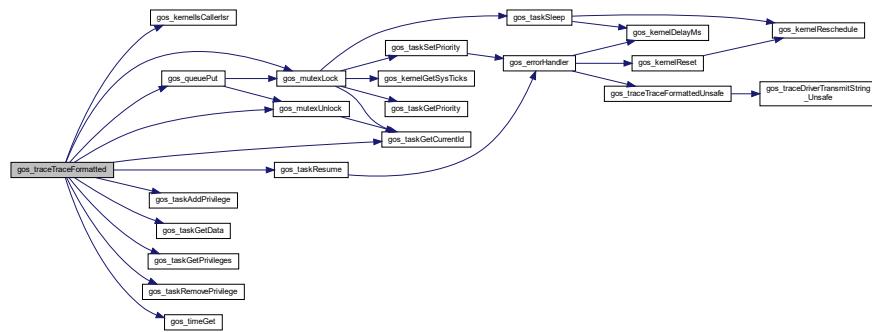
Result of formatted tracing.

Return values

<i>GOS_SUCCESS</i>	Formatted tracing successful.
<i>GOS_ERROR</i>	Queue put error.

Definition at line 266 of file gos_trace.c.

Here is the call graph for this function:



7.29.2.4 gos_traceTraceFormattedUnsafe()

```
gos_result_t gos_traceTraceFormattedUnsafe (
```

Traces a given formatted message.

Prints the formatted message into a local buffer and transmits it via the trace port.

Parameters

in	<i>traceFormat</i>	Formatter string.
in	...	Optional parameters.

Returns

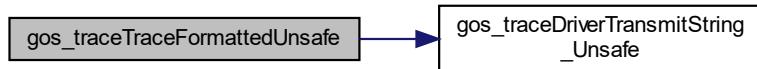
Result of formatted tracing.

Return values

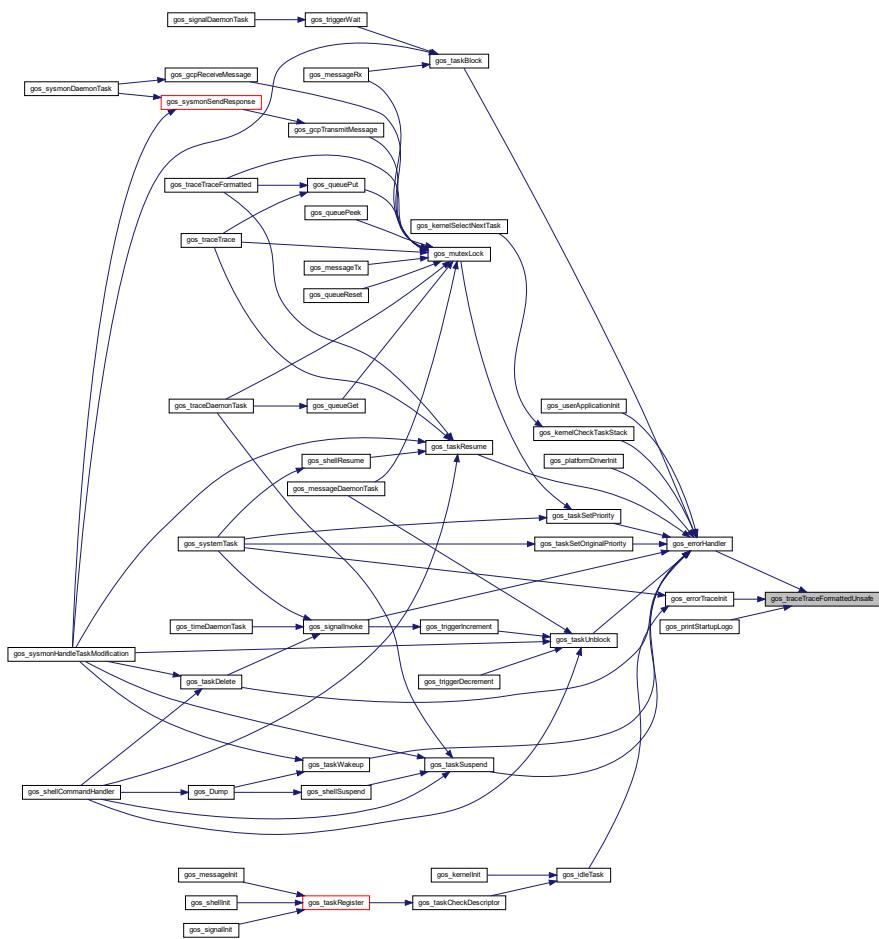
<i>GOS_SUCCESS</i>	Message traced successfully.
<i>GOS_ERROR</i>	Transmit error.

Definition at line 356 of file qos_trace.c.

Here is the call graph for this function:



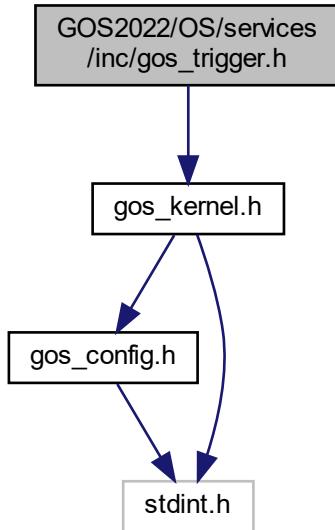
Here is the caller graph for this function:



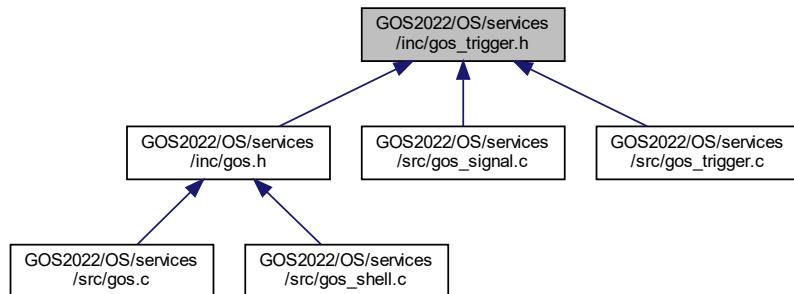
7.30 GOS2022/OS/services/inc/gos_trigger.h File Reference

GOS trigger service header.

```
#include <gos_kernel.h>
Include dependency graph for gos_trigger.h:
```



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



Data Structures

- struct [gos_trigger_t](#)

Macros

- `#define GOS_TRIGGER_ENDLESS_TMO (0xFFFFFFFF)`
- `#define GOS_TRIGGER_NO_TMO (0x00000000)`

Functions

- `gos_result_t gos_triggerInit (gos_trigger_t *pTrigger)`
Initializes the trigger instance.
- `gos_result_t gos_triggerReset (gos_trigger_t *pTrigger)`
Resets the trigger counter of the given trigger instance.
- `gos_result_t gos_triggerWait (gos_trigger_t *pTrigger, u32_t value, u32_t timeout)`
Waits for the trigger instance to reach the given trigger value.
- `gos_result_t gos_triggerIncrement (gos_trigger_t *pTrigger)`
Increments the trigger value of the given trigger.
- `gos_result_t gos_triggerDecrement (gos_trigger_t *pTrigger)`
Decrements the trigger value of the given trigger.

7.30.1 Detailed Description

GOS trigger service header.

Author

Ahmed Gazar

Date

2023-11-15

Version

2.4

Trigger service is a way of synchronizing tasks. A trigger instance works as a counter. A trigger can be incremented, and it can be waited on. When waiting for a trigger, the caller must specify a desired trigger value to be reached. Until the value is reached, the caller will wait in a non-blocking way. Once the value is reached, the caller returns.

7.30.2 Macro Definition Documentation

7.30.2.1 GOS_TRIGGER_ENDLESS_TMO

```
#define GOS_TRIGGER_ENDLESS_TMO ( 0xFFFFFFFF )
```

Mutex endless timeout.

Definition at line 72 of file gos_trigger.h.

7.30.2.2 GOS_TRIGGER_NO_TMO

```
#define GOS_TRIGGER_NO_TMO ( 0x00000000 )
```

Mutex no timeout.

Definition at line 77 of file gos_trigger.h.

7.30.3 Function Documentation

7.30.3.1 gos_triggerDecrement()

```
gos_result_t gos_triggerDecrement (
    gos_trigger_t * pTrigger )
```

Decrements the trigger value of the given trigger.

Decrements the trigger value of the given trigger.

Parameters

in, out	<i>pTrigger</i>	Pointer to the trigger instance.
---------	-----------------	----------------------------------

Returns

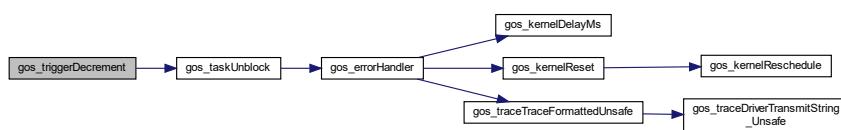
Result of trigger decrementing.

Return values

<i>GOS_SUCCESS</i>	Decrementing successful.
<i>GOS_ERROR</i>	Trigger is NULL pointer or value is already zero.

Definition at line 246 of file gos_trigger.c.

Here is the call graph for this function:



7.30.3.2 gos_triggerIncrement()

```
gos_result_t gos_triggerIncrement (
    gos_trigger_t * pTrigger )
```

Increments the trigger value of the given trigger.

Increments the trigger value of the given trigger.

Parameters

in, out	<i>pTrigger</i>	Pointer to the trigger instance.
---------	-----------------	----------------------------------

Returns

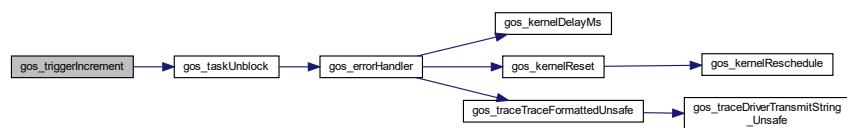
Result of trigger incrementing.

Return values

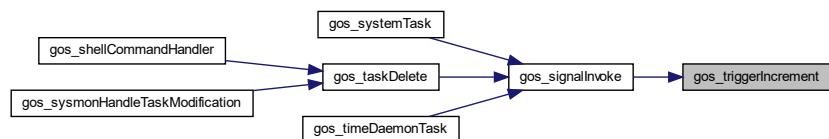
<i>GOS_SUCCESS</i>	Incrementing successful.
<i>GOS_ERROR</i>	Trigger is NULL pointer.

Definition at line 199 of file gos_trigger.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.30.3.3 gos_triggerInit()

```
gos_result_t gos_triggerInit (
    gos_trigger_t * pTrigger )
```

Initializes the trigger instance.

Calls the initializer for the trigger mutex.

Parameters

out	<i>pTrigger</i>	Pointer to the trigger to be initialized.
-----	-----------------	---

Returns

Result of trigger initializing.

Return values

<i>GOS_SUCCESS</i>	Trigger initialized successfully.
<i>GOS_ERROR</i>	Trigger descriptor or trigger mutex is NULL pointer.

Definition at line 78 of file gos_trigger.c.

Here is the caller graph for this function:



7.30.3.4 gos_triggerReset()

```
gos_result_t gos_triggerReset (
    gos_trigger_t * pTrigger )
```

Resets the trigger counter of the given trigger instance.

Sets the trigger counter of the given trigger instance to zero.

Parameters

out	<i>pTrigger</i>	Pointer to the trigger instance.
-----	-----------------	----------------------------------

Returns

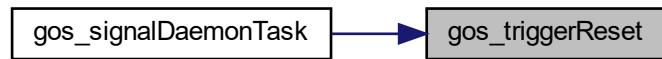
Result of trigger resetting.

Return values

<i>GOS_SUCCESS</i>	Trigger resetting successful.
<i>GOS_ERROR</i>	Trigger is NULL pointer.

Definition at line 107 of file gos_trigger.c.

Here is the caller graph for this function:

**7.30.3.5 gos_triggerWait()**

```
gos_result_t gos_triggerWait (
    gos_trigger_t * pTrigger,
    u32_t value,
    u32_t timeout )
```

Waits for the trigger instance to reach the given trigger value.

Increases the trigger waiter number, and waits in a non-blocking way until the desired trigger value is reached or the timeout is reached.

Parameters

in, out	<i>pTrigger</i>	Pointer to the trigger instance.
in	<i>value</i>	The desired trigger value.
in	<i>timeout</i>	Timeout value.

Returns

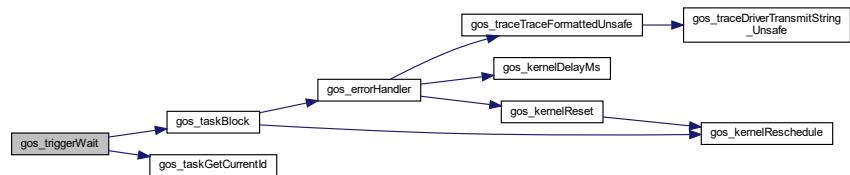
Result of trigger waiting.

Return values

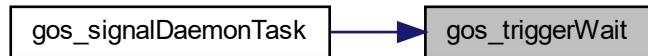
<i>GOS_SUCCESS</i>	Trigger value reached.
<i>GOS_ERROR</i>	Trigger value was not reached within the timeout value.

Definition at line 140 of file gos_trigger.c.

Here is the call graph for this function:



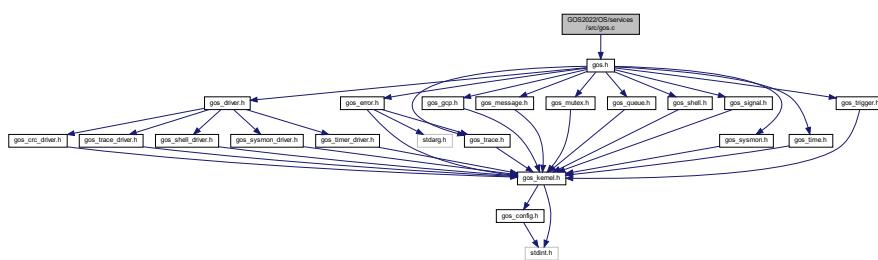
Here is the caller graph for this function:



7.31 GOS2022/OS/services/src/gos.c File Reference

GOS source.

```
#include <gos.h>
Include dependency graph for gos.c:
```



Data Structures

- struct gos initStruct t

Macros

- #define GOS SYS TASK SLEEP TIME (100u)

Typedefs

- `typedef gos_result_t(* gos_initFunc_t) (void_t)`

Functions

- `GOS_STATIC void_t gos_systemTask (void_t)`
Initializes the system.
- `GOS_STATIC gos_result_t gos_Start (void_t)`
Starts the OS.
- `int main (void_t)`
- `void_t gos_Dump (void_t)`
GOS dump.
- `gos_result_t gos_platformDriverInit (void_t)`
Platform driver initializer. Used for the platform-specific driver initializations.
- `gos_result_t gos_userApplicationInit (void_t)`
User application initializer. Used for the application-related initializations.

Variables

- `GOS_EXTERN gos_signallt kernelDumpReadySignal`
- `GOS_STATIC bool_t initError`
- `GOS_STATIC bool_t dumpRequired`
- `GOS_STATIC gos_initStruct_t initializers []`
- `GOS_STATIC gos_tid_t systemTaskId`
- `GOS_STATIC gos_taskDescriptor_t systemTaskDesc`

7.31.1 Detailed Description

GOS source.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.10

For a more detailed description of this service, please refer to [gos.h](#)

7.31.2 Macro Definition Documentation

7.31.2.1 GOS_SYS_TASK_SLEEP_TIME

```
#define GOS_SYS_TASK_SLEEP_TIME ( 100u )
```

System task sleep time.

Definition at line 76 of file gos.c.

7.31.3 Typedef Documentation

7.31.3.1 gos_initFunc_t

```
typedef gos_result_t(* gos_initFunc_t) (void_t)
```

Initializer function type.

Definition at line 84 of file gos.c.

7.31.4 Function Documentation

7.31.4.1 gos_Dump()

```
void_t gos_Dump (
    void_t   )
```

GOS dump.

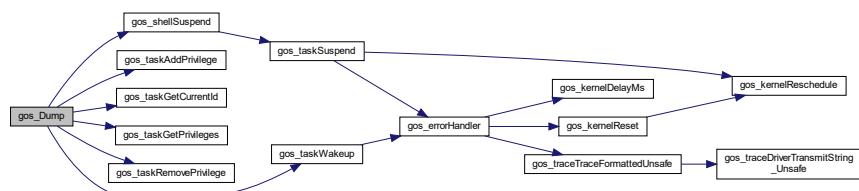
Sets the dump required flag and wakes up the system task that calls the individual dump functions.

Returns

-

Definition at line 207 of file gos.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.31.4.2 gos_Start()

```
GOS_STATIC gos_result_t gos_Start (
    void_t )
```

Starts the OS.

Checks whether the initializer function has set the error flag to GOS_FALSE, and if so, it starts the kernel (and thus the scheduling of tasks).

Returns

Result of OS starting.

Return values

<code>GOS_ERROR</code>	OS not started due to initialization error or kernel start error.
------------------------	---

Remarks

This function should only return with error. If the initialization is successful, the function is not expected to return.

Definition at line 273 of file gos.c.

Here is the call graph for this function:



7.31.4.3 gos_systemTask()

```
GOS_STATIC void_t gos_systemTask (
```

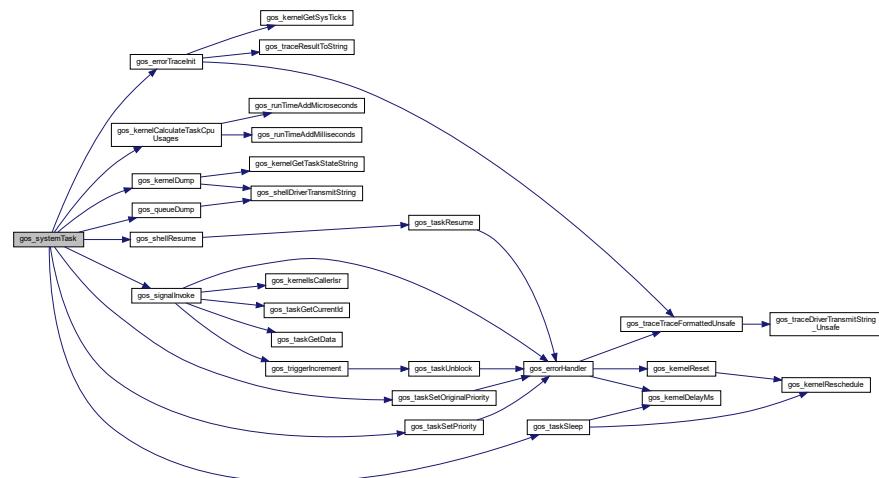
Initializes the system.

Calls the OS service initializer functions and the user application initializer, and deletes itself.

Returns

Definition at line 302 of file gos.c.

Here is the call graph for this function:



7.31.5 Variable Documentation

7.31.5.1 dumpRequired

```
GOS_STATIC bool_t dumpRequired
```

Dump required flag.

Definition at line 111 of file gos.c.

7.31.5.2 initError

```
GOS_STATIC bool_t initError
```

Initialization error flag.

Definition at line 106 of file gos.c.

7.31.5.3 initializers

```
GOS_STATIC gos_initStruct_t initializers[]
```

Initial value:

```
=  
{  
    {"Queue service initialization", gos_queueInit},  
    {"Trace service initialization", gos_traceInit},  
    {"Signal service initialization", gos_signalInit},  
    {"Time service initialization", gos_timeInit},  
    {"Message service initialization", gos_messageInit},  
    {"GCP service initialization", gos_gcpInit},  
    {"User application initialization", gos_userApplicationInit}  
}
```

Initializer function and description lookup table.

Definition at line 116 of file gos.c.

7.31.5.4 systemTaskDesc

```
GOS_STATIC gos_taskDescriptor_t systemTaskDesc
```

Initial value:

```
=  
{  
    .taskFunction      = gos_systemTask,  
    .taskName         = "gos_system_task",  
    .taskPriority     = 0u,  
    .taskStackSize    = CFG_SYSTEM_TASK_STACK_SIZE,  
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL  
}
```

Initializer task descriptor.

Definition at line 147 of file gos.c.

7.31.5.5 systemTaskId

```
GOS_STATIC gos_tid_t systemTaskId
```

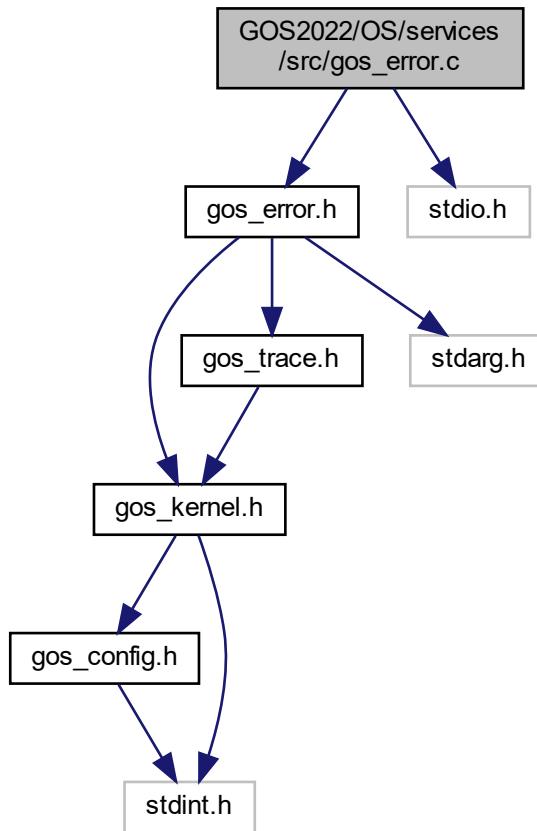
Initializer task ID.

Definition at line 136 of file gos.c.

7.32 GOS2022/OS/services/src/gos_error.c File Reference

GOS error handler service source.

```
#include <gos_error.h>
#include <stdio.h>
Include dependency graph for gos_error.c:
```



Macros

- #define SEPARATOR_LINE "+-----+\r\n"
- #define ERROR_BUFFER_SIZE (80u)
- #define RESULT_STRING_SUCCESS " OK "
- #define RESULT_STRING_ERROR " ERROR "
- #define RESULT_STRING_UNKNOWN "UNKNOWN"

Functions

- **GOS_STATIC char_t * gos_traceResultToString (gos_result_t *pResult)**
Converts the result enumerator to a string.

- `void_t gos_printStartupLogo (void_t)`
Prints the startup logo on the trace output.
- `void_t gos_errorHandler (gos_errorLevel_t errorLevel, const char_t *function, u32_t line, const char_t *errorMessage,...)`
Handles the given error.
- `gos_result_t gos_errorTraceInit (const char_t *initDescription, gos_result_t initResult)`
Traces an initialization message.

Variables

- `GOS_STATIC char_t errorBuffer [(80u)]`

7.32.1 Detailed Description

GOS error handler service source.

Author

Ahmed Gazar

Date

2025-04-06

Version

2.5

For a more detailed description of this service, please refer to [gos_error.h](#)

7.32.2 Macro Definition Documentation

7.32.2.1 ERROR_BUFFER_SIZE

```
#define ERROR_BUFFER_SIZE ( 80u )
```

Error buffer size.

Definition at line 75 of file gos_error.c.

7.32.2.2 RESULT_STRING_ERROR

```
#define RESULT_STRING_ERROR " ERROR "
```

Error result string.

Definition at line 85 of file gos_error.c.

7.32.2.3 RESULT_STRING_SUCCESS

```
#define RESULT_STRING_SUCCESS " OK "
```

Success result string.

Definition at line 80 of file gos_error.c.

7.32.2.4 RESULT_STRING_UNKNOWN

```
#define RESULT_STRING_UNKNOWN "UNKNOWN"
```

Unknown result string.

Definition at line 90 of file gos_error.c.

7.32.2.5 SEPARATOR_LINE

```
#define SEPARATOR_LINE "+-----+\r\n"
```

Separator line.

Definition at line 70 of file gos_error.c.

7.32.3 Function Documentation

7.32.3.1 gos_errorHandler()

```
void_t gos_errorHandler (
    gos_errorLevel_t errorLevel,
    const char_t * function,
    u32_t line,
    const char_t * errorMessage,
    ... )
```

Handles the given error.

Prints the formatted error message on the trace output, and based on the error level, it returns or stays in an infinite loop and disables scheduling.

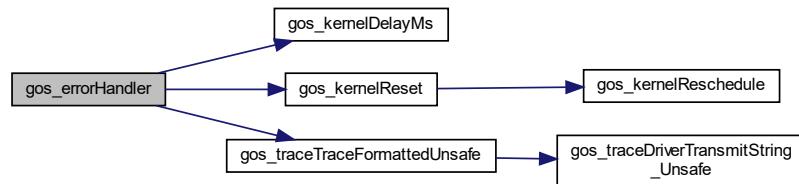
Parameters

in	<i>errorLevel</i>	Level of error (OS/user, warning/fatal).
in	<i>function</i>	Function name.
in	<i>line</i>	Line number.
in	<i>errorMessage</i>	Error message.
in	...	Formatter variable arguments.

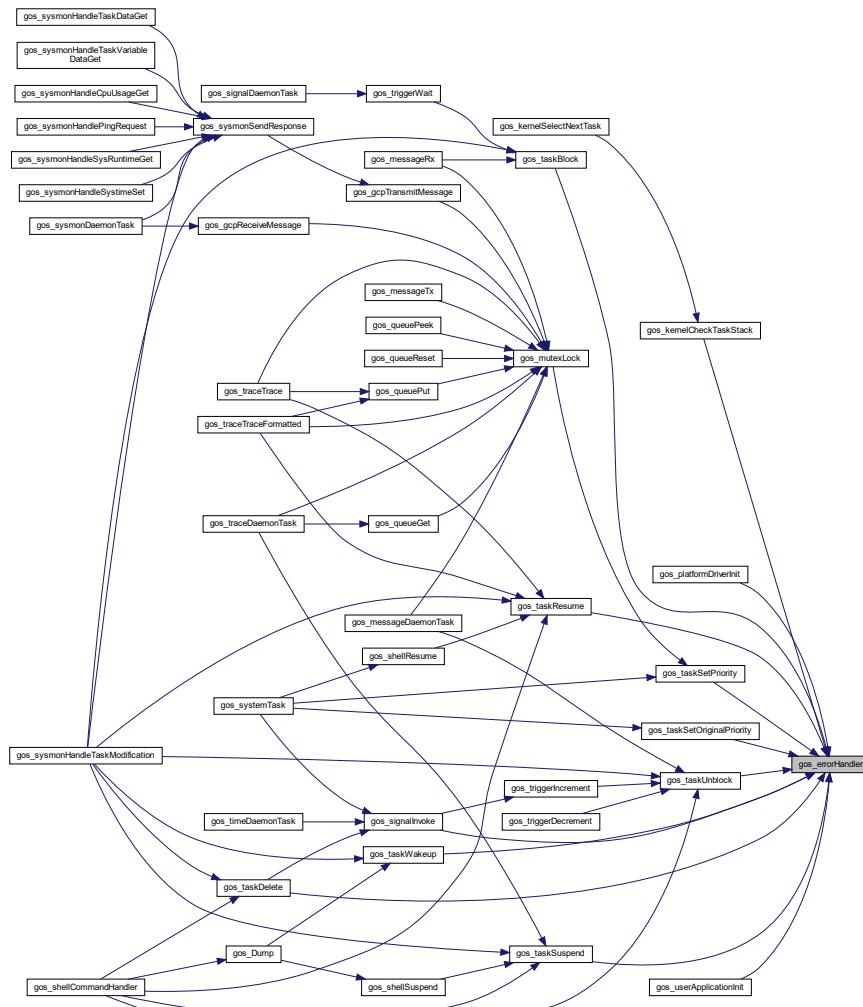
Returns

Definition at line 129 of file gos_error.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.3.2 gos_errorTraceInit()

```
gos_result_t gos_errorTraceInit (
    const char_t * initDescription,
    gos_result_t initResult )
```

Traces an initialization message.

Writes the formatted initialization message on the trace output.

Parameters

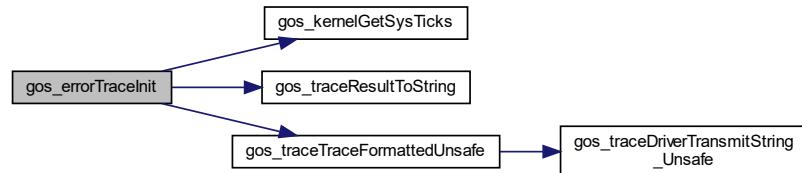
in	<i>initDescription</i>	Message to describe the initialization step.
in	<i>initResult</i>	Result of initialization.

Returns

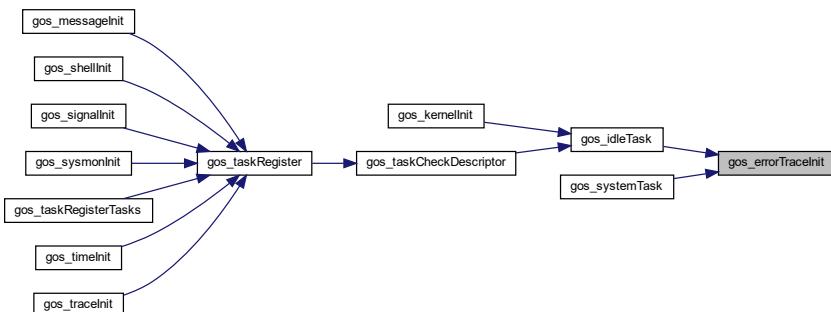
Result of initialization.

Definition at line 237 of file gos_error.c.

Here is the call graph for this function:



Here is the caller graph for this function:

**7.32.3.3 gos_printStartupLogo()**

```
void_t gos_printStartupLogo (
    void_t )
```

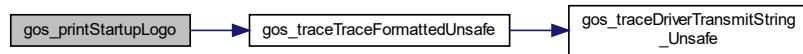
Prints the startup logo on the trace output.

Prints the GOS logo (similar to the ones used in the file headers) on the log output.

Returns

-
Definition at line 108 of file gos_error.c.

Here is the call graph for this function:

**7.32.3.4 gos_traceResultToString()**

```
GOS_STATIC char_t * gos_traceResultToString (
    gos_result_t * pResult )
```

Converts the result enumerator to a string.

Returns the formatted string version of the result enumerator.

Parameters

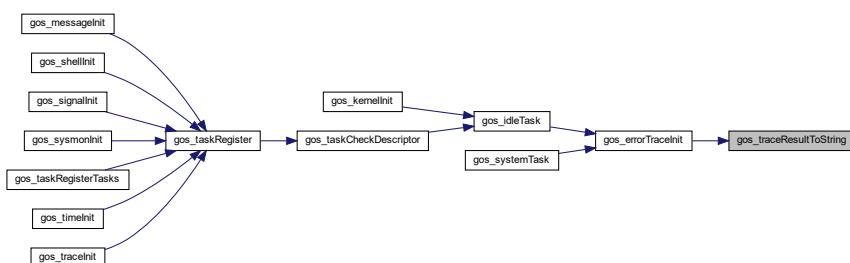
<code>out</code>	<code>pResult</code>	Pointer to the result variable.
------------------	----------------------	---------------------------------

Returns

Formatted string.

Definition at line 260 of file gos_error.c.

Here is the caller graph for this function:

**7.32.4 Variable Documentation**

7.32.4.1 errorBuffer

```
GOS_STATIC char_t errorBuffer[ (80u) ]
```

Buffer for error message formatting.

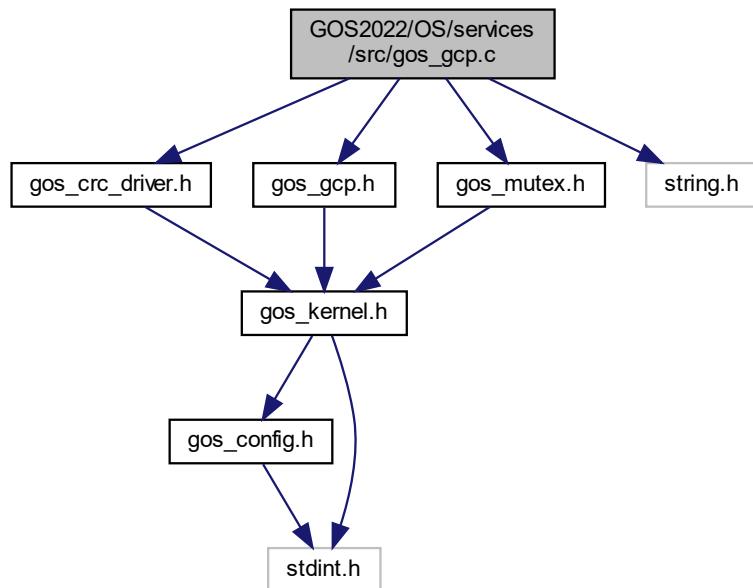
Definition at line 98 of file gos_error.c.

7.33 GOS2022/OS/services/src/gos_gcp.c File Reference

GOS General Communication Protocol handler service source.

```
#include <gos_crc_driver.h>
#include <gos_gcp.h>
#include <gos_mutex.h>
#include <string.h>
```

Include dependency graph for gos_gcp.c:



Data Structures

- struct [gos_gcpHeaderFrame_t](#)
- struct [gos_gcpChannelFunctions_t](#)

Macros

- #define [GCP_PROTOCOL_VERSION_MAJOR](#) (2)
- #define [GCP_PROTOCOL_VERSION_MINOR](#) (0)

Enumerations

- enum `gos_gcpAck_t` {
 `GCP_ACK_REQ` = 0, `GCP_ACK_OK` = 1, `GCP_ACK_CRC_ERROR` = 2, `GCP_ACK_RESEND` = 3,
`GCP_ACK_SIZE_ERROR` = 4, `GCP_ACK_PV_ERROR` = 5, `GCP_ACK_INVALID` = 6 }

Functions

- `GOS_STATIC GOS_INLINE gos_result_t gos_gcpTransmitMessageInternal (gos_gcpChannelNumber_t channel, u16_t messageId, void_t *pMessagePayload, u16_t payloadSize, u16_t maxChunkSize)`
Internal transmitter function (re-entrant).
- `GOS_STATIC GOS_INLINE gos_result_t gos_gcpReceiveMessageInternal (gos_gcpChannelNumber_t channel, u16_t *pMessageId, void_t *pPayloadTarget, u16_t targetSize, u16_t maxChunkSize)`
Internal receiver function (re-entrant).
- `GOS_STATIC gos_result_t gos_gcpValidateHeader (gos_gcpHeaderFrame_t *pHeader, gos_gcpAck_t *pAck)`
Validates the given GCP header.
- `GOS_STATIC gos_result_t gos_gcpValidateData (gos_gcpHeaderFrame_t *pHeader, void_t *pData, gos_gcpAck_t *pAck)`
Validates the given GCP payload/data.
- `gos_result_t gos_gcpInit (void_t)`
Initializes the GCP service.
- `gos_result_t gos_gcpRegisterPhysicalDriver (gos_gcpChannelNumber_t channelNumber, gos_gcpTransmitFunction_t transmitFunction, gos_gcpReceiveFunction_t receiveFunction)`
Registers the physical-layer transmit and receive driver functions.
- `gos_result_t gos_gcpTransmitMessage (gos_gcpChannelNumber_t channel, u16_t messageId, void_t *pMessagePayload, u16_t payloadSize, u16_t maxChunkSize)`
Transmits the given message via the GCP protocol.
- `gos_result_t gos_gcpReceiveMessage (gos_gcpChannelNumber_t channel, u16_t *pMessageId, void_t *pPayloadTarget, u16_t targetSize, u16_t maxChunkSize)`
Receives the given message via the GCP protocol.

Variables

- `GOS_STATIC gos_gcpChannelFunctions_t channelFunctions [CFG_GCP_CHANNELS_MAX_NUMBER]`
- `GOS_STATIC gos_mutex_t gcpRxMutexes [CFG_GCP_CHANNELS_MAX_NUMBER]`
- `GOS_STATIC gos_mutex_t gcpTxMutexes [CFG_GCP_CHANNELS_MAX_NUMBER]`

7.33.1 Detailed Description

GOS General Communication Protocol handler service source.

Author

Ahmed Gazar

Date

2025-04-06

Version

3.1

For a more detailed description of this service, please refer to [gos_gcp.h](#)

7.33.2 Macro Definition Documentation

7.33.2.1 GCP_PROTOCOL_VERSION_MAJOR

```
#define GCP_PROTOCOL_VERSION_MAJOR ( 2 )
```

GCP protocol version high byte.

Definition at line 72 of file gos_gcp.c.

7.33.2.2 GCP_PROTOCOL_VERSION_MINOR

```
#define GCP_PROTOCOL_VERSION_MINOR ( 0 )
```

GCP protocol version low byte.

Definition at line 77 of file gos_gcp.c.

7.33.3 Enumeration Type Documentation

7.33.3.1 gos_gcpAck_t

```
enum gos_gcpAck_t
```

GCP acknowledge type.

Enumerator

GCP_ACK_REQ	Request.
GCP_ACK_OK	OK.
GCP_ACK_CRC_ERROR	CRC error.
GCP_ACK_RESEND	Re-send request.
GCP_ACK_SIZE_ERROR	Size error.
GCP_ACK_PV_ERROR	Protocol version error.
GCP_ACK_INVALID	Invalid message.

Definition at line 85 of file gos_gcp.c.

7.33.4 Function Documentation

7.33.4.1 gos_gcpInit()

```
gos_result_t gos_gcpInit (
    void_t   )
```

Initializes the GCP service.

Creates the GCP lock.

Returns

Result of initialization.

Return values

GOS_SUCCESS	GCP service initialized successfully.
GOS_ERROR	Lock creation error.

Definition at line 171 of file gos_gcp.c.

Here is the call graph for this function:



7.33.4.2 gos_gcpReceiveMessage()

```
gos_result_t gos_gcpReceiveMessage (
    gos_gcpChannelNumber_t channel,
    u16_t * pMessageId,
    void_t * pPayloadTarget,
    u16_t targetSize,
    u16_t maxChunkSize )
```

Receives the given message via the GCP protocol.

Calls the internal receiver function.

Parameters

in	<i>channel</i>	GCP channel.
out	<i>pMessageId</i>	Pointer to a variable to store the message ID.
out	<i>pPayloadTarget</i>	Pointer to the payload target buffer.
in	<i>targetSize</i>	Size of the target buffer (in bytes).
in	<i>maxChunkSize</i>	Maximum size of payload chunks.

Returns

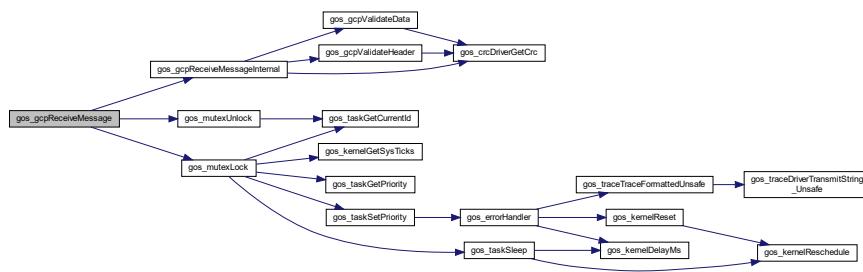
Result of message reception.

Return values

GOS_SUCCESS	Message received successfully.
GOS_ERROR	An error occurred during reception or validation.

Definition at line 258 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.4.3 gos_gcpReceiveMessageInternal()

```

GOS_STATIC GOS_INLINE gos_result_t gos_gcpReceiveMessageInternal (
    gos_gcpChannelNumber_t channel,
    u16_t * pMessageId,
    void_t * pPayloadTarget,
    u16_t targetSize,
    u16_t maxChunkSize )
  
```

Internal receiver function (re-entrant).

Receives a message over GCP (header, payload, and then transmits a response header).

Parameters

in	<i>channel</i>	GCP channel number.
out	<i>pMessageId</i>	Pointer to a variable to store the message ID.
out	<i>pPayloadTarget</i>	Pointer to a buffer to store the payload.
in	<i>targetSize</i>	Size of the payload buffer (in bytes).
in	<i>maxChunkSize</i>	Maximum chunk size (in bytes).

Returns

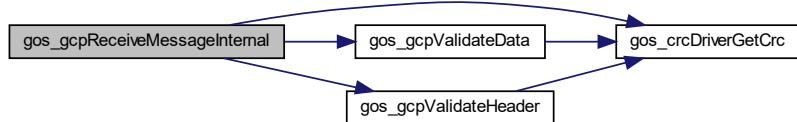
Result of message reception.

Return values

<i>GOS_SUCCESS</i>	Reception successful.
<i>GOS_ERROR</i>	One of the function parameters are invalid or request header validation failed or payload validation failed.

Definition at line 432 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.4.4 gos_gcpRegisterPhysicalDriver()

```
gos_result_t gos_gcpRegisterPhysicalDriver (
    gos_gcpChannelNumber_t channel,
    gos_gcpTransmitFunction_t transmitFunction,
    gos_gcpReceiveFunction_t receiveFunction )
```

Registers the physical-layer transmit and receive driver functions.

Registers the physical-layer transmit and receive driver functions.

Parameters

in	<i>channel</i>	GCP channel.
in	<i>transmitFunction</i>	Transmit function to register.
in	<i>receiveFunction</i>	Receive function to register.

Returns

Result of physical driver registration.

Return values

GOS_SUCCESS	Physical driver registration successful.
GOS_ERROR	Transmit or receive function is NULL.

Definition at line 194 of file gos_gcp.c.

Here is the caller graph for this function:

**7.33.4.5 gos_gcpTransmitMessage()**

```
gos_result_t gos_gcpTransmitMessage (
    gos_gcpChannelNumber_t channel,
    u16_t messageId,
    void_t * pMessagePayload,
    u16_t payloadSize,
    u16_t maxChunkSize )
```

Transmits the given message via the GCP protocol.

Calls the internal message transmitter function.

Parameters

in	<i>channel</i>	GCP channel.
in	<i>messageId</i>	Message ID.
in	<i>pMessagePayload</i>	Pointer to the message payload.
in	<i>payloadSize</i>	Size of the payload (in bytes).
in	<i>maxChunkSize</i>	Maximum size of payload chunks.

Returns

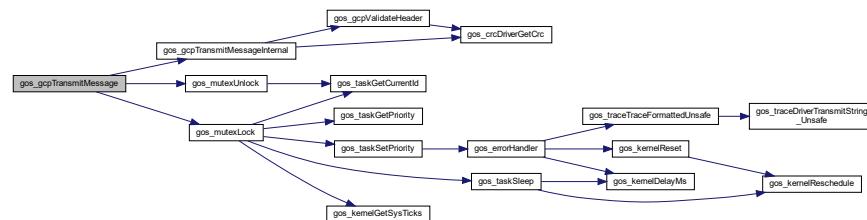
Result of message transmission.

Return values

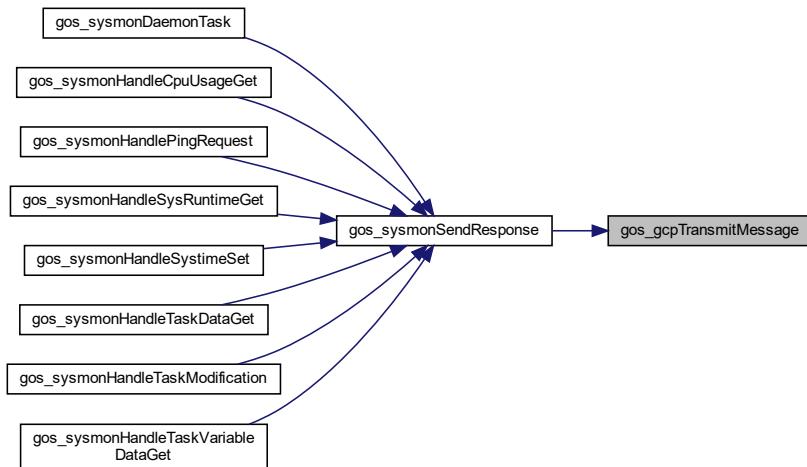
GOS_SUCCESS	Message transmitted successfully.
GOS_ERROR	An error occurred during transmission or validation.

Definition at line 225 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.4.6 gos_gcpTransmitMessageInternal()

```

GOS_STATIC GOS_INLINE gos_result_t gos_gcpTransmitMessageInternal (
    gos_gcpChannelNumber_t channel,
    u16_t messageId,
  
```

```
void_t * pMessagePayload,
u16_t payloadSize,
u16_t maxChunkSize )
```

Internal transmitter function (re-entrant).

Transmits a message over GCP (header request, payload, and then receives a response header).

Parameters

in	<i>channel</i>	GCP channel number.
in	<i>messageId</i>	ID of the message.
in	<i>pMessagePayload</i>	Pointer to the payload buffer.
in	<i>payloadSize</i>	Size of the payload (number of bytes).
in	<i>maxChunkSize</i>	Maximum chunk size (in bytes).

Returns

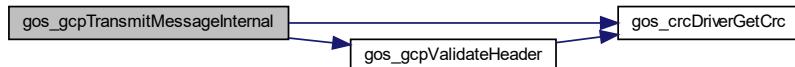
Result of message transmission.

Return values

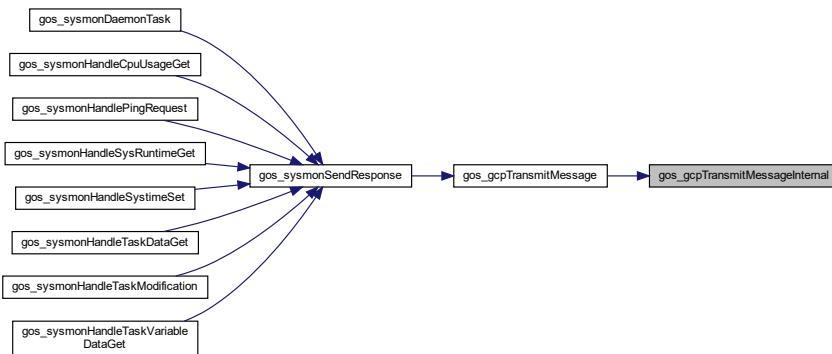
<i>GOS_SUCCESS</i>	Transmission successful.
<i>GOS_ERROR</i>	One of the function parameters are invalid or there was a transmission or reception error.

Definition at line 305 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.4.7 gos_gcpValidateData()

```
GOS_STATIC gos_result_t gos_gcpValidateData (
    gos_gcpHeaderFrame_t * pHeader,
    void_t * pData,
    gos_gcpAck_t * pAck )
```

Validates the given GCP payload/data.

Checks the CRC of the given payload buffer based on the GCP header passed as a parameter. Returns the acknowledge code in the variable passed as a pointer in case the validation fails.

Parameters

in	<i>pHeader</i>	Pointer to the GCP header containing the payload data.
in	<i>pData</i>	Pointer to the data buffer to validate.
out	<i>pAck</i>	Pointer to an acknowledge variable to store the result in.

Returns

Result of validation.

Return values

GOS_SUCCESS	Validation successful.
GOS_ERROR	CRC error or NULL pointer parameter.

Definition at line 628 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.4.8 gos_gcpValidateHeader()

```
GOS_STATIC gos_result_t gos_gcpValidateHeader (
    gos_gcpHeaderFrame_t * pHeader,
    gos_gcpAck_t * pAck )
```

Validates the given GCP header.

Checks the CRC and the protocol version of the header. Returns the acknowledge code in the variable passed as a pointer in case the validation fails.

Parameters

in	<i>pHeader</i>	Pointer to the GCP header to validate.
out	<i>pAck</i>	Pointer to an acknowledge variable to store the result in.

Returns

Result of validation.

Return values

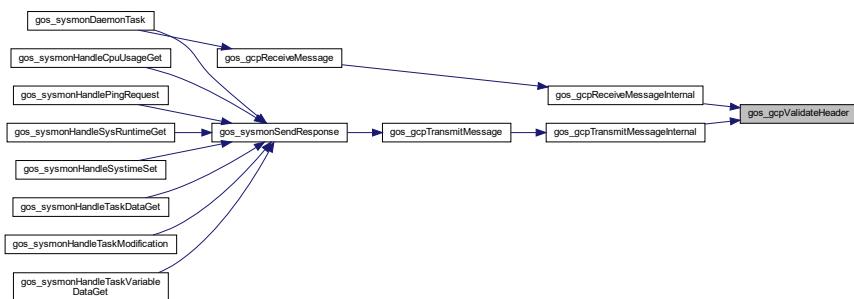
GOS_SUCCESS	Validation successful.
GOS_ERROR	CRC or PV error or NULL pointer parameter.

Definition at line 573 of file gos_gcp.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.5 Variable Documentation

7.33.5.1 channelFunctions

```
GOS_STATIC gos_gcpChannelFunctions_t channelFunctions[CFG_GCP_CHANNELS_MAX_NUMBER]
```

Channel functions.

Definition at line 126 of file gos_gcp.c.

7.33.5.2 gcpRxMutexes

```
GOS_STATIC gos_mutex_t gcpRxMutexes[CFG_GCP_CHANNELS_MAX_NUMBER]
```

GCP RX mutex array.

Definition at line 131 of file gos_gcp.c.

7.33.5.3 gcpTxMutexes

```
GOS_STATIC gos_mutex_t gcpTxMutexes[CFG_GCP_CHANNELS_MAX_NUMBER]
```

GCP TX mutex array.

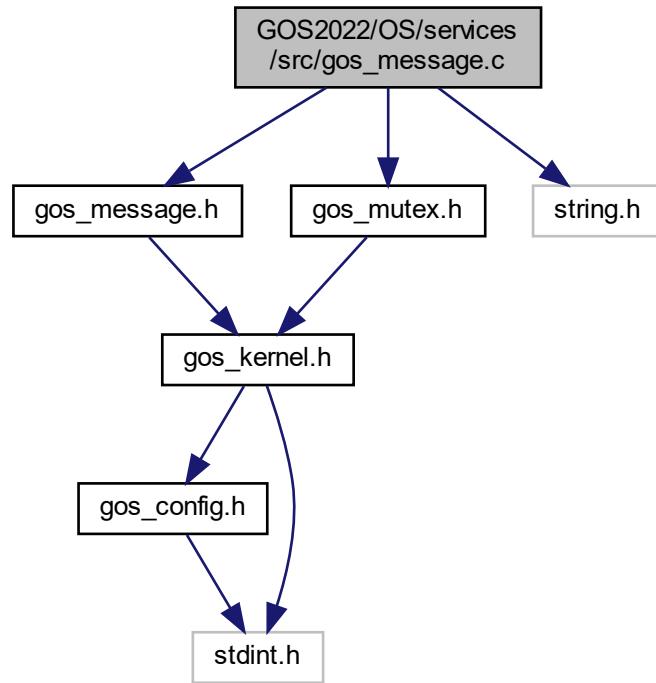
Definition at line 136 of file gos_gcp.c.

7.34 GOS2022/OS/services/src/gos_message.c File Reference

GOS message service source.

```
#include <gos_message.h>
#include <gos_mutex.h>
```

```
#include <string.h>
Include dependency graph for gos_message.c:
```



Data Structures

- struct `gos_messageWaiterDesc_t`

Macros

- #define `GOS_MESSAGE_DAEMON_POLL_TIME_MS` (50u)

Functions

- `GOS_STATIC void_t gos_messageDaemonTask (void_t)`
Message daemon task.
- `gos_result_t gos_messageInit (void_t)`
Initializes the message service.
- `GOS_INLINE gos_result_t gos_messageRx (gos_messageId_t *messagelIdArray, gos_message_t *target, gos_messageTimeout_t tmo)`
Receives the selected messages.
- `GOS_INLINE gos_result_t gos_messageTx (gos_message_t *message)`
Transmits a message.

Variables

- `GOS_STATIC gos_message_t messageArray [CFG_MESSAGE_MAX_NUMBER]`
- `GOS_STATIC gos_messageWaiterDesc_t messageWaiterArray [CFG_MESSAGE_MAX_WAITERS]`
- `GOS_STATIC gos_tid_t messageDaemonTaskId`
- `GOS_STATIC gos_messageIndex_t nextMessageIndex`
- `GOS_STATIC gos_messageWaiterIndex_t nextWaiterIndex`
- `GOS_STATIC gos_mutex_t messageMutex`
- `GOS_STATIC gos_taskDescriptor_t messageDaemonTaskDesc`

7.34.1 Detailed Description

GOS message service source.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.10

For a more detailed description of this service, please refer to [gos_message.h](#)

7.34.2 Macro Definition Documentation

7.34.2.1 GOS_MESSAGE_DAEMON_POLL_TIME_MS

```
#define GOS_MESSAGE_DAEMON_POLL_TIME_MS ( 50u )
```

Message daemon poll time in [ms].

Definition at line 79 of file gos_message.c.

7.34.3 Function Documentation

7.34.3.1 gos_messageDaemonTask()

```
GOS_STATIC void_t gos_messageDaemonTask (
    void_t   )
```

Message daemon task.

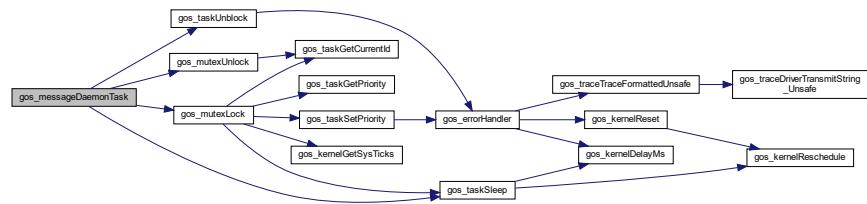
This task is responsible for message transfer between tasks. It loops through the internal message array and the waiter task array and if a message and a waiter matches, it copies the message to the target buffer and unblocks the previously blocked task.

Returns

-

Definition at line 376 of file gos_message.c.

Here is the call graph for this function:



7.34.3.2 gos_messageInit()

```
gos_result_t gos_messageInit (
    void_t   )
```

Initializes the message service.

Initializes the internal message and waiter arrays and registers the message daemon task.

Returns

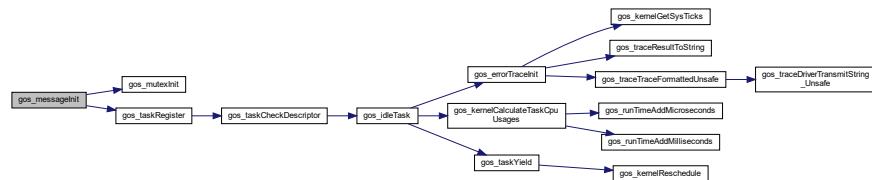
Result of initialization.

Return values

<code>GOS_SUCCESS</code>	Initialization successful.
<code>GOS_ERROR</code>	Message daemon task registration failed.

Definition at line 150 of file gos_message.c.

Here is the call graph for this function:



7.34.3.3 gos_messageRx()

```
GOS_INLINE gos_result_t gos_messageRx (
    gos_messageId_t * messageIdArray,
    gos_message_t * target,
    gos_messageTimeout_t tmo )
```

Receives the selected messages.

Based on the selected message IDs, this function receives the first available message. Until the message is received, the caller task is put to blocked state. The caller will be unblocked if the message is received or the timeout elapsed.

Parameters

in	<i>messageIdArray</i>	Array of messages IDs the function should receive. Must be terminated by a 0 element!
out	<i>target</i>	Pointer to the target message structure. Received data will be placed here.
in	<i>tmo</i>	Timeout value in [ms]. The resolution of timeout is 10ms! Do not use endless timeout if it is not guaranteed that the message will be received!

Returns

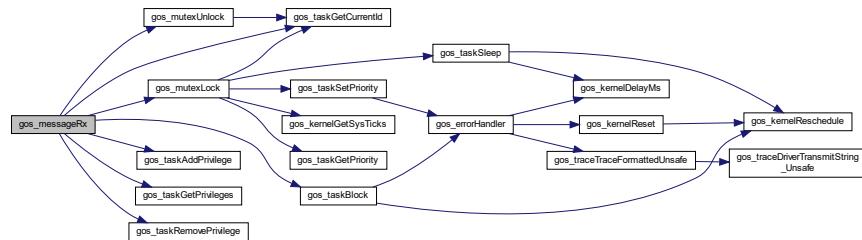
Result of message reception.

Return values

GOS_SUCCESS	Reception successful, data placed in the target structure.
GOS_ERROR	Reception failed because of invalid parameters or timeout.

Definition at line 193 of file gos_message.c.

Here is the call graph for this function:



7.34.3.4 gos_messageTx()

```
GOS_INLINE gos_result_t gos_messageTx (
    gos_message_t * message )
```

Transmits a message.

Copies the message in the internal message array for the message daemon to transfer it to the recipient task.

Parameters

in	<i>message</i>	Pointer to the message structure to be transmitted.
----	----------------	---

Returns

Result of message transmission.

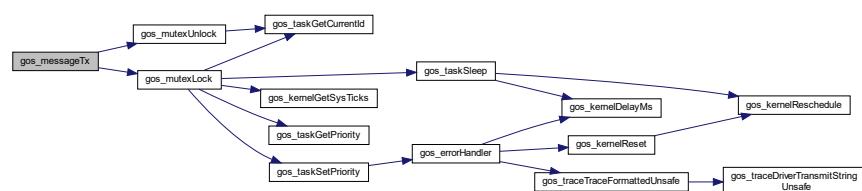
Return values

GOS_SUCCESS	Message transmission initiated successfully.
GOS_ERROR	Invalid message pointer or data or message array is full.

Function code.

Definition at line 306 of file gos_message.c.

Here is the call graph for this function:



7.34.4 Variable Documentation

7.34.4.1 messageArray

```
GOS_STATIC gos_message_t messageArray[CFG_MESSAGE_MAX_NUMBER]
```

Internal message array.

Definition at line 103 of file gos_message.c.

7.34.4.2 messageDaemonTaskDesc

```
GOS_STATIC gos_taskDescriptor_t messageDaemonTaskDesc
```

Initial value:

```
=  
{  
    .taskFunction      = gos_messageDaemonTask,  
    .taskName         = "gos_message_daemon",  
    .taskPriority     = CFG_TASK_MESSAGE_DAEMON_PRIO,  
    .taskStackSize    = CFG_TASK_MESSAGE_DAEMON_STACK,  
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL  
}
```

Message daemon task descriptor.

Definition at line 138 of file gos_message.c.

7.34.4.3 messageDaemonTaskId

```
GOS_STATIC gos_tid_t messageDaemonTaskId
```

Message daemon task ID.

Definition at line 113 of file gos_message.c.

7.34.4.4 messageMutex

```
GOS_STATIC gos_mutex_t messageMutex
```

Message mutex to protect the internal arrays as shared resources.

Definition at line 128 of file gos_message.c.

7.34.4.5 messageWaiterArray

```
GOS_STATIC gos_messageWaiterDesc_t messageWaiterArray[CFG_MESSAGE_MAX_WAITERS]
```

Internal message waiter array.

Definition at line 108 of file gos_message.c.

7.34.4.6 nextMessageIndex

```
GOS_STATIC gos_messageIndex_t nextMessageIndex
```

Index of the next message slot (for circular buffer).

Definition at line 118 of file gos_message.c.

7.34.4.7 nextWaiterIndex

```
GOS_STATIC gos_messageWaiterIndex_t nextWaiterIndex
```

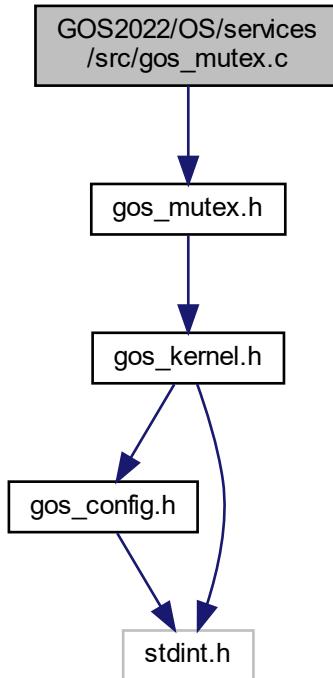
Index of the next waiter slot (for circular buffer).

Definition at line 123 of file gos_message.c.

7.35 GOS2022/OS/services/src/gos_mutex.c File Reference

GOS mutex service source.

```
#include <gos_mutex.h>
Include dependency graph for gos_mutex.c:
```



Macros

- `#define MUTEX_LOCK_SLEEP_MS (2u)`

Functions

- `gos_result_t gos_mutexInit (gos_mutex_t *pMutex)`
Initializes the mutex instance.
- `gos_result_t gos_mutexLock (gos_mutex_t *pMutex, u32_t timeout)`
Tries to lock the given mutex with the given timeout.
- `gos_result_t gos_mutexUnlock (gos_mutex_t *pMutex)`
Unlocks the mutex instance.

7.35.1 Detailed Description

GOS mutex service source.

Author

Ahmed Gazar

Date

2024-04-02

Version

1.8

For a more detailed description of this service, please refer to [gos_mutex.h](#)

7.35.2 Macro Definition Documentation

7.35.2.1 MUTEX_LOCK_SLEEP_MS

```
#define MUTEX_LOCK_SLEEP_MS ( 2u )
```

Sleep time in [ms] before re-attempting to lock mutex.

Definition at line 71 of file gos_mutex.c.

7.35.3 Function Documentation

7.35.3.1 gos_mutexInit()

```
gos_result_t gos_mutexInit (
    gos_mutex_t * pMutex )
```

Initializes the mutex instance.

Sets the mutex state to unlocked.

Parameters

in, out	pMutex	Pointer to the mutex to be initialized.
---------	--------	---

Returns

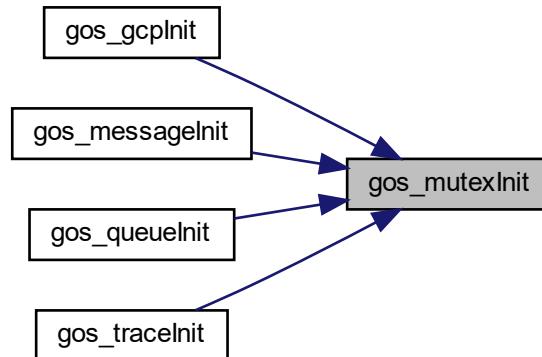
Result of initialization.

Return values

GOS_SUCCESS	Mutex initialized successfully.
GOS_ERROR	Mutex pointer is NULL.

Definition at line 76 of file gos_mutex.c.

Here is the caller graph for this function:



7.35.3.2 gos_mutexLock()

```
gos_result_t gos_mutexLock (
    gos_mutex_t * pMutex,
    u32_t timeout )
```

Tries to lock the given mutex with the given timeout.

Waits in an unblocking way until the mutex is unlocked or the timeout value is reached. If the mutex becomes unlocked within the timeout value, it locks it.

Parameters

in, out	<i>pMutex</i>	Pointer to the mutex to be locked.
in	<i>timeout</i>	Timeout value.

Returns

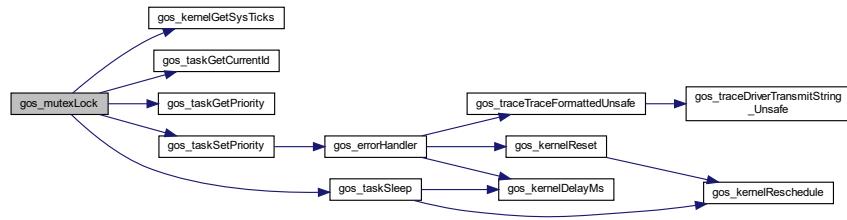
Result of mutex locking.

Return values

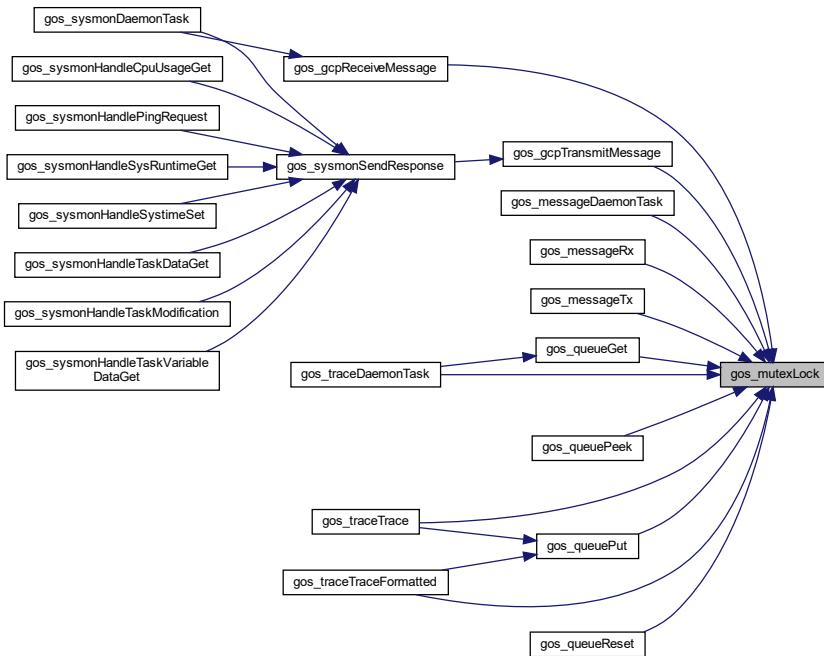
<i>GOS_SUCCESS</i>	Mutex locked successfully.
<i>GOS_ERROR</i>	Mutex could not be locked within the timeout value.

Definition at line 103 of file gos_mutex.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.35.3.3 gos_mutexUnlock()

```
gos_result_t gos_mutexUnlock (
    gos_mutex_t * pMutex )
```

Unlocks the mutex instance.

Sets the mutex state to unlocked.

Parameters

in, out	pMutex	Pointer to the mutex to be unlocked.
---------	--------	--------------------------------------

Returns

Result of mutex unlocking.

Return values

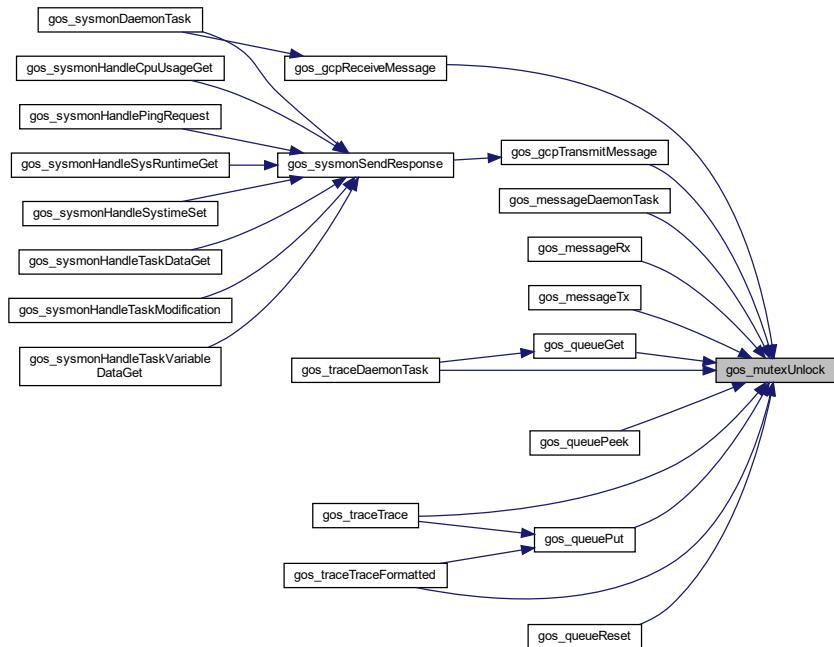
<i>GOS_SUCCESS</i>	Unlocking successful.
<i>GOS_ERROR</i>	Mutex is NULL or caller is not the owner of the mutex.

Definition at line 202 of file gos_mutex.c.

Here is the call graph for this function:



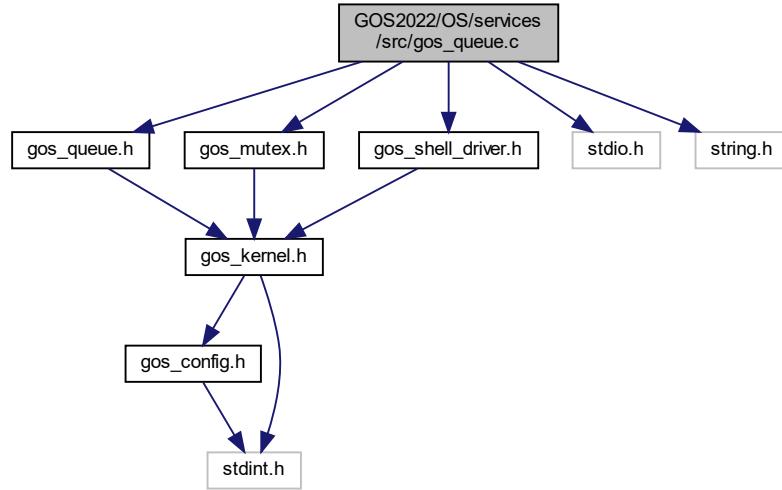
Here is the caller graph for this function:



7.36 GOS2022/OS/services/src/gos queue.c File Reference

GOS queue service source.

```
#include <gos_queue.h>
#include <gos_mutex.h>
#include <gos_shell_driver.h>
#include <stdio.h>
#include <string.h>
Include dependency graph for gos_queue.c:
```



Data Structures

- struct [gos_queueElement_t](#)
- struct [gos_queue_t](#)

Macros

- #define DUMP_SEPARATOR "+-----+\r\n"

Functions

- [gos_result_t gos_queueInit \(void_t\)](#)
This function initializes the queue service.
- [gos_result_t gos_queueCreate \(gos_queueDescriptor_t *pQueueDescriptor\)](#)
This function creates a new queue.
- [gos_result_t gos_queuePut \(gos_queueId_t queueId, void_t *element, gos_queueLength_t elementSize, u32_t timeout\)](#)
This function puts an element in the given queue.
- [gos_result_t gos_queueGet \(gos_queueId_t queueId, void_t *target, gos_queueLength_t targetSize, u32_t timeout\)](#)
This function gets the next element from the given queue.
- [gos_result_t gos_queuePeek \(gos_queueId_t queueId, void_t *target, gos_queueLength_t targetSize, u32_t timeout\)](#)
This function gets the next element from the given queue without removing it.

- [gos_result_t gos_queueReset \(gos_queueId_t queueId\)](#)
Resets the given queue.
- [gos_result_t gos_queueRegisterFullHook \(gos_queueFullHook fullHook\)](#)
This function registers a queue full hook function.
- [gos_result_t gos_queueRegisterEmptyHook \(gos_queueEmptyHook emptyHook\)](#)
This function registers a queue empty hook function.
- [gos_result_t gos_queueGetName \(gos_queueId_t queueId, gos_queueName_t queueName\)](#)
This function gets the name of the given queue.
- [gos_result_t gos_queueGetElementNumber \(gos_queueId_t queueId, gos_queueIndex_t *elementNumber\)](#)
This function gets the number of elements in the given queue.
- [void_t gos_queueDump \(void_t\)](#)
Queue dump.

Variables

- [GOS_STATIC gos_queue_t queues \[CFG_QUEUE_MAX_NUMBER\]](#)
- [GOS_STATIC gos_queueLength_t readCounters \[CFG_QUEUE_MAX_NUMBER\]](#)
- [GOS_STATIC gos_queueLength_t writeCounters \[CFG_QUEUE_MAX_NUMBER\]](#)
- [GOS_STATIC gos_mutex_t queueMutex](#)
- [GOS_STATIC gos_queueFullHook queueFullHook = NULL](#)
- [GOS_STATIC gos_queueEmptyHook queueEmptyHook = NULL](#)

7.36.1 Detailed Description

GOS queue service source.

Author

Ahmed Gazar

Date

2024-04-02

Version

1.8

For a more detailed description of this service, please refer to [gos_queue.h](#)

7.36.2 Macro Definition Documentation

7.36.2.1 DUMP_SEPARATOR

```
#define DUMP_SEPARATOR "-----+\r\n"
```

Dump separator line.

Definition at line 83 of file gos_queue.c.

7.36.3 Function Documentation

7.36.3.1 gos_queueCreate()

```
gos_result_t gos_queueCreate (
    gos_queueDescriptor_t * pQueueDescriptor )
```

This function creates a new queue.

This function loops through the internal queue array and registers the new queue in the next free slot.

Parameters

in, out	<i>pQueueDescriptor</i>	Pointer to queue descriptor variable with queue data.
---------	-------------------------	---

Returns

Result of queue creation.

Return values

<i>GOS_SUCCESS</i>	Queue creation successful.
<i>GOS_ERROR</i>	Queue descriptor is NULL pointer or queue array is full.

Definition at line 181 of file gos_queue.c.

Here is the caller graph for this function:



7.36.3.2 gos_queueDump()

```
void_t gos_queueDump (
    void_t )
```

Queue dump.

Prints the queue data of all queues to the trace output.

Returns

-

Definition at line 598 of file gos_queue.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.36.3.3 gos_queueGet()

```
gos_result_t gos_queueGet (
    gos_queueId_t queueId,
    void_t * target,
    gos_queueLength_t targetSize,
    u32_t timeout )
```

This function gets the next element from the given queue.

This function checks the queue state and gets the next element from the queue.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>target</i>	Pointer to target variable.
in	<i>targetSize</i>	Size of target.
in	<i>timeout</i>	Timeout for locking queue mutex.

Returns

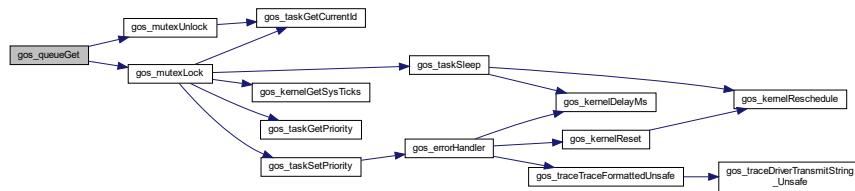
Result of element getting.

Return values

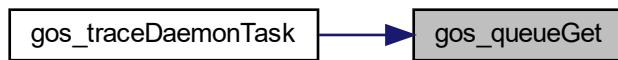
<i>GOS_SUCCESS</i>	Element successfully moved from queue to target.
<i>GOS_ERROR</i>	Invalid queue ID, invalid target size or queue is empty.

Definition at line 310 of file gos_queue.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.36.3.4 gos_queueGetElementNumber()

```
gos_result_t gos_queueGetElementNumber (
    gos_queueId_t queueId,
    gos_queueIndex_t * elementNumber )
```

This function gets the number of elements in the given queue.

This function copies the number of elements in the queue belonging to the given ID to the given variable.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>elementNumber</i>	Pointer to variable to store the element number in.

Returns

Result of queue element number getting.

Return values

<i>GOS_SUCCESS</i>	Element number getting successful.
<i>GOS_ERROR</i>	Invalid queue ID or element number variable is NULL.

Definition at line 564 of file gos_queue.c.

7.36.3.5 gos_queueGetName()

```
gos_result_t gos_queueGetName (
    gos_queueId_t queueId,
    gos_queueName_t queueName )
```

This function gets the name of the given queue.

This function copies the name of the queue belonging to the given ID to the given variable.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>queueName</i>	Queue name.

Returns

Result of queue name getting.

Return values

<i>GOS_SUCCESS</i>	Name getting successful.
<i>GOS_ERROR</i>	Invalid queue ID or queue name variable is NULL.

Definition at line 525 of file gos_queue.c.

7.36.3.6 gos_queueInit()

```
gos_result_t gos_queueInit (
    void_t     )
```

This function initializes the queue service.

Initializes the internal queue array, creates the queue lock, and registers the queue dump task in the kernel.

Returns

Result of initialization.

Return values

<i>GOS_SUCCESS</i>	Initialization successful.
<i>GOS_ERROR</i>	Lock creation, task registration or task suspension error.

Definition at line 147 of file gos_queue.c.

Here is the call graph for this function:



7.36.3.7 gos_queuePeek()

```
gos_result_t gos_queuePeek (
    gos_queueId_t queueId,
    void_t * target,
    gos_queueLength_t targetSize,
    u32_t timeout )
```

This function gets the next element from the given queue without removing it.

This function checks the queue state and returns the next element from the queue without modifying the queue counters.

Parameters

in	<i>queueId</i>	Queue ID.
out	<i>target</i>	Pointer to target variable.
in	<i>targetSize</i>	Size of target.
in	<i>timeout</i>	Timeout for locking queue mutex.

Returns

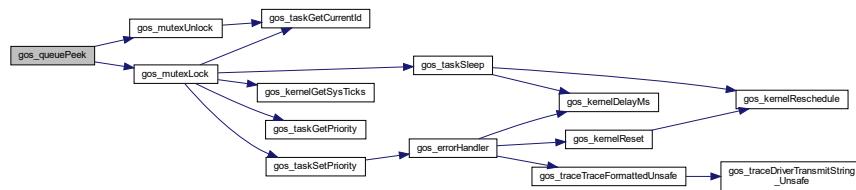
Result of element getting.

Return values

GOS_SUCCESS	Element successfully copied from queue to target.
GOS_ERROR	Invalid queue ID, invalid target size or queue is empty.

Definition at line 385 of file gos_queue.c.

Here is the call graph for this function:

**7.36.3.8 gos_queuePut()**

```

gos_result_t gos_queuePut (
    gos_queueId_t queueId,
    void_t * element,
    gos_queueLength_t elementSize,
    u32_t timeout )
  
```

This function puts an element in the given queue.

This function checks the queue state and places the given element in the next queue element.

Parameters

in	<i>queueId</i>	Queue ID.
in	<i>element</i>	Pointer to element.
in	<i>elementSize</i>	Size of element.
in	<i>timeout</i>	Timeout for locking queue mutex.

Returns

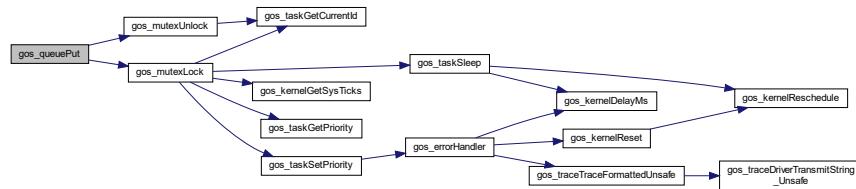
Result of element putting.

Return values

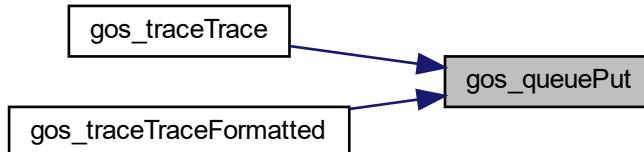
GOS_SUCCESS	Element successfully put in the queue.
GOS_ERROR	Invalid queue ID, invalid element size or queue is full.

Definition at line 231 of file gos_queue.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.36.3.9 gos_queueRegisterEmptyHook()

```
gos_result_t gos_queueRegisterEmptyHook (
    gos_queueEmptyHook emptyHook )
```

This function registers a queue empty hook function.

This function checks whether a hook has been already registered, and if not, it saves the given hook function.

Parameters

in	<i>emptyHook</i>	Hook function.
----	------------------	----------------

Returns

Result of hook registration.

Return values

GOS_SUCCESS	Hook registration successful.
GOS_ERROR	Hook already exists or parameter is NULL pointer.

Definition at line 499 of file gos_queue.c.

7.36.3.10 gos_queueRegisterFullHook()

```
gos_result_t gos_queueRegisterFullHook (
    gos_queueFullHook fullHook )
```

This function registers a queue full hook function.

This function checks whether a hook has been already registered, and if not, it saves the given hook function.

Parameters

in	<i>fullHook</i>	Hook function.
----	-----------------	----------------

Returns

Result of hook registration.

Return values

<i>GOS_SUCCESS</i>	Hook registration successful.
<i>GOS_ERROR</i>	Hook already exists or parameter is NULL pointer.

Definition at line 473 of file gos_queue.c.

7.36.3.11 gos_queueReset()

```
gos_result_t gos_queueReset (
    gos_queueId_t queueId )
```

Resets the given queue.

Sets the read and write counter to zero, making the queue empty.

Parameters

in	<i>queueId</i>	Queue ID.
----	----------------	-----------

Returns

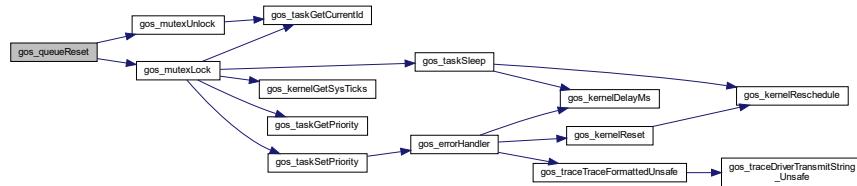
Result of queue resetting.

Return values

<code>GOS_SUCCESS</code>	Reset successful.
<code>GOS_ERROR</code>	Invalid queue ID.

Definition at line 435 of file gos_queue.c.

Here is the call graph for this function:



7.36.4 Variable Documentation

7.36.4.1 queueEmptyHook

```
GOS_STATIC gos_queueEmptyHook queueEmptyHook = NULL
```

Queue empty hook.

Definition at line 142 of file gos_queue.c.

7.36.4.2 queueFullHook

```
GOS_STATIC gos_queueFullHook queueFullHook = NULL
```

Queue full hook.

Definition at line 137 of file gos_queue.c.

7.36.4.3 queueMutex

```
GOS_STATIC gos_mutex_t queueMutex
```

Queue mutex.

Definition at line 132 of file gos_queue.c.

7.36.4.4 queues

```
GOS_STATIC gos_queue_t queues[CFG_QUEUE_MAX_NUMBER]
```

Internal queue array.

Definition at line 117 of file gos_queue.c.

7.36.4.5 readCounters

```
GOS_STATIC gos_queueLength_t readCounters[CFG_QUEUE_MAX_NUMBER]
```

Read counter array (next queue element to read).

Definition at line 122 of file gos_queue.c.

7.36.4.6 writeCounters

```
GOS_STATIC gos_queueLength_t writeCounters[CFG_QUEUE_MAX_NUMBER]
```

Write counter array (next queue element to write).

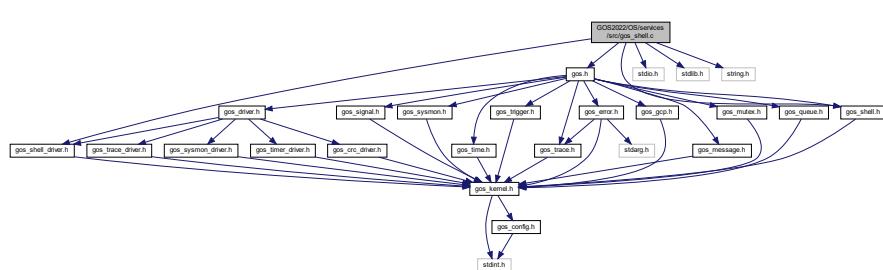
Definition at line 127 of file gos_queue.c.

7.37 GOS2022/OS/services/src/gos_shell.c File Reference

GOS shell service source.

```
#include <gos.h>
#include <gos_shell.h>
#include <gos_shell_driver.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for gos_shell.c:



Macros

- #define GOS_SHELL_DAEMON_POLL_TIME_MS (50u)
- #define GOS_SHELL_DISPLAY_TEXT ("[\x1B[1m]\x1B[33mgos shell\x1B[0m]>> ")

Functions

- GOS_STATIC void_t gos_shellDaemonTask (void_t)
Shell daemon task.
- GOS_STATIC void_t gos_shellCommandHandler (char_t *params)
Shell command handler.
- gos_result_t gos_shellInit (void_t)
This function initializes the shell service.
- gos_result_t gos_shellRegisterCommands (gos_shellCommand_t *commands, u16_t arraySize)
This function registers an array of commands in the shell.
- gos_result_t gos_shellRegisterCommand (gos_shellCommand_t *command)
This function registers a command in the shell.
- gos_result_t gos_shellSuspend (void_t)
Suspends the shell daemon task.
- gos_result_t gos_shellResume (void_t)
Resumes the shell daemon task.
- gos_result_t gos_shellEchoOn (void_t)
Turns the shell echoing on.
- gos_result_t gos_shellEchoOff (void_t)
Turns the shell echoing off.

Variables

- GOS_STATIC gos_shellCommand_t shellCommands [CFG_SHELL_MAX_COMMAND_NUMBER]
- GOS_STATIC gos_tid_t shellDaemonTaskId
- GOS_STATIC char_t commandBuffer [CFG_SHELL_COMMAND_BUFFER_SIZE]
- GOS_STATIC u16_t commandBufferIndex
- GOS_STATIC bool_t useEcho
- GOS_STATIC char_t actualCommand [CFG_SHELL_MAX_COMMAND_LENGTH]
- GOS_STATIC char_t commandParams [CFG_SHELL_MAX_PARAMS_LENGTH]
- GOS_STATIC gos_taskDescriptor_t shellDaemonTaskDesc
- GOS_STATIC gos_shellCommand_t shellCommand

7.37.1 Detailed Description

GOS shell service source.

Author

Ahmed Gazar

Date

2024-06-28

Version

1.9

For a more detailed description of this service, please refer to [gos_shell.h](#)

7.37.2 Macro Definition Documentation

7.37.2.1 GOS_SHELL_DAEMON_POLL_TIME_MS

```
#define GOS_SHELL_DAEMON_POLL_TIME_MS ( 50u )
```

Shell daemon poll time [ms].

Definition at line 77 of file gos_shell.c.

7.37.2.2 GOS_SHELL_DISPLAY_TEXT

```
#define GOS_SHELL_DISPLAY_TEXT ("[\x1B[1m\x1B[33mgos shell\x1B[0m]>> ")
```

Shell display text.

Definition at line 82 of file gos_shell.c.

7.37.3 Function Documentation

7.37.3.1 gos_shellCommandHandler()

```
GOS_STATIC void_t gos_shellCommandHandler (
    char_t * params )
```

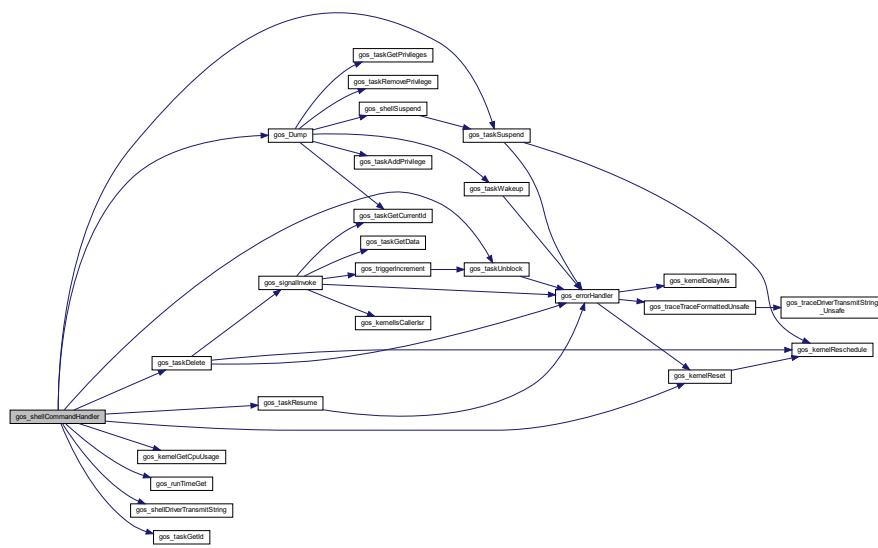
Shell command handler.

Handles the built-in shell command.

Returns

Definition at line 481 of file gos_shell.c.

Here is the call graph for this function:

**7.37.3.2 gos_shellDaemonTask()**

```
GOS_STATIC void_t gos_shellDaemonTask (
    void_t
)
```

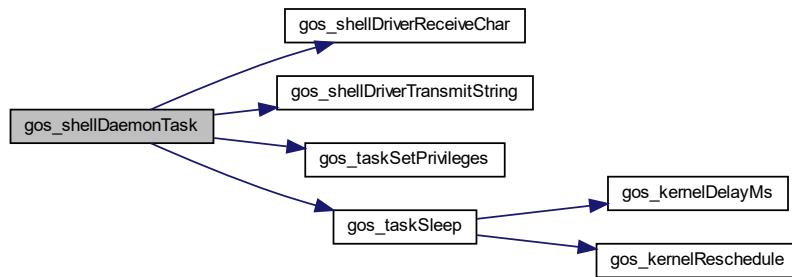
Shell daemon task.

Receives a character from the log serial line, if the echoing is on, then sends the same character back. When an enter key is received, it processes the command typed in, and loops through the command array. When the command is found, it calls the command handler function with the parameter list as a string.

Returns

Definition at line 349 of file gos_shell.c.

Here is the call graph for this function:



7.37.3.3 gos_shellEchoOff()

```
gos_result_t gos_shellEchoOff (
    void_t )
```

Turns the shell echoing off.

Resets the internal echo flag.

Returns

Result of turning echoing off.

Return values

<code>GOS_SUCCESS</code>	Echoing turned off successfully.
--------------------------	----------------------------------

Definition at line 324 of file `gos_shell.c`.

7.37.3.4 gos_shellEchoOn()

```
gos_result_t gos_shellEchoOn (
    void_t )
```

Turns the shell echoing on.

Sets the internal echo flag.

Returns

Result of turning echoing on.

Return values

GOS_SUCCESS	Echoing turned on successfully.
--------------------	---------------------------------

Definition at line 306 of file gos_shell.c.

7.37.3.5 gos_shellInit()

```
gos_result_t gos_shellInit (
    void_t    )
```

This function initializes the shell service.

Initializes the internal command structure, and registers the shell daemon task in the kernel.

Returns

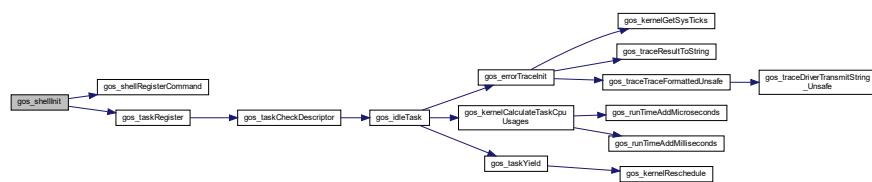
Result of initialization.

Return values

GOS_SUCCESS	Shell initialization successful.
GOS_ERROR	Shell daemon task registration failed.

Definition at line 153 of file gos_shell.c.

Here is the call graph for this function:



7.37.3.6 gos_shellRegisterCommand()

```
gos_result_t gos_shellRegisterCommand (
    gos_shellCommand_t * command )
```

This function registers a command in the shell.

Checks the command structure and registers the command in the next empty slot in the internal command array.

Parameters

in	<i>command</i>	Pointer to the command structure to register.
----	----------------	---

Returns

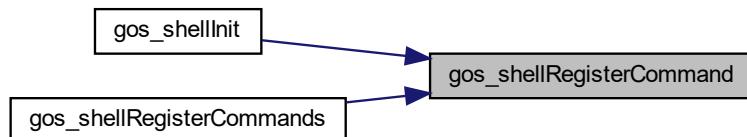
Result of shell command registration.

Return values

<i>GOS_SUCCESS</i>	Command registration successful.
<i>GOS_ERROR</i>	Command function or name is NULL or internal command array is full.

Definition at line 235 of file gos_shell.c.

Here is the caller graph for this function:

**7.37.3.7 gos_shellRegisterCommands()**

```
gos_result_t gos_shellRegisterCommands (
    gos_shellCommand_t * commands,
    u16_t arraySize )
```

This function registers an array of commands in the shell.

Checks the array pointer and registers the commands one by one.

Parameters

in	<i>commands</i>	Pointer to the command structure array to register.
in	<i>arraySize</i>	Size of the array in bytes.

Returns

Result of shell command registration.

Return values

GOS_SUCCESS	Command registration successful.
GOS_ERROR	Command function or name is NULL or internal command array is full.

Definition at line 187 of file gos_shell.c.

Here is the call graph for this function:



7.37.3.8 gos_shellResume()

```
gos_result_t gos_shellResume (
    void_t   )
```

Resumes the shell daemon task.

Resumes the shell daemon task.

Returns

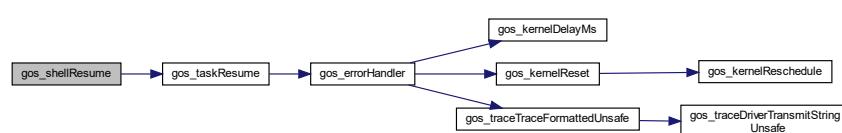
Result of shell resumption.

Return values

GOS_SUCCESS	Shell resumed successfully.
GOS_ERROR	Task resumption failed.

Definition at line 288 of file gos_shell.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.37.3.9 gos_shellSuspend()

```
gos_result_t gos_shellSuspend (
    void_t )
```

Suspends the shell daemon task.

Suspends the shell daemon task.

Returns

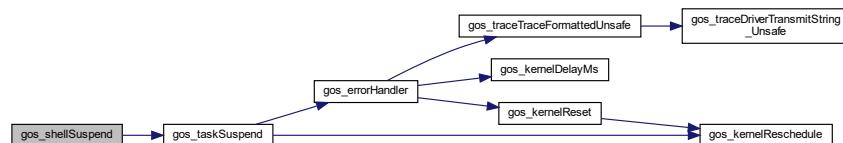
Result of shell suspension.

Return values

<i>GOS_SUCCESS</i>	Shell suspended successfully.
<i>GOS_ERROR</i>	Task suspension failed.

Definition at line 270 of file gos_shell.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.37.4 Variable Documentation

7.37.4.1 actualCommand

```
GOS_STATIC char_t actualCommand[CFG_SHELL_MAX_COMMAND_LENGTH]
```

Actual command buffer.

Definition at line 115 of file gos_shell.c.

7.37.4.2 commandBuffer

```
GOS_STATIC char_t commandBuffer[CFG_SHELL_COMMAND_BUFFER_SIZE]
```

Command buffer.

Definition at line 100 of file gos_shell.c.

7.37.4.3 commandBufferIndex

```
GOS_STATIC u16_t commandBufferIndex
```

Command buffer index.

Definition at line 105 of file gos_shell.c.

7.37.4.4 commandParams

```
GOS_STATIC char_t commandParams[CFG_SHELL_MAX_PARAMS_LENGTH]
```

Command parameters buffer.

Definition at line 120 of file gos_shell.c.

7.37.4.5 shellCommand

```
GOS_STATIC gos_shellCommand_t shellCommand
```

Initial value:

```
=
{
    .command      = "shell",
    .commandHandler = gos_shellCommandHandler,
    .commandHandlerPrivileges = GOS_TASK_PRIVILEGE_KERNEL
}
```

Shell info command.

Definition at line 143 of file gos_shell.c.

7.37.4.6 shellCommands

```
GOS_STATIC gos_shellCommand_t shellCommands[CFG_SHELL_MAX_COMMAND_NUMBER]
```

Shell command array.

Definition at line 90 of file gos_shell.c.

7.37.4.7 shellDaemonTaskDesc

```
GOS_STATIC gos_taskDescriptor_t shellDaemonTaskDesc
```

Initial value:

```
=
{
    .taskFunction      = gos_shellDaemonTask,
    .taskName          = "gos_shell_daemon",
    .taskPriority      = CFG_TASK_SHELL_DAEMON_PRIO,
    .taskStackSize     = CFG_TASK_SHELL_DAEMON_STACK,
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL
}
```

Shell daemon task descriptor.

Definition at line 131 of file gos_shell.c.

7.37.4.8 shellDaemonTaskId

```
GOS_STATIC gos_tid_t shellDaemonTaskId
```

Shell daemon task ID.

Definition at line 95 of file gos_shell.c.

7.37.4.9 useEcho

```
GOS_STATIC bool_t useEcho
```

Shell echo flag.

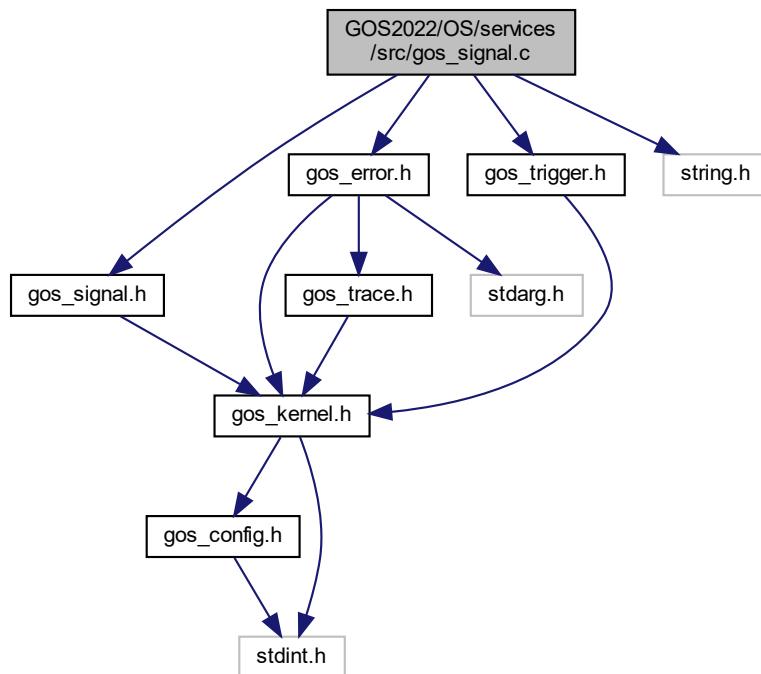
Definition at line 110 of file gos_shell.c.

7.38 GOS2022/OS/services/src/gos_signal.c File Reference

GOS signal service source.

```
#include <gos_signal.h>
#include <gos_error.h>
#include <gos_trigger.h>
#include <string.h>
```

Include dependency graph for gos_signal.c:



Data Structures

- struct [gos_signalDescriptor_t](#)
- struct [gos_signallInvokeDescriptor](#)

Macros

- #define [GOS_SIGNAL_DAEMON_POLL_TIME_MS](#) (50u)

Functions

- [GOS_STATIC void_t gos_signalDaemonTask \(void_t\)](#)
Signal daemon task.
- [gos_result_t gos_signallInit \(void_t\)](#)
Initializes the signal service.
- [gos_result_t gos_signalCreate \(gos_signallId_t *pSignal\)](#)
Creates a new signal.
- [gos_result_t gos_signalSubscribe \(gos_signallId_t signallId, gos_signalHandler_t signalHandler, gos_taskPrivilegeLevel_t signalHandlerPrivileges\)](#)
Subscribes to the given signal.
- [GOS_INLINE gos_result_t gos_signallInvoke \(gos_signallId_t signallId, gos_signalSenderId_t senderId\)](#)
Invokes the given signal.

Variables

- [GOS_STATIC gos_signalDescriptor_t signalArray \[CFG_SIGNAL_MAX_NUMBER\]](#)
- [GOS_STATIC gos_trigger_t signallInvokeTrigger](#)
- [GOS_STATIC gos_taskDescriptor_t callerTaskDesc = {0}](#)
- [GOS_EXTERN gos_signallId_t kernelTaskDeleteSignal](#)
- [GOS_EXTERN gos_signallId_t kernelDumpReadySignal](#)
- [GOS_STATIC gos_taskDescriptor_t signalDaemonTaskDescriptor](#)

7.38.1 Detailed Description

GOS signal service source.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.10

For a more detailed description of this service, please refer to [gos_signal.h](#)

7.38.2 Macro Definition Documentation

7.38.2.1 GOS_SIGNAL_DAEMON_POLL_TIME_MS

```
#define GOS_SIGNAL_DAEMON_POLL_TIME_MS ( 50u )
```

Signal daemon poll time [ms].

Definition at line 76 of file gos_signal.c.

7.38.3 Function Documentation

7.38.3.1 gos_signalCreate()

```
gos_result_t gos_signalCreate (
    gos_signalId_t * pSignal )
```

Creates a new signal.

Finds the next free slot in the signal array and registers the new signal there.

Parameters

out	<i>pSignal</i>	Pointer to a signal identifier.
-----	----------------	---------------------------------

Returns

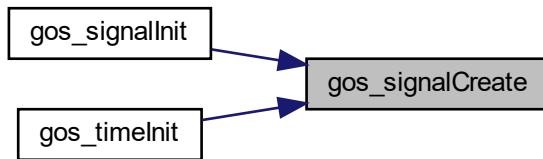
Success of signal creation.

Return values

<i>GOS_SUCCESS</i>	Creation successful.
<i>GOS_ERROR</i>	Signal array full.

Definition at line 183 of file gos_signal.c.

Here is the caller graph for this function:



7.38.3.2 gos_signalDaemonTask()

```
GOS_STATIC void_t gos_signalDaemonTask (
```

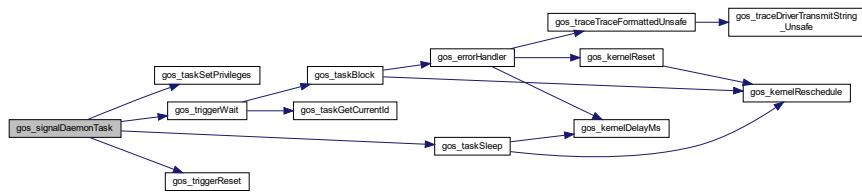
Signal daemon task.

Polls the signal invoke queue, and completes the necessary signal invokings.

Returns

Definition at line 308 of file gos_signal.c.

Here is the call graph for this function:



7.38.3.3 gos_signallInit()

```
gos_result_t gos_signalInit (
```

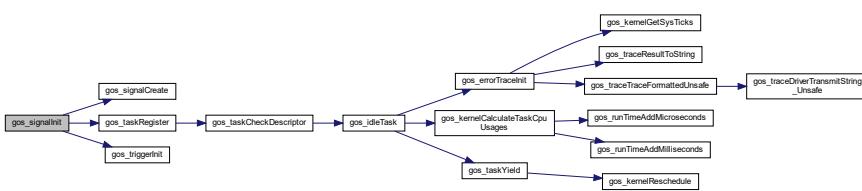
Initializes the signal service.

Initializes the internal signal array, creates a signal queue, registers the signal daemon task, creates the kernel dump signal, and subscribes the necessary components for the dump signal.

Returns

Definition at line 146 of file gos_signal.c.

Here is the call graph for this function:



7.38.3.4 gos_signalInvoke()

```
GOS_INLINE gos_result_t gos_signalInvoke (
    gos_signalId_t signalId,
    gos_signalSenderId_t senderId )
```

Invokes the given signal.

Places the given signal in the invoke queue (for the signal daemon to actually invoke the signal in the background).

Parameters

in	<i>signalId</i>	Signal identifier.
in	<i>senderId</i>	Sender identifier (or data to pass).

Returns

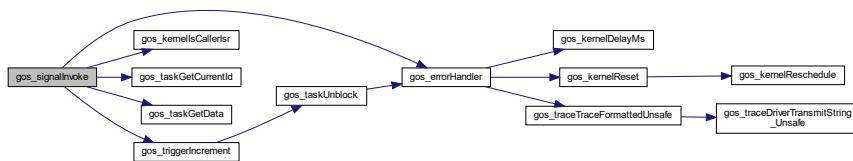
Success of signal invoking.

Return values

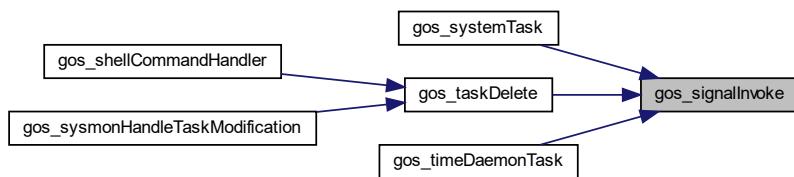
<i>GOS_SUCCESS</i>	Invoking successful.
<i>GOS_ERROR</i>	Invalid signal ID or signal unused.

Definition at line 260 of file gos_signal.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.5 gos_signalSubscribe()

```
gos_result_t gos_signalSubscribe (
    gos_signalId_t signalId,
    gos_signalHandler_t signalHandler,
    gos_taskPrivilegeLevel_t signalHandlerPrivileges )
```

Subscribes to the given signal.

Finds the next free slot in the signal handler array and registers the signal handler there.

Parameters

in	<i>signalId</i>	Signal identifier.
in	<i>signalHandler</i>	Signal handler function pointer.
in	<i>signalHandlerPrivileges</i>	Signal handler privilege level.

Returns

Success of signal subscription.

Return values

<i>GOS_SUCCESS</i>	Subscription successful.
<i>GOS_ERROR</i>	Invalid signal ID, signal handler NULL pointer, or handler array full.

Definition at line 217 of file gos_signal.c.

7.38.4 Variable Documentation

7.38.4.1 callerTaskDesc

```
GOS_STATIC gos_taskDescriptor_t callerTaskDesc = {0}
```

Caller task descriptor.

Definition at line 118 of file gos_signal.c.

7.38.4.2 signalArray

```
GOS_STATIC gos_signalDescriptor_t signalArray[CFG_SIGNAL_MAX_NUMBER]
```

Internal signal descriptor array.

Definition at line 108 of file gos_signal.c.

7.38.4.3 signalDaemonTaskDescriptor

```
GOS_STATIC gos_taskDescriptor_t signalDaemonTaskDescriptor
```

Initial value:

```
=
{
    .taskFunction      = gos_signalDaemonTask,
    .taskName         = "gos_signal_daemon",
    .taskStackSize     = CFG_TASK_SIGNAL_DAEMON_STACK,
    .taskPriority      = CFG_TASK_SIGNAL_DAEMON_PRIO,
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL
}
```

Signal daemon task descriptor.

Definition at line 134 of file gos_signal.c.

7.38.4.4 signallInvokeTrigger

```
GOS_STATIC gos_trigger_t signalInvokeTrigger
```

Invoke trigger (to count the number of signal invokings).

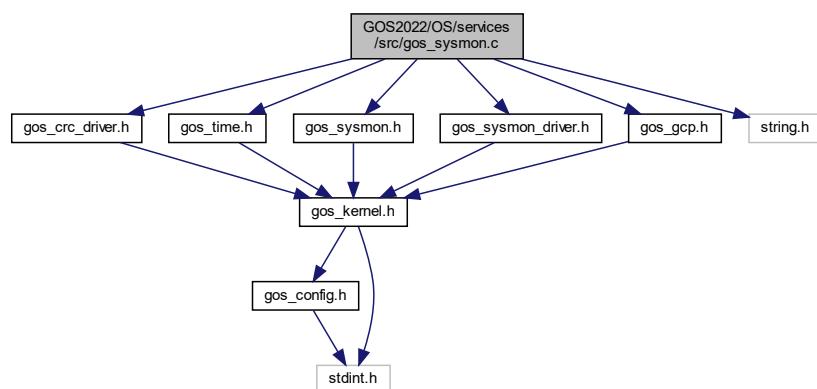
Definition at line 113 of file gos_signal.c.

7.39 GOS2022/OS/services/src/gos_sysmon.c File Reference

GOS system monitoring service source.

```
#include <gos_crc_driver.h>
#include <gos_time.h>
#include <gos_sysmon.h>
#include <gos_sysmon_driver.h>
#include <gos_gcp.h>
#include <string.h>
```

Include dependency graph for gos_sysmon.c:



Data Structures

- struct `gos_sysmonTaskData_t`
- struct `gos_sysmonTaskVariableData`
- struct `gos_sysmonPingMessage_t`
- struct `gos_sysmonCpuUsageMessage_t`
- struct `gos_sysmonTaskDataGetMessage_t`
- struct `gos_sysmonTaskDataMessage_t`
- struct `gos_sysmonTaskVariableDataMessage_t`
- struct `gos_sysmonTaskModifyMessage_t`
- struct `gos_sysmonTaskModifyResultMessage_t`
- struct `gos_sysmonSysruntimeGetResultMessage_t`
- struct `gos_sysmonSystimeSetMessage_t`
- struct `gos_sysmonSystimeSetResultMessage_t`
- struct `gos_sysmonLut_t`

Macros

- `#define RECEIVE_BUFFER_SIZE (4096u)`

Typedefs

- `typedef void_t(* gos_sysmonMessageHandler_t) (gos_sysmonMessageEnum_t)`

Enumerations

- enum `gos_sysmonMessageId_t` {

`GOS_SYSMON_MSG_UNKNOWN_ID = 0, GOS_SYSMON_MSG_PING_ID = 0x0001, GOS_SYSMON_MSG_PING_RESP_ID = 0xA01,`
`GOS_SYSMON_MSG_CPU_USAGE_GET_ID = 0x0002, GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_ID = 0xA02,`
`GOS_SYSMON_MSG_TASK_GET_DATA_ID = 0x0003, GOS_SYSMON_MSG_TASK_GET_DATA_RESP_ID = 0xA03,`
`GOS_SYSMON_MSG_TASK_GET_VAR_DATA_ID = 0x0004, GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_ID = 0xA04,`
`GOS_SYSMON_MSG_TASK_MODIFY_STATE_ID = 0x0005, GOS_SYSMON_MSG_TASK_MODIFY_STATE_RESP_ID = 0xA05,`
`GOS_SYSMON_MSG_SYSRUNTIME_GET_ID = 0x0006, GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_ID = 0xA06,`
`GOS_SYSMON_MSG_SYSRUNTIME_SET_ID = 0x0007, GOS_SYSMON_MSG_SYSRUNTIME_SET_RESP_ID = 0xA07, GOS_SYSMON_MSG_RESET_REQ_ID = 0xFFFF }`
- enum `gos_sysmonMessageEnum_t` {

`GOS_SYSMON_MSG_UNKNOWN = 0, GOS_SYSMON_MSG_PING, GOS_SYSMON_MSG_PING_RESP,`
`GOS_SYSMON_MSG_CPU_USAGE_GET,`
`GOS_SYSMON_MSG_CPU_USAGE_GET_RESP, GOS_SYSMON_MSG_TASK_GET_DATA, GOS_SYSMON_MSG_TASK_GET_VAR_DATA,`
`GOS_SYSMON_MSG_TASK_MODIFY_STATE, GOS_SYSMON_MSG_TASK_MODIFY_STATE_RESP,`
`GOS_SYSMON_MSG_SYSRUNTIME_GET, GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP,`
`GOS_SYSMON_MSG_SYSRUNTIME_SET, GOS_SYSMON_MSG_SYSRUNTIME_SET_RESP,`
`GOS_SYSMON_MSG_RESET_REQ,`
`GOS_SYSMON_MSG_NUM_OF_MESSAGES }`

```

• enum gos_sysmonMessagePv_t {
    GOS_SYSMON_MSG_UNKNOWN_PV = 1, GOS_SYSMON_MSG_PING_PV = 1, GOS_SYSMON_MSG_PING_ACK_PV
    = 1, GOS_SYSMON_MSG_CPU_USAGE_GET_PV = 1,
    GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_PV = 1, GOS_SYSMON_MSG_TASK_GET_DATA_PV
    = 1, GOS_SYSMON_MSG_TASK_GET_DATA_RESP_PV = 1, GOS_SYSMON_MSG_TASK_GET_VAR_DATA_PV
    = 1,
    GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_PV = 1, GOS_SYSMON_MSG_TASK MODIFY STATE_PV
    = 1, GOS_SYSMON_MSG_TASK MODIFY STATE_RESP_PV = 1, GOS_SYSMON_MSG_SYSRUNTIME_GET_PV
    = 1,
    GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_PV = 1, GOS_SYSMON_MSG_SYSTIME_SET_PV =
    1, GOS_SYSMON_MSG_SYSTIME_SET_RESP_PV = 1, GOS_SYSMON_MSG_RESET_REQ_PV = 1 }
• enum gos_sysmonMessageResult_t { GOS_SYSMON_MSG_RES_OK = 40, GOS_SYSMON_MSG_RES_ERROR
= 99, GOS_SYSMON_MSG_INV_PV = 35, GOS_SYSMON_MSG_INV_PAYLOAD_CRC = 28 }
• enum gos_sysmonTaskModifyType_t {
    GOS_SYSMON_TASK_MOD_TYPE_SUSPEND = 12, GOS_SYSMON_TASK_MOD_TYPE_RESUME =
    34, GOS_SYSMON_TASK_MOD_TYPE_DELETE = 49, GOS_SYSMON_TASK_MOD_TYPE_BLOCK = 52,
    GOS_SYSMON_TASK_MOD_TYPE_UNBLOCK = 63, GOS_SYSMON_TASK_MOD_TYPE_WAKEUP = 74
}

```

Functions

- **GOS_STATIC void_t gos_sysmonDaemonTask (void_t)**
System monitoring daemon task.
- **GOS_STATIC gos_sysmonMessageEnum_t gos_sysmonGetLutIndex (gos_sysmonMessageId_t messageId)**
Gets the LUT index of the given message.
- **GOS_STATIC void_t gos_sysmonSendResponse (gos_sysmonMessageEnum_t lutIndex)**
Sends the response to the given message.
- **GOS_STATIC void_t gos_sysmonHandlePingRequest (gos_sysmonMessageEnum_t lutIndex)**
Handles the ping request.
- **GOS_STATIC void_t gos_sysmonHandleCpuUsageGet (gos_sysmonMessageEnum_t lutIndex)**
Handles the CPU usage get request.
- **GOS_STATIC void_t gos_sysmonHandleTaskDataGet (gos_sysmonMessageEnum_t lutIndex)**
Handles the task data get request.
- **GOS_STATIC void_t gos_sysmonHandleTaskVariableDataGet (gos_sysmonMessageEnum_t lutIndex)**
Handles the task variable data get request.
- **GOS_STATIC void_t gos_sysmonHandleTaskModification (gos_sysmonMessageEnum_t lutIndex)**
Handles the task modification request.
- **GOS_STATIC void_t gos_sysmonHandleSysRuntimeGet (gos_sysmonMessageEnum_t lutIndex)**
Handles the system runtime get request.
- **GOS_STATIC void_t gos_sysmonHandleSystimeSet (gos_sysmonMessageEnum_t lutIndex)**
Handles the system time set request.
- **GOS_STATIC void_t gos_sysmonHandleResetRequest (gos_sysmonMessageEnum_t lutIndex)**
Handles the system reset request.
- **GOS_STATIC gos_sysmonMessageResult_t gos_sysmonCheckMessage (gos_sysmonMessageEnum_t lutIndex)**
Checks the high-level message parameters.
- **gos_result_t gos_sysmonInit (void_t)**
This function initializes the system monitoring service.
- **gos_result_t gos_sysmonRegisterUserMessage (gos_sysmonUserMessageDescriptor_t *pDesc)**
This function registers a custom user sysmon message.

Variables

- GOS_STATIC u8_t receiveBuffer [(4096u)]
- GOS_STATIC gos_sysmonPingMessage_t pingMessage = {0}
- GOS_STATIC gos_sysmonCpuUsageMessage_t cpuMessage = {0}
- GOS_STATIC gos_sysmonTaskDataGetMessage_t taskDataGetMsg = {0}
- GOS_STATIC gos_sysmonTaskDataMessage_t taskDataMsg = {0}
- GOS_STATIC gos_sysmonTaskVariableDataMessage_t taskVariableDataMsg = {0}
- GOS_STATIC gos_taskDescriptor_t taskDesc = {0}
- GOS_STATIC gos_sysmonTaskModifyMessage_t taskModifyMessage = {0}
- GOS_STATIC gos_sysmonTaskModifyResultMessage_t taskModifyResultMessage = {0}
- GOS_STATIC gos_sysmonSysruntimeGetResultMessage_t sysRuntimeGetResultMessage = {0}
- GOS_STATIC gos_sysmonSystimeSetMessage_t sysTimeSetMessage = {0}
- GOS_STATIC gos_sysmonSystimeSetResultMessage_t sysTimeSetResultMessage = {0}
- GOS_STATIC gos_sysmonUserMessageDescriptor_t userMessages [CFG_SYSMON_MAX_USER_MESSAGES]
- GOS_STATIC gos_taskDescriptor_t sysmonDaemonTaskDesc
- GOS_STATIC GOS_CONST gos_sysmonLut_t sysmonLut [GOS_SYSMON_MSG_NUM_OF_MESSAGES]

7.39.1 Detailed Description

GOS system monitoring service source.

Author

Ahmed Gazar

Date

2025-03-26

Version

1.6

For a more detailed description of this service, please refer to [gos_sysmon.h](#)

7.39.2 Macro Definition Documentation

7.39.2.1 RECEIVE_BUFFER_SIZE

```
#define RECEIVE_BUFFER_SIZE ( 4096u )
```

Receive buffer size.

Definition at line 74 of file gos_sysmon.c.

7.39.3 Typedef Documentation

7.39.3.1 gos_sysmonMessageHandler_t

```
typedef void_t (* gos_sysmonMessageHandler_t) (gos_sysmonMessageEnum_t)
```

Message handler function type.

Definition at line 303 of file gos_sysmon.c.

7.39.4 Enumeration Type Documentation

7.39.4.1 gos_sysmonMessageEnum_t

```
enum gos_sysmonMessageEnum_t
```

System monitoring message index enumerator.

Enumerator

GOS_SYSMON_MSG_UNKNOWN	Unknown message LUT index.
GOS_SYSMON_MSG_PING	Ping message LUT index.
GOS_SYSMON_MSG_PING_RESP	Ping response message LUT index.
GOS_SYSMON_MSG_CPU_USAGE_GET	CPU usage get message LUT index.
GOS_SYSMON_MSG_CPU_USAGE_GET_RESP	CPU usage get response message LUT index.
GOS_SYSMON_MSG_TASK_GET_DATA	Task data get message LUT index.
GOS_SYSMON_MSG_TASK_GET_DATA_RESP	Task data get response message LUT index.
GOS_SYSMON_MSG_TASK_GET_VAR_DATA	Task variable data get message LUT index.
GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP	Task variable data get response message LUT index.
GOS_SYSMON_MSG_TASK_MODIFY_STATE	Task modify message LUT index.
GOS_SYSMON_MSG_TASK_MODIFY_STATE_RESP	Task modify response message LUT index.
GOS_SYSMON_MSG_SYSRUNTIME_GET	System runtime get message LUT index.
GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP	System runtime get response message LUT index.
GOS_SYSMON_MSG_SYSTIME_SET	System time set message LUT index.
GOS_SYSMON_MSG_SYSTIME_SET_RESP	System time set response message LUT index.
GOS_SYSMON_MSG_RESET_REQ	System reset message LUT index.
GOS_SYSMON_MSG_NUM_OF_MESSAGES	Number of messages.

Definition at line 105 of file gos_sysmon.c.

7.39.4.2 gos_sysmonMessageId_t

enum `gos_sysmonMessageId_t`

System monitoring message ID enum.

Enumerator

<code>GOS_SYSMON_MSG_UNKNOWN_ID</code>	Unknown message ID.
<code>GOS_SYSMON_MSG_PING_ID</code>	Ping message ID.
<code>GOS_SYSMON_MSG_PING_RESP_ID</code>	Ping response message ID.
<code>GOS_SYSMON_MSG_CPU_USAGE_GET_ID</code>	CPU usage get message ID.
<code>GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_ID</code>	CPU usage get response message ID.
<code>GOS_SYSMON_MSG_TASK_GET_DATA_ID</code>	Task data get message ID.
<code>GOS_SYSMON_MSG_TASK_GET_DATA_RESP_ID</code>	Task data get response message ID.
<code>GOS_SYSMON_MSG_TASK_GET_VAR_DATA_ID</code>	Task variable data get message ID.
<code>GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_ID</code>	Task variable data get response message ID.
<code>GOS_SYSMON_MSG_TASK MODIFY STATE_ID</code>	Task modify state message ID.
<code>GOS_SYSMON_MSG_TASK MODIFY STATE RESP_ID</code>	Task modify state response ID.
<code>GOS_SYSMON_MSG_SYSRUNTIME_GET_ID</code>	System runtime get message ID.
<code>GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_ID</code>	System runtime get response message ID.
<code>GOS_SYSMON_MSG_SYSTIME_SET_ID</code>	System time set message ID.
<code>GOS_SYSMON_MSG_SYSTIME_SET_RESP_ID</code>	System time set response ID.
<code>GOS_SYSMON_MSG_RESET_REQ_ID</code>	System reset request ID.

Definition at line 82 of file `gos_sysmon.c`.

7.39.4.3 gos_sysmonMessagePv_t

enum `gos_sysmonMessagePv_t`

System monitoring message protocol version enum.

Enumerator

<code>GOS_SYSMON_MSG_UNKNOWN_PV</code>	Unknown protocol version.
<code>GOS_SYSMON_MSG_PING_PV</code>	Ping message protocol version.
<code>GOS_SYSMON_MSG_PING_ACK_PV</code>	Ping acknowledge message protocol version.
<code>GOS_SYSMON_MSG_CPU_USAGE_GET_PV</code>	CPU usage get message protocol version.
<code>GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_PV</code>	CPU usage get response message protocol version.
<code>GOS_SYSMON_MSG_TASK_GET_DATA_PV</code>	Task data get message protocol version.
<code>GOS_SYSMON_MSG_TASK_GET_DATA_RESP_PV</code>	Task data get response message protocol version.
<code>GOS_SYSMON_MSG_TASK_GET_VAR_DATA_PV</code>	Task variable data get message protocol version.
<code>GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_PV</code>	Task variable data get response message protocol version.
<code>GOS_SYSMON_MSG_TASK MODIFY STATE_PV</code>	Task modify state message protocol version.

Enumerator

GOS_SYSMON_MSG_TASK_MODIFY_STATE_R↔ESP_PV	Task modify state response message protocol version.
GOS_SYSMON_MSG_SYSRUNTIME_GET_PV	System runtime get message protocol version.
GOS_SYSMON_MSG_SYSRUNTIME_GET_RES↔P_PV	System runtime get response protocol version.
GOS_SYSMON_MSG_SYSTIME_SET_PV	System time set message protocol version.
GOS_SYSMON_MSG_SYSTIME_SET_RESP_PV	System time set response message protocol version.
GOS_SYSMON_MSG_RESET_REQ_PV	System reset request message protocol version.

Definition at line 129 of file gos_sysmon.c.

7.39.4.4 gos_sysmonMessageResult_t

```
enum gos_sysmonMessageResult_t
```

Message result enum.

Enumerator

GOS_SYSMON_MSG_RES_OK	Message result OK.
GOS_SYSMON_MSG_RES_ERROR	Message result ERROR.
GOS_SYSMON_MSG_INV_PV	Invalid protocol version.
GOS_SYSMON_MSG_INV_PAYLOAD_CRC	Invalid payload CRC.

Definition at line 152 of file gos_sysmon.c.

7.39.4.5 gos_sysmonTaskModifyType_t

```
enum gos_sysmonTaskModifyType_t
```

State modification enum.

Enumerator

GOS_SYSMON_TASK_MOD_TYPE_SUSPEND	Task suspend.
GOS_SYSMON_TASK_MOD_TYPE_RESUME	Task resume.
GOS_SYSMON_TASK_MOD_TYPE_DELETE	Task delete.
GOS_SYSMON_TASK_MOD_TYPE_BLOCK	Task block.
GOS_SYSMON_TASK_MOD_TYPE_UNBLOCK	Task unblock.
GOS_SYSMON_TASK_MOD_TYPE_WAKEUP	Task wakeup.

Definition at line 163 of file gos_sysmon.c.

7.39.5 Function Documentation

7.39.5.1 gos_sysmonCheckMessage()

```
GOS_STATIC gos_sysmonMessageResult_t gos_sysmonCheckMessage (
    gos_sysmonMessageEnum_t lutIndex )
```

Checks the high-level message parameters.

Checks the protocol version and the payload CRC value.

Parameters

in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

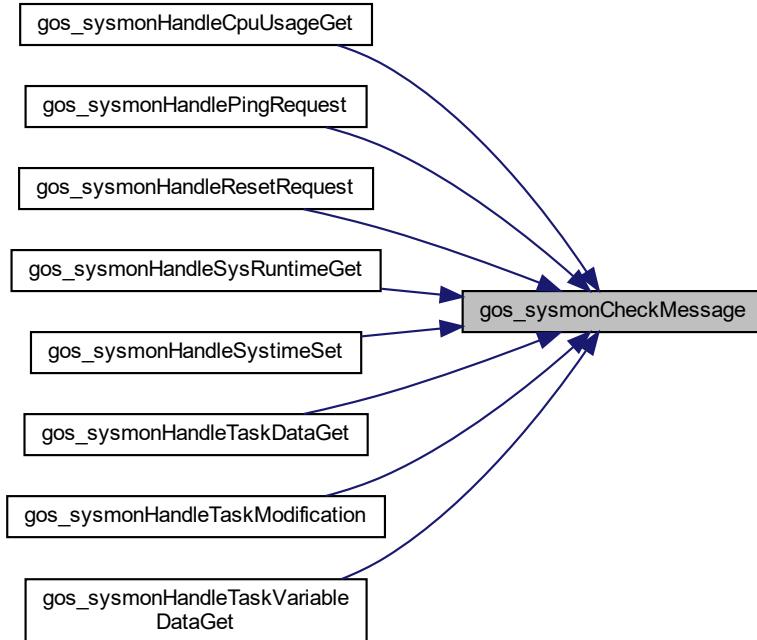
Result of message checking.

Return values

<i>GOS_SYSMON_MSG_RES_OK</i>	Message OK.
<i>GOS_SYSMON_MSG_INV_PV</i>	Invalid protocol version.
<i>GOS_SYSMON_MSG_INV_PAYLOAD_CRC</i>	Payload CRC mismatch.

Definition at line 1192 of file gos_sysmon.c.

Here is the caller graph for this function:



7.39.5.2 gos_sysmonDaemonTask()

```
GOS_STATIC void_t gos_sysmonDaemonTask (
    void_t )

```

System monitoring daemon task.

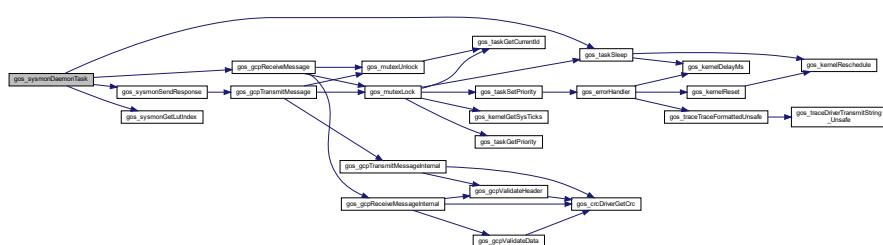
Serves the incoming system monitoring messages.

Returns

-

Definition at line 619 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.3 gos_sysmonGetLutIndex()

```
GOS_STATIC gos_sysmonMessageEnum_t gos_sysmonGetLutIndex (
    gos_sysmonMessageId_t messageId )
```

Gets the LUT index of the given message.

Returns the look-up table index that belongs to the given message ID.

Parameters

in	<i>messageId</i>	ID of the message to get the index for.

Returns

Look-up table index of the message.

Definition at line 703 of file gos_sysmon.c.

Here is the caller graph for this function:



7.39.5.4 gos_sysmonHandleCpuUsageGet()

```
GOS_STATIC void_t gos_sysmonHandleCpuUsageGet (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the CPU usage get request.

Sends out the current CPU usage.

Parameters

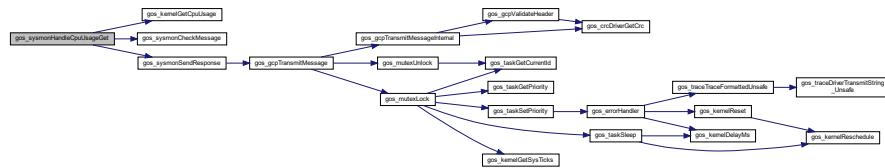
in	<i>lutIndex</i>	Look-up table index of the message.

Returns

-

Definition at line 774 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.5 gos_sysmonHandlePingRequest()

```
GOS_STATIC void_t gos_sysmonHandlePingRequest (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the ping request.

Sends a ping response.

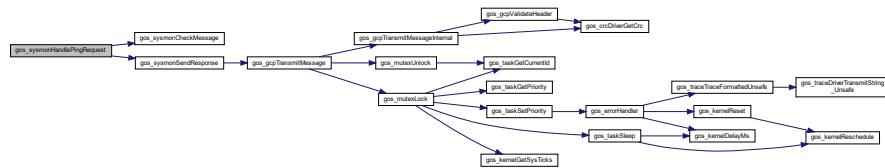
Parameters

in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

Definition at line 757 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.6 gos_sysmonHandleResetRequest()

```
GOS_STATIC void_t gos_sysmonHandleResetRequest (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the system reset request.

Resets the system.

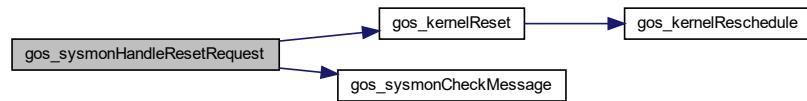
Parameters

in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

Definition at line 1165 of file gos_sysmon.c.

Here is the call graph for this function:

**7.39.5.7 gos_sysmonHandleSysRuntimeGet()**

```
GOS_STATIC void_t gos_sysmonHandleSysRuntimeGet (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the system runtime get request.

Sends out the total system runtime since startup.

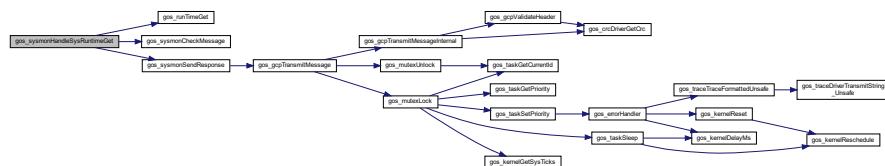
Parameters

in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

Definition at line 1097 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.8 gos_sysmonHandleSystimeSet()

```
GOS_STATIC void_t gos_sysmonHandleSystimeSet (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the system time set request.

Sets the system time to the given value.

Parameters

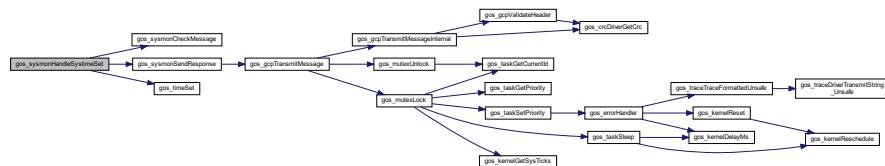
in *lutIndex* Look-up table index of the message.

Returns

-

Definition at line 1131 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.9 gos sysmonHandleTaskDataGet()

```
GOS_STATIC void_t gos_sysmonHandleTaskDataGet (  
    qos_sysmonMessageEnum_t lutIndex )
```

Handles the task data get request.

Sends out the static task data for the requested task.

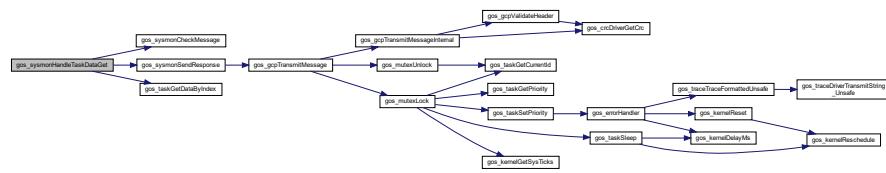
Parameters

in *lutIndex* Look-up table index of the message.

Returns

Definition at line 801 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.10 gos_sysmonHandleTaskModification()

```
GOS_STATIC void_t gos_sysmonHandleTaskModification (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the task modification request.

Performs the requested kind of modification on the requested task.

Parameters

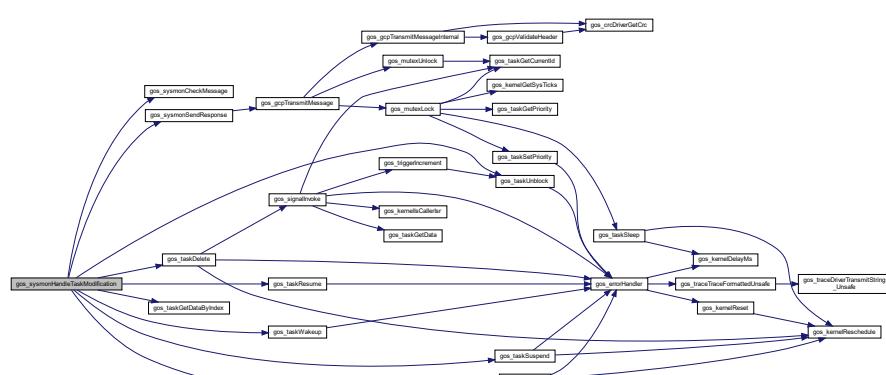
in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

-

Definition at line 988 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.11 gos_sysmonHandleTaskVariableDataGet()

```
GOS_STATIC void_t gos_sysmonHandleTaskVariableDataGet (
    gos_sysmonMessageEnum_t lutIndex )
```

Handles the task variable data get request.

Sends out the variable task data related to the request task.

Parameters

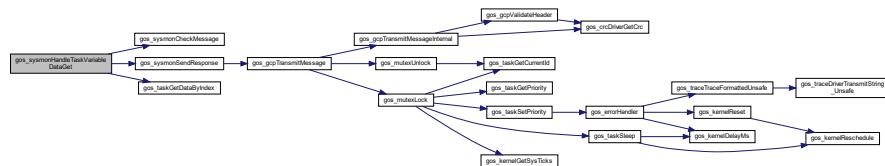
in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

-

Definition at line 900 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.12 gos_sysmonInit()

```
gos_result_t gos_sysmonInit (
    void_t )
```

This function initializes the system monitoring service.

Registers the GCP physical driver and the sysmon daemon task in the kernel.

Returns

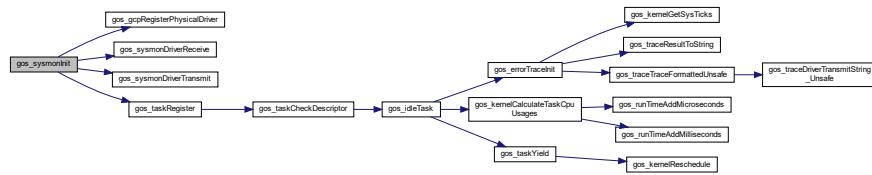
Result of initialization.

Return values

GOS_SUCCESS	Sysmon initialization successful.
GOS_ERROR	Sysmon daemon task registration failed.

Definition at line 543 of file gos_sysmon.c.

Here is the call graph for this function:



7.39.5.13 gos_sysmonRegisterUserMessage()

```
gos_result_t gos_sysmonRegisterUserMessage (
    gos_sysmonUserMessageDescriptor_t * pDesc )
```

This function registers a custom user sysmon message.

Registers a custom sysmon message given by its ID and PV. The message will only be processed if the required ID is not an existing sysmon ID. When a message is received with the given ID and PV, the message content will be copied into the buffer defined in the descriptor structure, and the registered callback function will be called.

Recommended ID range: 0x6000 ... 0x9999.

Parameters

in	<i>pDesc</i>	Pointer to a sysmon user message descriptor.
----	--------------	--

Returns

Result of registration.

Return values

<i>GOS_SUCCESS</i>	User message registered successfully.
<i>GOS_ERROR</i>	Descriptor or callback is NULL or maximum number of user messages has been reached.

Definition at line 580 of file gos_sysmon.c.

7.39.5.14 gos_sysmonSendResponse()

```
GOS_STATIC void_t gos_sysmonSendResponse (
    gos_sysmonMessageEnum_t lutIndex )
```

Sends the response to the given message.

Fills out the remaining transmit header parameters (except for the result field), and transmits the message with the corresponding payload.

Parameters

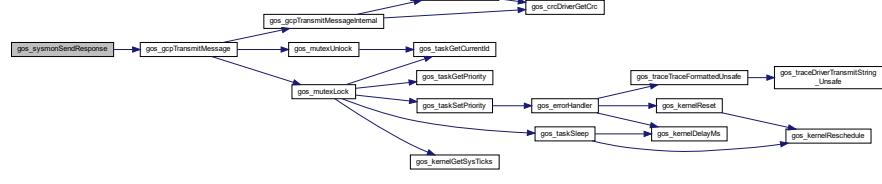
in	<i>lutIndex</i>	Look-up table index of the message.
----	-----------------	-------------------------------------

Returns

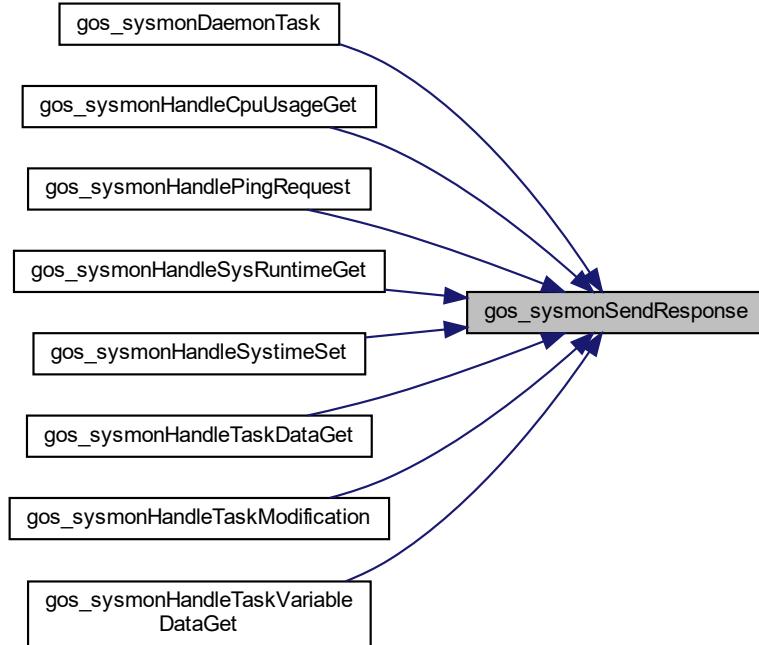
-

Definition at line 737 of file gos_sysmon.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.39.6 Variable Documentation

7.39.6.1 cpuMessage

```
GOS_STATIC gos_sysmonCpuUsageMessage_t cpuMessage = {0}
```

CPU load message structure.

Definition at line 333 of file gos_sysmon.c.

7.39.6.2 pingMessage

```
GOS_STATIC gos_sysmonPingMessage_t pingMessage = {0}
```

Ping message structure.

Definition at line 328 of file gos_sysmon.c.

7.39.6.3 receiveBuffer

```
GOS_STATIC u8_t receiveBuffer[ (4096u) ]
```

Receive buffer.

Definition at line 323 of file gos_sysmon.c.

7.39.6.4 sysmonDaemonTaskDesc

```
GOS_STATIC gos_taskDescriptor_t sysmonDaemonTaskDesc
```

Initial value:

```
=  
{  
    .taskFunction      = gos_sysmonDaemonTask,  
    .taskName         = "gos_sysmon_daemon",  
    .taskPriority     = CFG_TASK_SYSMON_DAEMON_PRIO,  
    .taskStackSize    = CFG_TASK_SYSMON_DAEMON_STACK,  
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL  
}
```

Sysmon daemon task descriptor.

Definition at line 404 of file gos_sysmon.c.

7.39.6.5 sysmonLut

```
GOS_STATIC GOS_CONST gos_sysmonLut_t sysmonLut [GOS_SYSMON_MSG_NUM_OF_MESSAGES]
```

Sysmon look-up table.

Definition at line 416 of file gos_sysmon.c.

7.39.6.6 sysRuntimeGetResultMessage

```
GOS_STATIC gos_sysmonSysruntimeGetResultMessage_t sysRuntimeGetResultMessage = {0}
```

System runtime get message.

Definition at line 368 of file gos_sysmon.c.

7.39.6.7 sysTimeSetMessage

```
GOS_STATIC gos_sysmonSystimeSetMessage_t sysTimeSetMessage = {0}
```

System time set message.

Definition at line 373 of file gos_sysmon.c.

7.39.6.8 sysTimeSetResultMessage

```
GOS_STATIC gos_sysmonSystimeSetResultMessage_t sysTimeSetResultMessage = {0}
```

System time set result message.

Definition at line 378 of file gos_sysmon.c.

7.39.6.9 taskDataGetMsg

```
GOS_STATIC gos_sysmonTaskDataGetMessage_t taskDataGetMsg = {0}
```

Task data get message structure.

Definition at line 338 of file gos_sysmon.c.

7.39.6.10 taskDataMsg

```
GOS_STATIC gos_sysmonTaskDataMessage_t taskDataMsg = {0}
```

Task data message structure.

Definition at line 343 of file gos_sysmon.c.

7.39.6.11 taskDesc

```
GOS_STATIC gos_taskDescriptor_t taskDesc = {0}
```

Task descriptor structure.

Definition at line 353 of file gos_sysmon.c.

7.39.6.12 taskModifyMessage

```
GOS_STATIC gos_sysmonTaskModifyMessage_t taskModifyMessage = {0}
```

Task modify message.

Definition at line 358 of file gos_sysmon.c.

7.39.6.13 taskModifyResultMessage

```
GOS_STATIC gos_sysmonTaskModifyResultMessage_t taskModifyResultMessage = {0}
```

Task modify result message.

Definition at line 363 of file gos_sysmon.c.

7.39.6.14 taskVariableDataMsg

```
GOS_STATIC gos_sysmonTaskVariableDataMessage_t taskVariableDataMsg = {0}
```

Task variable data message structure.

Definition at line 348 of file gos_sysmon.c.

7.39.6.15 userMessages

```
GOS_STATIC gos_sysmonUserMessageDescriptor_t userMessages[CFG_SYSMON_MAX_USER_MESSAGES]
```

Sysmon user messages.

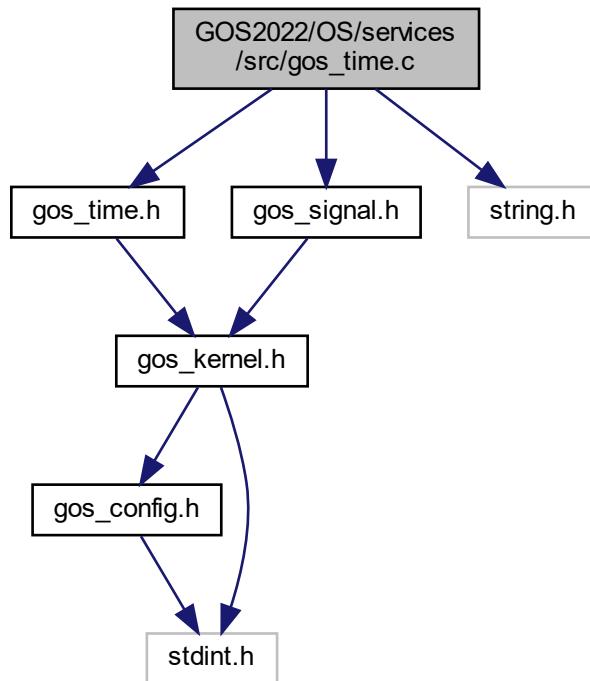
Definition at line 383 of file gos_sysmon.c.

7.40 GOS2022/OS/services/src/gos_time.c File Reference

GOS time service source.

```
#include <gos_time.h>
#include <gos_signal.h>
#include <string.h>
```

Include dependency graph for gos_time.c:



Macros

- #define TIME_DEFAULT_YEAR (2023)
- #define TIME_DEFAULT_MONTH (GOS_TIME_JANUARY)
- #define TIME_DEFAULT_DAY (1)
- #define TIME_SLEEP_TIME_MS (500u)

Functions

- [GOS_STATIC void_t gos_timeDaemonTask \(void_t\)](#)
Time daemon task.
- [gos_result_t gos_timeInit \(void_t\)](#)
This function initializes the time service.
- [gos_result_t gos_timeGet \(gos_time_t *pTime\)](#)
This function gets the system time.
- [gos_result_t gos_timeSet \(gos_time_t *pTime\)](#)
This function sets the system time.
- [gos_result_t gos_runTimeGet \(gos_runtime_t *pRunTime\)](#)
This function gets the system run-time.
- [gos_result_t gos_timeCompare \(gos_time_t *pTime1, gos_time_t *pTime2, gos_timeCompareResult_t *result\)](#)
This function compares two time structures.
- [gos_result_t gos_timeAddMilliseconds \(gos_time_t *pTime, u16_t milliseconds\)](#)
This function adds the given number of milliseconds to the given time structure.
- [gos_result_t gos_timeAddSeconds \(gos_time_t *pTime, u16_t seconds\)](#)
This function adds the given number of seconds to the given time variable.
- [gos_result_t gos_runTimeAddMicroseconds \(gos_runtime_t *pRunTime1, gos_runtime_t *pRunTime2, u16_t microseconds\)](#)
This function adds the given number of microseconds to the given time variables.
- [gos_result_t gos_runTimeAddMilliseconds \(gos_runtime_t *pRunTime, u16_t milliseconds\)](#)
This function adds the given number of milliseconds to the given time variables.
- [gos_result_t gos_runTimeAddSeconds \(gos_runtime_t *pRunTime, u32_t seconds\)](#)
This function adds the given number of seconds to the given run-time variable.
- [gos_result_t gos_timeIncreaseSystemTime \(u16_t milliseconds\)](#)
This function increases the system time and runtime with the given value of milliseconds.

Variables

- [GOS_STATIC gos_time_t systemTime](#)
- [GOS_STATIC gos_runtime_t systemRunTime](#)
- [GOS_STATIC GOS_CONST gos_day_t dayLookupTable \[GOS_TIME_NUMBER_OF_MONTHS\]](#)
- [GOS_STATIC gos_tid_t timeDaemonTaskId](#)
- [gos_signallId_t timeSignallId](#)
- [GOS_STATIC gos_taskDescriptor_t timeDaemonTaskDesc](#)

7.40.1 Detailed Description

GOS time service source.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.8

For a more detailed description of this service, please refer to [gos_time.h](#)

7.40.2 Macro Definition Documentation

7.40.2.1 TIME_DEFAULT_DAY

```
#define TIME_DEFAULT_DAY ( 1 )
```

Default day.

Definition at line 86 of file gos_time.c.

7.40.2.2 TIME_DEFAULT_MONTH

```
#define TIME_DEFAULT_MONTH ( GOS_TIME_JANUARY )
```

Default month.

Definition at line 81 of file gos_time.c.

7.40.2.3 TIME_DEFAULT_YEAR

```
#define TIME_DEFAULT_YEAR ( 2023 )
```

Default year.

Definition at line 76 of file gos_time.c.

7.40.2.4 TIME_SLEEP_TIME_MS

```
#define TIME_SLEEP_TIME_MS ( 500u )
```

Time task sleep time in [ms].

Definition at line 91 of file gos_time.c.

7.40.3 Function Documentation

7.40.3.1 gos_runTimeAddMicroseconds()

```
gos_result_t gos_runTimeAddMicroseconds (
    gos_runtime_t * pRuntime1,
    gos_runtime_t * pRuntime2,
    u16_t microseconds )
```

This function adds the given number of microseconds to the given time variables.

This function adds the given number of microseconds to the given time variables.

Parameters

in, out	<i>pRunTime1</i>	Pointer to the time variable.
in, out	<i>pRunTime2</i>	Pointer to the time variable.
in	<i>microseconds</i>	Number of microseconds to add.

Returns

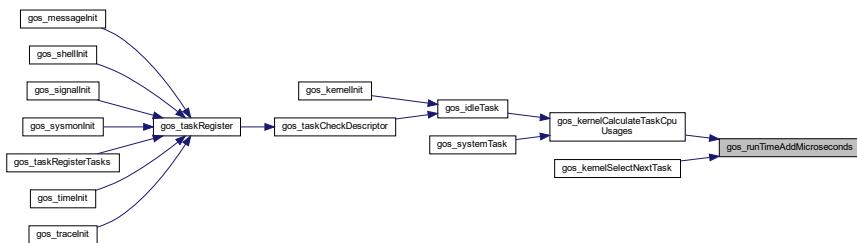
Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Microseconds added successfully.
<i>GOS_ERROR</i>	Time variable is NULL pointer.

Definition at line 498 of file gos_time.c.

Here is the caller graph for this function:

**7.40.3.2 gos_runTimeAddMilliseconds()**

```

gos_result_t gos_runTimeAddMilliseconds (
    gos_runtime_t * pRunTime,
    u16_t milliseconds )
  
```

This function adds the given number of milliseconds to the given time variables.

This function adds the given number of milliseconds to the given time variables.

Parameters

in, out	<i>pRunTime</i>	Pointer to the time variable.
in	<i>milliseconds</i>	Number of milliseconds to add.

Returns

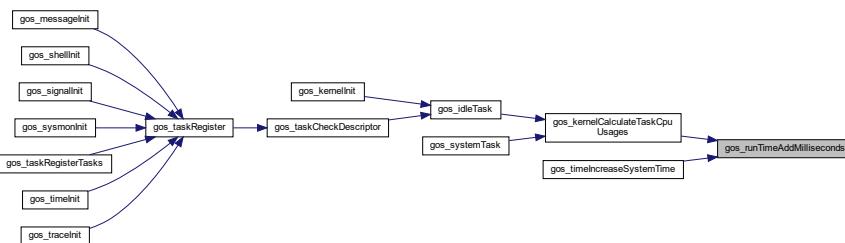
Result of time increasing.

Return values

GOS_SUCCESS	Milliseconds added successfully.
GOS_ERROR	Time variable is NULL pointer.

Definition at line 638 of file gos_time.c.

Here is the caller graph for this function:

**7.40.3.3 gos_runTimeAddSeconds()**

```
gos_result_t gos_runTimeAddSeconds (
    gos_runtime_t * pRunTime,
    u32_t seconds )
```

This function adds the given number of seconds to the given run-time variable.

This function adds the given number of seconds to the given run-time variable.

Parameters

in, out	<i>pRunTime</i>	Pointer to a run-time variable.
in	<i>seconds</i>	Number of seconds to add.

Returns

Result of time increasing.

Return values

GOS_SUCCESS	Seconds added successfully.
GOS_ERROR	Run-time variable is NULL pointer.

Definition at line 707 of file gos_time.c.

7.40.3.4 gos_runTimeGet()

```
gos_result_t gos_runTimeGet (
    gos_runtime_t * pRunTime )
```

This function gets the system run-time.

This function gets the system run-time.

Parameters

<code>out</code>	<code>pRunTime</code>	Pointer to a run-time variable to store the system run-time in.
------------------	-----------------------	---

Returns

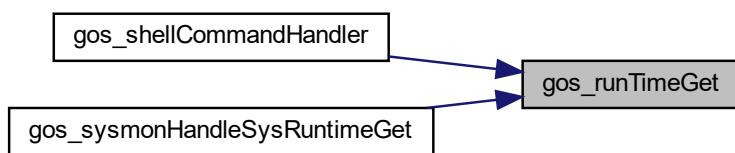
Result of run-time getting.

Return values

<code>GOS_SUCCESS</code>	Run-time getting is successful.
<code>GOS_ERROR</code>	Run-time variable is NULL pointer.

Definition at line 241 of file gos_time.c.

Here is the caller graph for this function:



7.40.3.5 gos_timeAddMilliseconds()

```
gos_result_t gos_timeAddMilliseconds (
    gos_time_t * pTime,
    u16_t milliseconds )
```

This function adds the given number of milliseconds to the given time structure.

This function adds the given number of milliseconds to the given time structure.

Parameters

in, out	<i>pTime</i>	Pointer to the time structure.
in	<i>milliseconds</i>	Number of milliseconds to add.

Returns

Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Increasing successful.
<i>GOS_ERROR</i>	Time structure is NULL pointer.

Definition at line 312 of file gos_time.c.

Here is the caller graph for this function:

**7.40.3.6 gos_timeAddSeconds()**

```
gos_result_t gos_timeAddSeconds (
    gos_time_t * pTime,
    u16_t seconds )
```

This function adds the given number of seconds to the given time variable.

This function adds the given number of seconds to the given time variable.

Parameters

in, out	<i>pTime</i>	Pointer to the time variable.
in	<i>seconds</i>	Number of seconds to add.

Returns

Result of time increasing.

Return values

<i>GOS_SUCCESS</i>	Seconds added successfully.
<i>GOS_ERROR</i>	Time variable is NULL pointer.

Definition at line 408 of file gos_time.c.

7.40.3.7 gos_timeCompare()

```
gos_result_t gos_timeCompare (
    gos_time_t * pTime1,
    gos_time_t * pTime2,
    gos_timeComprareResult_t * result )
```

This function compares two time structures.

This function compares two time structures.

Parameters

in	<i>pTime1</i>	Pointer to the first time variable.
in	<i>pTime2</i>	Pointer to the second time variable.
out	<i>result</i>	Pointer to the comparison result variable.

Returns

Result of time comparison.

Return values

<i>GOS_SUCCESS</i>	Time comparison successful.
<i>GOS_ERROR</i>	Either time structure and/or result variable is NULL pointer.

Definition at line 268 of file gos_time.c.

7.40.3.8 gos_timeDaemonTask()

```
GOS_STATIC void_t gos_timeDaemonTask (
    void_t )
```

Time daemon task.

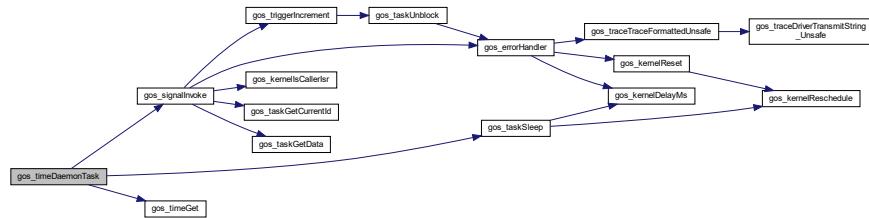
Increases the system time approximately every second and invokes the elapsed signals.

Returns

-

Definition at line 798 of file gos_time.c.

Here is the call graph for this function:



7.40.3.9 gos_timeGet()

```
gos_result_t gos_timeGet (
    gos_time_t * pTime )
```

This function gets the system time.

This function copies the system time value to the given time variable.

Parameters

<code>out</code>	<code>pTime</code>	Pointer to a time variable to store the system time value in.
------------------	--------------------	---

Returns

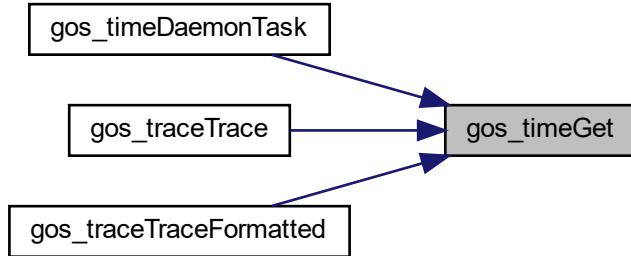
Result of time getting.

Return values

<code>GOS_SUCCESS</code>	Time getting successful.
<code>GOS_ERROR</code>	Time variable is NULL pointer.

Definition at line 187 of file gos_time.c.

Here is the caller graph for this function:



7.40.3.10 `gos_timeIncreaseSystemTime()`

```
gos_result_t gos_timeIncreaseSystemTime (
    u16_t milliseconds )
```

This function increases the system time and runtime with the given value of milliseconds.

This function increases the system time and runtime with the given value of milliseconds.

Parameters

in	<code>milliseconds</code>	Number of milliseconds to add to system time.
----	---------------------------	---

Returns

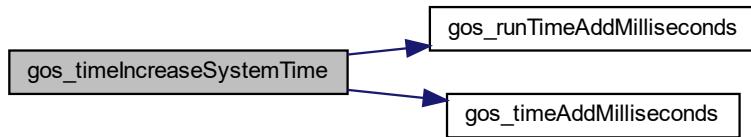
Result of system time increasing.

Return values

<code>GOS_SUCCESS</code>	Increasing successful.
<code>GOS_ERROR</code>	System time or runtime increasing failed.

Definition at line 768 of file `gos_time.c`.

Here is the call graph for this function:



7.40.3.11 gos_timeInit()

```
gos_result_t gos_timeInit (
    void_t )
```

This function initializes the time service.

Creates the time signal and registers the time daemon in the kernel.

Returns

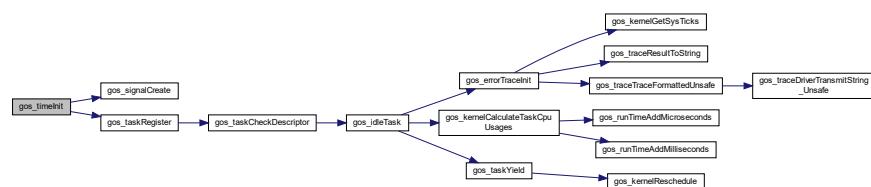
Result of initialization.

Return values

<code>GOS_SUCCESS</code>	Initialization successful.
<code>GOS_ERROR</code>	Time signal registration error or time daemon registration error.

Definition at line 160 of file `gos_time.c`.

Here is the call graph for this function:



7.40.3.12 gos_timeSet()

```
gos_result_t gos_timeSet (
    gos_time_t * pTime )
```

This function sets the system time.

This function copies the time value from the given time variable to the system time variable.

Parameters

in	<i>pTime</i>	Pointer to a time variable holding the desired time value.
----	--------------	--

Returns

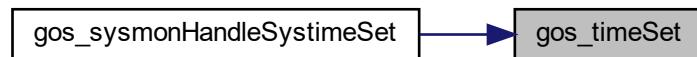
Result of time setting.

Return values

<i>GOS_SUCCESS</i>	Time setting successful.
<i>GOS_ERROR</i>	Time structure is NULL pointer.

Definition at line 214 of file gos_time.c.

Here is the caller graph for this function:



7.40.4 Variable Documentation

7.40.4.1 dayLookupTable

```
GOS_STATIC GOS_CONST gos_day_t dayLookupTable[GOS_TIME_NUMBER_OF_MONTHS]
```

Initial value:

```
=
{
    [GOS_TIME_JANUARY] - 1] = 31,
    [GOS_TIME_FEBRUARY] - 1] = 28,
    [GOS_TIME_MARCH] - 1] = 31,
    [GOS_TIME_APRIIL] - 1] = 30,
    [GOS_TIME_MAY] - 1] = 31,
```

```
[GOS_TIME_JUNE      - 1] = 30,
[GOS_TIME_JULY      - 1] = 31,
[GOS_TIME_AUGUST    - 1] = 31,
[GOS_TIME_SEPTEMBER - 1] = 30,
[GOS_TIME_OCTOBER   - 1] = 31,
[GOS_TIME_NOVEMBER  - 1] = 30,
[GOS_TIME_DECEMBER  - 1] = 31
}
```

Number of days in each month - lookup table.

Definition at line 114 of file gos_time.c.

7.40.4.2 systemRunTime

```
GOS_STATIC gos_runtime_t systemRunTime
```

System run-time.

Definition at line 109 of file gos_time.c.

7.40.4.3 systemTime

```
GOS_STATIC gos_time_t systemTime
```

Initial value:

```
=
{
    .years    = ( 2023 ) ,
    .months   = ( GOS_TIME_JANUARY ) ,
    .days     = ( 1 )
}
```

System time.

Definition at line 99 of file gos_time.c.

7.40.4.4 timeDaemonTaskDesc

```
GOS_STATIC gos_taskDescriptor_t timeDaemonTaskDesc
```

Initial value:

```
=
{
    .taskFunction      = gos_timeDaemonTask,
    .taskName         = "gos_time_daemon",
    .taskStackSize    = CFG_TASK_TIME_DAEMON_STACK,
    .taskPriority     = CFG_TASK_TIME_DAEMON_PRIO,
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL
}
```

Time task descriptor.

Definition at line 148 of file gos_time.c.

7.40.4.5 timeDaemonTaskId

```
GOS_STATIC gos_tid_t timeDaemonTaskId
```

Time task ID.

Definition at line 133 of file gos_time.c.

7.40.4.6 timeSignalId

```
gos_signalId_t timeSignalId
```

Time signal ID. This signal is invoked when a second has elapsed.

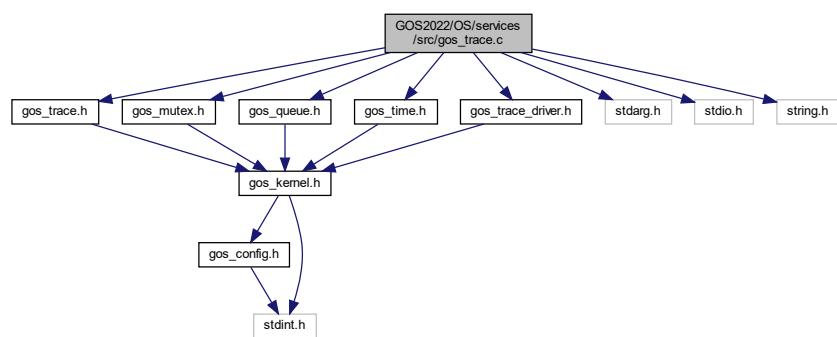
Definition at line 138 of file gos_time.c.

7.41 GOS2022/OS/services/src/gos_trace.c File Reference

GOS trace service source.

```
#include <gos_trace.h>
#include <gos_mutex.h>
#include <gos_queue.h>
#include <gos_time.h>
#include <gos_trace_driver.h>
#include <stdarg.h>
#include <stdio.h>
#include <string.h>
```

Include dependency graph for gos_trace.c:



Macros

- #define **GOS_TRACE_TIMESTAMP_FORMAT** "[\"TRACE_FG_YELLOW_START\"%04d-%02d-%02d %02d:%02d:%02d.%03d\"TRACE_FORMAT_RESET\"]t"
- #define **GOS_TRACE_TIMESTAMP_LENGTH** (46u)
- #define **GOS_TRACE_QUEUE_TMO_MS** (2000u)
- #define **GOS_TRACE_MUTEX_TMO_MS** (2000u)

Functions

- [GOS_STATIC void_t gos_traceDaemonTask \(void_t\)](#)
Trace daemon task.
- [gos_result_t gos_tracelInit \(void_t\)](#)
Initializes the trace service.
- [GOS_INLINE gos_result_t gos_traceTrace \(bool_t addTimeStamp, char_t *traceMessage\)](#)
Traces a given message.
- [gos_result_t gos_traceTraceFormatted \(bool_t addTimeStamp, GOS_CONST char_t *traceFormat,...\)](#)
Traces a given formatted message.
- [gos_result_t gos_traceTraceFormattedUnsafe \(GOS_CONST char_t *traceFormat,...\)](#)
Traces a given formatted message.

Variables

- [GOS_STATIC gos_queueDescriptor_t traceQueue](#)
- [GOS_STATIC char_t traceLine \[CFG_TRACE_MAX_LENGTH\]](#)
- [GOS_STATIC char_t formattedBuffer \[CFG_TRACE_MAX_LENGTH\]](#)
- [GOS_STATIC char_t timeStampBuffer \[\(46u\)\]](#)
- [GOS_STATIC gos_mutex_t traceMutex](#)
- [GOS_STATIC gos_taskDescriptor_t traceDaemonTaskDesc](#)

7.41.1 Detailed Description

GOS trace service source.

Author

Ahmed Gazar

Date

2025-04-06

Version

1.13

For a more detailed description of this service, please refer to [gos_trace.h](#)

7.41.2 Macro Definition Documentation

7.41.2.1 GOS_TRACE_MUTEX_TMO_MS

```
#define GOS_TRACE_MUTEX_TMO_MS ( 2000u )
```

Timeout value in [ms] for mutex operations.

Definition at line 97 of file gos_trace.c.

7.41.2.2 GOS_TRACE_QUEUE_TMO_MS

```
#define GOS_TRACE_QUEUE_TMO_MS ( 2000u )
```

Timeout value in [ms] for queue operations.

Definition at line 92 of file gos_trace.c.

7.41.2.3 GOS_TRACE_TIMESTAMP_FORMAT

```
#define GOS_TRACE_TIMESTAMP_FORMAT "[ "TRACE_FG_YELLOW_START"%04d-%02d-%02d %02d:%02d:%02d.%03d"TRACE_FORMAT_RESET" ]\t"
```

Trace timestamp formatter.

Definition at line 82 of file gos_trace.c.

7.41.2.4 GOS_TRACE_TIMESTAMP_LENGTH

```
#define GOS_TRACE_TIMESTAMP_LENGTH ( 46u )
```

Trace timestamp length.

Definition at line 87 of file gos_trace.c.

7.41.3 Function Documentation

7.41.3.1 gos_traceDaemonTask()

```
GOS_STATIC void_t gos_traceDaemonTask (
    void_t
)
```

Trace daemon task.

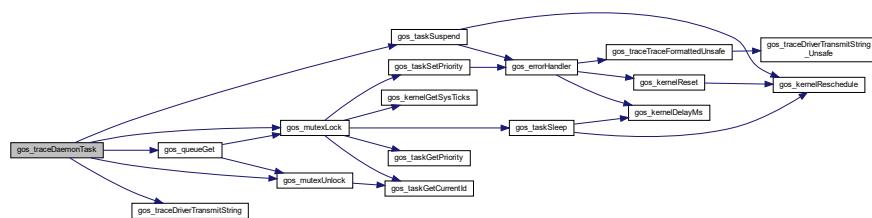
Polls the trace queue and transmits the elements in the trace queue via the registered trace driver.

Returns

-

Definition at line 394 of file gos_trace.c.

Here is the call graph for this function:



7.41.3.2 gos_traceInit()

```
gos_result_t gos_traceInit (
    void_t
)
```

Initializes the trace service.

Creates a trace queue and registers the trace daemon in the kernel.

Returns

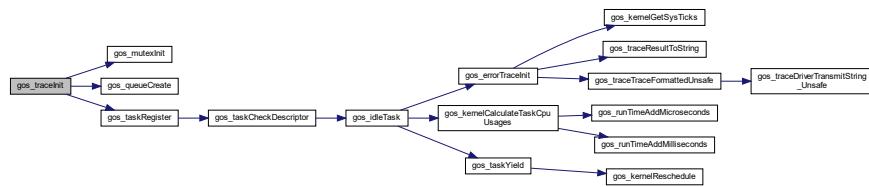
Result of initialization.

Return values

GOS_SUCCESS	Initialization successful.
GOS_ERROR	Queue creation or task registration error.

Definition at line 152 of file gos_trace.c.

Here is the call graph for this function:



7.41.3.3 gos_traceTrace()

```
GOS_INLINE gos_result_t gos_traceTrace (
    bool_t addTimeStamp,
    char_t * traceMessage )
```

Traces a given message.

Places the given message to the trace queue (for the trace daemon to print it).

Parameters

in	<i>addTimeStamp</i>	Flag to indicate whether to add time-stamp or not.
in	<i>traceMessage</i>	String to trace.

Returns

Result of tracing.

Return values

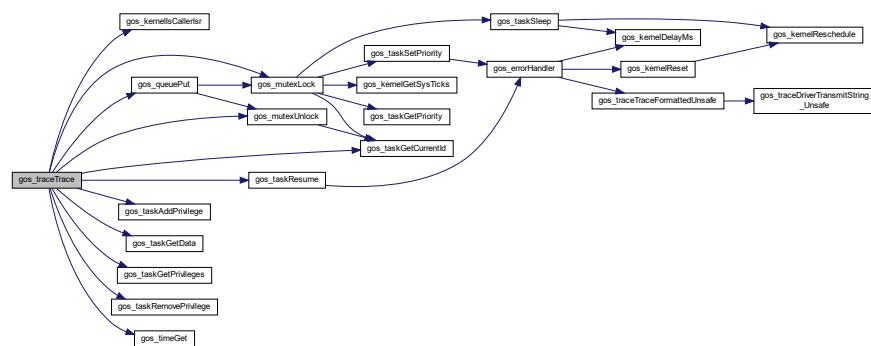
<i>GOS_SUCCESS</i>	Tracing successful.
<i>GOS_ERROR</i>	Queue put error.

Remarks

This function uses the queue service.

Definition at line 177 of file gos_trace.c.

Here is the call graph for this function:



7.41.3.4 gos_traceTraceFormatted()

```
gos_result_t gos_traceTraceFormatted (
    bool_t addTimeStamp,
    GOS_CONST char_t * traceFormat,
    ...
)
```

Traces a given formatted message.

Prints the formatted message into a local buffer and places it to the trace queue (for the trace daemon to print it).

Parameters

in	<i>addTimeStamp</i>	Flag to indicate whether to add time-stamp or not.
in	<i>traceFormat</i>	Formatter string.
in	...	Optional parameters.

Returns

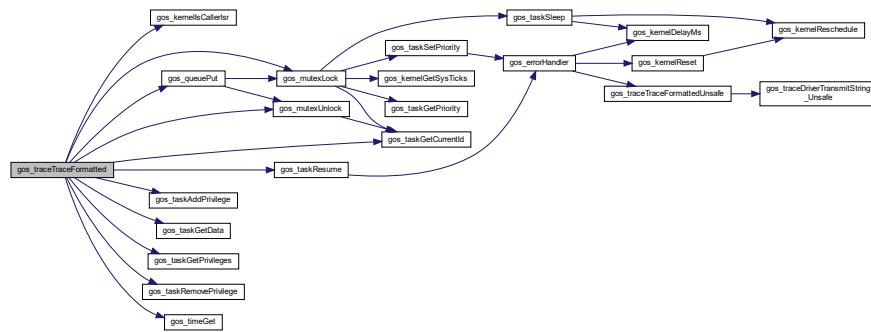
Result of formatted tracing.

Return values

<i>GOS_SUCCESS</i>	Formatted tracing successful.
<i>GOS_ERROR</i>	Queue put error.

Definition at line 266 of file gos_trace.c.

Here is the call graph for this function:



7.41.3.5 gos_traceTraceFormattedUnsafe()

```
gos_result_t gos_traceTraceFormattedUnsafe (
    GOS_CONST char_t * traceFormat,
    ...
)
```

Traces a given formatted message.

Prints the formatted message into a local buffer and transmits it via the trace port.

Parameters

in	<i>traceFormat</i>	Formatter string.
in	...	Optional parameters.

Returns

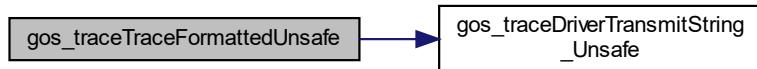
Result of formatted tracing.

Return values

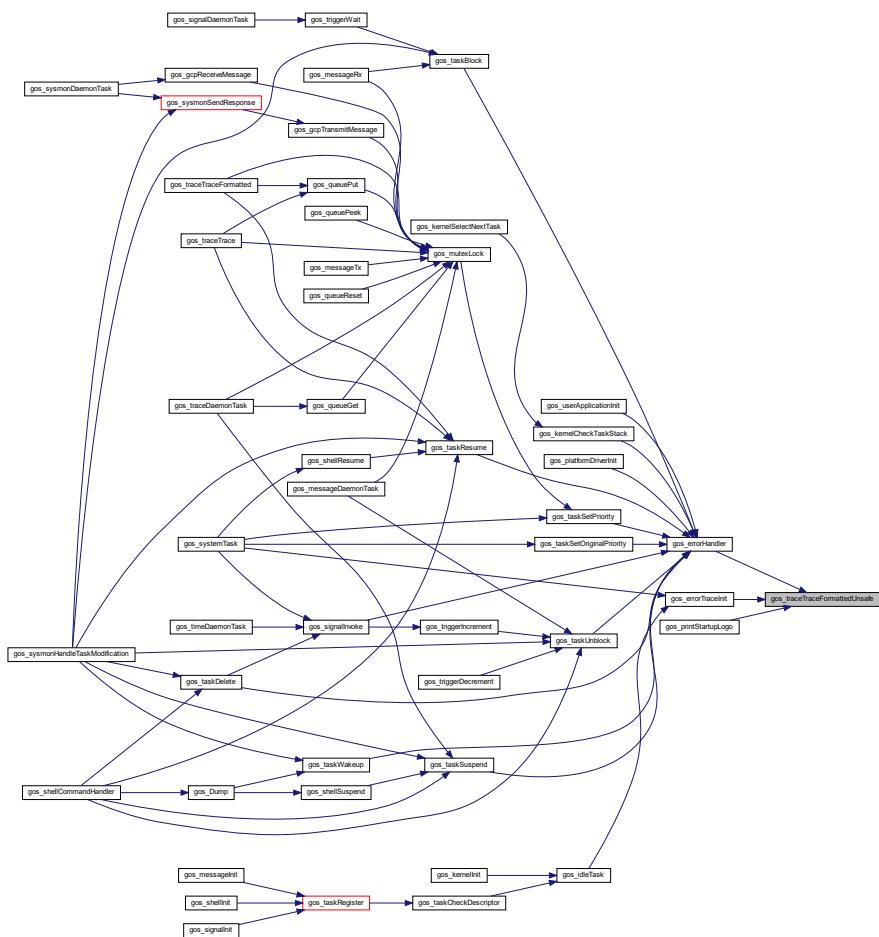
<i>GOS_SUCCESS</i>	Message traced successfully.
<i>GOS_ERROR</i>	Transmit error.

Definition at line 356 of file gos_trace.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.41.4 Variable Documentation

7.41.4.1 formattedBuffer

```
GOS_STATIC char_t formattedBuffer[CFG_TRACE_MAX_LENGTH]
```

Trace formatted buffer for message formatting.

Definition at line 120 of file gos_trace.c.

7.41.4.2 timeStampBuffer

```
GOS_STATIC char_t timeStampBuffer[(46u)]
```

Buffer for timestamp printing.

Definition at line 125 of file gos_trace.c.

7.41.4.3 traceDaemonTaskDesc

```
GOS_STATIC gos_taskDescriptor_t traceDaemonTaskDesc
```

Initial value:

```
=  
{  
    .taskFunction      = gos_traceDaemonTask,  
    .taskName         = "gos_trace_daemon",  
    .taskPriority     = CFG_TASK_TRACE_DAEMON_PRIO,  
    .taskStackSize    = CFG_TASK_TRACE_DAEMON_STACK,  
    .taskPrivilegeLevel = GOS_TASK_PRIVILEGE_KERNEL  
}
```

Trace daemon task descriptor.

Definition at line 140 of file gos_trace.c.

7.41.4.4 traceLine

```
GOS_STATIC char_t traceLine[CFG_TRACE_MAX_LENGTH]
```

Trace line buffer.

Definition at line 115 of file gos_trace.c.

7.41.4.5 traceMutex

```
GOS_STATIC gos_mutex_t traceMutex
```

Trace mutex.

Definition at line 130 of file gos_trace.c.

7.41.4.6 traceQueue

```
GOS_STATIC gos_queueDescriptor_t traceQueue
```

Initial value:

```
=  
{  
}
```

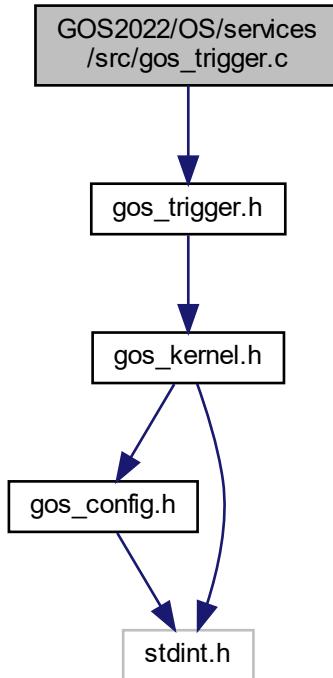
Trace queue.

Definition at line 105 of file gos_trace.c.

7.42 GOS2022/OS/services/src/gos_trigger.c File Reference

GOS trigger service source.

```
#include <gos_trigger.h>  
Include dependency graph for gos_trigger.c:
```



Functions

- `gos_result_t gos_triggerInit (gos_trigger_t *pTrigger)`
Initializes the trigger instance.
- `GOS_INLINE gos_result_t gos_triggerReset (gos_trigger_t *pTrigger)`
Resets the trigger counter of the given trigger instance.
- `GOS_INLINE gos_result_t gos_triggerWait (gos_trigger_t *pTrigger, u32_t value, u32_t timeout)`
Waits for the trigger instance to reach the given trigger value.
- `GOS_INLINE gos_result_t gos_triggerIncrement (gos_trigger_t *pTrigger)`
Increments the trigger value of the given trigger.
- `GOS_INLINE gos_result_t gos_triggerDecrement (gos_trigger_t *pTrigger)`
Decrements the trigger value of the given trigger.

7.42.1 Detailed Description

GOS trigger service source.

Author

Ahmed Gazar

Date

2025-03-28

Version

2.10

For a more detailed description of this service, please refer to [gos_trigger.h](#)

7.42.2 Function Documentation

7.42.2.1 gos_triggerDecrement()

```
GOS_INLINE gos_result_t gos_triggerDecrement (
    gos_trigger_t * pTrigger )
```

Decrements the trigger value of the given trigger.

Decrements the trigger value of the given trigger.

Parameters

in, out	<code>pTrigger</code>	Pointer to the trigger instance.
---------	-----------------------	----------------------------------

Returns

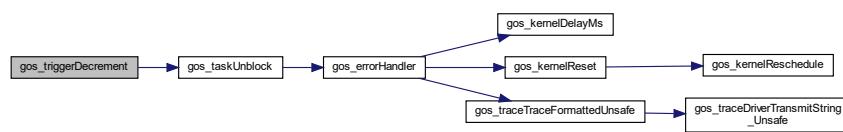
Result of trigger decrementing.

Return values

<i>GOS_SUCCESS</i>	Decrementing successful.
<i>GOS_ERROR</i>	Trigger is NULL pointer or value is already zero.

Definition at line 246 of file gos_trigger.c.

Here is the call graph for this function:

**7.42.2.2 gos_triggerIncrement()**

```
GOS_INLINE gos_result_t gos_triggerIncrement (
    gos_trigger_t * pTrigger )
```

Increments the trigger value of the given trigger.

Increments the trigger value of the given trigger.

Parameters

<i>in, out</i>	<i>pTrigger</i>	Pointer to the trigger instance.
----------------	-----------------	----------------------------------

Returns

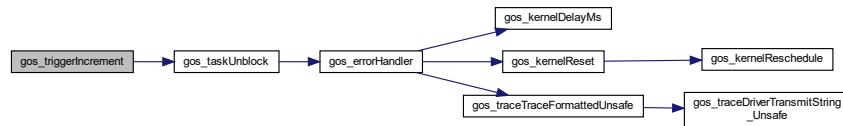
Result of trigger incrementing.

Return values

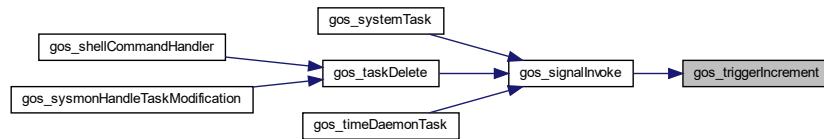
<i>GOS_SUCCESS</i>	Incrementing successful.
<i>GOS_ERROR</i>	Trigger is NULL pointer.

Definition at line 199 of file gos_trigger.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.2.3 gos_triggerInit()

```
gos_result_t gos_triggerInit (
    gos_trigger_t * pTrigger )
```

Initializes the trigger instance.

Calls the initializer for the trigger mutex.

Parameters

out	<i>pTrigger</i>	Pointer to the trigger to be initialized.
-----	-----------------	---

Returns

Result of trigger initializing.

Return values

<i>GOS_SUCCESS</i>	Trigger initialized successfully.
<i>GOS_ERROR</i>	Trigger descriptor or trigger mutex is NULL pointer.

Definition at line 78 of file gos_trigger.c.

Here is the caller graph for this function:



7.42.2.4 gos_triggerReset()

```
GOS_INLINE gos_result_t gos_triggerReset (
    gos_trigger_t * pTrigger )
```

Resets the trigger counter of the given trigger instance.

Sets the trigger counter of the given trigger instance to zero.

Parameters

out	<i>pTrigger</i>	Pointer to the trigger instance.
-----	-----------------	----------------------------------

Returns

Result of trigger resetting.

Return values

<i>GOS_SUCCESS</i>	Trigger resetting successful.
<i>GOS_ERROR</i>	Trigger is NULL pointer.

Definition at line 107 of file gos_trigger.c.

Here is the caller graph for this function:



7.42.2.5 gos_triggerWait()

```
GOS_INLINE gos_result_t gos_triggerWait (
    gos_trigger_t * pTrigger,
    u32_t value,
    u32_t timeout )
```

Waits for the trigger instance to reach the given trigger value.

Increases the trigger waiter number, and waits in a non-blocking way until the desired trigger value is reached or the timeout is reached.

Parameters

in, out	<i>pTrigger</i>	Pointer to the trigger instance.
in	<i>value</i>	The desired trigger value.
in	<i>timeout</i>	Timeout value.

Returns

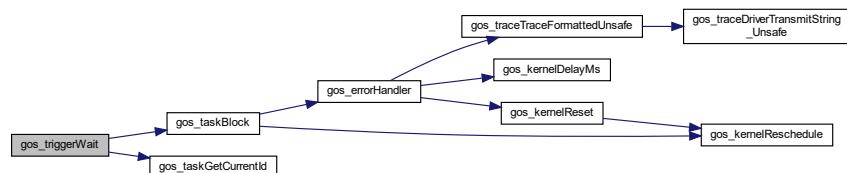
Result of trigger waiting.

Return values

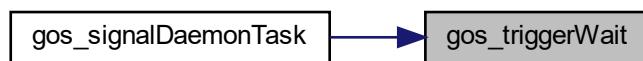
<i>GOS_SUCCESS</i>	Trigger value reached.
<i>GOS_ERROR</i>	Trigger value was not reached within the timeout value.

Definition at line 140 of file gos_trigger.c.

Here is the call graph for this function:



Here is the caller graph for this function:



Index

actualCommand
 gos_shell.c, 313

ARM_CORTEX_M4
 gos_bootloader_config.h, 123
 gos_config.h, 134

atomicCntr
 gos_kernel.c, 166

Basic data type definitions to be used in a GOS system,
 18

BINARY_PATTERN
 gos_kernel.c, 161

callerTaskDesc
 gos_signal.c, 320

CFG_GCP_CHANNELS_MAX_NUMBER
 gos_bootloader_config.h, 123
 gos_config.h, 135

CFG_IDLE_TASK_STACK_SIZE
 gos_bootloader_config.h, 123
 gos_config.h, 135

CFG_MESSAGE_MAX_ADDRESSEES
 gos_config.h, 135

CFG_MESSAGE_MAX_LENGTH
 gos_bootloader_config.h, 123
 gos_config.h, 135

CFG_MESSAGE_MAX_NUMBER
 gos_bootloader_config.h, 124
 gos_config.h, 135

CFG_MESSAGE_MAX_WAITER_IDS
 gos_bootloader_config.h, 124
 gos_config.h, 136

CFG_MESSAGE_MAX_WAITERS
 gos_bootloader_config.h, 124
 gos_config.h, 136

CFG_QUEUE_MAX_ELEMENTS
 gos_bootloader_config.h, 124
 gos_config.h, 136

CFG_QUEUE_MAX_LENGTH
 gos_bootloader_config.h, 124
 gos_config.h, 136

CFG_QUEUE_MAX_NAME_LENGTH
 gos_bootloader_config.h, 125
 gos_config.h, 136

CFG_QUEUE_MAX_NUMBER
 gos_bootloader_config.h, 125
 gos_config.h, 137

CFG_QUEUE_USE_NAME
 gos_bootloader_config.h, 125
 gos_config.h, 137

CFG_RESET_ON_ERROR
 gos_bootloader_config.h, 125
 gos_config.h, 137

CFG_RESET_ON_ERROR_DELAY_MS
 gos_bootloader_config.h, 125
 gos_config.h, 137

CFG_SCHED_COOPERATIVE
 gos_bootloader_config.h, 126
 gos_config.h, 137

CFG_SHELL_COMMAND_BUFFER_SIZE
 gos_bootloader_config.h, 126
 gos_config.h, 138

CFG_SHELL_MAX_COMMAND_LENGTH
 gos_bootloader_config.h, 126
 gos_config.h, 138

CFG_SHELL_MAX_COMMAND_NUMBER
 gos_bootloader_config.h, 126
 gos_config.h, 138

CFG_SHELL_MAX_PARAMS_LENGTH
 gos_bootloader_config.h, 126
 gos_config.h, 138

CFG_SHELL_STARTUP_DELAY_MS
 gos_bootloader_config.h, 127
 gos_config.h, 138

CFG_SHELL_USE_SERVICE
 gos_bootloader_config.h, 127
 gos_config.h, 139

CFG_SIGNAL_MAX_NUMBER
 gos_bootloader_config.h, 127
 gos_config.h, 139

CFG_SIGNAL_MAX_SUBSCRIBERS
 gos_bootloader_config.h, 127
 gos_config.h, 139

CFG_SYSMON_GCP_CHANNEL_NUM
 gos_bootloader_config.h, 127
 gos_config.h, 139

CFG_SYSMON_MAX_USER_MESSAGES
 gos_bootloader_config.h, 128
 gos_config.h, 139

CFG_SYSMON_USE_SERVICE
 gos_bootloader_config.h, 128
 gos_config.h, 140

CFG_SYSTEM_TASK_STACK_SIZE
 gos_bootloader_config.h, 128
 gos_config.h, 140

CFG_TARGET_CPU
 gos_bootloader_config.h, 128
 gos_config.h, 140

CFG_TASK_MAX_NAME_LENGTH

gos_bootloader_config.h, 128
 gos_config.h, 140
CFG_TASK_MAX_NUMBER
 gos_bootloader_config.h, 129
 gos_config.h, 140
CFG_TASK_MAX_STACK_SIZE
 gos_bootloader_config.h, 129
 gos_config.h, 141
CFG_TASK_MESSAGE_DAEMON_PRIO
 gos_bootloader_config.h, 129
 gos_config.h, 141
CFG_TASK_MESSAGE_DAEMON_STACK
 gos_bootloader_config.h, 129
 gos_config.h, 141
CFG_TASK_MIN_STACK_SIZE
 gos_bootloader_config.h, 129
 gos_config.h, 141
CFG_TASK_SHELL_DAEMON_PRIO
 gos_bootloader_config.h, 130
 gos_config.h, 141
CFG_TASK_SHELL_DAEMON_STACK
 gos_bootloader_config.h, 130
 gos_config.h, 142
CFG_TASK_SIGNAL_DAEMON_PRIO
 gos_bootloader_config.h, 130
 gos_config.h, 142
CFG_TASK_SIGNAL_DAEMON_STACK
 gos_bootloader_config.h, 130
 gos_config.h, 142
CFG_TASK_SYS_PRIO
 gos_bootloader_config.h, 130
 gos_config.h, 142
CFG_TASK_SYSMON_DAEMON_PRIO
 gos_bootloader_config.h, 131
 gos_config.h, 142
CFG_TASK_SYSMON_DAEMON_STACK
 gos_bootloader_config.h, 131
 gos_config.h, 143
CFG_TASK_TIME_DAEMON_PRIO
 gos_bootloader_config.h, 131
 gos_config.h, 143
CFG_TASK_TIME_DAEMON_STACK
 gos_bootloader_config.h, 131
 gos_config.h, 143
CFG_TASK_TRACE_DAEMON_PRIO
 gos_bootloader_config.h, 131
 gos_config.h, 143
CFG_TASK_TRACE_DAEMON_STACK
 gos_bootloader_config.h, 132
 gos_config.h, 143
CFG_TRACE_MAX_LENGTH
 gos_bootloader_config.h, 132
 gos_config.h, 144
CFG_USE_PRIO_INHERITANCE
 gos_bootloader_config.h, 132
 gos_config.h, 144
channelFunctions
 gos_gcp.c, 280

 commandBuffer
 gos_shell.c, 313
commandBufferIndex
 gos_shell.c, 313
commandParams
 gos_shell.c, 313
CONFIG_DUMP_SEPARATOR
 gos_kernel.c, 161
Control macros for scheduling and ISR state setting, 14
 GOS_ATOMIC_ENTER, 14
 GOS_ATOMIC_EXIT, 14
 GOS_DISABLE_SCHED, 14
 GOS_ENABLE_SCHED, 15
 GOS_ISR_ENTER, 15
 GOS_ISR_EXIT, 15
cpuMessage
 gos_sysmon.c, 337
cpuUseLimit
 gos_kernel.c, 166
CRC_INITIAL_VALUE
 gos_crc_driver.c, 107
CRC_POLYNOMIAL_VALUE
 gos_crc_driver.c, 107
currentTaskIndex
 gos_kernel.c, 167

dayLookupTable
 gos_time.c, 352
DUMP_SEPARATOR
 gos_queue.c, 294
dumpRequired
 gos.c, 260

ERROR_BUFFER_SIZE
 gos_error.c, 263
errorBuffer
 gos_error.c, 268

formattedBuffer
 gos_shell_driver.c, 112
 gos_trace.c, 361

GCP_ACK_CRC_ERROR
 gos_gcp.c, 271
GCP_ACK_INVALID
 gos_gcp.c, 271
GCP_ACK_OK
 gos_gcp.c, 271
GCP_ACK_PV_ERROR
 gos_gcp.c, 271
GCP_ACK_REQ
 gos_gcp.c, 271
GCP_ACK_RESEND
 gos_gcp.c, 271
GCP_ACK_SIZE_ERROR
 gos_gcp.c, 271
GCP_PROTOCOL_VERSION_MAJOR
 gos_gcp.c, 271
GCP_PROTOCOL_VERSION_MINOR

gos_gcp.c, 271
gcpRxMutexes
 gos_gcp.c, 280
gcpTxMutexes
 gos_gcp.c, 280
Global definitions, 9
 GLOBAL_STACK, 9
 GOS_ASM, 10
 GOS_CONST, 10
 GOS_DEFAULT_TASK_ID, 10
 GOS_EXTERN, 10
 GOS_INLINE, 10
 GOS_INVALID_TASK_ID, 11
 GOS_NAKED, 11
 GOS_NOP, 11
 GOS_STATIC, 11
 GOS_STATIC_INLINE, 11
 GOS_TASK_IDLE_PRIO, 12
 GOS_TASK_MAX_BLOCK_TIME_MS, 12
 GOS_TASK_MAX_PRIO_LEVELS, 12
 GOS_UNUSED, 12
 MAIN_STACK, 12
 NULL, 13
 RAM_SIZE, 13
 RAM_START, 13
GLOBAL_STACK
 Global definitions, 9
gos.c
 dumpRequired, 260
 gos_Dump, 258
 gos_initFunc_t, 258
 gos_Start, 259
 GOS_SYS_TASK_SLEEP_TIME, 257
 gos_systemTask, 259
 initError, 260
 initializers, 261
 systemTaskDesc, 261
 systemTaskId, 261
gos.h
 gos_Dump, 178
 GOS_VERSION_MAJOR, 177
 GOS_VERSION_MINOR, 178
GOS2022/OS/driver/inc/gos_crc_driver.h, 87
GOS2022/OS/driver/inc/gos_driver.h, 89
GOS2022/OS/driver/inc/gos_shell_driver.h, 91
GOS2022/OS/driver/inc/gos_sysmon_driver.h, 95
GOS2022/OS/driver/inc/gos_timer_driver.h, 99
GOS2022/OS/driver/inc/gos_trace_driver.h, 101
GOS2022/OS/driver/src/gos_crc_driver.c, 105
GOS2022/OS/driver/src/gos_driver.c, 108
GOS2022/OS/driver/src/gos_shell_driver.c, 110
GOS2022/OS/driver/src/gos_sysmon_driver.c, 113
GOS2022/OS/driver/src/gos_timer_driver.c, 116
GOS2022/OS/driver/src/gos_trace_driver.c, 118
GOS2022/OS/kernel/inc/gos_bootloader_config.h, 121
GOS2022/OS/kernel/inc/gos_config.h, 133
GOS2022/OS/kernel/inc/gos_kernel.h, 144
GOS2022/OS/kernel/inc/gos_port.h, 153
GOS2022/OS/kernel/src/gos_kernel.c, 158
GOS2022/OS/kernel/src/gos_task.c, 170
GOS2022/OS/services/inc/gos.h, 176
GOS2022/OS/services/inc/gos_error.h, 179
GOS2022/OS/services/inc/gos_gcp.h, 184
GOS2022/OS/services/inc/gos_message.h, 190
GOS2022/OS/services/inc/gos_mutex.h, 195
GOS2022/OS/services/inc/gos_queue.h, 201
GOS2022/OS/services/inc/gos_shell.h, 214
GOS2022/OS/services/inc/gos_signal.h, 222
GOS2022/OS/services/inc/gos_sysmon.h, 228
GOS2022/OS/services/inc/gos_time.h, 231
GOS2022/OS/services/inc/gos_trace.h, 243
GOS2022/OS/services/inc/gos_trigger.h, 249
GOS2022/OS/services/src/gos.c, 256
GOS2022/OS/services/src/gos_error.c, 262
GOS2022/OS/services/src/gos_gcp.c, 269
GOS2022/OS/services/src/gos_message.c, 280
GOS2022/OS/services/src/gos_mutex.c, 287
GOS2022/OS/services/src/gos_queue.c, 292
GOS2022/OS/services/src/gos_shell.c, 304
GOS2022/OS/services/src/gos_signal.c, 315
GOS2022/OS/services/src/gos_sysmon.c, 321
GOS2022/OS/services/src/gos_time.c, 341
GOS2022/OS/services/src/gos_trace.c, 354
GOS2022/OS/services/src/gos_trigger.c, 363
GOS_ASM
 Global definitions, 10
GOS_ATOMIC_ENTER
 Control macros for scheduling and ISR state setting, 14
GOS_ATOMIC_EXIT
 Control macros for scheduling and ISR state setting, 14
gos_boolValue_t
 gos_kernel.h, 150
gos_bootloader_config.h
 ARM_CORTEX_M4, 123
 CFG_GCP_CHANNELS_MAX_NUMBER, 123
 CFG_IDLE_TASK_STACK_SIZE, 123
 CFG_MESSAGE_MAX_LENGTH, 123
 CFG_MESSAGE_MAX_NUMBER, 124
 CFG_MESSAGE_MAX_WAITER_IDS, 124
 CFG_MESSAGE_MAX_WAITERS, 124
 CFG_QUEUE_MAX_ELEMENTS, 124
 CFG_QUEUE_MAX_LENGTH, 124
 CFG_QUEUE_MAX_NAME_LENGTH, 125
 CFG_QUEUE_MAX_NUMBER, 125
 CFG_QUEUE_USE_NAME, 125
 CFG_RESET_ON_ERROR, 125
 CFG_RESET_ON_ERROR_DELAY_MS, 125
 CFG_SCHED_COOPERATIVE, 126
 CFG_SHELL_COMMAND_BUFFER_SIZE, 126
 CFG_SHELL_MAX_COMMAND_LENGTH, 126
 CFG_SHELL_MAX_COMMAND_NUMBER, 126
 CFG_SHELL_MAX_PARAMS_LENGTH, 126
 CFG_SHELL_STARTUP_DELAY_MS, 127
 CFG_SHELL_USE_SERVICE, 127

CFG_SIGNAL_MAX_NUMBER, 127
 CFG_SIGNAL_MAX_SUBSCRIBERS, 127
 CFG_SYSMON_GCP_CHANNEL_NUM, 127
 CFG_SYSMON_MAX_USER_MESSAGES, 128
 CFG_SYSMON_USE_SERVICE, 128
 CFG_SYSTEM_TASK_STACK_SIZE, 128
 CFG_TARGET_CPU, 128
 CFG_TASK_MAX_NAME_LENGTH, 128
 CFG_TASK_MAX_NUMBER, 129
 CFG_TASK_MAX_STACK_SIZE, 129
 CFG_TASK_MESSAGE_DAEMON_PRIO, 129
 CFG_TASK_MESSAGE_DAEMON_STACK, 129
 CFG_TASK_MIN_STACK_SIZE, 129
 CFG_TASK_SHELL_DAEMON_PRIO, 130
 CFG_TASK_SHELL_DAEMON_STACK, 130
 CFG_TASK_SIGNAL_DAEMON_PRIO, 130
 CFG_TASK_SIGNAL_DAEMON_STACK, 130
 CFG_TASK_SYS_PRIO, 130
 CFG_TASK_SYSMON_DAEMON_PRIO, 131
 CFG_TASK_SYSMON_DAEMON_STACK, 131
 CFG_TASK_TIME_DAEMON_PRIO, 131
 CFG_TASK_TIME_DAEMON_STACK, 131
 CFG_TASK_TRACE_DAEMON_PRIO, 131
 CFG_TASK_TRACE_DAEMON_STACK, 132
 CFG_TRACE_MAX_LENGTH, 132
 CFG_USE_PRIO_INHERITANCE, 132
GOS_BUSY
 gos_kernel.h, 151
GOS_CONCAT_RESULT
 gos_kernel.h, 150
gos_config.h
 ARM_CORTEX_M4, 134
 CFG_GCP_CHANNELS_MAX_NUMBER, 135
 CFG_IDLE_TASK_STACK_SIZE, 135
 CFG_MESSAGE_MAX_ADDRESSEES, 135
 CFG_MESSAGE_MAX_LENGTH, 135
 CFG_MESSAGE_MAX_NUMBER, 135
 CFG_MESSAGE_MAX_WAITER_IDS, 136
 CFG_MESSAGE_MAX_WAITERS, 136
 CFG_QUEUE_MAX_ELEMENTS, 136
 CFG_QUEUE_MAX_LENGTH, 136
 CFG_QUEUE_MAX_NAME_LENGTH, 136
 CFG_QUEUE_MAX_NUMBER, 137
 CFG_QUEUE_USE_NAME, 137
 CFG_RESET_ON_ERROR, 137
 CFG_RESET_ON_ERROR_DELAY_MS, 137
 CFG_SCHED_COOPERATIVE, 137
 CFG_SHELL_COMMAND_BUFFER_SIZE, 138
 CFG_SHELL_MAX_COMMAND_LENGTH, 138
 CFG_SHELL_MAX_COMMAND_NUMBER, 138
 CFG_SHELL_MAX_PARAMS_LENGTH, 138
 CFG_SHELL_STARTUP_DELAY_MS, 138
 CFG_SHELL_USE_SERVICE, 139
 CFG_SIGNAL_MAX_NUMBER, 139
 CFG_SIGNAL_MAX_SUBSCRIBERS, 139
 CFG_SYSMON_GCP_CHANNEL_NUM, 139
 CFG_SYSMON_MAX_USER_MESSAGES, 139
 CFG_SYSMON_USE_SERVICE, 140
 CFG_SYSTEM_TASK_STACK_SIZE, 140
 CFG_TARGET_CPU, 140
 CFG_TASK_MAX_NAME_LENGTH, 140
 CFG_TASK_MAX_NUMBER, 140
 CFG_TASK_MAX_STACK_SIZE, 141
 CFG_TASK_MESSAGE_DAEMON_PRIO, 141
 CFG_TASK_MESSAGE_DAEMON_STACK, 141
 CFG_TASK_MIN_STACK_SIZE, 141
 CFG_TASK_SHELL_DAEMON_PRIO, 141
 CFG_TASK_SHELL_DAEMON_STACK, 142
 CFG_TASK_SIGNAL_DAEMON_PRIO, 142
 CFG_TASK_SIGNAL_DAEMON_STACK, 142
 CFG_TASK_SYS_PRIO, 142
 CFG_TASK_SYSMON_DAEMON_PRIO, 142
 CFG_TASK_SYSMON_DAEMON_STACK, 143
 CFG_TASK_TIME_DAEMON_PRIO, 143
 CFG_TASK_TIME_DAEMON_STACK, 143
 CFG_TASK_TRACE_DAEMON_PRIO, 143
 CFG_TASK_TRACE_DAEMON_STACK, 143
 CFG_TRACE_MAX_LENGTH, 144
 CFG_USE_PRIO_INHERITANCE, 144
GOS_CONST
 Global definitions, 10
gos_crc_driver.c
 CRC_INITIAL_VALUE, 107
 CRC_POLYNOMIAL_VALUE, 107
 gos_crcDriverGetCrc, 107
gos_crc_driver.h
 gos_crcDriverGetCrc, 88
gos_crcDriverGetCrc
 gos_crc_driver.c, 107
 gos_crc_driver.h, 88
GOS_DEFAULT_QUEUE_ID
 gos_queue.h, 204
GOS_DEFAULT_TASK_ID
 Global definitions, 10
GOS_DISABLE_SCHED
 Control macros for scheduling and ISR state setting, 14
gos_driver.c
 gos_driverInit, 109
gos_driver.h
 gos_driverInit, 90
gos_driver_functions_t, 67
gos_driverInit
 gos_driver.c, 109
 gos_driver.h, 90
gos_Dump
 gos.c, 258
 gos.h, 178
GOS_ENABLE_SCHED
 Control macros for scheduling and ISR state setting, 15
GOS_ERROR
 gos_kernel.h, 151
gos_error.c
 ERROR_BUFFER_SIZE, 263
 errorBuffer, 268

gos_errorHandler, 264
gos_errorTraceInit, 266
gos_printStartupLogo, 267
gos_traceResultToString, 268
RESULT_STRING_ERROR, 263
RESULT_STRING_SUCCESS, 264
RESULT_STRING_UNKNOWN, 264
SEPARATOR_LINE, 264

gos_error.h
 GOS_ERROR_LEVEL_OS_FATAL, 180
 GOS_ERROR_LEVEL_OS_WARNING, 180
 GOS_ERROR_LEVEL_USER_FATAL, 180
 GOS_ERROR_LEVEL_USER_WARNING, 180
 gos_errorHandler, 181
 gos_errorLevel_t, 180
 gos_errorTraceInit, 182
 gos_printStartupLogo, 183

GOS_ERROR_LEVEL_OS_FATAL
 gos_error.h, 180

GOS_ERROR_LEVEL_OS_WARNING
 gos_error.h, 180

GOS_ERROR_LEVEL_USER_FATAL
 gos_error.h, 180

GOS_ERROR_LEVEL_USER_WARNING
 gos_error.h, 180

gos_errorHandler
 gos_error.c, 264
 gos_error.h, 181

gos_errorLevel_t
 gos_error.h, 180

gos_errorTraceInit
 gos_error.c, 266
 gos_error.h, 182

GOS_EXTERN
 Global definitions, 10

GOS_FALSE
 gos_kernel.h, 151

gos_gcp.c
 channelFunctions, 280
 GCP_ACK_CRC_ERROR, 271
 GCP_ACK_INVALID, 271
 GCP_ACK_OK, 271
 GCP_ACK_PV_ERROR, 271
 GCP_ACK_REQ, 271
 GCP_ACK_RESEND, 271
 GCP_ACK_SIZE_ERROR, 271
 GCP_PROTOCOL_VERSION_MAJOR, 271
 GCP_PROTOCOL_VERSION_MINOR, 271
 gcpRxMutexes, 280
 gcpTxMutexes, 280
 gos_gcpAck_t, 271
 gos_gcpInit, 271
 gos_gcpReceiveMessage, 272
 gos_gcpReceiveMessageInternal, 273
 gos_gcpRegisterPhysicalDriver, 274
 gos_gcpTransmitMessage, 275
 gos_gcpTransmitMessageInternal, 276
 gos_gcpValidateData, 278
 gos_gcpValidateHeader, 278

gos_gcp.h
 gos_gcpInit, 186
 gos_gcpReceiveFunction_t, 185
 gos_gcpReceiveMessage, 186
 gos_gcpRegisterPhysicalDriver, 187
 gos_gcpTransmitFunction_t, 185
 gos_gcpTransmitMessage, 188

gos_gcpAck_t
 gos_gcp.c, 271

gos_gcpChannelFunctions_t, 68

gos_gcpHeaderFrame_t, 68

gos_gcpInit
 gos_gcp.c, 271
 gos_gcp.h, 186

gos_gcpReceiveFunction_t
 gos_gcp.h, 185

gos_gcpReceiveMessage
 gos_gcp.c, 272
 gos_gcp.h, 186

gos_gcpReceiveMessageInternal
 gos_gcp.c, 273

gos_gcpRegisterPhysicalDriver
 gos_gcp.c, 274
 gos_gcp.h, 187

gos_gcpTransmitFunction_t
 gos_gcp.h, 185

gos_gcpTransmitMessage
 gos_gcp.c, 275
 gos_gcp.h, 188

gos_gcpTransmitMessageInternal
 gos_gcp.c, 276

gos_gcpValidateData
 gos_gcp.c, 278

gos_gcpValidateHeader
 gos_gcp.c, 278

gos_idleTask
 gos_kernel.c, 163
 gos_task.c, 173

gos_initFunc_t
 gos.c, 258

gos_initStruct_t, 69

GOS_INLINE
 Global definitions, 10

GOS_INVALID_QUEUE_ID
 gos_queue.h, 204

GOS_INVALID_TASK_ID
 Global definitions, 11

GOS_ISR_ENTER
 Control macros for scheduling and ISR state setting, 15

GOS_ISR_EXIT
 Control macros for scheduling and ISR state setting, 15

gos_kernel.c
 atomicCntr, 166
 BINARY_PATTERN, 161
 CONFIG_DUMP_SEPARATOR, 161

cpuUseLimit, 166
currentTaskIndex, 167
gos_idleTask, 163
gos_kernelCheckTaskStack, 163
gos_kernelGetCurrentPsp, 164
gos_kernelGetTaskStateString, 164
gos_kernelProcessorReset, 165
gos_kernelSaveCurrentPsp, 165
gos_kernelSelectNextTask, 166
ICSR, 161
inlsr, 167
isKernelRunning, 167
kernelDumpReadySignal, 167
kernelDumpSignal, 167
kernelPreResetHookFunction, 168
kernelPrivilegedHookFunction, 168
kernelSwapHookFunction, 168
kernelSysTickHookFunction, 168
MAX_CPU_DUMP_SEPARATOR, 161
monitoringTime, 168
previousTick, 169
primask, 169
privilegedModeSetRequired, 169
resetRequired, 169
schedDisableCntr, 169
SHCSR, 162
STACK_STATS_SEPARATOR, 162
sysTicks, 170
sysTimerValue, 170
TASK_DUMP_SEPARATOR, 162
TO_BINARY, 162
gos_kernel.h
 gos_boolValue_t, 150
 GOS_BUSY, 151
 GOS_CONCAT_RESULT, 150
 GOS_ERROR, 151
 GOS_FALSE, 151
 gos_kernel_privilege_t, 151
 GOS_PRIVILEGED, 151
 gos_result_t, 151
 GOS_SUCCESS, 151
 GOS_TASK_BLOCKED, 152
 GOS_TASK_PRIVILEGE_KERNEL, 152
 GOS_TASK_PRIVILEGE_SUPERVISOR, 152
 GOS_TASK_PRIVILEGE_USER, 152
 GOS_TASK_PRIVILEGED_USER, 152
 GOS_TASK_READY, 152
 GOS_TASK_SLEEPING, 152
 GOS_TASK_SUSPENDED, 152
 GOS_TASK_ZOMBIE, 152
 gos_taskPrivilegeLevel_t, 151
 gos_taskState_t, 152
 GOS_TRUE, 151
 GOS_UNPRIVILEGED, 151
gos_kernel_privilege_t
 gos_kernel.h, 151
gos_kernelCalculateTaskCpuUsages
 Kernel functions, 23
gos_kernelCheckTaskStack
 gos_kernel.c, 163
gos_kernelDelayMs
 Kernel functions, 24
gos_kernelDelayUs
 Kernel functions, 25
gos_kernelDump
 Kernel functions, 25
gos_kernelGetCpuUsage
 Kernel functions, 26
gos_kernelGetCurrentPsp
 gos_kernel.c, 164
gos_kernelGetMaxCpuLoad
 Kernel functions, 27
gos_kernelGetSysTicks
 Kernel functions, 27
gos_kernelGetTaskStateString
 gos_kernel.c, 164
gos_kernellInit
 Kernel functions, 28
gos_kernellsCallerLsr
 Kernel functions, 29
gos_kernelPrivilegedModeSetRequired
 Kernel functions, 30
gos_kernelProcessorReset
 gos_kernel.c, 165
gos_kernelRegisterIdleHook
 Kernel functions, 30
gos_kernelRegisterPreResetHook
 Kernel functions, 31
gos_kernelRegisterPrivilegedHook
 Kernel functions, 31
gos_kernelRegisterSwapHook
 Kernel functions, 32
gos_kernelRegisterSysTickHook
 Kernel functions, 32
gos_kernelReschedule
 Kernel functions, 33
gos_kernelReset
 Kernel functions, 34
gos_kernelSaveCurrentPsp
 gos_kernel.c, 165
gos_kernelSelectNextTask
 gos_kernel.c, 166
gos_kernelSetMaxCpuLoad
 Kernel functions, 35
gos_kernelStart
 Kernel functions, 36
gos_kernelSubscribeDumpReadySignal
 Kernel functions, 37
gos_message.c
 GOS_MESSAGE_DAEMON_POLL_TIME_MS,
 282
 gos_messageDaemonTask, 282
 gos_messageInit, 283
 gos_messageRx, 284
 gos_messageTx, 285
 messageArray, 286

messageDaemonTaskDesc, 286
messageDaemonTaskId, 286
messageMutex, 286
messageWaiterArray, 286
nextMessageIndex, 287
nextWaiterIndex, 287
gos_message.h
 GOS_MESSAGE_ENDLESS_TMO, 192
 GOS_MESSAGE_INVALID_ID, 192
 gos_messageId_t, 192
 gos_messageInit, 192
 gos_messageRx, 193
 gos_messageTimeout_t, 192
 gos_messageTx, 194
GOS_MESSAGE_DAEMON_POLL_TIME_MS
 gos_message.c, 282
GOS_MESSAGE_ENDLESS_TMO
 gos_message.h, 192
GOS_MESSAGE_INVALID_ID
 gos_message.h, 192
gos_message_t, 69
gos_messageDaemonTask
 gos_message.c, 282
gos_messageId_t
 gos_message.h, 192
gos_messageInit
 gos_message.c, 283
 gos_message.h, 192
gos_messageRx
 gos_message.c, 284
 gos_message.h, 193
gos_messageTimeout_t
 gos_message.h, 192
gos_messageTx
 gos_message.c, 285
 gos_message.h, 194
gos_messageWaiterDesc_t, 70
gos_mutex.c
 gos_mutexInit, 289
 gos_mutexLock, 290
 gos_mutexUnlock, 291
 MUTEX_LOCK_SLEEP_MS, 289
gos_mutex.h
 GOS_MUTEX_ENDLESS_TMO, 196
 GOS_MUTEX_LOCKED, 197
 GOS_MUTEX_NO_TMO, 197
 GOS_MUTEX_UNLOCKED, 197
 gos_mutexInit, 197
 gos_mutexLock, 198
 gos_mutexState_t, 197
 gos_mutexUnlock, 200
GOS_MUTEX_ENDLESS_TMO
 gos_mutex.h, 196
GOS_MUTEX_LOCKED
 gos_mutex.h, 197
GOS_MUTEX_NO_TMO
 gos_mutex.h, 197
gos_mutex_t, 70
GOS_MUTEX_UNLOCKED
 gos_mutex.h, 197
gos_mutexInit
 gos_mutex.c, 289
 gos_mutex.h, 197
gos_mutexLock
 gos_mutex.c, 290
 gos_mutex.h, 198
gos_mutexLock
 gos_mutex.c, 290
 gos_mutex.h, 198
gos_mutexState_t
 gos_mutex.h, 197
gos_mutexUnlock
 gos_mutex.c, 291
 gos_mutex.h, 200
GOS_NAKED
 Global definitions, 11
GOS_NOP
 Global definitions, 11
gos_platformDriverInit
 Platform initializer weak functions, 63
gos_port.h
 gos_ported_doContextSwitch, 154
 gos_ported_enableFaultHandlers, 155
 gos_ported_handleSVC, 155
 gos_ported_handleSVCMain, 155
 gos_ported_kernelStartInit, 156
 gos_ported_pendSVHandler, 156
 gos_ported_procReset, 156
 gos_ported_reschedule, 157
 gos_ported_svHandler, 157
 gos_ported_svHandlerMain, 157
 gos_ported_sysTickInterrupt, 158
gos_ported_doContextSwitch
 gos_port.h, 154
gos_ported_enableFaultHandlers
 gos_port.h, 155
gos_ported_handleSVC
 gos_port.h, 155
gos_ported_handleSVCMain
 gos_port.h, 155
gos_ported_kernelStartInit
 gos_port.h, 156
gos_ported_pendSVHandler
 gos_port.h, 156
gos_ported_procReset
 gos_port.h, 156
gos_ported_reschedule
 gos_port.h, 157
gos_ported_svHandler
 gos_port.h, 157
gos_ported_svHandlerMain
 gos_port.h, 157
gos_ported_sysTickInterrupt
 gos_port.h, 158
gos_printStartupLogo
 gos_error.c, 267
 gos_error.h, 183
GOS_PRIV_RESERVED_3
 Privilege bit definitions, 16

GOS_PRIV_RESERVED_4
 Privilege bit definitions, 16

GOS_PRIV_RESERVED_5
 Privilege bit definitions, 16

GOS_PRIV_SIGNALING
 Privilege bit definitions, 16

GOS_PRIV_TASK_MANIPULATE
 Privilege bit definitions, 17

GOS_PRIV_TASK_PRIO_CHANGE
 Privilege bit definitions, 17

GOS_PRIV_TRACE
 Privilege bit definitions, 17

GOS_PRIVILEGED
`gos_kernel.h`, 151

gos_queue.c
DUMP_SEPARATOR, 294
`gos_queueCreate`, 295
`gos_queueDump`, 295
`gos_queueGet`, 296
`gos_queueGetElementNumber`, 297
`gos_queueGetName`, 298
`gos_queueInit`, 298
`gos_queuePeek`, 299
`gos_queuePut`, 300
`gos_queueRegisterEmptyHook`, 301
`gos_queueRegisterFullHook`, 302
`gos_queueReset`, 302
`queueEmptyHook`, 303
`queueFullHook`, 303
`queueMutex`, 303
`queues`, 303
`readCounters`, 304
`writeCounters`, 304

gos_queue.h
GOS_DEFAULT_QUEUE_ID, 204
GOS_INVALID_QUEUE_ID, 204
`gos_queueByte_t`, 204
`gos_queueCreate`, 205
`gos_queueDump`, 206
`gos_queueEmptyHook`, 204
`gos_queueFullHook`, 205
`gos_queueGet`, 206
`gos_queueGetElementNumber`, 208
`gos_queueGetName`, 208
`gos_queueInit`, 209
`gos_queuePeek`, 209
`gos_queuePut`, 210
`gos_queueRegisterEmptyHook`, 211
`gos_queueRegisterFullHook`, 212
`gos_queueReset`, 212

gos_queue_t, 71

gos_queueByte_t
`gos_queue.h`, 204

gos_queueCreate
`gos_queue.c`, 295
`gos_queue.h`, 205

gos_queueDescriptor_t, 72

gos_queueDump

gos_queue.c
`gos_queue.h`, 206
`gos_queueElement_t`, 72
`gos_queueEmptyHook`
`gos_queue.h`, 204
`gos_queueFullHook`
`gos_queue.h`, 205
`gos_queueGet`
`gos_queue.c`, 296
`gos_queue.h`, 206
`gos_queueGetElementNumber`
`gos_queue.c`, 297
`gos_queue.h`, 208
`gos_queueGetName`
`gos_queue.c`, 298
`gos_queue.h`, 208
`gos_queueInit`
`gos_queue.c`, 298
`gos_queue.h`, 209
`gos_queuePeek`
`gos_queue.c`, 299
`gos_queue.h`, 209
`gos_queuePut`
`gos_queue.c`, 300
`gos_queue.h`, 210
`gos_queueRegisterEmptyHook`
`gos_queue.c`, 301
`gos_queue.h`, 211
`gos_queueRegisterFullHook`
`gos_queue.c`, 302
`gos_queue.h`, 212
`gos_queueReset`
`gos_queue.c`, 302
`gos_queue.h`, 212

gos_result_t
`gos_kernel.h`, 151

gos_runtime_t, 73

gos_runTimeAddMicroseconds
`gos_time.c`, 343
`gos_time.h`, 234

gos_runTimeAddMilliseconds
`gos_time.c`, 344
`gos_time.h`, 235

gos_runTimeAddSeconds
`gos_time.c`, 345
`gos_time.h`, 236

gos_runTimeGet
`gos_time.c`, 346
`gos_time.h`, 237

gos_shell.c
`actualCommand`, 313
`commandBuffer`, 313
`commandBufferIndex`, 313
`commandParams`, 313
GOS_SHELL_DAEMON_POLL_TIME_MS, 306
GOS_SHELL_DISPLAY_TEXT, 306
`gos_shellCommandHandler`, 306
`gos_shellDaemonTask`, 307

gos_shellEchoOff, 308
gos_shellEchoOn, 308
gos_shellInit, 309
gos_shellRegisterCommand, 309
gos_shellRegisterCommands, 310
gos_shellResume, 311
gos_shellSuspend, 312
shellCommand, 313
shellCommands, 314
shellDaemonTaskDesc, 314
shellDaemonTaskId, 314
useEcho, 314
gos_shell.h
 gos_shellEchoOff, 217
 gos_shellEchoOn, 217
 gos_shellFunction, 216
 gos_shellInit, 218
 gos_shellRegisterCommand, 218
 gos_shellRegisterCommands, 219
 gos_shellResume, 220
 gos_shellSuspend, 221
GOS_SHELL_DAEMON_POLL_TIME_MS
 gos_shell.c, 306
GOS_SHELL_DISPLAY_TEXT
 gos_shell.c, 306
gos_shell_driver.c
 formattedBuffer, 112
 gos_shellDriverReceiveChar, 111
 gos_shellDriverTransmitString, 111
gos_shell_driver.h
 gos_shellDriverReceiveChar, 93
 gos_shellDriverReceiveChar_t, 92
 gos_shellDriverTransmitString, 94
 gos_shellDriverTransmitString_t, 93
gos_shellCommand_t, 73
gos_shellCommandHandler
 gos_shell.c, 306
gos_shellDaemonTask
 gos_shell.c, 307
gos_shellDriverReceiveChar
 gos_shell_driver.c, 111
 gos_shell_driver.h, 93
gos_shellDriverReceiveChar_t
 gos_shell_driver.h, 92
gos_shellDriverTransmitString
 gos_shell_driver.c, 111
 gos_shell_driver.h, 94
gos_shellDriverTransmitString_t
 gos_shell_driver.h, 93
gos_shellEchoOff
 gos_shell.c, 308
 gos_shell.h, 217
gos_shellEchoOn
 gos_shell.c, 308
 gos_shell.h, 217
gos_shellFunction
 gos_shell.h, 216
gos_shellInit
 gos_shell.c, 309
 gos_shell.h, 218
gos_shellRegisterCommand
 gos_shell.c, 309
 gos_shell.h, 218
gos_shellRegisterCommands
 gos_shell.c, 310
 gos_shell.h, 219
gos_shellResume
 gos_shell.c, 311
 gos_shell.h, 220
gos_shellSuspend
 gos_shell.c, 312
 gos_shell.h, 221
gos_signal.c
 callerTaskDesc, 320
 GOS_SIGNAL_DAEMON_POLL_TIME_MS, 316
 gos_signalCreate, 317
 gos_signalDaemonTask, 318
 gos_signallInit, 318
 gos_signallInvoke, 318
 gos_signalSubscribe, 319
 signalArray, 320
 signalDaemonTaskDescriptor, 320
 signallInvokeTrigger, 321
gos_signal.h
 gos_signalCreate, 223
 gos_signallInit, 225
 gos_signallInvoke, 226
 gos_signalSubscribe, 227
GOS_SIGNAL_DAEMON_POLL_TIME_MS
 gos_signal.c, 316
gos_signalCreate
 gos_signal.c, 317
 gos_signal.h, 223
gos_signalDaemonTask
 gos_signal.c, 318
gos_signalDescriptor_t, 74
gos_signallInit
 gos_signal.c, 318
 gos_signal.h, 225
gos_signallInvoke
 gos_signal.c, 318
 gos_signal.h, 226
gos_signallInvokeDescriptor, 74
gos_signalSubscribe
 gos_signal.c, 319
 gos_signal.h, 227
gos_Start
 gos.c, 259
GOS_STATIC
 Global definitions, 11
GOS_STATIC_INLINE
 Global definitions, 11
GOS_SUCCESS
 gos_kernel.h, 151
GOS_SYS_TASK_SLEEP_TIME
 gos.c, 257

gos_sysmon.c
 cpuMessage, 337
 GOS_SYSMON_MSG_CPU_USAGE_GET, 325
 GOS_SYSMON_MSG_CPU_USAGE_GET_ID,
 326
 GOS_SYSMON_MSG_CPU_USAGE_GET_PV,
 326
 GOS_SYSMON_MSG_CPU_USAGE_GET_RESP,
 325
 GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_ID,
 326
 GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_PV,
 326
 GOS_SYSMON_MSG_INV_PAYLOAD_CRC, 327
 GOS_SYSMON_MSG_INV_PV, 327
 GOS_SYSMON_MSG_NUM_OF_MESSAGES,
 325
 GOS_SYSMON_MSG_PING, 325
 GOS_SYSMON_MSG_PING_ACK_PV, 326
 GOS_SYSMON_MSG_PING_ID, 326
 GOS_SYSMON_MSG_PING_PV, 326
 GOS_SYSMON_MSG_PING_RESP, 325
 GOS_SYSMON_MSG_PING_RESP_ID, 326
 GOS_SYSMON_MSG_RES_ERROR, 327
 GOS_SYSMON_MSG_RESET_REQ, 325
 GOS_SYSMON_MSG_RESET_REQ_ID, 326
 GOS_SYSMON_MSG_RESET_REQ_PV, 327
 GOS_SYSMON_MSG_SYSRUNTIME_GET, 325
 GOS_SYSMON_MSG_SYSRUNTIME_GET_ID,
 326
 GOS_SYSMON_MSG_SYSRUNTIME_GET_PV,
 327
 GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP,
 325
 GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_ID,
 326
 GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_PV,
 327
 GOS_SYSMON_MSG_SYSTIME_SET, 325
 GOS_SYSMON_MSG_SYSTIME_SET_ID, 326
 GOS_SYSMON_MSG_SYSTIME_SET_PV, 327
 GOS_SYSMON_MSG_SYSTIME_SET_RESP,
 325
 GOS_SYSMON_MSG_SYSTIME_SET_RESP_ID,
 326
 GOS_SYSMON_MSG_SYSTIME_SET_RESP_PV,
 327
 GOS_SYSMON_MSG_TASK_GET_DATA, 325
 GOS_SYSMON_MSG_TASK_GET_DATA_ID, 326
 GOS_SYSMON_MSG_TASK_GET_DATA_PV,
 326
 GOS_SYSMON_MSG_TASK_GET_DATA_RESP,
 325
 GOS_SYSMON_MSG_TASK_GET_DATA_RESP_ID,
 326
 GOS_SYSMON_MSG_TASK_GET_DATA_RESP_PV,
 326
 GOS_SYSMON_MSG_TASK_GET_VAR_DATA,
 325
 GOS_SYSMON_MSG_TASK_GET_VAR_DATA_ID,
 326
 GOS_SYSMON_MSG_TASK_GET_VAR_DATA_PV,
 326
 GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP,
 325
 GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_ID,
 326
 GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_PV,
 326
 GOS_SYSMON_MSG_TASK_MODIFY_STATE,
 325
 GOS_SYSMON_MSG_TASK_MODIFY_STATE_ID,
 326
 GOS_SYSMON_MSG_TASK_MODIFY_STATE_PV,
 326
 GOS_SYSMON_MSG_TASK_MODIFY_STATE_RESP,
 325
 GOS_SYSMON_MSG_TASK_MODIFY_STATE_RESP_ID,
 326
 GOS_SYSMON_MSG_TASK_MODIFY_STATE_RESP_PV,
 327
 GOS_SYSMON_MSG_UNKNOWN, 325
 GOS_SYSMON_MSG_UNKNOWN_ID, 326
 GOS_SYSMON_MSG_UNKNOWN_PV, 326
 GOS_SYSMON_TASK_MOD_TYPE_BLOCK, 327
 GOS_SYSMON_TASK_MOD_TYPE_DELETE,
 327
 GOS_SYSMON_TASK_MOD_TYPE_RESUME,
 327
 GOS_SYSMON_TASK_MOD_TYPE_SUSPEND,
 327
 GOS_SYSMON_TASK_MOD_TYPE_UNBLOCK,
 327
 GOS_SYSMON_TASK_MOD_TYPE_WAKEUP,
 327
 gos_sysmonCheckMessage, 328
 gos_sysmonDaemonTask, 329
 gos_sysmonGetLutIndex, 329
 gos_sysmonHandleCpuUsageGet, 330
 gos_sysmonHandlePingRequest, 331
 gos_sysmonHandleResetRequest, 331
 gos_sysmonHandleSysRuntimeGet, 332
 gos_sysmonHandleSystimeSet, 332
 gos_sysmonHandleTaskDataGet, 333
 gos_sysmonHandleTaskModification, 334
 gos_sysmonHandleTaskVariableDataGet, 334
 gos_sysmonInit, 335
 gos_sysmonMessageEnum_t, 325
 gos_sysmonMessageHandler_t, 325
 gos_sysmonMessageId_t, 325
 gos_sysmonMessagePv_t, 326
 gos_sysmonMessageResult_t, 327
 gos_sysmonRegisterUserMessage, 336
 gos_sysmonSendResponse, 336
 gos_sysmonTaskModifyType_t, 327

pingMessage, 338
RECEIVE_BUFFER_SIZE, 324
receiveBuffer, 338
sysmonDaemonTaskDesc, 338
sysmonLut, 338
sysRuntimeGetResultMessage, 339
sysTimeSetMessage, 339
sysTimeSetResultMessage, 339
taskDataGetMsg, 339
taskDataMsg, 339
taskDesc, 340
taskModifyMessage, 340
taskModifyResultMessage, 340
taskVariableDataMsg, 340
userMessages, 340
gos_sysmon.h
 gos_sysmonInit, 230
 gos_sysmonMessageReceivedCallback, 229
 gos_sysmonRegisterUserMessage, 230
gos_sysmon_driver.c
 gos_sysmonDriverReceive, 114
 gos_sysmonDriverTransmit, 114
gos_sysmon_driver.h
 gos_sysmonDriverReceive, 97
 gos_sysmonDriverReceive_t, 96
 gos_sysmonDriverTransmit, 98
 gos_sysmonDriverTransmit_t, 97
GOS_SYSMON_MSG_CPU_USAGE_GET
 gos_sysmon.c, 325
GOS_SYSMON_MSG_CPU_USAGE_GET_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_CPU_USAGE_GET_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_CPU_USAGE_GET_RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_CPU_USAGE_GET_RESP_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_INV_PAYLOAD_CRC
 gos_sysmon.c, 327
GOS_SYSMON_MSG_INV_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG_NUM_OF_MESSAGES
 gos_sysmon.c, 325
GOS_SYSMON_MSG_PING
 gos_sysmon.c, 325
GOS_SYSMON_MSG_PING_ACK_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_PING_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_PING_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_PING_RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_PING_RESP_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_RES_ERROR
 gos_sysmon.c, 327
GOS_SYSMON_MSG_RES_OK
 gos_sysmon.c, 327
GOS_SYSMON_MSG_RESET_REQ
 gos_sysmon.c, 325
GOS_SYSMON_MSG_RESET_REQ_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_RESET_REQ_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG_SYSRUNTIME_GET
 gos_sysmon.c, 325
GOS_SYSMON_MSG_SYSRUNTIME_GET_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_SYSRUNTIME_GET_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_SYSRUNTIME_GET_RESP_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG_SYSTIME_SET
 gos_sysmon.c, 325
GOS_SYSMON_MSG_SYSTIME_SET_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_SYSTIME_SET_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG_SYSTIME_SET_RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_SYSTIME_SET_RESP_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_SYSTIME_SET_RESP_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG_TASK_GET_DATA
 gos_sysmon.c, 325
GOS_SYSMON_MSG_TASK_GET_DATA_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_DATA_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_DATA_RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_TASK_GET_DATA_RESP_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_DATA_RESP_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_VAR_DATA
 gos_sysmon.c, 325
GOS_SYSMON_MSG_TASK_GET_VAR_DATA_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_VAR_DATA_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_GET_VAR_DATA_RESP_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK_MODIFY_STATE

gos_sysmon.c, 325
GOS_SYSMON_MSG_TASK MODIFY STATE_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK MODIFY STATE_PV
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK MODIFY STATE RESP
 gos_sysmon.c, 325
GOS_SYSMON_MSG_TASK MODIFY STATE RESP_ID
 gos_sysmonLut_t, 75
 gos_sysmon.c, 326
GOS_SYSMON_MSG_TASK MODIFY STATE RESP_PV
 gos_sysmon.c, 327
GOS_SYSMON_MSG UNKNOWN
 gos_sysmon.c, 325
GOS_SYSMON_MSG UNKNOWN_ID
 gos_sysmon.c, 326
GOS_SYSMON_MSG UNKNOWN_PV
 gos_sysmon.c, 326
GOS_SYSMON_TASK MOD_TYPE_BLOCK
 gos_sysmon.c, 327
GOS_SYSMON_TASK MOD_TYPE_DELETE
 gos_sysmon.c, 327
GOS_SYSMON_TASK MOD_TYPE_RESUME
 gos_sysmon.c, 327
GOS_SYSMON_TASK MOD_TYPE_SUSPEND
 gos_sysmon.c, 327
GOS_SYSMON_TASK MOD_TYPE_UNBLOCK
 gos_sysmon.c, 327
GOS_SYSMON_TASK MOD_TYPE_WAKEUP
 gos_sysmon.c, 327
gos_sysmonCheckMessage
 gos_sysmon.c, 328
gos_sysmonCpuUsageMessage_t, 75
gos_sysmonDaemonTask
 gos_sysmon.c, 329
gos_sysmonDriverReceive
 gos_sysmon_driver.c, 114
 gos_sysmon_driver.h, 97
gos_sysmonDriverReceive_t
 gos_sysmon_driver.h, 96
gos_sysmonDriverTransmit
 gos_sysmon_driver.c, 114
 gos_sysmon_driver.h, 98
gos_sysmonDriverTransmit_t
 gos_sysmon_driver.h, 97
gos_sysmonGetLutIndex
 gos_sysmon.c, 329
gos_sysmonHandleCpuUsageGet
 gos_sysmon.c, 330
gos_sysmonHandlePingRequest
 gos_sysmon.c, 331
gos_sysmonHandleResetRequest
 gos_sysmon.c, 331
gos_sysmonHandleSysRuntimeGet
 gos_sysmon.c, 332
gos_sysmonHandleSystimeSet
 gos_sysmon.c, 332
gos_sysmonHandleTaskDataGet
 gos_sysmon.c, 333
gos_sysmonHandleTaskModification
 gos_sysmon.c, 334
gos_sysmonHandleTaskVariableDataGet
 gos_sysmon.c, 334
gos_sysmonInit
 gos_sysmon.c, 335
 gos_sysmon.h, 230
gos_sysmonMessageEnum_t
 gos_sysmon.c, 325
gos_sysmonMessageHandler_t
 gos_sysmon.c, 325
gos_sysmonMessageId_t
 gos_sysmon.c, 325
gos_sysmonMessagePv_t
 gos_sysmon.c, 326
gos_sysmonMessageReceivedCallback
 gos_sysmon.h, 229
gos_sysmonMessageResult_t
 gos_sysmon.c, 327
gos_sysmonPingMessage_t, 76
gos_sysmonRegisterUserMessage
 gos_sysmon.c, 336
 gos_sysmon.h, 230
gos_sysmonSendResponse
 gos_sysmon.c, 336
gos_sysmonSysruntimeGetResultMessage_t, 76
gos_sysmonSystimeSetMessage_t, 77
gos_sysmonSystimeSetResultMessage_t, 78
gos_sysmonTaskData_t, 78
gos_sysmonTaskDataGetMessage_t, 79
gos_sysmonTaskDataMessage_t, 80
gos_sysmonTaskModifyMessage_t, 80
gos_sysmonTaskModifyResultMessage_t, 81
gos_sysmonTaskModifyType_t
 gos_sysmon.c, 327
gos_sysmonTaskVariableData, 81
gos_sysmonTaskVariableDataMessage_t, 82
gos_sysmonUserMessageDescriptor_t, 83
gos_systemTask
 gos.c, 259
gos_task.c
 gos_idleTask, 173
 gos_taskCheckDescriptor, 174
 kernelIdleHookFunction, 175
 kernelTaskDeleteSignal, 175
 taskDescriptors, 176
GOS_TASK_BLOCKED
 gos_kernel.h, 152
GOS_TASK_IDLE_PRIO
 Global definitions, 12
GOS_TASK_MAX_BLOCK_TIME_MS
 Global definitions, 12
GOS_TASK_MAX_PRIO_LEVELS
 Global definitions, 12
GOS_TASK_PRIVILEGE_KERNEL
 gos_kernel.h, 152
GOS_TASK_PRIVILEGE_SUPERVISOR

gos_kernel.h, 152
GOS_TASK_PRIVILEGE_USER
gos_kernel.h, 152
GOS_TASK_PRIVILEGED_USER
gos_kernel.h, 152
GOS_TASK_READY
gos_kernel.h, 152
GOS_TASK_SLEEPING
gos_kernel.h, 152
GOS_TASK_SUSPENDED
gos_kernel.h, 152
GOS_TASK_ZOMBIE
gos_kernel.h, 152
gos_taskAddPrivilege
Task-related functions, 40
gos_taskBlock
Task-related functions, 40
gos_taskCheckDescriptor
gos_task.c, 174
gos_taskDelete
Task-related functions, 41
gos_taskDescriptor_t, 84
gos_taskGetCurrentId
Task-related functions, 42
gos_taskGetData
Task-related functions, 44
gos_taskGetDataByIndex
Task-related functions, 45
gos_taskGetId
Task-related functions, 46
gos_taskGetName
Task-related functions, 47
gos_taskGetNumber
Task-related functions, 47
gos_taskGetOriginalPriority
Task-related functions, 48
gos_taskGetPriority
Task-related functions, 48
gos_taskGetPrivileges
Task-related functions, 49
gos_taskPrivilegeLevel_t
gos_kernel.h, 151
gos_taskRegister
Task-related functions, 50
gos_taskRegisterTasks
Task-related functions, 51
gos_taskRemovePrivilege
Task-related functions, 52
gos_taskResume
Task-related functions, 53
gos_taskSetOriginalPriority
Task-related functions, 54
gos_taskSetPriority
Task-related functions, 55
gos_taskSetPrivileges
Task-related functions, 56
gos_taskSleep
Task-related functions, 57
gos_taskState_t
gos_kernel.h, 152
gos_taskSubscribeDeleteSignal
Kernel functions, 37
gos_taskSuspend
Task-related functions, 59
gos_taskUnblock
Task-related functions, 60
gos_taskWakeUp
Task-related functions, 61
gos_taskYield
Task-related functions, 61
gos_time.c
dayLookupTable, 352
gos_runTimeAddMicroseconds, 343
gos_runTimeAddMilliseconds, 344
gos_runTimeAddSeconds, 345
gos_runTimeGet, 346
gos_timeAddMilliseconds, 346
gos_timeAddSeconds, 347
gos_timeCompare, 348
gos_timeDaemonTask, 348
gos_timeGet, 349
gos_timeIncreaseSystemTime, 350
gos_timeInit, 351
gos_timeSet, 351
systemRunTime, 353
systemTime, 353
TIME_DEFAULT_DAY, 343
TIME_DEFAULT_MONTH, 343
TIME_DEFAULT_YEAR, 343
TIME_SLEEP_TIME_MS, 343
timeDaemonTaskDesc, 353
timeDaemonTaskId, 353
timeSignalId, 354
gos_time.h
gos_runTimeAddMicroseconds, 234
gos_runTimeAddMilliseconds, 235
gos_runTimeAddSeconds, 236
gos_runTimeGet, 237
GOS_TIME_APRIIL, 234
GOS_TIME_AUGUST, 234
GOS_TIME_DAY_ELAPSED_SENDER_ID, 234
GOS_TIME_DECEMBER, 234
GOS_TIME_EARLIER, 233
GOS_TIME_EQUAL, 233
GOS_TIME_FEBRUARY, 234
GOS_TIME_HOUR_ELAPSED_SENDER_ID, 234
GOS_TIME_JANUARY, 234
GOS_TIME_JULY, 234
GOS_TIME_JUNE, 234
GOS_TIME_LATER, 233
GOS_TIME_MARCH, 234
GOS_TIME_MAY, 234
GOS_TIME_MINUTE_ELAPSED_SENDER_ID,
234
GOS_TIME_MONTH_ELAPSED_SENDER_ID,
234

GOS_TIME_NOVEMBER, 234
 GOS_TIME_NUMBER_OF_MONTHS, 234
 GOS_TIME_OCTOBER, 234
 GOS_TIME_SECOND_ELAPSED_SENDER_ID, 234
 GOS_TIME_SEPTEMBER, 234
 GOS_TIME_YEAR_ELAPSED_SENDER_ID, 234
 gos_timeAddMilliseconds, 237
 gos_timeAddSeconds, 238
 gos_timeCompare, 239
 gos_timeComprareResult_t, 233
 gos_timeElapsedSenderId_t, 233
 gos_timeGet, 239
 gos_timeIncreaseSystemTime, 240
 gos_timeInit, 241
 gos_timeMonthEnum_t, 234
 gos_timeSet, 242
 GOS_TIME_APRL
 gos_time.h, 234
 GOS_TIME_AUGUST
 gos_time.h, 234
 GOS_TIME_DAY_ELAPSED_SENDER_ID
 gos_time.h, 234
 GOS_TIME_DECEMBER
 gos_time.h, 234
 GOS_TIME_EARLIER
 gos_time.h, 233
 GOS_TIME_EQUAL
 gos_time.h, 233
 GOS_TIME_FEBRUARY
 gos_time.h, 234
 GOS_TIME_HOUR_ELAPSED_SENDER_ID
 gos_time.h, 234
 GOS_TIME_JANUARY
 gos_time.h, 234
 GOS_TIME_JULY
 gos_time.h, 234
 GOS_TIME_JUNE
 gos_time.h, 234
 GOS_TIME_LATER
 gos_time.h, 233
 GOS_TIME_MARCH
 gos_time.h, 234
 GOS_TIME_MAY
 gos_time.h, 234
 GOS_TIME_MINUTE_ELAPSED_SENDER_ID
 gos_time.h, 234
 GOS_TIME_MONTH_ELAPSED_SENDER_ID
 gos_time.h, 234
 GOS_TIME_NOVEMBER
 gos_time.h, 234
 GOS_TIME_NUMBER_OF_MONTHS
 gos_time.h, 234
 GOS_TIME_OCTOBER
 gos_time.h, 234
 GOS_TIME_SECOND_ELAPSED_SENDER_ID
 gos_time.h, 234
 GOS_TIME_SEPTEMBER
 gos_time.h, 234
 gos_time.h, 234
 gos_time_t, 85
 GOS_TIME_YEAR_ELAPSED_SENDER_ID
 gos_time.h, 234
 gos_timeAddMilliseconds
 gos_time.c, 346
 gos_time.h, 237
 gos_timeAddSeconds
 gos_time.c, 347
 gos_time.h, 238
 gos_timeCompare
 gos_time.c, 348
 gos_time.h, 239
 gos_timeComprareResult_t
 gos_time.h, 233
 gos_timeDaemonTask
 gos_time.c, 348
 gos_timeElapsedSenderId_t
 gos_time.h, 233
 gos_timeGet
 gos_time.c, 349
 gos_time.h, 239
 gos_timeIncreaseSystemTime
 gos_time.c, 350
 gos_time.h, 240
 gos_timeInit
 gos_time.c, 351
 gos_time.h, 241
 gos_timeMonthEnum_t
 gos_time.h, 234
 gos_timer_driver.c
 gos_timerDriverSysTimerGet, 117
 gos_timer_driver.h
 gos_timerDriverSysTimerGet, 100
 gos_timerDriverSysTimerGetVal_t, 100
 gos_timerDriverSysTimerGet
 gos_timer_driver.c, 117
 gos_timer_driver.h, 100
 gos_timerDriverSysTimerGetVal_t
 gos_timer_driver.h, 100
 gos_timeSet
 gos_time.c, 351
 gos_time.h, 242
 gos_trace.c
 formattedBuffer, 361
 GOS_TRACE_MUTEX_TMO_MS, 355
 GOS_TRACE_QUEUE_TMO_MS, 356
 GOS_TRACE_TIMESTAMP_FORMAT, 356
 GOS_TRACE_TIMESTAMP_LENGTH, 356
 gos_traceDaemonTask, 356
 gos_tracelInit, 357
 gos_traceTrace, 358
 gos_traceTraceFormatted, 359
 gos_traceTraceFormattedUnsafe, 360
 timeStampBuffer, 362
 traceDaemonTaskDesc, 362
 traceLine, 362
 traceMutex, 362

traceQueue, 362
gos_trace.h
 gos_traceInit, 245
 gos_traceTrace, 246
 gos_traceTraceFormatted, 247
 gos_traceTraceFormattedUnsafe, 248
gos_trace_driver.c
 gos_traceDriverTransmitString, 119
 gos_traceDriverTransmitString_Unsafe, 120
gos_trace_driver.h
 gos_traceDriverTransmitString, 103
 gos_traceDriverTransmitString_t, 102
 gos_traceDriverTransmitString_Unsafe, 104
 gos_traceDriverTransmitString_Unsafe_t, 103
GOS_TRACE_MUTEX_TMO_MS
 gos_trace.c, 355
GOS_TRACE_QUEUE_TMO_MS
 gos_trace.c, 356
GOS_TRACE_TIMESTAMP_FORMAT
 gos_trace.c, 356
GOS_TRACE_TIMESTAMP_LENGTH
 gos_trace.c, 356
gos_traceDaemonTask
 gos_trace.c, 356
gos_traceDriverTransmitString
 gos_trace_driver.c, 119
 gos_trace_driver.h, 103
gos_traceDriverTransmitString_t
 gos_trace_driver.h, 102
gos_traceDriverTransmitString_Unsafe
 gos_trace_driver.c, 120
 gos_trace_driver.h, 104
gos_traceDriverTransmitString_Unsafe_t
 gos_trace_driver.h, 103
gos_tracelInit
 gos_trace.c, 357
 gos_trace.h, 245
gos_traceResultToString
 gos_error.c, 268
gos_traceTrace
 gos_trace.c, 358
 gos_trace.h, 246
gos_traceTraceFormatted
 gos_trace.c, 359
 gos_trace.h, 247
gos_traceTraceFormattedUnsafe
 gos_trace.c, 360
 gos_trace.h, 248
gos_trigger.c
 gos_triggerDecrement, 364
 gos_triggerIncrement, 365
 gos_triggerInit, 366
 gos_triggerReset, 367
 gos_triggerWait, 367
gos_trigger.h
 GOS_TRIGGER_ENDLESS_TMO, 251
 GOS_TRIGGER_NO_TMO, 251
 gos_triggerDecrement, 252
 gos_triggerIncrement, 252
 gos_triggerInit, 253
 gos_triggerReset, 254
 gos_triggerWait, 255
GOS_TRIGGER_ENDLESS_TMO
 gos_trigger.h, 251
GOS_TRIGGER_NO_TMO
 gos_trigger.h, 251
gos_trigger_t, 86
gos_triggerDecrement
 gos_trigger.c, 364
 gos_trigger.h, 252
gos_triggerIncrement
 gos_trigger.c, 365
 gos_trigger.h, 252
gos_triggerInit
 gos_trigger.c, 366
 gos_trigger.h, 253
gos_triggerReset
 gos_trigger.c, 367
 gos_trigger.h, 254
gos_triggerWait
 gos_trigger.c, 367
 gos_trigger.h, 255
GOS_TRUE
 gos_kernel.h, 151
GOS_UNPRIVILEGED
 gos_kernel.h, 151
GOS_UNUSED
 Global definitions, 12
gos_userApplicationInit
 Platform initializer weak functions, 63
GOS_VERSION_MAJOR
 gos.h, 177
GOS_VERSION_MINOR
 gos.h, 178
Hook function type definitions, 20
ICSR
 gos_kernel.c, 161
inlSr
 gos_kernel.c, 167
initError
 gos.c, 260
initializers
 gos.c, 261
isKernelRunning
 gos_kernel.c, 167
Kernel functions, 22
 gos_kernelCalculateTaskCpuUsages, 23
 gos_kernelDelayMs, 24
 gos_kernelDelayUs, 25
 gos_kernelDump, 25
 gos_kernelGetCpuUsage, 26
 gos_kernelGetMaxCpuLoad, 27
 gos_kernelGetSysTicks, 27
 gos_kernellInit, 28

gos_kernelsCallerIsr, 29
 gos_kernelPrivilegedModeSetRequired, 30
 gos_kernelRegisterIdleHook, 30
 gos_kernelRegisterPreResetHook, 31
 gos_kernelRegisterPrivilegedHook, 31
 gos_kernelRegisterSwapHook, 32
 gos_kernelRegisterSysTickHook, 32
 gos_kernelReschedule, 33
 gos_kernelReset, 34
 gos_kernelSetMaxCpuLoad, 35
 gos_kernelStart, 36
 gos_kernelSubscribeDumpReadySignal, 37
 gos_taskSubscribeDeleteSignal, 37
 kernelDumpReadySignal
 gos_kernel.c, 167
 kernelDumpSignal
 gos_kernel.c, 167
 kernelIdleHookFunction
 gos_task.c, 175
 kernelPreResetHookFunction
 gos_kernel.c, 168
 kernelPrivilegedHookFunction
 gos_kernel.c, 168
 kernelSwapHookFunction
 gos_kernel.c, 168
 kernelSysTickHookFunction
 gos_kernel.c, 168
 kernelTaskDeleteSignal
 gos_task.c, 175
 MAIN_STACK
 Global definitions, 12
 MAX_CPU_DUMP_SEPARATOR
 gos_kernel.c, 161
 messageArray
 gos_message.c, 286
 messageDaemonTaskDesc
 gos_message.c, 286
 messageDaemonTaskId
 gos_message.c, 286
 messageMutex
 gos_message.c, 286
 messageWaiterArray
 gos_message.c, 286
 monitoringTime
 gos_kernel.c, 168
 MUTEX_LOCK_SLEEP_MS
 gos_mutex.c, 289
 nextMessageIndex
 gos_message.c, 287
 nextWaiterIndex
 gos_message.c, 287
 NULL
 Global definitions, 13
 pingMessage
 gos_sysmon.c, 338
 Platform initializer weak functions, 63
 gos_platformDriverInit, 63
 gos_userApplicationInit, 63
 previousTick
 gos_kernel.c, 169
 primask
 gos_kernel.c, 169
 Privilege bit definitions, 16
 GOS_PRIV_RESERVED_3, 16
 GOS_PRIV_RESERVED_4, 16
 GOS_PRIV_RESERVED_5, 16
 GOS_PRIV_SIGNALING, 16
 GOS_PRIV_TASK_MANIPULATE, 17
 GOS_PRIV_TASK_PRIO_CHANGE, 17
 GOS_PRIV_TRACE, 17
 privilegedModeSetRequired
 gos_kernel.c, 169
 queueEmptyHook
 gos_queue.c, 303
 queueFullHook
 gos_queue.c, 303
 queueMutex
 gos_queue.c, 303
 queues
 gos_queue.c, 303
 RAM_SIZE
 Global definitions, 13
 RAM_START
 Global definitions, 13
 readCounters
 gos_queue.c, 304
 RECEIVE_BUFFER_SIZE
 gos_sysmon.c, 324
 receiveBuffer
 gos_sysmon.c, 338
 resetRequired
 gos_kernel.c, 169
 RESULT_STRING_ERROR
 gos_error.c, 263
 RESULT_STRING_SUCCESS
 gos_error.c, 264
 RESULT_STRING_UNKNOWN
 gos_error.c, 264
 schedDisableCntr
 gos_kernel.c, 169
 SEPARATOR_LINE
 gos_error.c, 264
 SHCSR
 gos_kernel.c, 162
 shellCommand
 gos_shell.c, 313
 shellCommands
 gos_shell.c, 314
 shellDaemonTaskDesc
 gos_shell.c, 314
 shellDaemonTaskId
 gos_shell.c, 314

signalArray
 gos_signal.c, 320
signalDaemonTaskDescriptor
 gos_signal.c, 320
signalInvokeTrigger
 gos_signal.c, 321
STACK_STATS_SEPARATOR
 gos_kernel.c, 162
sysmonDaemonTaskDesc
 gos_sysmon.c, 338
sysmonLut
 gos_sysmon.c, 338
sysRuntimeGetResultMessage
 gos_sysmon.c, 339
System monitoring message structures, 66
systemRunTime
 gos_time.c, 353
systemTaskDesc
 gos.c, 261
systemTaskId
 gos.c, 261
systemTime
 gos_time.c, 353
sysTicks
 gos_kernel.c, 170
sysTimerValue
 gos_kernel.c, 170
sysTimeSetMessage
 gos_sysmon.c, 339
sysTimeSetResultMessage
 gos_sysmon.c, 339

Task-related functions, 39
 gos_taskAddPrivilege, 40
 gos_taskBlock, 40
 gos_taskDelete, 41
 gos_taskGetCurrentId, 42
 gos_taskGetData, 44
 gos_taskGetDataByIndex, 45
 gos_taskGetId, 46
 gos_taskGetName, 47
 gos_taskGetNumber, 47
 gos_taskGetOriginalPriority, 48
 gos_taskGetPriority, 48
 gos_taskGetPrivileges, 49
 gos_taskRegister, 50
 gos_taskRegisterTasks, 51
 gos_taskRemovePrivilege, 52
 gos_taskResume, 53
 gos_taskSetOriginalPriority, 54
 gos_taskSetPriority, 55
 gos_taskSetPrivileges, 56
 gos_taskSleep, 57
 gos_taskSuspend, 59
 gos_taskUnblock, 60
 gos_taskWakeUp, 61
 gos_taskYield, 61

Task-related type definitions, 19
TASK_DUMP_SEPARATOR

 gos_kernel.c, 162
taskDataGetMsg
 gos_sysmon.c, 339
taskDataMsg
 gos_sysmon.c, 339
taskDesc
 gos_sysmon.c, 340
taskDescriptors
 gos_task.c, 176
taskModifyMessage
 gos_sysmon.c, 340
taskModifyResultMessage
 gos_sysmon.c, 340
taskVariableDataMsg
 gos_sysmon.c, 340
Time-related definitions, 21
TIME_DEFAULT_DAY
 gos_time.c, 343
TIME_DEFAULT_MONTH
 gos_time.c, 343
TIME_DEFAULT_YEAR
 gos_time.c, 343
TIME_SLEEP_TIME_MS
 gos_time.c, 343
timeDaemonTaskDesc
 gos_time.c, 353
timeDaemonTaskId
 gos_time.c, 353
timeSignalId
 gos_time.c, 354
timeStampBuffer
 gos_trace.c, 362
TO_BINARY
 gos_kernel.c, 162
Trace formatter macros, 65
traceDaemonTaskDesc
 gos_trace.c, 362
traceLine
 gos_trace.c, 362
traceMutex
 gos_trace.c, 362
traceQueue
 gos_trace.c, 362

useEcho
 gos_shell.c, 314
userMessages
 gos_sysmon.c, 340

writeCounters
 gos_queue.c, 304