

LPC845-C++

2.0

Generated by Doxygen 1.13.1

1 KITLPC845	1
1.1 Informacion:	1
1.1.1 Drivers/Periféricos ya creados:	1
1.1.2 Para crear un proyecto nuevo:	2
1.1.3 Para utilizar este proyecto:	2
2 Topic Index	3
2.1 Topics	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	11
5.1 File List	11
6 Topic Documentation	15
6.1 Misc	15
6.1.1 Detailed Description	15
6.2 Drivers	15
6.2.1 Detailed Description	17
6.2.2 Typedef Documentation	17
6.2.2.1 Timer_Handler	17
6.2.3 Variable Documentation	17
6.2.3.1 g_gpiohandler	17
6.2.3.2 g_Handler	17
6.2.3.3 g_MRThandler	18
6.2.3.4 IOCON_INDEX_PIO0	18
6.2.3.5 IOCON_INDEX_PIO1	18
6.3 Abstracta	18
6.3.1 Detailed Description	19
6.4 REGISTERS Peripheral Access Layer	19
6.4.1 Detailed Description	20
6.4.2 Macro Definition Documentation	20
6.4.2.1 FCLKIN	20
6.4.2.2 FREQ_PRINCIPAL	20
6.4.3 SYSCON Peripheral Access Layer	20
6.4.3.1 Detailed Description	21
6.4.3.2 Macro Definition Documentation	21
6.4.4 GPIO Peripheral Access Layer	22
6.4.4.1 Detailed Description	22
6.4.4.2 Macro Definition Documentation	22

6.4.5 IOCON Peripheral Access Layer	22
6.4.5.1 Detailed Description	23
6.4.5.2 Macro Definition Documentation	23
6.4.6 SYSTICK Peripheral Access Layer	23
6.4.6.1 Detailed Description	24
6.4.6.2 Macro Definition Documentation	24
6.4.7 MRT Peripheral Access Layer	24
6.4.7.1 Detailed Description	25
6.4.7.2 Macro Definition Documentation	25
6.4.8 NVIC Peripheral Access Layer	26
6.4.8.1 Detailed Description	26
6.4.8.2 Macro Definition Documentation	26
6.4.9 PIN_INT Peripheral Access Layer	27
6.4.9.1 Detailed Description	27
6.4.9.2 Macro Definition Documentation	27
6.4.10 SCT Peripheral Access Layer	28
6.4.10.1 Detailed Description	28
6.4.10.2 Macro Definition Documentation	28
6.4.11 USART Peripheral Access Layer	28
6.4.11.1 Detailed Description	29
6.4.11.2 Macro Definition Documentation	29
6.4.12 SWM Peripheral Access Layer	31
6.4.12.1 Detailed Description	31
6.4.12.2 Macro Definition Documentation	31
6.4.13 ADC Peripheral Access Layer	31
6.4.13.1 Detailed Description	32
6.4.13.2 Macro Definition Documentation	32
6.4.13.3 ADC Register Masks	33
6.4.14 DAC Peripheral Access Layer	56
6.4.14.1 Detailed Description	57
6.4.14.2 Macro Definition Documentation	57
6.4.15 I2C Peripheral Access Layer	57
6.4.15.1 Detailed Description	58
6.4.15.2 Macro Definition Documentation	58
6.4.15.3 I2C Register Masks	59
6.4.16 SPI Peripheral Access Layer	72
6.4.16.1 Detailed Description	72
6.4.16.2 Macro Definition Documentation	72
6.4.16.3 SPI Register Masks	73
7 Class Documentation	81
7.1 ADC Class Reference	81

7.1.1 Detailed Description	82
7.1.2 Constructor & Destructor Documentation	82
7.1.2.1 ADC()	82
7.1.3 Member Function Documentation	83
7.1.3.1 Get()	83
7.1.3.2 Initialize()	83
7.1.3.3 IsResultReady()	83
7.1.3.4 Trigger()	83
7.2 ADC_Group Class Reference	84
7.2.1 Detailed Description	84
7.2.2 Member Typedef Documentation	85
7.2.2.1 adc_isr	85
7.2.2.2 error_t	85
7.2.2.3 irq_source_inten	85
7.2.3 Member Enumeration Documentation	85
7.2.3.1 adc_isr	85
7.2.3.2 error_t	85
7.2.3.3 irq_source_inten	85
7.2.4 Constructor & Destructor Documentation	85
7.2.4.1 ADC_Group()	85
7.2.4.2 ~ADC_Group()	86
7.2.5 Member Function Documentation	86
7.2.5.1 DisableIrq()	86
7.2.5.2 EnableIrq()	86
7.2.5.3 GetValue()	86
7.2.5.4 Handler()	87
7.2.5.5 InitADCChanel()	87
7.2.5.6 IsResultReady()	87
7.2.5.7 RemoveADCChanel()	87
7.2.5.8 SetLowPowerMode()	88
7.3 ADC_Type Struct Reference	88
7.3.1 Detailed Description	89
7.3.2 Member Data Documentation	89
7.3.2.1 CHAN_THRSEL	89
7.3.2.2 CTRL	89
7.3.2.3 DAT	89
7.3.2.4 FLAGS	89
7.3.2.5 INTEN	89
7.3.2.6 RESERVED_0	89
7.3.2.7 RESERVED_1	90
7.3.2.8 SEQ_CTRL	90
7.3.2.9 SEQ_GDAT	90

7.3.2.10 THR0_HIGH	90
7.3.2.11 THR0_LOW	90
7.3.2.12 THR1_HIGH	90
7.3.2.13 THR1_LOW	90
7.3.2.14 TRM	91
7.4 barrido Class Reference	91
7.4.1 Detailed Description	91
7.4.2 Member Function Documentation	91
7.4.2.1 Initialize()	91
7.4.2.2 SetDigito()	92
7.5 Callback Class Reference	92
7.5.1 Detailed Description	92
7.5.2 Member Function Documentation	93
7.5.2.1 SetInterrupt()	93
7.5.2.2 SWhandler()	93
7.5.2.3 UnSetInterrupt()	93
7.6 CircularBuffer< T > Class Template Reference	93
7.6.1 Detailed Description	94
7.6.2 Constructor & Destructor Documentation	94
7.6.2.1 CircularBuffer()	94
7.6.3 Member Function Documentation	94
7.6.3.1 isFull()	94
7.6.3.2 pop()	94
7.6.3.3 push()	95
7.7 ComunicacionAsincronica Class Reference	95
7.7.1 Detailed Description	95
7.7.2 Constructor & Destructor Documentation	95
7.7.2.1 ComunicacionAsincronica()	95
7.7.2.2 ~ComunicacionAsincronica()	96
7.7.3 Member Function Documentation	96
7.7.3.1 Read()	96
7.7.3.2 UART_IRQHandler()	96
7.7.3.3 Write() [1/2]	96
7.7.3.4 Write() [2/2]	96
7.8 ComunicacionSincronica Class Reference	97
7.8.1 Detailed Description	98
7.8.2 Constructor & Destructor Documentation	98
7.8.2.1 ComunicacionSincronica()	98
7.8.2.2 ~ComunicacionSincronica()	98
7.8.3 Member Function Documentation	98
7.8.3.1 Read()	98
7.8.3.2 Write()	98

7.8.4 Member Data Documentation	99
7.8.4.1 m_scl	99
7.9 DAC Class Reference	99
7.9.1 Detailed Description	101
7.9.2 Member Typedef Documentation	101
7.9.2.1 dac_channel	101
7.9.2.2 dac_error	101
7.9.3 Member Enumeration Documentation	101
7.9.3.1 dac_channel	101
7.9.3.2 dac_error	101
7.9.4 Constructor & Destructor Documentation	101
7.9.4.1 DAC()	101
7.9.4.2 ~DAC()	102
7.9.5 Member Function Documentation	102
7.9.5.1 Get()	102
7.9.5.2 GetMaxRange()	102
7.9.5.3 Initialize()	102
7.9.5.4 operator"!=()"	102
7.9.5.5 operator<()	103
7.9.5.6 operator<=()	103
7.9.5.7 operator=()	103
7.9.5.8 operator==()	104
7.9.5.9 operator>()	104
7.9.5.10 operator>=()	104
7.9.5.11 Set()	105
7.9.5.12 SetMaxRange()	105
7.10 DAC_t Struct Reference	105
7.10.1 Detailed Description	106
7.10.2 Member Data Documentation	106
7.10.2.1 BIAS	106
7.10.2.2 CNT_ENA	106
7.10.2.3 CNTVAL	106
7.10.2.4 CR	106
7.10.2.5 CTRL	106
7.10.2.6 DBLBUF_ENA	107
7.10.2.7 DMA_ENA	107
7.10.2.8 INT_DMA_REQ	107
7.10.2.9 RESERVED0	107
7.10.2.10 RESERVED1	107
7.10.2.11 RESERVED2	107
7.10.2.12 VALUE	107
7.11 digit0 Class Reference	108

7.11.1 Detailed Description	108
7.11.2 Member Enumeration Documentation	108
7.11.2.1 <code>codigo_t</code>	108
7.11.2.2 <code>modo_t</code>	108
7.11.2.3 SIMBOLOS	108
7.11.3 Constructor & Destructor Documentation	109
7.11.3.1 <code>digito()</code>	109
7.11.4 Member Function Documentation	109
7.11.4.1 <code>Clear()</code>	109
7.11.4.2 <code>Get()</code>	109
7.11.4.3 <code>Set()</code>	109
7.12 Display Class Reference	110
7.12.1 Detailed Description	110
7.12.2 Constructor & Destructor Documentation	110
7.12.2.1 <code>Display()</code>	110
7.12.2.2 <code>~Display()</code>	110
7.12.3 Member Function Documentation	111
7.12.3.1 <code>Clear()</code>	111
7.12.3.2 <code>Write()</code>	111
7.13 display7Segmentos Class Reference	111
7.13.1 Detailed Description	112
7.13.2 Constructor & Destructor Documentation	113
7.13.2.1 <code>display7Segmentos()</code>	113
7.13.2.2 <code>~display7Segmentos()</code>	113
7.13.3 Member Function Documentation	113
7.13.3.1 <code>Clear()</code>	113
7.13.3.2 <code>Set()</code>	113
7.13.3.3 <code>SWhandler()</code>	114
7.13.3.4 <code>Write()</code>	114
7.14 distancia Class Reference	114
7.14.1 Detailed Description	115
7.14.2 Constructor & Destructor Documentation	115
7.14.2.1 <code>distancia()</code>	115
7.14.2.2 <code>~distancia()</code>	115
7.14.3 Member Function Documentation	115
7.14.3.1 <code>GetDistancia()</code>	115
7.14.3.2 <code>operator==()</code>	115
7.15 ESP8266 Class Reference	116
7.15.1 Detailed Description	117
7.15.2 Member Enumeration Documentation	117
7.15.2.1 <code>conection_type</code>	117
7.15.2.2 <code>status_type</code>	117

7.15.3 Member Function Documentation	117
7.15.3.1 ConnectToServer()	117
7.15.3.2 ConnectToWifi()	117
7.15.3.3 DisconnectToServer()	118
7.15.3.4 DisconnectToWifi()	118
7.15.3.5 GetIP()	118
7.15.3.6 GetStatus()	118
7.15.3.7 Initialize()	119
7.15.3.8 IsConnectedToServer()	119
7.15.3.9 IsConnectedToWifi()	119
7.15.3.10 Read()	119
7.15.3.11 SetIP()	119
7.15.3.12 Write() [1/2]	120
7.15.3.13 Write() [2/2]	120
7.16 gpio Class Reference	121
7.16.1 Detailed Description	123
7.16.2 Constructor & Destructor Documentation	123
7.16.2.1 gpio()	123
7.16.2.2 ~gpio()	124
7.16.3 Member Function Documentation	124
7.16.3.1 ClrPin()	124
7.16.3.2 GetPin()	124
7.16.3.3 operator=()	124
7.16.3.4 SetDir()	125
7.16.3.5 SetPin()	125
7.16.3.6 SetPinMode()	125
7.16.3.7 SetPinResistor()	126
7.16.3.8 SetToggleDir()	126
7.16.3.9 SetTogglePin()	126
7.16.4 Member Data Documentation	126
7.16.4.1 m_activity	126
7.16.4.2 m_direction	127
7.16.4.3 m_mode	127
7.17 GPIO_Type Struct Reference	127
7.17.1 Detailed Description	128
7.17.2 Member Data Documentation	128
7.17.2.1 B	128
7.17.2.2 CLR	128
7.17.2.3 DIR	128
7.17.2.4 DIRCLR	128
7.17.2.5 DIRNOT	128
7.17.2.6 DIRSET	128

7.17.2.7 MASK	129
7.17.2.8 MPIN	129
7.17.2.9 NOT	129
7.17.2.10 PIN	129
7.17.2.11 RESERVED_0	129
7.17.2.12 RESERVED_1	129
7.17.2.13 RESERVED_10	129
7.17.2.14 RESERVED_2	129
7.17.2.15 RESERVED_3	130
7.17.2.16 RESERVED_4	130
7.17.2.17 RESERVED_5	130
7.17.2.18 RESERVED_6	130
7.17.2.19 RESERVED_7	130
7.17.2.20 RESERVED_8	130
7.17.2.21 RESERVED_9	130
7.17.2.22 SET	130
7.17.2.23 W	131
7.18 gruposdedigitos Struct Reference	131
7.18.1 Detailed Description	131
7.18.2 Constructor & Destructor Documentation	131
7.18.2.1 gruposdedigitos()	131
7.18.3 Member Data Documentation	131
7.18.3.1 m_cantidad	131
7.18.3.2 m_comienzo	132
7.19 HC_SR04 Class Reference	132
7.19.1 Detailed Description	133
7.19.2 Constructor & Destructor Documentation	133
7.19.2.1 HC_SR04()	133
7.19.3 Member Function Documentation	134
7.19.3.1 GetDistancia()	134
7.19.3.2 Initialize()	134
7.19.3.3 Off()	134
7.19.3.4 On()	134
7.19.3.5 operator<()	134
7.19.3.6 operator<=()	135
7.19.3.7 operator==()	135
7.19.3.8 operator>()	135
7.19.3.9 operator>=()	136
7.20 I2C Class Reference	136
7.20.1 Detailed Description	138
7.20.2 Member Enumeration Documentation	138
7.20.2.1 I2C_states_t	138

7.20.3 Constructor & Destructor Documentation	138
7.20.3.1 I2C()	138
7.20.4 Member Function Documentation	139
7.20.4.1 ACK()	139
7.20.4.2 ACKAddr()	139
7.20.4.3 Continue()	139
7.20.4.4 GetState()	139
7.20.4.5 I2C_IRQHandler()	139
7.20.4.6 Initialize()	139
7.20.4.7 operator=()	140
7.20.4.8 Read() [1/2]	140
7.20.4.9 Read() [2/2]	140
7.20.4.10 SetTimeOut()	141
7.20.4.11 Start()	141
7.20.4.12 Stop()	141
7.20.4.13 Write()	142
7.21 I2C_Type Struct Reference	143
7.21.1 Detailed Description	143
7.21.2 Member Data Documentation	143
7.21.2.1 CFG	143
7.21.2.2 CLKDIV	144
7.21.2.3 INTENCLR	144
7.21.2.4 INTENSET	144
7.21.2.5 INTSTAT	144
7.21.2.6 MONRXDAT	144
7.21.2.7 MSTCTL	144
7.21.2.8 MSTDAT	144
7.21.2.9 MSTTIME	145
7.21.2.10 RESERVED_0	145
7.21.2.11 RESERVED_1	145
7.21.2.12 RESERVED_2	145
7.21.2.13 SLVADR	145
7.21.2.14 SLVCTL	145
7.21.2.15 SLVDAT	145
7.21.2.16 SLVQUAL0	145
7.21.2.17 STAT	146
7.21.2.18 TIMEOUT	146
7.22 I2CMaster Class Reference	146
7.22.1 Detailed Description	148
7.22.2 Constructor & Destructor Documentation	148
7.22.2.1 I2CMaster()	148
7.22.3 Member Function Documentation	149

7.22.3.1 Initialize()	149
7.22.3.2 isIdle()	149
7.22.3.3 Read()	149
7.22.3.4 RequestRead()	150
7.22.3.5 Write() [1/2]	150
7.22.3.6 Write() [2/2]	150
7.23 I4017 Class Reference	151
7.23.1 Detailed Description	152
7.23.2 Constructor & Destructor Documentation	152
7.23.2.1 I4017()	152
7.23.3 Member Function Documentation	152
7.23.3.1 SetClock()	152
7.23.3.2 SetDigito()	152
7.23.3.3 SetReset()	153
7.24 I4511 Class Reference	153
7.24.1 Detailed Description	154
7.24.2 Constructor & Destructor Documentation	154
7.24.2.1 I4511()	154
7.24.3 Member Function Documentation	154
7.24.3.1 SetSegmentos()	154
7.25 InOut Class Reference	155
7.25.1 Detailed Description	155
7.25.2 Constructor & Destructor Documentation	156
7.25.2.1 InOut()	156
7.25.2.2 ~InOut()	156
7.25.3 Member Function Documentation	156
7.25.3.1 ClrPin()	156
7.25.3.2 GetPin()	156
7.25.3.3 SetDir()	156
7.25.3.4 SetPin()	156
7.25.3.5 SetPinMode()	157
7.25.3.6 SetPinResistor()	157
7.25.3.7 SetToggleDir()	157
7.25.3.8 SetTogglePin()	157
7.26 Input Class Reference	158
7.26.1 Detailed Description	161
7.26.2 Constructor & Destructor Documentation	161
7.26.2.1 Input()	161
7.26.2.2 ~Input()	161
7.26.3 Member Function Documentation	162
7.26.3.1 get()	162
7.26.3.2 Initialize()	162

7.26.3.3 operator"!=()	162
7.26.3.4 operator==()	162
7.26.3.5 SWhandler()	163
7.26.4 Friends And Related Symbol Documentation	163
7.26.4.1 operator==	163
7.27 IOCON_Type Struct Reference	163
7.27.1 Detailed Description	164
7.27.2 Member Data Documentation	164
7.27.2.1 PIO	164
7.28 L298N Class Reference	164
7.28.1 Detailed Description	166
7.28.2 Constructor & Destructor Documentation	166
7.28.2.1 L298N()	166
7.28.2.2 ~L298N()	166
7.28.3 Member Function Documentation	166
7.28.3.1 Avanzar()	166
7.28.3.2 Frenar()	167
7.28.3.3 Girar()	167
7.28.3.4 GirarDer()	167
7.28.3.5 GirarIzq()	167
7.28.3.6 Initialize()	168
7.28.3.7 Retroceder()	168
7.29 LCD Class Reference	168
7.29.1 Detailed Description	170
7.29.2 Member Enumeration Documentation	170
7.29.2.1 anonymous enum	170
7.29.3 Constructor & Destructor Documentation	170
7.29.3.1 LCD()	170
7.29.4 Member Function Documentation	170
7.29.4.1 Clear()	170
7.29.4.2 Initialize()	171
7.29.4.3 operator=()	171
7.29.4.4 SWhandler()	172
7.29.4.5 Write() [1/2]	172
7.29.4.6 Write() [2/2]	172
7.29.4.7 WriteAt() [1/2]	172
7.29.4.8 WriteAt() [2/2]	173
7.30 MRT_t Struct Reference	173
7.30.1 Detailed Description	173
7.30.2 Member Data Documentation	173
7.30.2.1 MODE	173
7.30.2.2 RESERVED0	174

7.30.2.3 RESERVED1	174
7.30.2.4 RESERVED2	174
7.30.2.5 RUN	174
7.31 MRTHandler Class Reference	174
7.31.1 Detailed Description	175
7.31.2 Constructor & Destructor Documentation	175
7.31.2.1 MRTHandler()	175
7.31.2.2 ~MRTHandler()	175
7.31.3 Member Function Documentation	176
7.31.3.1 MRT_get_time()	176
7.31.3.2 MRT_reset_time()	176
7.31.3.3 MRTHandler()	176
7.31.4 Member Data Documentation	176
7.31.4.1 m_timer_channel	176
7.32 NVIC_Type Struct Reference	176
7.32.1 Detailed Description	177
7.32.2 Member Data Documentation	177
7.32.2.1 ICER	177
7.32.2.2 IP	177
7.32.2.3 ISER	177
7.32.2.4 ISPR	177
7.32.2.5 RESERVED0	178
7.32.2.6 RESERVED2	178
7.32.2.7 RESERVED3	178
7.32.2.8 RESERVED4	178
7.32.2.9 RSERVED1	178
7.33 Output Class Reference	179
7.33.1 Detailed Description	182
7.33.2 Constructor & Destructor Documentation	182
7.33.2.1 Output()	182
7.33.2.2 ~Output()	182
7.33.3 Member Function Documentation	182
7.33.3.1 Initialize()	182
7.33.3.2 Off()	183
7.33.3.3 On()	183
7.33.3.4 operator=()	183
7.33.3.5 operator==()	183
7.33.3.6 SWhandler()	184
7.34 Pin Class Reference	184
7.34.1 Detailed Description	185
7.34.2 Member Typedef Documentation	185
7.34.2.1 error_t	185

7.34.2.2 port_t	185
7.34.3 Member Enumeration Documentation	185
7.34.3.1 error_t	185
7.34.3.2 port_t	186
7.34.4 Constructor & Destructor Documentation	186
7.34.4.1 Pin()	186
7.34.5 Member Data Documentation	186
7.34.5.1 m_bit	186
7.34.5.2 m_error	186
7.34.5.3 m_port	186
7.35 PIN_INTERRUPT_t Struct Reference	187
7.35.1 Detailed Description	187
7.35.2 Member Data Documentation	187
7.35.2.1 CIENF	187
7.35.2.2 CIENR	187
7.35.2.3 FALL	187
7.35.2.4 IENF	188
7.35.2.5 IENR	188
7.35.2.6 ISEL	188
7.35.2.7 IST	188
7.35.2.8 PMCFG	188
7.35.2.9 PMCTRL	188
7.35.2.10 PMSRC	188
7.35.2.11 RISE	188
7.35.2.12 SIENF	189
7.35.2.13 SIENR	189
7.36 PinInterrupt Class Reference	189
7.36.1 Detailed Description	192
7.36.2 Constructor & Destructor Documentation	192
7.36.2.1 PinInterrupt()	192
7.36.2.2 ~PinInterrupt()	193
7.36.3 Member Function Documentation	193
7.36.3.1 DisableInterrupt()	193
7.36.3.2 EnableInterrupt()	193
7.36.3.3 GpioHandler()	193
7.36.3.4 PinInterrupt_Inicializar()	193
7.36.4 Member Data Documentation	193
7.36.4.1 m_cant	193
7.36.4.2 m_interrupt_mode	194
7.36.4.3 m_interrupt_number	194
7.37 Puente_H Class Reference	194
7.37.1 Detailed Description	195

7.37.2 Member Enumeration Documentation	195
7.37.2.1 anonymous enum	195
7.37.3 Constructor & Destructor Documentation	195
7.37.3.1 Puente_H()	195
7.37.3.2 ~Puente_H()	195
7.37.4 Member Function Documentation	195
7.37.4.1 Avanzar()	195
7.37.4.2 Frenar()	196
7.37.4.3 Girar()	196
7.37.4.4 GirarDer()	196
7.37.4.5 Girarlzq()	196
7.37.4.6 Initialize()	196
7.37.4.7 Retroceder()	197
7.38 Pwm Class Reference	197
7.38.1 Detailed Description	199
7.38.2 Member Enumeration Documentation	199
7.38.2.1 activity_t	199
7.38.2.2 pwm_channel_t	199
7.38.2.3 pwm_time_unit_t	200
7.38.3 Constructor & Destructor Documentation	200
7.38.3.1 Pwm()	200
7.38.3.2 ~Pwm()	200
7.38.4 Member Function Documentation	200
7.38.4.1 Initialize()	200
7.38.4.2 Off()	201
7.38.4.3 On()	201
7.38.4.4 SetPeriod()	201
7.38.4.5 SetTon()	201
7.38.5 Member Data Documentation	202
7.38.5.1 m_activity	202
7.38.5.2 m_pwm_channel	202
7.38.5.3 m_toff	202
7.38.5.4 m_ton	202
7.39 PWM_Reader Class Reference	202
7.39.1 Detailed Description	206
7.39.2 Constructor & Destructor Documentation	207
7.39.2.1 PWM_Reader()	207
7.39.3 Member Function Documentation	208
7.39.3.1 GetPulseOn()	208
7.39.3.2 GpioHandler()	208
7.39.3.3 Initialize()	208
7.39.3.4 Off()	208

7.39.3.5 On()	208
7.40 Reloj Class Reference	209
7.40.1 Detailed Description	210
7.40.2 Constructor & Destructor Documentation	210
7.40.2.1 Reloj()	210
7.40.3 Member Function Documentation	210
7.40.3.1 GetHour()	210
7.40.3.2 GetMin()	210
7.40.3.3 GetSeg()	210
7.40.3.4 Reset()	211
7.40.3.5 SetTime()	211
7.40.3.6 SWhandler()	211
7.41 SCT_t Struct Reference	211
7.41.1 Detailed Description	212
7.41.2 Member Data Documentation	212
7.41.2.1 CAP	212
7.41.2.2 CAPCTRL	212
7.41.2.3 CLR	212
7.41.2.4 CONEN	212
7.41.2.5 CONFIG	212
7.41.2.6 CONFLAG	212
7.41.2.7 COUNT	213
7.41.2.8 CTRL	213
7.41.2.9 DMAREQ0	213
7.41.2.10 DMAREQ1	213
7.41.2.11 [struct]	213
7.41.2.12 EVEN	213
7.41.2.13 EVFLAG	213
7.41.2.14 HALT	213
7.41.2.15 INPUT	213
7.41.2.16 LIMIT	213
7.41.2.17 MATCH	213
7.41.2.18 MATCHREL	213
7.41.2.19 [struct]	214
7.41.2.20 OUTPUT	214
7.41.2.21 OUTPUTDIRCTRL	214
7.41.2.22 REGMODE	214
7.41.2.23 RES	214
7.41.2.24 RESERVED_0	214
7.41.2.25 RESERVED_1	214
7.41.2.26 RESERVED_2	214
7.41.2.27 RESERVED_3	214

7.41.2.28 RESERVED_4	214
7.41.2.29 SET	214
7.41.2.30 START	214
7.41.2.31 STATE	215
7.41.2.32 STOP	215
7.42 SCtimer Class Reference	215
7.42.1 Detailed Description	215
7.42.2 Constructor & Destructor Documentation	216
7.42.2.1 ~SCtimer()	216
7.42.3 Member Function Documentation	216
7.42.3.1 SetAutoLimit()	216
7.42.3.2 SetSwitchMatrizSCTOUT()	216
7.42.3.3 SetTime()	216
7.42.3.4 SetUnify()	216
7.42.3.5 StartTimer()	217
7.42.3.6 StopTimer()	217
7.43 segmentos Class Reference	217
7.43.1 Detailed Description	217
7.43.2 Member Function Documentation	217
7.43.2.1 Initialize()	217
7.43.2.2 SetSegmentos()	218
7.44 SPI Class Reference	218
7.44.1 Detailed Description	220
7.44.2 Member Enumeration Documentation	220
7.44.2.1 SPI_mode_t	220
7.44.2.2 SPI_role_t	220
7.44.3 Constructor & Destructor Documentation	220
7.44.3.1 SPI()	220
7.44.3.2 ~SPI()	221
7.44.4 Member Function Documentation	221
7.44.4.1 AddSSEL()	221
7.44.4.2 ClearEOF()	221
7.44.4.3 ClearEOT()	221
7.44.4.4 ClearSSEL()	221
7.44.4.5 forceFinish()	222
7.44.4.6 Initialize()	222
7.44.4.7 isRxReady()	222
7.44.4.8 isTxReady()	222
7.44.4.9 Read()	222
7.44.4.10 RemoveSSEL()	223
7.44.4.11 SetEOT()	223
7.44.4.12 SetSSEL()	223

7.44.4.13 Write()	223
7.44.5 Member Data Documentation	224
7.44.5.1 m_SPI_register	224
7.45 SPI_Type Struct Reference	224
7.45.1 Detailed Description	224
7.45.2 Member Data Documentation	224
7.45.2.1 CFG	224
7.45.2.2 DIV	224
7.45.2.3 DLY	224
7.45.2.4 INTENCLR	224
7.45.2.5 INTENSET	225
7.45.2.6 INTSTAT	225
7.45.2.7 RXDAT	225
7.45.2.8 STAT	225
7.45.2.9 TXCTL	225
7.45.2.10 TXDAT	225
7.45.2.11 TXDATCTL	225
7.46 SPIMaster Class Reference	225
7.46.1 Detailed Description	228
7.46.2 Constructor & Destructor Documentation	228
7.46.2.1 SPIMaster()	228
7.46.3 Member Function Documentation	228
7.46.3.1 AddSSEL()	228
7.46.3.2 hasData()	229
7.46.3.3 Initialize()	229
7.46.3.4 IsIdle()	229
7.46.3.5 Read()	229
7.46.3.6 RemoveSSEL()	229
7.46.3.7 RequestRead()	230
7.46.3.8 Write() [1/2]	230
7.46.3.9 Write() [2/2]	230
7.47 SWM_t Struct Reference	231
7.47.1 Detailed Description	231
7.47.2 Member Data Documentation	231
7.47.2.1 PINASSIGN0	231
7.47.2.2 PINASSIGN1	231
7.47.2.3 PINASSIGN10	231
7.47.2.4 PINASSIGN11	231
7.47.2.5 PINASSIGN12	231
7.47.2.6 PINASSIGN13	231
7.47.2.7 PINASSIGN14	231
7.47.2.8 PINASSIGN2	231

7.47.2.9 PINASSIGN3	232
7.47.2.10 PINASSIGN4	232
7.47.2.11 PINASSIGN5	232
7.47.2.12 PINASSIGN6	232
7.47.2.13 PINASSIGN7	232
7.47.2.14 PINASSIGN8	232
7.47.2.15 PINASSIGN9	232
7.47.2.16 PINASSIGN_DATA	232
7.47.2.17 PINENABLE0	232
7.47.2.18 PINENABLE1	232
7.47.2.19 RESERVED_0	232
7.48 SYSCON_Type Struct Reference	233
7.48.1 Detailed Description	234
7.48.2 Member Data Documentation	234
7.48.2.1 ADCCLKDIV	234
7.48.2.2 ADCCLKSEL	234
7.48.2.3 BODCTRL	234
7.48.2.4 CAPTCLKSEL	234
7.48.2.5 CLKOUTDIV	234
7.48.2.6 CLKOUTSEL	235
7.48.2.7 DEVICE_ID	235
7.48.2.8 EXTCLKSEL	235
7.48.2.9 EXTRACECMD	235
7.48.2.10 FCLKSEL	235
7.48.2.11 [struct]	235
7.48.2.12 FRGCLKSEL	235
7.48.2.13 FRGDIV	235
7.48.2.14 FRGMULT	235
7.48.2.15 FRODIRECTCLKUEN	235
7.48.2.16 FROOSCCTRL	235
7.48.2.17 IOCONCLKDIV0	235
7.48.2.18 IOCONCLKDIV1	236
7.48.2.19 IOCONCLKDIV2	236
7.48.2.20 IOCONCLKDIV3	236
7.48.2.21 IOCONCLKDIV4	236
7.48.2.22 IOCONCLKDIV5	236
7.48.2.23 IOCONCLKDIV6	236
7.48.2.24 IRQLATENCY	236
7.48.2.25 MAINCLKPLLSEL	236
7.48.2.26 MAINCLKPLLUEN	236
7.48.2.27 MAINCLKSEL	236
7.48.2.28 MAINCLKUEN	236

7.48.2.29 NMISRC	236
7.48.2.30 PDAWAKECFG	237
7.48.2.31 PDRUNCFG	237
7.48.2.32 PDSLEEPcfg	237
7.48.2.33 PINTSEL	237
7.48.2.34 PIOPORCAP	237
7.48.2.35 PRESETCTRL0	237
7.48.2.36 PRESETCTRL1	237
7.48.2.37 RESERVED_0	237
7.48.2.38 RESERVED_1	237
7.48.2.39 RESERVED_10	237
7.48.2.40 RESERVED_11	237
7.48.2.41 RESERVED_12	237
7.48.2.42 RESERVED_13	238
7.48.2.43 RESERVED_14	238
7.48.2.44 RESERVED_2	238
7.48.2.45 RESERVED_3	238
7.48.2.46 RESERVED_4	238
7.48.2.47 RESERVED_5	238
7.48.2.48 RESERVED_6	238
7.48.2.49 RESERVED_7	238
7.48.2.50 RESERVED_8	238
7.48.2.51 RESERVED_9	238
7.48.2.52 SCTCLKDIV	238
7.48.2.53 SCTCLKSEL	238
7.48.2.54 STARTERP0	239
7.48.2.55 STARTERP1	239
7.48.2.56 SYSAHBCLKCTRL0	239
7.48.2.57 SYSAHBCLKCTRL1	239
7.48.2.58 SYSAHBCLKDIV	239
7.48.2.59 SYSMEMREMAP	239
7.48.2.60 SYSOSCCTRL	239
7.48.2.61 SYSPLLCLKSEL	239
7.48.2.62 SYSPLLCLKUEN	239
7.48.2.63 SYSPLLCTRL	239
7.48.2.64 SYSPLLSTAT	239
7.48.2.65 SYSRSTSTAT	239
7.48.2.66 SYSTCKCAL	240
7.48.2.67 WDTOSCCTRL	240
7.49 SysTick_t Struct Reference	240
7.49.1 Detailed Description	240
7.49.2 Member Data Documentation	240

7.49.2.1 CALIB	240
7.49.2.2 CTRL	240
7.49.2.3 CURR	240
7.49.2.4 RELOAD	240
7.50 teclado Class Reference	241
7.50.1 Detailed Description	242
7.50.2 Constructor & Destructor Documentation	242
7.50.2.1 teclado()	242
7.50.2.2 ~teclado()	242
7.50.3 Member Function Documentation	242
7.50.3.1 Get()	242
7.50.3.2 Initialize()	242
7.50.3.3 SWhandler()	242
7.51 Timer Class Reference	243
7.51.1 Detailed Description	245
7.51.2 Member Typedef Documentation	245
7.51.2.1 bases_t	245
7.51.3 Member Enumeration Documentation	245
7.51.3.1 bases_t	245
7.51.4 Constructor & Destructor Documentation	245
7.51.4.1 Timer() [1/2]	245
7.51.4.2 Timer() [2/2]	245
7.51.5 Member Function Documentation	245
7.51.5.1 GetmrStandBy()	245
7.51.5.2 GetTimer()	246
7.51.5.3 GetTmrEvent()	246
7.51.5.4 GetTmrRun()	246
7.51.5.5 operator bool()	246
7.51.5.6 operator"!"()	246
7.51.5.7 operator=()	246
7.51.5.8 operator==()	247
7.51.5.9 SetmrStandBy()	247
7.51.5.10 SetTimer()	247
7.51.5.11 SetTimerBase()	247
7.51.5.12 SetTmrHandler()	247
7.51.5.13 StandByTimer()	248
7.51.5.14 TimerStart() [1/2]	248
7.51.5.15 TimerStart() [2/2]	248
7.51.5.16 TmrEvent()	248
7.51.6 Friends And Related Symbol Documentation	248
7.51.6.1 operator==	248
7.51.7 Member Data Documentation	249

7.51.7.1 m_TmrBase	249
7.51.7.2 m_TmrEvent	249
7.51.7.3 m_TmrHandler	249
7.51.7.4 m_TmrRun	249
7.51.7.5 m_TmrStandBy	249
7.52 timers Class Reference	249
7.52.1 Detailed Description	249
7.52.2 Constructor & Destructor Documentation	250
7.52.2.1 timers()	250
7.52.2.2 ~timers()	250
7.52.3 Member Function Documentation	250
7.52.3.1 operator<<()	250
7.52.3.2 TmrEvent()	250
7.53 Uart Class Reference	250
7.53.1 Detailed Description	250
7.54 USART_Type Struct Reference	250
7.54.1 Detailed Description	251
7.54.2 Member Data Documentation	251
7.54.2.1 ADDR	251
7.54.2.2 BRG	251
7.54.2.3 CFG	251
7.54.2.4 CTL	251
7.54.2.5 INTENCLR	251
7.54.2.6 INTENSET	251
7.54.2.7 INTSTAT	251
7.54.2.8 OSR	252
7.54.2.9 RXDAT	252
7.54.2.10 RXDATSTAT	252
7.54.2.11 STAT	252
7.54.2.12 TXDAT	252
8 File Documentation	253
8.1 Comunicacion.cpp File Reference	253
8.1.1 Detailed Description	253
8.2 Comunicacion.h File Reference	253
8.2.1 Detailed Description	254
8.3 Comunicacion.h	254
8.4 inicializar.cpp File Reference	255
8.4.1 Detailed Description	256
8.5 inicializar.h File Reference	256
8.5.1 Detailed Description	257
8.6 inicializar.h	257

8.7 Semaforo.cpp File Reference	258
8.7.1 Detailed Description	258
8.8 Semaforo.h File Reference	258
8.8.1 Detailed Description	259
8.9 Semaforo.h	259
8.10 CircularBuffer.h File Reference	260
8.10.1 Detailed Description	261
8.11 CircularBuffer.h	262
8.12 Drivers.h File Reference	263
8.12.1 Detailed Description	264
8.13 Drivers.h	264
8.14 teclado.cpp File Reference	265
8.14.1 Detailed Description	265
8.15 teclado.h File Reference	265
8.15.1 Detailed Description	267
8.15.2 Macro Definition Documentation	267
8.15.2.1 NO_KEY	267
8.16 teclado.h	268
8.17 Barrido.h File Reference	268
8.17.1 Detailed Description	269
8.18 Barrido.h	269
8.19 I4017.cpp File Reference	270
8.19.1 Detailed Description	271
8.20 I4017.h File Reference	271
8.20.1 Detailed Description	272
8.21 I4017.h	273
8.22 I4511.cpp File Reference	274
8.22.1 Detailed Description	274
8.23 I4511.h File Reference	274
8.23.1 Detailed Description	275
8.24 I4511.h	276
8.25 Segmentos.h File Reference	276
8.25.1 Detailed Description	278
8.26 Segmentos.h	278
8.27 Dígito.cpp File Reference	278
8.27.1 Detailed Description	279
8.27.2 Variable Documentation	279
8.27.2.1 Tabla_Dígitos_BCD_7seg	279
8.28 Dígito.h File Reference	279
8.28.1 Detailed Description	281
8.29 Dígito.h	281
8.30 Display7Segmentos.cpp File Reference	282

8.30.1 Detailed Description	282
8.31 Display7Segmentos.h File Reference	282
8.31.1 Detailed Description	284
8.31.2 Macro Definition Documentation	284
8.31.2.1 UPDATE_TICKS	284
8.32 Display7Segmentos.h	284
8.33 GrupoDeDigitos.h File Reference	285
8.33.1 Detailed Description	286
8.34 GrupoDeDigitos.h	286
8.35 Display.h File Reference	287
8.35.1 Detailed Description	288
8.36 Display.h	288
8.37 LCD.cpp File Reference	289
8.37.1 Detailed Description	289
8.38 LCD.h File Reference	290
8.38.1 Detailed Description	291
8.38.2 Macro Definition Documentation	292
8.38.2.1 INSTRUCTION_8BITS	292
8.38.2.2 INSTRUCTION_CLEAR_DISP	292
8.38.2.3 INSTRUCTION_DISP_CTRL	292
8.38.2.4 INSTRUCTION_ENTRY_MODE_SET	292
8.38.2.5 INSTRUCTION_FUNC_SET	292
8.38.2.6 INSTRUCTION_SET_POS	292
8.38.2.7 MAX_PIN_COUNT	292
8.38.2.8 ROW_OFFSET	292
8.39 LCD.h	292
8.40 L298N.cpp File Reference	293
8.40.1 Detailed Description	294
8.41 L298N.h File Reference	294
8.41.1 Detailed Description	296
8.42 L298N.h	296
8.43 PuenteH.h File Reference	297
8.43.1 Detailed Description	298
8.44 PuenteH.h	299
8.45 Reloj.cpp File Reference	299
8.45.1 Detailed Description	300
8.46 Reloj.h File Reference	300
8.46.1 Detailed Description	302
8.47 Reloj.h	302
8.48 Pwm.cpp File Reference	303
8.48.1 Detailed Description	304
8.49 Pwm.h File Reference	304

8.49.1 Detailed Description	306
8.50 Pwm.h	306
8.51 PWMReader.cpp File Reference	307
8.51.1 Detailed Description	308
8.52 PWMReader.h File Reference	308
8.52.1 Detailed Description	310
8.53 PWMReader.h	310
8.54 distancia.h File Reference	311
8.54.1 Detailed Description	313
8.55 distancia.h	313
8.56 HCSR04.cpp File Reference	313
8.56.1 Detailed Description	314
8.57 HCSR04.h File Reference	315
8.57.1 Detailed Description	316
8.57.2 Macro Definition Documentation	317
8.57.2.1 CALC_DISTANCIA	317
8.57.2.2 DISTANCIA_MAX	317
8.57.2.3 PERIODO	317
8.58 HCSR04.h	317
8.59 ESP8266.cpp File Reference	318
8.59.1 Detailed Description	318
8.60 ESP8266.h File Reference	318
8.60.1 Detailed Description	320
8.60.2 Macro Definition Documentation	320
8.60.2.1 DEFAULT_ESP01_BAUDRATE	320
8.60.2.2 SEG_ESP01_TIMEOUT	320
8.61 ESP8266.h	320
8.62 Pin.cpp File Reference	321
8.62.1 Detailed Description	322
8.63 Pin.h File Reference	322
8.63.1 Detailed Description	323
8.64 Pin.h	323
8.65 gpio.cpp File Reference	324
8.65.1 Detailed Description	325
8.66 gpio.h File Reference	325
8.66.1 Detailed Description	326
8.67 gpio.h	327
8.68 InOut.h File Reference	328
8.68.1 Detailed Description	328
8.69 InOut.h	329
8.70 Pininterrupt.cpp File Reference	329
8.70.1 Detailed Description	330

8.71 Pininterrupt.h File Reference	331
8.71.1 Detailed Description	332
8.71.2 Macro Definition Documentation	333
8.71.2.1 MAX_PININTERRUPT	333
8.72 Pininterrupt.h	333
8.73 Input.cpp File Reference	334
8.73.1 Detailed Description	335
8.73.2 Function Documentation	335
8.73.2.1 operator==()	335
8.74 Input.h File Reference	335
8.74.1 Detailed Description	337
8.74.2 Macro Definition Documentation	337
8.74.2.1 MAX_BOUNCE	337
8.75 Input.h	337
8.76 Output.cpp File Reference	338
8.76.1 Detailed Description	339
8.77 Output.h File Reference	340
8.77.1 Detailed Description	341
8.78 Output.h	341
8.79 Callback.cpp File Reference	342
8.79.1 Detailed Description	343
8.80 Callback.h File Reference	343
8.80.1 Detailed Description	345
8.80.2 Macro Definition Documentation	345
8.80.2.1 TICK_MICROSECONDS	345
8.80.2.2 TICK_MILISECONDS	345
8.80.2.3 TICK_SECONDS	345
8.81 Callback.h	345
8.82 Timer.h File Reference	346
8.82.1 Detailed Description	347
8.83 Timer.h	347
8.84 Timers.cpp File Reference	349
8.84.1 Detailed Description	349
8.85 Timers.h File Reference	350
8.85.1 Detailed Description	351
8.86 Timers.h	351
8.87 MRTHandler.cpp File Reference	352
8.87.1 Detailed Description	353
8.88 MRTHandler.h File Reference	353
8.88.1 Detailed Description	355
8.88.2 Macro Definition Documentation	355
8.88.2.1 MAX_MRT_CHANNEL	355

8.88.3 Enumeration Type Documentation	355
8.88.3.1 MRT_MODES	355
8.88.3.2 MRT_timer_channels	355
8.89 MRTHandler.h	355
8.90 SCtimer.cpp File Reference	356
8.90.1 Detailed Description	357
8.91 SCtimer.h File Reference	358
8.91.1 Detailed Description	359
8.92 SCtimer.h	360
8.93 DAC.cpp File Reference	360
8.93.1 Detailed Description	361
8.94 DAC.h File Reference	362
8.94.1 Detailed Description	363
8.94.2 Macro Definition Documentation	364
8.94.2.1 MAX_DAC_CHANNEL	364
8.94.2.2 MAX_DAC_VALUE	364
8.95 DAC.h	364
8.96 ADC.cpp File Reference	365
8.96.1 Detailed Description	366
8.96.2 Variable Documentation	366
8.96.2.1 pin_index	366
8.97 ADC.h File Reference	367
8.97.1 Detailed Description	368
8.98 ADC.h	368
8.99 ADCGroup.cpp File Reference	369
8.99.1 Detailed Description	370
8.100 ADCGroup.h	370
8.101 ComunicacionAsincronica.h File Reference	372
8.101.1 Detailed Description	373
8.102 ComunicacionAsincronica.h	373
8.103 UART.cpp File Reference	374
8.103.1 Detailed Description	375
8.103.2 Variable Documentation	375
8.103.2.1 g_usart	375
8.104 UART.h File Reference	375
8.104.1 Detailed Description	376
8.105 UART.h	377
8.106 ComunicacionSincronica.h File Reference	378
8.106.1 Detailed Description	379
8.107 ComunicacionSincronica.h	380
8.108 I2C.cpp File Reference	380
8.108.1 Detailed Description	381

8.108.2 Macro Definition Documentation	382
8.108.2.1 MAX_IC2	382
8.108.3 Variable Documentation	382
8.108.3.1 g_i2c	382
8.109 I2C.h File Reference	382
8.109.1 Detailed Description	383
8.109.2 Macro Definition Documentation	384
8.109.2.1 I2C_MAX_FREQ	384
8.110 I2C.h	384
8.111 I2CMaster.cpp File Reference	385
8.111.1 Detailed Description	386
8.112 I2CMaster.h File Reference	386
8.112.1 Detailed Description	388
8.113 I2CMaster.h	389
8.114 SPI.cpp File Reference	389
8.114.1 Detailed Description	390
8.114.2 Macro Definition Documentation	391
8.114.2.1 MAX_SPI	391
8.114.3 Variable Documentation	391
8.114.3.1 g_spi	391
8.115 SPI.h File Reference	391
8.115.1 Detailed Description	392
8.115.2 Macro Definition Documentation	393
8.115.2.1 MAX_SSEL_SPI0	393
8.115.2.2 MAX_SSEL_SPI1	393
8.116 SPI.h	393
8.117 SPIMaster.cpp File Reference	394
8.117.1 Detailed Description	395
8.118 SPIMaster.h File Reference	395
8.118.1 Detailed Description	397
8.118.2 Macro Definition Documentation	398
8.118.2.1 NO_DATA	398
8.119 SPIMaster.h	398
8.120 dr_pll.cpp File Reference	399
8.120.1 Detailed Description	399
8.120.2 Function Documentation	400
8.120.2.1 Inicializar_PLL()	400
8.121 dr_pll.h File Reference	400
8.121.1 Detailed Description	401
8.121.2 Function Documentation	401
8.121.2.1 Inicializar_PLL()	401
8.122 dr_pll.h	402

8.123 systick.cpp File Reference	402
8.123.1 Detailed Description	403
8.123.2 Macro Definition Documentation	404
8.123.2.1 MAX_TICKS	404
8.123.3 Function Documentation	404
8.123.3.1 Inicializar_SysTick()	404
8.123.4 Variable Documentation	404
8.123.4.1 g_systick_freq	404
8.124 systick.h File Reference	404
8.124.1 Detailed Description	405
8.124.2 Function Documentation	406
8.124.2.1 Inicializar_SysTick()	406
8.124.3 Variable Documentation	406
8.124.3.1 g_systick_freq	406
8.125 systick.h	406
8.126 LPC845.h File Reference	407
8.126.1 Detailed Description	413
8.127 LPC845.h	414
8.128 tipos.h File Reference	432
8.128.1 Detailed Description	432
8.128.2 Macro Definition Documentation	433
8.128.2.1 __R	433
8.128.2.2 __RW	433
8.128.2.3 __W	433
8.128.2.4 INT32_MAX	433
8.128.2.5 INT32_MIN	433
8.128.2.6 INT8_MAX	433
8.128.2.7 INT8_MIN	433
8.128.2.8 UINT32_MAX	433
8.128.2.9 UINT8_MAX	433
8.128.3 Typedef Documentation	433
8.128.3.1 int16_t	433
8.128.3.2 int32_t	433
8.128.3.3 int8_t	434
8.128.3.4 uint16_t	434
8.128.3.5 uint32_t	434
8.128.3.6 uint8_t	434
8.129 tipos.h	434
Index	437

Chapter 1

KITLPC845

Librerías propias comenzadas a realizar en Informática II y continuadas de forma particular. Cubren la mayoría de los periféricos del Kit de desarrollo LPC845. No poseen implementaciones de DMA.

1.1 Información:



EN CASO DE QUERER UTILIZAR ESTA PLAQUETA:

- Nombre de la placa: LPC845.
- Alimentación de 3,3V. (Para alimentar externamente GND en pin 20 y 3,3v en pin 40).
- Programada con "MCUexpresso".
- Lenguaje C++.

1.1.1 Drivers/Periféricos ya creados:

- [Input](#).
- [Output](#).
- [Timer](#).
- [Reloj](#).
- [ADC](#).
- [DAC](#).
- [PWM](#).
- Lector de ancho de pulso ([PWM_Reader](#)).
- [UART](#) (comunicación serie).
- [Display](#) de 7 segmentos.
- [Display LCD](#).

- Teclado matricial.
- Puente H para motores.
- Ultrasónico.
- ESP01 (Wifi arduino con comandos AT).
- [I2C](#).
- [SPI](#).

1.1.2 Para crear un proyecto nuevo:

- File.
- New.
- New C/C++ Projetc.
- (Se selecciona la placa a utilizar) Next >.
- C++ Projetc.
- (Se escribe el nombre del proyecto).
- Se toca "Next >" hasta terminar.

1.1.3 Para utilizar este proyecto:

- Import project(s) from file system.
- (Se selecciona el archivo .zip o esta carpeta) Next > .
- Finish.
- Se puede renombrar al proyecto con cualquier nombre. (Los includes se autoconfiguran para seguir funcionando). (Con la placa conectada)
- El "bicho azul" carga el programa la primera vez.
- El "bicho verde" carga el programa las veces siguientes.

En caso de requerir los nombres de cada pata y su función, buscar hoja de datos en internet.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Misc	15
Drivers	15
Abstracta	18
REGISTERS Peripheral Access Layer	19
SYSCON Peripheral Access Layer	20
GPIO Peripheral Access Layer	22
IOCON Peripheral Access Layer	22
SYSTICK Peripheral Access Layer	23
MRT Peripheral Access Layer	24
NVIC Peripheral Access Layer	26
PIN_INT Peripheral Access Layer	27
SCT Peripheral Access Layer	28
USART Peripheral Access Layer	28
SWM Peripheral Access Layer	31
ADC Peripheral Access Layer	31
ADC Register Masks	33
DAC Peripheral Access Layer	56
I2C Peripheral Access Layer	57
I2C Register Masks	59
SPI Peripheral Access Layer	72
SPI Register Masks	73

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ADC_Group	84
ADC_Type	88
barido	91
I4017	151
Callback	92
Input	158
LCD	168
Output	179
Reloj	209
Timer	243
display7Segmentos	111
teclado	241
CircularBuffer< T >	93
CircularBuffer< uint8_t >	93
ComunicacionAsincronica	95
ComunicacionSincronica	97
I2C	136
I2CMaster	146
SPI	218
SPIMaster	225
DAC_t	105
digito	108
Display	110
LCD	168
display7Segmentos	111
distancia	114
HC_SR04	132
GPIO_Type	127
gruposdedigitos	131
I2C_Type	143
InOut	155
gpio	121
Input	158

Output	179
PinInterrupt	189
PWM_Reader	202
IOCON_Type	163
MRT_t	173
MRThandler	174
PWM_Reader	202
NVIC_Type	176
Pin	184
ADC	81
DAC	99
Pwm	197
gpio	121
PIN_INTERRUPT_t	187
Puente_H	194
L298N	164
SCT_t	211
SCtimer	215
Pwm	197
segmentos	217
I4511	153
SPI_Type	224
SWM_t	231
SYSCON_Type	233
SysTick_t	240
timers	249
Uart	250
USART_Type	250

Chapter 4

Class Index

4.1 Class List

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

ADC	Clase del objeto ADC FUNCIONAMIENTO: La clase ADC utiliza el ADCGroup para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del ADC existen. La configuración de conversión se realiza automáticamente con el primer objeto ADC creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo	81
ADC_Group	Clase del objeto ADC_Group FUNCIONAMIENTO: Solo debe crearse UN objeto ADC . Todos los canales y distintos ADC son manejados por el mismo objeto. Realiza un barrido y guarda a todos los ADC en un vector de resultados	84
ADC_Type	Structure type to access the Analog Digital Convertor (ADC)	88
barrido	Clase del objeto barrido Clase abstracta pura para la generación de barridos	91
Callback	Clase del objeto Callback	92
CircularBuffer< T >	Clase del objeto CircularBuffer	93
ComunicacionAsincronica	Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de UART .	95
ComunicacionSincronica	Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI	97
DAC	Clase del objeto DAC FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0	99
DAC_t	Structure type to access the Digital Analog Convertor (DAC)	105
digito	Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos	108
Display	Clase del objeto display Clase abstracta pura para la generación de displays	110

display7Segmentos	Clase del objeto display7Segmentos El objeto display7Segmentos permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor	111
distancia	Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04	114
ESP8266	Clase del objeto ESP8266 El objeto ESP8266 permite la simple utilización del módulo arduino ESP8266 y el ESP01 mediante comandos AT. El módulo debe estar por defecto en la velocidad DEFAULT_ESP01_BAUDRATE. El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Sí se probó la inicialización y conexión a internet, no se probó la conexión a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver	116
gpio	Clase del objeto gpio	121
GPIO_Type	Structure type to access the General Purpose Input Output (GPIO)	127
gruposdedigitos	Estructura de grupo de dígitos	131
HC_SR04	Clase del objeto HC_SR04 El objeto HC_SR04 Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades	132
I2C	Clase del objeto I2C El objeto I2C genera una comunicación sincrónica de tipo I2C. Posee las funciones básicas como start, stop, write y read	136
I2C_Type	143
I2CMaster	Clase del objeto I2CMaster El objeto I2CMaster genera una comunicación tipo master de I2C utilizando buffers de recepción y transmisión con interrupciones	146
I4017	Clase del objeto I4017 El objeto I4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a través de sus patas	151
I4511	Clase del objeto I4511 El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo	153
InOut	Clase del objeto InOut	155
Input	Clase del objeto Input	158
IOCON_Type	Structure type to access the IOCON	163
L298N	Clase del objeto L298N El objeto L298N realiza las acciones de control de dos motores controlados por el correspondiente periférico	164
LCD	Clase del objeto LCD El objeto LCD permite el manejo de displays digitales mediante comunicación de 4 bits	168
MRT_t	Structure type to access the MRT timer	173
MRTHandler	Clase del objeto MRTHandler El objeto MRTHandler debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer	174
NVIC_Type	Structure type to access the Nested Vectored Interrupt Controller (NVIC)	176

Output	Clase del objeto outputs	179
Pin	Clase del objeto Pin	184
PIN_INTERRUPT_t	Structure type to access the Pin Interrupt	187
PinInterrupt	Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interrupciones por pin	189
Puente_H	Clase del objeto Puente_H El objeto Puente_H es la interfaz abstracta pura de cualquier puente H que se desee realizar	194
Pwm	Clase del objeto Pwm	197
PWM_Reader	Clase del objeto PWM_Reader	202
Reloj	Clase del objeto Reloj	209
SCT_t	Structure type to access the SCT	211
SCtimer	Clase del objeto SCtimer El objeto SCtimer debe ser heredado por quienes deseen utilizar las interrupciones o funcionalidades del SCtimer	215
segmentos	Clase del objeto segmentos Clase abstracta pura para la generación de segmentos	217
SPI	Clase del objeto SPI	218
SPI_Type		224
SPIMaster	Clase del objeto SPIMaster	225
SWM_t	Structure type to access the Switch Matriz (SWM)	231
SYSCON_Type	Structure type to access the SYSCON	233
SysTick_t	Structure type to access the System Timer (SysTick)	240
teclado	Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla	241
Timer	Clase del objeto timer	243
timers	Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código	249
Uart	Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART	250
USART_Type	Structure type to access the Universal Serial Asincronical Reciver Transmitter (USART)	250

Chapter 5

File Index

5.1 File List

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

Comunicacion.cpp	Clase máquina de estados de comunicación del proyecto	253
Comunicacion.h	Clase máquina de estados de comunicación del proyecto	253
inicializar.cpp	Archivo inicializador de la placa	255
inicializar.h	Archivo inicializador de la placa	256
Semaforo.cpp	Clases semáforo de ejemplo utilizando led tricolor propio de la placa	258
Semaforo.h	Clases semáforo de ejemplo utilizando led tricolor propio de la placa	258
CircularBuffer.h	Modulo de comunicacion CircularBuffer	260
Drivers.h	Archivo Master con todos los includes de perifericos del Kit	263
teclado.cpp	Modulo de teclado matricial	265
teclado.h	Modulo de teclado matricial	265
Barrido.h	Clase abstracta de manejo barridos de datos	268
I4017.cpp	Objeto de control del integrado I4017	270
I4017.h	Objeto de control del integrado I4017	271
I4511.cpp	Objeto de control del integrado I4511	274
I4511.h	Objeto de control del integrado I4511	274
Segmentos.h	Clase abstracta de manejo de segmentos binarios	276
Digito.cpp	Objeto dígito genérico para implementaciones posteriores	278
Digito.h	Objeto dígito genérico para implementaciones posteriores	279

Display7Segmentos.cpp	Clase para la creación de displays de 7 segmentos	282
Display7Segmentos.h	Clase para la creación de displays de 7 segmentos	282
GrupoDeDigitos.h	Clase para la agrupación de dígitos	285
Display.h	Clase base para objetos del tipo pantallas/displays	287
LCD.cpp	Clase para manejar un LCD con comunicación de 4 patas (solo escritura)	289
LCD.h	Clase para manejar un LCD con comunicación de 4 patas (solo escritura)	290
L298N.cpp	Clase del módulo de puente H LN298N	293
L298N.h	Clase del módulo de puente H LN298N	294
PuenteH.h	Clase base para objetos del tipo Puente H	297
Reloj.cpp	Objeto que guardará el tiempo desde que se creo	299
Reloj.h	Objeto que guardará el tiempo desde que se creo	300
Pwm.cpp	Generador de PWM sin interrupción	303
Pwm.h	Generador de PWM sin interrupción	304
PWMReader.cpp	Pata que lee tamaños de pulsos de entrada	307
PWMReader.h	Pata que lee tamaños de pulsos de entrada	308
distancia.h	Clase base para objetos medidores de distancias	311
HCSR04.cpp	Clase del sensor ultrasónico HCSR04	313
HCSR04.h	Clase del sensor ultrasónico HCSR04	315
ESP8266.cpp	Modulo de comunicacion WIFI con ESP8266	318
ESP8266.h	Modulo de comunicacion WIFI con ESP8266	318
Pin.cpp	Clase Abstracta de cualquier pin del microcontrolador	321
Pin.h	Clase Abstracta de cualquier pin del microcontrolador	322
gpio.cpp	Módulo con clase de manejo de GPIO	324
gpio.h	Módulo con clase de manejo de GPIO	325
InOut.h	Clase Abstracta Pura de las GPIO	328
Pininterrupt.cpp	Clase para entradas con interrupciones por flanco	329
Pininterrupt.h	Clase para entradas con interrupciones por flanco	331
Input.cpp	Funciones miembro de la clase Input	334
Input.h	Funciones miembro de la clase Input	335

Output.cpp	Clase del tipo Output o salida digital	338
Output.h	Clase del tipo Output o salida digital	340
Callback.cpp	Funciones miembro de la clase Callback	342
Callback.h	Clase virtual pura. Esta clase debe ser heredada por las clases que se tienen que enganchar del Systick Timer	343
Timer.h	Clase para creacion de temporizadores	346
Timers.cpp	Funciones miembro de la clase timers	349
Timers.h	Modulo para crear conjuntos de Timer	350
MRTHandler.cpp	Handler del timer MRT	352
MRTHandler.h	Handler del timer MRT	353
SCtimer.cpp	Modulo de Salida autonoma temporizada	356
SCtimer.h	Modulo de Salida autonoma temporizada	358
DAC.cpp	Objeto DAC	360
DAC.h	Objeto DAC	362
ADC.cpp	Funciones miembro de ADC	365
ADC.h	Modulo de ADC	367
ADCGroup.cpp	Funciones miembro de ADCGroup	369
ADCGroup.h	Modulo de ADCGroup	370
ComunicacionAsincronica.h	Objeto base para la creacion de comunicaciones asincrónicas	372
UART.cpp	Funciones miembro de Uart	374
UART.h	Modulo de Comunicacion Uart	375
ComunicacionSincronica.h	Objeto base para la creacion de comunicaciones sincrónicas	378
I2C.cpp	Funciones miembro de I2C	380
I2C.h	Modulo de comunicacion I2C	382
I2CMaster.cpp	Modulo para master I2C	385
I2CMaster.h	Modulo para master I2C	386
SPI.cpp	Funciones miembro de SPI	389
SPI.h	Modulo de comunicacion SPI	391
SPIMaster.cpp	Modulo de comunicacion SPIMaster	394
SPIMaster.h	Modulo de comunicacion SPIMaster	395

dr_pll.cpp	Descripcion del modulo	399
dr_pll.h	Descripcion del modulo	400
systick.cpp	Firmware del systick	402
systick.h	Firmware del systick	404
LPC845.h	Registros del LPC845	407
tipos.h	Definiciones de tipos de variables	432

Chapter 6

Topic Documentation

6.1 Misc

Classes

- class [CircularBuffer< T >](#)
Clase del objeto CircularBuffer.

6.1.1 Detailed Description

6.2 Drivers

Classes

- class [teclado](#)
Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.
- class [I4017](#)
Clase del objeto I4017 El objeto I4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a través de sus patas.
- class [I4511](#)
Clase del objeto I4511 El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.
- class [digito](#)
Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.
- class [display7Segmentos](#)
Clase del objeto display7Segmentos El objeto display7Segmentos permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor.
- struct [gruposdedigitos](#)
Estructura de grupo de dígitos.
- class [LCD](#)

Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

- class [L298N](#)

Clase del objeto L298N El objeto [L298N](#) realiza las acciones de control de dos motores controlados por el correspondiente periférico.

- class [Reloj](#)

Clase del objeto Reloj.

- class [Pwm](#)

Clase del objeto Pwm.

- class [PWM_Reader](#)

Clase del objeto PWM_Reader.

- class [HC_SR04](#)

Clase del objeto HC_SR04 El objeto [HC_SR04](#) Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

- class [Pin](#)

Clase del objeto Pin.

- class [gpio](#)

Clase del objeto gpio.

- class [PinInterrupt](#)

Clase del objeto Pin_interrupt El objeto [Pin_interrupt](#) debe ser heredado por cualquier objeto que desee tener interrupciones por pin.

- class [Input](#)

Clase del objeto Input.

- class [Output](#)

Clase del objeto outputs.

- class [Callback](#)

Clase del objeto Callback.

- class [Timer](#)

Clase del objeto timer.

- class [timers](#)

Clase del objeto timers El objeto [timers](#) permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.

- class [MRThandler](#)

Clase del objeto MRThandler El objeto [MRThandler](#) debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.

- class [SCTimer](#)

Clase del objeto SCTimer El objeto [SCTimer](#) debe ser heredado por quienes deseen utilizar las interrupciones o funcionalidades del [SCTimer](#).

- class [DAC](#)

Clase del objeto DAC FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

- class [ADC](#)

Clase del objeto ADC FUNCIONAMIENTO: La clase [ADC](#) utiliza el [ADCGroup](#) para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del [ADC](#) existen. La configuración de conversión se realiza automáticamente con el primer objeto [ADC](#) creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

- class [ADC_Group](#)

Clase del objeto ADC_Group FUNCIONAMIENTO: Solo debe crearse UN objeto [ADC](#). Todos los canales y distintos [ADC](#) son manejados por el mismo objeto. Realiza un barrido y guarda a todos los [ADC](#) en un vector de resultados.

- class [I2C](#)

Clase del objeto I2C El objeto [I2C](#) genera una comunicación sincrónica de tipo [I2C](#). Posee las funciones basicas como start, stop, write y read.

- class [I2CMaster](#)

Clase del objeto `I2CMaster` El objeto `I2CMaster` genera una comunicación tipo master de `I2C` utilizando buffers de recepción y transmisión con interrupciones.

- class `SPI`

Clase del objeto `SPI`.

- class `SPIMaster`

Clase del objeto `SPIMaster`.

- class `Uart`

Clase del objeto `uart` El objeto `uart` genera una comunicación asíncrona de tipo `UART`.

TypeDefs

- `typedef void(* Timer_Handler) (void)`

Variables

- `const uint8_t IOCON_INDEX_PIO0[] = { 17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,20,0,0,35 };`
- `const uint8_t IOCON_INDEX_PIO1[] = { 36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,0,0,50,51 };`
- `PinInterrupt * g_gpiohandler [MAX_PININTERRUPT]`
- `vector< Callback * > g_Handler`
- `std::vector< MRTHandler * > g_MRTHandler`

6.2.1 Detailed Description

6.2.2 Typedef Documentation

6.2.2.1 Timer_Handler

```
typedef void(* Timer_Handler) (void)
```

Tipo de dato: función a ejecutar al terminar el timer.

6.2.3 Variable Documentation

6.2.3.1 g_gpiohandler

```
PinInterrupt* g_gpiohandler[MAX_PININTERRUPT] [extern]
```

Vector de interrupciones del PIN INTERRUPT

6.2.3.2 g_Handler

```
vector<Callback*> g_Handler [extern]
```

Vector de interrupciones del systick

6.2.3.3 g_MRThandler

```
std::vector<MRThandler * > g_MRThandler [extern]
```

Vector de interrupciones del MRT [Timer](#)

6.2.3.4 IOCON_INDEX_PIO0

```
const uint8_t IOCON_INDEX_PIO0[] = { 17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,
```

Index for the IOCON Register Port0

6.2.3.5 IOCON_INDEX_PIO1

```
const uint8_t IOCON_INDEX_PIO1[] = { 36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,
```

Index for the IOCON Register Port1

6.3 Abstracta

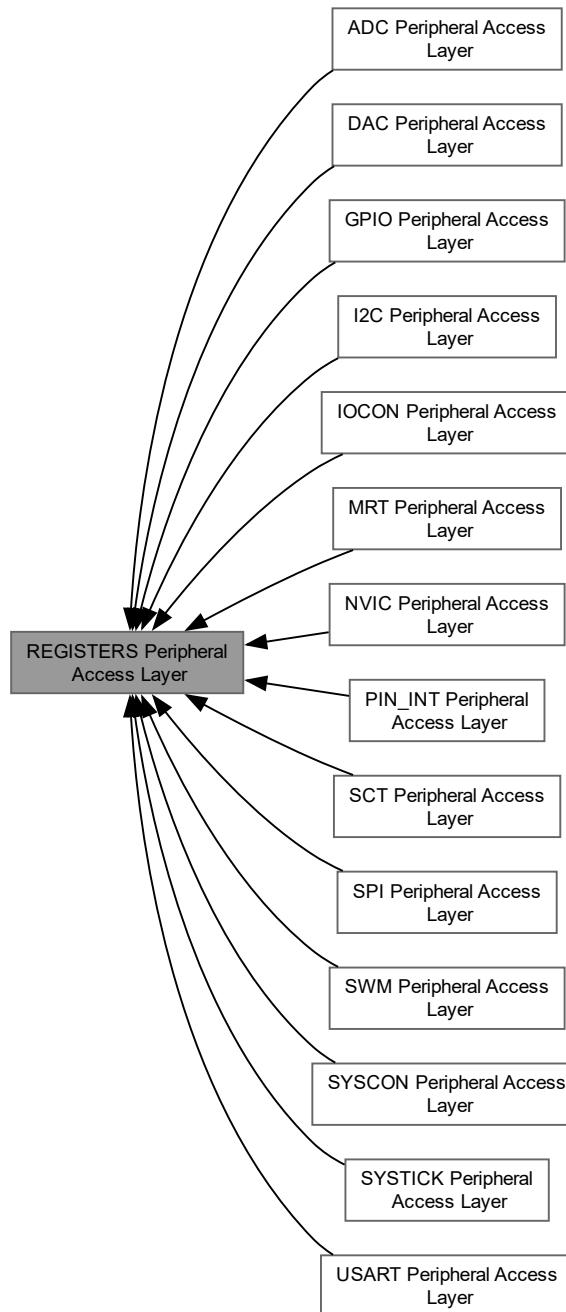
Clases

- class [barrido](#)
Clase del objeto barrido Clase abstracta pura para la generación de barridos.
- class [segmentos](#)
Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.
- class [Display](#)
Clase del objeto display Clase abstracta pura para la generación de displays.
- class [Puente_H](#)
Clase del objeto Puente_H El objeto Puente_H es la interfaz abstracta pura de cualquier puente H que se desee realizar.
- class [distancia](#)
Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.
- class [InOut](#)
Clase del objeto InOut.
- class [ComunicacionAsincronica](#)
Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de UART.
- class [ComunicacionSincronica](#)
Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI.

6.3.1 Detailed Description

6.4 REGISTERS Peripheral Access Layer

Collaboration diagram for REGISTERS Peripheral Access Layer:



Topics

- SYSCON Peripheral Access Layer

- GPIO Peripheral Access Layer
- IOCON Peripheral Access Layer
- SYSTICK Peripheral Access Layer
- MRT Peripheral Access Layer
- NVIC Peripheral Access Layer
- PIN_INT Peripheral Access Layer
- SCT Peripheral Access Layer
- USART Peripheral Access Layer
- SWM Peripheral Access Layer
- ADC Peripheral Access Layer
- DAC Peripheral Access Layer
- I2C Peripheral Access Layer
- SPI Peripheral Access Layer

Macros

- `#define FCLKIN (12000000UL)`
- `#define FREQ_PRINCIPAL 48000000UL`

6.4.1 Detailed Description

6.4.2 Macro Definition Documentation

6.4.2.1 FCLKIN

```
#define FCLKIN (12000000UL)
```

Frecuencia base, 12 MHz

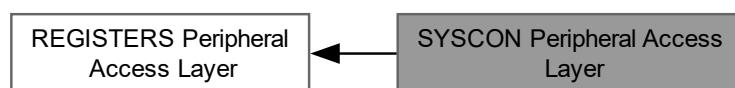
6.4.2.2 FREQ_PRINCIPAL

```
#define FREQ_PRINCIPAL 48000000UL
```

Frecuencia principal del LPC845

6.4.3 SYSCON Peripheral Access Layer

Collaboration diagram for SYSCON Peripheral Access Layer:



Classes

- struct [SYSCON_Type](#)
Structure type to access the SYSCON.

Macros

- #define [SYSCON_BASE](#) (0x40048000u)
- #define [SYSCON](#) (([SYSCON_Type](#) *)[SYSCON_BASE](#))
- #define [SYSCON_BASE_ADDRS](#) { [SYSCON_BASE](#) }
- #define [SYSCON_BASE_PTRS](#) { [SYSCON](#) }
- #define [SYSCON IRQS](#) { [BOD_IRQn](#) }

6.4.3.1 Detailed Description

6.4.3.2 Macro Definition Documentation

6.4.3.2.1 SYSCON

```
#define SYSCON ((SYSCON_Type *)SYSCON_BASE)
```

Peripheral SYSCON base pointer

6.4.3.2.2 SYSCON_BASE

```
#define SYSCON_BASE (0x40048000u)
```

Registers for the SYSCON SYSCON - Peripheral instance base addresses Peripheral SYSCON base address

6.4.3.2.3 SYSCON_BASE_ADDRS

```
#define SYSCON_BASE_ADDRS { SYSCON_BASE }
```

Array initializer of SYSCON peripheral base addresses

6.4.3.2.4 SYSCON_BASE_PTRS

```
#define SYSCON_BASE_PTRS { SYSCON }
```

Array initializer of SYSCON peripheral base pointers

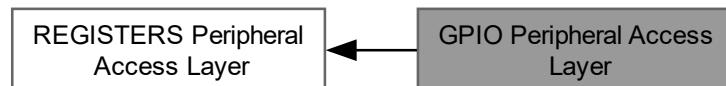
6.4.3.2.5 SYSCON_IRQS

```
#define SYSCON_IRQS { BOD_IRQn }
```

Interrupt vectors for the SYSCON peripheral type

6.4.4 GPIO Peripheral Access Layer

Collaboration diagram for GPIO Peripheral Access Layer:



Classes

- struct [GPIO_Type](#)

Structure type to access the General Purpose [Input Output](#) (GPIO).

Macros

- `#define GPIO ((GPIO_Type*) 0xA0000000UL)`

6.4.4.1 Detailed Description

6.4.4.2 Macro Definition Documentation

6.4.4.2.1 GPIO

```
#define GPIO ( (GPIO_Type*) 0xA0000000UL )
```

Registers for the GPIO

6.4.5 IOCON Peripheral Access Layer

Collaboration diagram for IOCON Peripheral Access Layer:



Classes

- struct [IOCON_Type](#)

Structure type to access the IOCON.

Macros

- `#define IOCON ((IOCON_Type*) 0x40044000UL)`

6.4.5.1 Detailed Description

6.4.5.2 Macro Definition Documentation

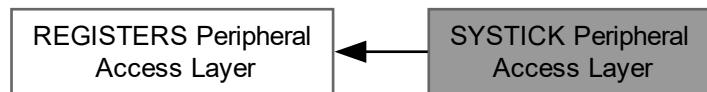
6.4.5.2.1 IOCON

```
#define IOCON ( (IOCON_Type*) 0x40044000UL )
```

Registers for the IOCON

6.4.6 SYSTICK Peripheral Access Layer

Collaboration diagram for SYSTICK Peripheral Access Layer:



Classes

- struct [SysTick_t](#)

Structure type to access the System [Timer](#) (SysTick).

Macros

- `#define SysTick ((SysTick_t *) 0xE000E010UL)`
- `#define FREQ_SYSTICK (1000)`

6.4.6.1 Detailed Description

6.4.6.2 Macro Definition Documentation

6.4.6.2.1 FREQ_SYSTICK

```
#define FREQ_SYSTICK (1000)
```

Frecuencia de empleo del Systick

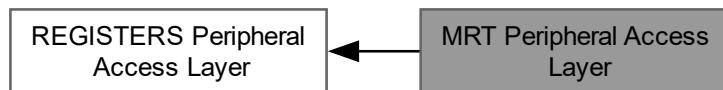
6.4.6.2.2 SysTick

```
#define SysTick ( (SysTick_t *) 0xE000E010UL)
```

Registers for the SYSTICK SysTick configuration struct

6.4.7 MRT Peripheral Access Layer

Collaboration diagram for MRT Peripheral Access Layer:



Classes

- struct [MRT_t](#)

Structure type to access the MRT timer.

Macros

- #define [MRT](#) ((MRT_t *) 0x40004000UL)
- #define [MRT0](#) ((MRT_t *) [MRT](#))
- #define [MRT1](#) ((MRT_t *) [MRT0](#) + 1)
- #define [MRT2](#) ((MRT_t *) [MRT1](#) + 1)
- #define [MRT3](#) ((MRT_t *) [MRT2](#) + 1)
- #define [MRT_IDLE_CH](#) ((__R uint32_t *) 0x400040F4UL)
- #define [MRT_IRQ_FLAG](#) ((__RW uint32_t *) 0x400040F8UL)

6.4.7.1 Detailed Description

6.4.7.2 Macro Definition Documentation

6.4.7.2.1 MRT

```
#define MRT ( (MRT_t *) 0x40004000UL)
```

Registers for the MRTTIMER

6.4.7.2.2 MRT0

```
#define MRT0 ( (MRT_t *) MRT )
```

Registers for the MRTTIMER0

6.4.7.2.3 MRT1

```
#define MRT1 ( (MRT_t *) MRT0 + 1 )
```

Registers for the MRTTIMER1

6.4.7.2.4 MRT2

```
#define MRT2 ( (MRT_t *) MRT1 + 1 )
```

Registers for the MRTTIMER2

6.4.7.2.5 MRT3

```
#define MRT3 ( (MRT_t *) MRT2 + 1 )
```

Registers for the MRTTIMER3

6.4.7.2.6 MRT_IDLE_CH

```
#define MRT_IDLE_CH ( (__R uint32_t *) 0x400040F4UL )
```

MRT CHANNEL IDLE

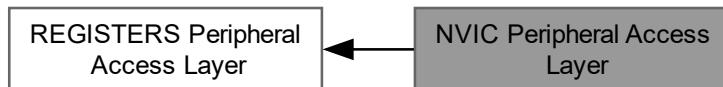
6.4.7.2.7 MRT_IRQ_FLAG

```
#define MRT_IRQ_FLAG ( (__RW uint32_t *) 0x400040F8UL )
```

MRT IRQ FLAGS REGISTER

6.4.8 NVIC Peripheral Access Layer

Collaboration diagram for NVIC Peripheral Access Layer:



Classes

- struct [NVIC_Type](#)
Structure type to access the Nested Vectored Interrupt Controller (NVIC).

Macros

- #define [SCS_BASE](#) (0xE000E000UL)
- #define [SysTick_BASE](#) ([SCS_BASE](#) + 0x0010UL)
- #define [NVIC_BASE](#) ([SCS_BASE](#) + 0x0100UL)
- #define [SCB_BASE](#) ([SCS_BASE](#) + 0x0D00UL)
- #define [NVIC](#) (([NVIC_Type](#) *) [NVIC_BASE](#))

6.4.8.1 Detailed Description

6.4.8.2 Macro Definition Documentation

6.4.8.2.1 NVIC

```
#define NVIC ((NVIC_Type *) NVIC_BASE )
```

NVIC configuration struct

6.4.8.2.2 NVIC_BASE

```
#define NVIC_BASE (SCS_BASE + 0x0100UL)
```

NVIC Base Address

6.4.8.2.3 SCB_BASE

```
#define SCB_BASE (SCS_BASE + 0x0D00UL)
```

System Control Block Base Address

6.4.8.2.4 SCS_BASE

```
#define SCS_BASE (0xE000E000UL)
```

Registers for the NVIC System Control Space Base Address

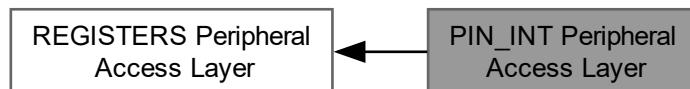
6.4.8.2.5 SysTick_BASE

```
#define SysTick_BASE (SCS_BASE + 0x0010UL)
```

SysTick Base Address

6.4.9 PIN_INT Peripheral Access Layer

Collaboration diagram for PIN_INT Peripheral Access Layer:



Classes

- struct [PIN_INTERRUPT_t](#)
Structure type to access the Pin Interrupt.

Macros

- #define [PIN_INTERRUPT](#) (([PIN_INTERRUPT_t](#) *) 0xA0004000UL)

6.4.9.1 Detailed Description

6.4.9.2 Macro Definition Documentation

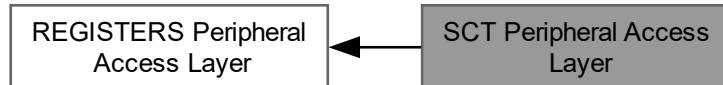
6.4.9.2.1 PIN_INTERRUPT

```
#define PIN_INTERRUPT ( (PIN\_INTERRUPT\_t *) 0xA0004000UL )
```

Registers for the PIN_INTERRUPT

6.4.10 SCT Peripheral Access Layer

Collaboration diagram for SCT Peripheral Access Layer:



Classes

- struct [SCT_t](#)
Structure type to access the SCT.

Macros

- #define [SCT](#) (([SCT_t](#) *) 0x50004000UL)

6.4.10.1 Detailed Description

6.4.10.2 Macro Definition Documentation

6.4.10.2.1 SCT

```
#define SCT ( (SCT\_t *) 0x50004000UL )
```

Registers for the SCTTIMER Register for the SCT0

6.4.11 USART Peripheral Access Layer

Collaboration diagram for USART Peripheral Access Layer:



Classes

- struct **USART_Type**

Structure type to access the Universal Serial Asincronical Reciver Transmitter (USART).

Macros

- #define USART0_BASE (0x40064000u)
- #define USART0 ((USART_Type *)USART0_BASE)
- #define USART1_BASE (0x40068000u)
- #define USART1 ((USART_Type *)USART1_BASE)
- #define USART2_BASE (0x4006C000u)
- #define USART2 ((USART_Type *)USART2_BASE)
- #define USART3_BASE (0x40070000u)
- #define USART3 ((USART_Type *)USART3_BASE)
- #define USART4_BASE (0x40074000u)
- #define USART4 ((USART_Type *)USART4_BASE)
- #define USART_BASE_ADDRS { USART0_BASE, USART1_BASE, USART2_BASE, USART3_BASE, USART4_BASE }
- #define USART_BASE_PTRS { USART0, USART1, USART2, USART3, USART4 }
- #define USART IRQS { USART0_IRQHandler, USART1_IRQHandler, USART2_IRQHandler, PIN_INT6_USART3_IRQHandler, PININT7_USART4_IRQHandler }

6.4.11.1 Detailed Description

6.4.11.2 Macro Definition Documentation

6.4.11.2.1 USART0

```
#define USART0 ((USART_Type *)USART0_BASE)
```

Peripheral USART0 base pointer

6.4.11.2.2 USART0_BASE

```
#define USART0_BASE (0x40064000u)
```

Registers for the USART Peripheral USART0 base address

6.4.11.2.3 USART1

```
#define USART1 ((USART_Type *)USART1_BASE)
```

Peripheral USART1 base pointer

6.4.11.2.4 USART1_BASE

```
#define USART1_BASE (0x40068000u)
```

Peripheral USART1 base address

6.4.11.2.5 USART2

```
#define USART2 ((USART_Type *)USART2_BASE)
```

Peripheral USART2 base pointer

6.4.11.2.6 USART2_BASE

```
#define USART2_BASE (0x4006C000u)
```

Peripheral USART2 base address

6.4.11.2.7 USART3

```
#define USART3 ((USART_Type *)USART3_BASE)
```

Peripheral USART3 base pointer

6.4.11.2.8 USART3_BASE

```
#define USART3_BASE (0x40070000u)
```

Peripheral USART3 base address

6.4.11.2.9 USART4

```
#define USART4 ((USART_Type *)USART4_BASE)
```

Peripheral USART4 base pointer

6.4.11.2.10 USART4_BASE

```
#define USART4_BASE (0x40074000u)
```

Peripheral USART4 base address

6.4.11.2.11 USART_BASE_ADDRS

```
#define USART_BASE_ADDRS { USART0_BASE, USART1_BASE, USART2_BASE, USART3_BASE, USART4_BASE }
```

Array initializer of USART peripheral base addresses

6.4.11.2.12 USART_BASE_PTRS

```
#define USART_BASE_PTRS { USART0, USART1, USART2, USART3, USART4 }
```

Array initializer of USART peripheral base pointers

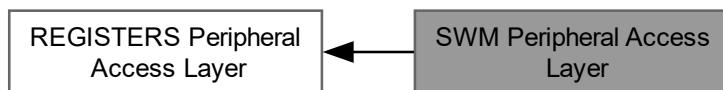
6.4.11.2.13 USART IRQS

```
#define USART_IRQS { USART0_IRQn, USART1_IRQn, USART2_IRQn, PIN_INT6_USART3_IRQn, PIN_INT7_←
USART4_IRQn }
```

Interrupt vectors for the USART peripheral type

6.4.12 SWM Peripheral Access Layer

Collaboration diagram for SWM Peripheral Access Layer:



Classes

- struct [SWM_t](#)
Structure type to access the Switch Matrix (SWM).

Macros

- #define [SWM](#) ([SWM_t](#) *) 0x4000C000UL)

6.4.12.1 Detailed Description

6.4.12.2 Macro Definition Documentation

6.4.12.2.1 SWM

```
#define SWM ( SWM\_t * ) 0x4000C000UL )
```

Register for the SWM Register for the SWM0

6.4.13 ADC Peripheral Access Layer

Collaboration diagram for ADC Peripheral Access Layer:



Topics

- ADC Register Masks

Classes

- struct `ADC_Type`

Structure type to access the Analog Digital Convertos (`ADC`).

Macros

- `#define ADC0_BASE (0x4001C000u)`
- `#define ADC0 ((ADC_Type *)ADC0_BASE)`
- `#define ADC_BASE_ADDRS { ADC0_BASE }`
- `#define ADC_BASE_PTRS { ADC0 }`
- `#define ADC_SEQ IRQS { ADC0_SEQA_IRQn, ADC0_SEQB_IRQn }`

6.4.13.1 Detailed Description

6.4.13.2 Macro Definition Documentation

6.4.13.2.1 ADC0

```
#define ADC0 ((ADC_Type *)ADC0_BASE)
```

Peripheral ADC0 base pointer

6.4.13.2.2 ADC0_BASE

```
#define ADC0_BASE (0x4001C000u)
```

Peripheral ADC0 base address

6.4.13.2.3 ADC_BASE_ADDRS

```
#define ADC_BASE_ADDRS { ADC0_BASE }
```

Array initializer of `ADC` peripheral base addresses

6.4.13.2.4 ADC_BASE_PTRS

```
#define ADC_BASE_PTRS { ADC0 }
```

Array initializer of `ADC` peripheral base pointers

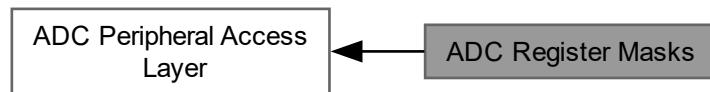
6.4.13.2.5 ADC_SEQ_IRQS

```
#define ADC_SEQ IRQS { ADC0_SEQA_IRQn, ADC0_SEQB_IRQn }
```

Interrupt vectors for the [ADC](#) peripheral type

6.4.13.3 ADC Register Masks

Collaboration diagram for ADC Register Masks:



Macros

- #define [ADC_SEQ_CTRL_COUNT](#) (2U)
- #define [ADC_SEQ_GDAT_COUNT](#) (2U)
- #define [ADC_DAT_COUNT](#) (12U)

CTRL - ADC Control register. Contains the clock divide value, resolution selection, sampling time selection, and mode controls.

- #define [ADC_CTRL_CLKDIV](#)(x)
- #define [ADC_CTRL_ASYNMODE](#)(x)
- #define [ADC_CTRL_LPWRMODE](#)(x)
- #define [ADC_CTRL_CALMODE](#)(x)

SEQ_CTRL - ADC Conversion Sequence-n control register: Controls triggering and channel selection for conversion sequence-n. Also specifies interrupt mode for sequence-n.

- #define [ADC_SEQ_CTRL_CHANNELS](#)(x)
- #define [ADC_SEQ_CTRL_TRIGGER](#)(x)
- #define [ADC_SEQ_CTRL_TRIGPOL](#)(x)
- #define [ADC_SEQ_CTRL_SYNCBYPASS](#)(x)
- #define [ADC_SEQ_CTRL_START](#)(x)
- #define [ADC_SEQ_CTRL_BURST](#)(x)
- #define [ADC_SEQ_CTRL_SINGLESTEP](#)(x)
- #define [ADC_SEQ_CTRL_LOWPRIOR](#)(x)
- #define [ADC_SEQ_CTRL_MODE](#)(x)
- #define [ADC_SEQ_CTRL_SEQ_ENA](#)(x)

SEQ_GDAT - ADC Sequence-n Global Data register. This register contains the result of the most recent ADC conversion performed under sequence-n.

- #define ADC_SEQ_GDAT_RESULT(x)
- #define ADC_SEQ_GDAT_THCMPRANGE(x)
- #define ADC_SEQ_GDAT_THCMPCROSS(x)
- #define ADC_SEQ_GDAT_CHN(x)
- #define ADC_SEQ_GDAT_OVERRUN(x)
- #define ADC_SEQ_GDAT_DATAVALID(x)

DAT - ADC Channel N Data register. This register contains the result of the most recent conversion completed on channel N.

- #define ADC_DAT_RESULT(x)
- #define ADC_DAT_THCMPRANGE(x)
- #define ADC_DAT_THCMPCROSS(x)
- #define ADC_DAT_CHANNEL(x)
- #define ADC_DAT_OVERRUN(x)
- #define ADC_DAT_DATAVALID(x)

THR0_LOW - ADC Low Compare Threshold register 0: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0.

- #define ADC_THR0_LOW_THRLOW(x)

THR1_LOW - ADC Low Compare Threshold register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1.

- #define ADC_THR1_LOW_THRLOW(x)

THR0_HIGH - ADC High Compare Threshold register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0.

- #define ADC_THR0_HIGH_THRHIGH(x)

THR1_HIGH - ADC High Compare Threshold register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1.

- #define ADC_THR1_HIGH_THRHIGH(x)

CHAN_THRSEL - ADC Channel-Threshold Select register. Specifies which set of threshold compare registers are to be used for each channel

- #define ADC_CHAN_THRSEL_CH0_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH1_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH2_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH3_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH4_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH5_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH6_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH7_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH8_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH9_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH10_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH11_THRSEL(x)

INTEN - ADC Interrupt Enable register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated.

- #define ADC_INTEN_SEQA_INTEN(x)
- #define ADC_INTEN_SEQB_INTEN(x)
- #define ADC_INTEN_OVR_INTEN(x)
- #define ADC_INTEN_ADCMPINTEN0(x)
- #define ADC_INTEN_ADCMPINTEN1(x)
- #define ADC_INTEN_ADCMPINTEN2(x)
- #define ADC_INTEN_ADCMPINTEN3(x)
- #define ADC_INTEN_ADCMPINTEN4(x)
- #define ADC_INTEN_ADCMPINTEN5(x)
- #define ADC_INTEN_ADCMPINTEN6(x)
- #define ADC_INTEN_ADCMPINTEN7(x)
- #define ADC_INTEN_ADCMPINTEN8(x)
- #define ADC_INTEN_ADCMPINTEN9(x)
- #define ADC_INTEN_ADCMPINTEN10(x)
- #define ADC_INTEN_ADCMPINTEN11(x)

FLAGS - ADC Flags register. Contains the four interrupt/DMA trigger flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers).

- #define ADC_FLAGS_THCMP0(x)
- #define ADC_FLAGS_THCMP1(x)
- #define ADC_FLAGS_THCMP2(x)
- #define ADC_FLAGS_THCMP3(x)
- #define ADC_FLAGS_THCMP4(x)
- #define ADC_FLAGS_THCMP5(x)
- #define ADC_FLAGS_THCMP6(x)
- #define ADC_FLAGS_THCMP7(x)
- #define ADC_FLAGS_THCMP8(x)
- #define ADC_FLAGS_THCMP9(x)
- #define ADC_FLAGS_THCMP10(x)
- #define ADC_FLAGS_THCMP11(x)
- #define ADC_FLAGS_OVERRUN0(x)
- #define ADC_FLAGS_OVERRUN1(x)
- #define ADC_FLAGS_OVERRUN2(x)
- #define ADC_FLAGS_OVERRUN3(x)
- #define ADC_FLAGS_OVERRUN4(x)
- #define ADC_FLAGS_OVERRUN5(x)
- #define ADC_FLAGS_OVERRUN6(x)
- #define ADC_FLAGS_OVERRUN7(x)
- #define ADC_FLAGS_OVERRUN8(x)
- #define ADC_FLAGS_OVERRUN9(x)
- #define ADC_FLAGS_OVERRUN10(x)
- #define ADC_FLAGS_OVERRUN11(x)
- #define ADC_FLAGS_SEQA_OVR(x)
- #define ADC_FLAGS_SEQB_OVR(x)
- #define ADC_FLAGS_SEQA_INT(x)
- #define ADC_FLAGS_SEQB_INT(x)
- #define ADC_FLAGS_THCMP_INT(x)
- #define ADC_FLAGS_OVR_INT(x)

TRM - ADC Startup register.

- #define ADC_TRM_VRANGE(x)

6.4.13.3.1 Detailed Description

6.4.13.3.2 Macro Definition Documentation

6.4.13.3.2.1 ADC_CHAN_THRSEL_CH0_THRSEL

```
#define ADC_CHAN_THRSEL_CH0_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CHAN_THRSEL_CH0_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH0_THRSEL_MASK)
```

CH0_THRSEL - Threshold select for channel 0. 0b0..Threshold 0. Results for this channel will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers. 0b1..Threshold 1. Results for this channel will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers.

6.4.13.3.2.2 ADC_CHAN_THRSEL_CH10_THRSEL

```
#define ADC_CHAN_THRSEL_CH10_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CHAN_THRSEL_CH10_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH10_THRSEL_MASK)
```

CH10_THRSEL - Threshold select for channel 10. See description for channel 0.

6.4.13.3.2.3 ADC_CHAN_THRSEL_CH11_THRSEL

```
#define ADC_CHAN_THRSEL_CH11_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CHAN_THRSEL_CH11_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH11_THRSEL_MASK)
```

CH11_THRSEL - Threshold select for channel 11. See description for channel 0.

6.4.13.3.2.4 ADC_CHAN_THRSEL_CH1_THRSEL

```
#define ADC_CHAN_THRSEL_CH1_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CHAN_THRSEL_CH1_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH1_THRSEL_MASK)
```

CH1_THRSEL - Threshold select for channel 1. See description for channel 0.

6.4.13.3.2.5 ADC_CHAN_THRSEL_CH2_THRSEL

```
#define ADC_CHAN_THRSEL_CH2_THRSEL( x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH2_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH2_THRSEL_MASK)
```

CH2_THRSEL - Threshold select for channel 2. See description for channel 0.

6.4.13.3.2.6 ADC_CHAN_THRSEL_CH3_THRSEL

```
#define ADC_CHAN_THRSEL_CH3_THRSEL( x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH3_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH3_THRSEL_MASK)
```

CH3_THRSEL - Threshold select for channel 3. See description for channel 0.

6.4.13.3.2.7 ADC_CHAN_THRSEL_CH4_THRSEL

```
#define ADC_CHAN_THRSEL_CH4_THRSEL( x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH4_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH4_THRSEL_MASK)
```

CH4_THRSEL - Threshold select for channel 4. See description for channel 0.

6.4.13.3.2.8 ADC_CHAN_THRSEL_CH5_THRSEL

```
#define ADC_CHAN_THRSEL_CH5_THRSEL( x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH5_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH5_THRSEL_MASK)
```

CH5_THRSEL - Threshold select for channel 5. See description for channel 0.

6.4.13.3.2.9 ADC_CHAN_THRSEL_CH6_THRSEL

```
#define ADC_CHAN_THRSEL_CH6_THRSEL( x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH6_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH6_THRSEL_MASK)
```

CH6_THRSEL - Threshold select for channel 6. See description for channel 0.

6.4.13.3.2.10 ADC_CHAN_THRSEL_CH7_THRSEL

```
#define ADC_CHAN_THRSEL_CH7_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH7_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH7_THRSEL_MASK)
```

CH7_THRSEL - Threshold select for channel 7. See description for channel 0.

6.4.13.3.2.11 ADC_CHAN_THRSEL_CH8_THRSEL

```
#define ADC_CHAN_THRSEL_CH8_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH8_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH8_THRSEL_MASK)
```

CH8_THRSEL - Threshold select for channel 8. See description for channel 0.

6.4.13.3.2.12 ADC_CHAN_THRSEL_CH9_THRSEL

```
#define ADC_CHAN_THRSEL_CH9_THRSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CHAN_THRSEL_CH9_THRSEL_SHIFT) & ADC_CHAN_THRSEL_CH9_THRSEL_MASK)
```

CH9_THRSEL - Threshold select for channel 9. See description for channel 0.

6.4.13.3.2.13 ADC_CTRL_ASYNMODE

```
#define ADC_CTRL_ASYNMODE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_CTRL_ASYNMODE_SHIFT) & ADC_CTRL_ASYNMODE_MASK)
```

ASYNMODE - Select clock mode. 0b0..Synchronous mode. The [ADC](#) clock is derived from the system clock based on the divide value selected in the CLKDIV field. The [ADC](#) clock will be started in a controlled fashion in response to a trigger to eliminate any uncertainty in the launching of an [ADC](#) conversion in response to any synchronous (on-chip) trigger. In Synchronous mode with the SYNCBYPASS bit (in a sequence control register) set, sampling of the [ADC](#) input and start of conversion will initiate 2 system clocks after the leading edge of a (synchronous) trigger pulse. 0b1..Asynchronous mode. The [ADC](#) clock is based on the output of the [ADC](#) clock divider ADCCLKSEL in the SYSCON block.

6.4.13.3.2.14 ADC_CTRL_CALMODE

```
#define ADC_CTRL_CALMODE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CTRL_CALMODE_SHIFT)) & ADC_CTRL_CALMODE_MASK)
```

CALMODE - Writing a '1' to this bit will initiate a self-calibration cycle. This bit will be automatically cleared by hardware after the calibration cycle is complete. Note: Other bits of this register may be written to concurrently with setting this bit, however once this bit has been set no further writes to this register are permitted until the full calibration cycle has ended.

6.4.13.3.2.15 ADC_CTRL_CLKDIV

```
#define ADC_CTRL_CLKDIV(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CTRL_CLKDIV_SHIFT)) & ADC_CTRL_CLKDIV_MASK)
```

CLKDIV - In synchronous mode only, the system clock is divided by this value plus one to produce the clock for the [ADC](#) converter, which should be less than or equal to 72 MHz. Typically, software should program the smallest value in this field that yields this maximum clock rate or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable. This field is ignored in the asynchronous operating mode.

6.4.13.3.2.16 ADC_CTRL_LPWRMODE

```
#define ADC_CTRL_LPWRMODE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_CTRL_LPWRMODE_SHIFT)) & ADC_CTRL_LPWRMODE_MASK)
```

LPWRMODE - The low-power [ADC](#) mode 0b0..The low-power [ADC](#) mode is disabled. The analog circuitry remains activated even when no conversions are requested. 0b1..The low-power [ADC](#) mode is enabled. The analog circuitry is automatically powered-down when no conversions are taking place. When any (hardware or software) triggering event is detected, the analog circuitry is enabled. After the required start-up time, the requested conversion will be launched. Once the conversion completes, the analog-circuitry will again be powered-down provided no further conversions are pending. Using this mode can save an appreciable amount of current (approximately 2.5 mA) when conversions are required relatively infrequently. The penalty for using this mode is an approximately FIFTEEN [ADC](#) CLOCK delay (30 clocks in 10-bit mode), based on the frequency specified in the CLKDIV field, from the time the trigger event occurs until sampling of the A/D input commences. Note: This mode will NOT power-up the A/D if the ADC_ENA bit is low.

6.4.13.3.2.17 ADC_DAT_CHANNEL

```
#define ADC_DAT_CHANNEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_DAT_CHANNEL_SHIFT)) & ADC_DAT_CHANNEL_MASK)
```

CHANNEL - This field is hard-coded to contain the channel number that this particular register relates to (i.e. this field will contain 0b0000 for the DAT0 register, 0b0001 for the DAT1 register, etc)

6.4.13.3.2.18 ADC_DAT_COUNT

```
#define ADC_DAT_COUNT (12U)
```

The count of ADC_DAT

6.4.13.3.2.19 ADC_DAT_DATAVALID

```
#define ADC_DAT_DATAVALID(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_DAT_DATAVALID_SHIFT)) & ADC_DAT_DATAVALID_MASK)
```

DATAVALID - This bit is set to 1 when an **ADC** conversion on this channel completes. This bit is cleared whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.

6.4.13.3.2.20 ADC_DAT_OVERRUN

```
#define ADC_DAT_OVERRUN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_DAT_OVERRUN_SHIFT)) & ADC_DAT_OVERRUN_MASK)
```

OVERRUN - This bit will be set to a 1 if a new conversion on this channel completes and overwrites the previous contents of the RESULT field before it has been read - i.e. while the DONE bit is set. This bit is cleared, along with the DONE bit, whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. This bit (in any of the 12 registers) will cause an overrun interrupt/DMA trigger to be asserted if the overrun interrupt is enabled. While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.

6.4.13.3.2.21 ADC_DAT_RESULT

```
#define ADC_DAT_RESULT(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_DAT_RESULT_SHIFT)) & ADC_DAT_RESULT_MASK)
```

RESULT - This field contains the 12-bit **ADC** conversion result from the last conversion performed on this channel. This will be a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP.

6.4.13.3.2.22 ADC_DAT_THCMPCROSS

```
#define ADC_DAT_THCMPCROSS (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_DAT_THCMPCROSS_SHIFT) & ADC_DAT_THCMPCROSS_MASK)
```

THCMPCROSS - Threshold Crossing Comparison result. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 0x2 = Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 0x3 = Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred

- i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.

6.4.13.3.2.23 ADC_DAT_THCMPRANGE

```
#define ADC_DAT_THCMPRANGE (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_DAT_THCMPRANGE_SHIFT) & ADC_DAT_THCMPRANGE_MASK)
```

THCMPRANGE - Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.

6.4.13.3.2.24 ADC_FLAGS_OVERRUN0

```
#define ADC_FLAGS_OVERRUN0 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN0_SHIFT) & ADC_FLAGS_OVERRUN0_MASK)
```

OVERRUN0 - Mirrors the OVERRUN status flag from the result register for [ADC](#) channel 0

6.4.13.3.2.25 ADC_FLAGS_OVERRUN1

```
#define ADC_FLAGS_OVERRUN1 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN1_SHIFT) & ADC_FLAGS_OVERRUN1_MASK)
```

OVERRUN1 - Mirrors the OVERRUN status flag from the result register for [ADC](#) channel 1

6.4.13.3.2.26 ADC_FLAGS_OVERRUN10

```
#define ADC_FLAGS_OVERRUN10 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_FLAGS_OVERRUN10_SHIFT) & ADC_FLAGS_OVERRUN10_MASK)
```

OVERRUN10 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 10

6.4.13.3.2.27 ADC_FLAGS_OVERRUN11

```
#define ADC_FLAGS_OVERRUN11 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_FLAGS_OVERRUN11_SHIFT) & ADC_FLAGS_OVERRUN11_MASK)
```

OVERRUN11 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 11

6.4.13.3.2.28 ADC_FLAGS_OVERRUN2

```
#define ADC_FLAGS_OVERRUN2 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_FLAGS_OVERRUN2_SHIFT) & ADC_FLAGS_OVERRUN2_MASK)
```

OVERRUN2 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 2

6.4.13.3.2.29 ADC_FLAGS_OVERRUN3

```
#define ADC_FLAGS_OVERRUN3 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_FLAGS_OVERRUN3_SHIFT) & ADC_FLAGS_OVERRUN3_MASK)
```

OVERRUN3 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 3

6.4.13.3.2.30 ADC_FLAGS_OVERRUN4

```
#define ADC_FLAGS_OVERRUN4 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_FLAGS_OVERRUN4_SHIFT) & ADC_FLAGS_OVERRUN4_MASK)
```

OVERRUN4 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 4

6.4.13.3.2.31 ADC_FLAGS_OVERRUN5

```
#define ADC_FLAGS_OVERRUN5 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN5_SHIFT)) & ADC_FLAGS_OVERRUN5_MASK)
```

OVERRUN5 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 5

6.4.13.3.2.32 ADC_FLAGS_OVERRUN6

```
#define ADC_FLAGS_OVERRUN6 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN6_SHIFT)) & ADC_FLAGS_OVERRUN6_MASK)
```

OVERRUN6 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 6

6.4.13.3.2.33 ADC_FLAGS_OVERRUN7

```
#define ADC_FLAGS_OVERRUN7 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN7_SHIFT)) & ADC_FLAGS_OVERRUN7_MASK)
```

OVERRUN7 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 7

6.4.13.3.2.34 ADC_FLAGS_OVERRUN8

```
#define ADC_FLAGS_OVERRUN8 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN8_SHIFT)) & ADC_FLAGS_OVERRUN8_MASK)
```

OVERRUN8 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 8

6.4.13.3.2.35 ADC_FLAGS_OVERRUN9

```
#define ADC_FLAGS_OVERRUN9 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_OVERRUN9_SHIFT)) & ADC_FLAGS_OVERRUN9_MASK)
```

OVERRUN9 - Mirrors the OVERRRUN status flag from the result register for [ADC](#) channel 9

6.4.13.3.2.36 ADC_FLAGS_OVR_INT

```
#define ADC_FLAGS_OVR_INT (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << ADC_FLAGS_OVR_INT_SHIFT)) & ADC_FLAGS_OVR_INT_MASK)
```

OVR_INT - Overrun Interrupt flag. Any overrun bit in any of the individual channel data registers will cause this interrupt. In addition, if the MODE bit in either of the SEQn_CTRL registers is 0 then the OVERRUN bit in the corresponding SEQn_GDAT register will also cause this interrupt. This interrupt must be enabled in the INTEN register. This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.

6.4.13.3.2.37 ADC_FLAGS_SEQA_INT

```
#define ADC_FLAGS_SEQA_INT (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << ADC_FLAGS_SEQA_INT_SHIFT)) & ADC_FLAGS_SEQA_INT_MASK)
```

SEQA_INT - Sequence A interrupt/DMA trigger. If the MODE bit in the SEQA_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQA_GDAT), which is set at the end of every ADC conversion performed as part of sequence A. It will be cleared automatically when the SEQA_GDAT register is read. If the MODE bit in the SEQA_CTRL register is 1, this flag will be set upon completion of an entire A sequence. In this case it must be cleared by writing a 1 to this SEQA_INT bit. This interrupt must be enabled in the INTEN register.

6.4.13.3.2.38 ADC_FLAGS_SEQA_OVR

```
#define ADC_FLAGS_SEQA_OVR (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << ADC_FLAGS_SEQA_OVR_SHIFT)) & ADC_FLAGS_SEQA_OVR_MASK)
```

SEQA_OVR - Mirrors the global OVERRUN status flag in the SEQA_GDAT register

6.4.13.3.2.39 ADC_FLAGS_SEQB_INT

```
#define ADC_FLAGS_SEQB_INT (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << ADC_FLAGS_SEQB_INT_SHIFT)) & ADC_FLAGS_SEQB_INT_MASK)
```

SEQB_INT - Sequence B interrupt/DMA trigger. If the MODE bit in the SEQB_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence B global data register (SEQB_GDAT), which is set at the end of every ADC conversion performed as part of sequence B. It will be cleared automatically when the SEQB_GDAT register is read. If the MODE bit in the SEQB_CTRL register is 1, this flag will be set upon completion of an entire B sequence. In this case it must be cleared by writing a 1 to this SEQB_INT bit. This interrupt must be enabled in the INTEN register.

6.4.13.3.2.40 ADC_FLAGS_SEQB_OVR

```
#define ADC_FLAGS_SEQB_OVR (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_SEQB_OVR_SHIFT)) & ADC_FLAGS_SEQB_OVR_MASK)
```

SEQB_OVR - Mirrors the global OVERRUN status flag in the SEQB_GDAT register

6.4.13.3.2.41 ADC_FLAGS_THCMP0

```
#define ADC_FLAGS_THCMP0 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP0_SHIFT)) & ADC_FLAGS_THCMP0_MASK)
```

THCMP0 - Threshold comparison event on Channel 0. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.

6.4.13.3.2.42 ADC_FLAGS_THCMP1

```
#define ADC_FLAGS_THCMP1 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP1_SHIFT)) & ADC_FLAGS_THCMP1_MASK)
```

THCMP1 - Threshold comparison event on Channel 1. See description for channel 0.

6.4.13.3.2.43 ADC_FLAGS_THCMP10

```
#define ADC_FLAGS_THCMP10 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP10_SHIFT)) & ADC_FLAGS_THCMP10_MASK)
```

THCMP10 - Threshold comparison event on Channel 10. See description for channel 0.

6.4.13.3.2.44 ADC_FLAGS_THCMP11

```
#define ADC_FLAGS_THCMP11 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP11_SHIFT)) & ADC_FLAGS_THCMP11_MASK)
```

THCMP11 - Threshold comparison event on Channel 11. See description for channel 0.

6.4.13.3.2.45 ADC_FLAGS_THCMP2

```
#define ADC_FLAGS_THCMP2 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP2_SHIFT)) & ADC_FLAGS_THCMP2_MASK)
```

THCMP2 - Threshold comparison event on Channel 2. See description for channel 0.

6.4.13.3.2.46 ADC_FLAGS_THCMP3

```
#define ADC_FLAGS_THCMP3 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP3_SHIFT)) & ADC_FLAGS_THCMP3_MASK)
```

THCMP3 - Threshold comparison event on Channel 3. See description for channel 0.

6.4.13.3.2.47 ADC_FLAGS_THCMP4

```
#define ADC_FLAGS_THCMP4 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP4_SHIFT)) & ADC_FLAGS_THCMP4_MASK)
```

THCMP4 - Threshold comparison event on Channel 4. See description for channel 0.

6.4.13.3.2.48 ADC_FLAGS_THCMP5

```
#define ADC_FLAGS_THCMP5 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP5_SHIFT)) & ADC_FLAGS_THCMP5_MASK)
```

THCMP5 - Threshold comparison event on Channel 5. See description for channel 0.

6.4.13.3.2.49 ADC_FLAGS_THCMP6

```
#define ADC_FLAGS_THCMP6 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP6_SHIFT)) & ADC_FLAGS_THCMP6_MASK)
```

THCMP6 - Threshold comparison event on Channel 6. See description for channel 0.

6.4.13.3.2.50 ADC_FLAGS_THCMP7

```
#define ADC_FLAGS_THCMP7 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP7_SHIFT)) & ADC_FLAGS_THCMP7_MASK)
```

THCMP7 - Threshold comparison event on Channel 7. See description for channel 0.

6.4.13.3.2.51 ADC_FLAGS_THCMP8

```
#define ADC_FLAGS_THCMP8 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP8_SHIFT)) & ADC_FLAGS_THCMP8_MASK)
```

THCMP8 - Threshold comparison event on Channel 8. See description for channel 0.

6.4.13.3.2.52 ADC_FLAGS_THCMP9

```
#define ADC_FLAGS_THCMP9 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP9_SHIFT)) & ADC_FLAGS_THCMP9_MASK)
```

THCMP9 - Threshold comparison event on Channel 9. See description for channel 0.

6.4.13.3.2.53 ADC_FLAGS_THCMP_INT

```
#define ADC_FLAGS_THCMP_INT (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_FLAGS_THCMP_INT_SHIFT)) & ADC_FLAGS_THCMP_INT_MASK)
```

THCMP_INT - Threshold Comparison Interrupt. This bit will be set if any of the THCMP flags in the lower bits of this register are set to 1 (due to an enabled out-of-range or threshold-crossing event on any channel). Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt. This bit will be cleared when all of the individual threshold flags are cleared via writing 1s to those bits.

6.4.13.3.2.54 ADC_INTEN_ADCMPINTENO

```
#define ADC_INTEN_ADCMPINTENO (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTENO_SHIFT)) & ADC_INTEN_ADCMPINTENO_MASK)
```

ADCMPINTENO - Threshold comparison interrupt enable for channel 0. 0b00..Disabled. 0b01..Outside threshold. 0b10..Crossing threshold. 0b11..Reserved

6.4.13.3.2.55 ADC_INTEN_ADCMPINTEN1

```
#define ADC_INTEN_ADCMPINTEN1 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN1_SHIFT)) & ADC_INTEN_ADCMPINTEN1_MASK)
```

ADCMPINTEN1 - Channel 1 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.56 ADC_INTEN_ADCMPINTEN10

```
#define ADC_INTEN_ADCMPINTEN10 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN10_SHIFT)) & ADC_INTEN_ADCMPINTEN10_MASK)
```

ADCMPINTEN10 - Channel 10 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.57 ADC_INTEN_ADCMPINTEN11

```
#define ADC_INTEN_ADCMPINTEN11 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN11_SHIFT)) & ADC_INTEN_ADCMPINTEN11_MASK)
```

ADCMPINTEN11 - Channel 21 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.58 ADC_INTEN_ADCMPINTEN2

```
#define ADC_INTEN_ADCMPINTEN2 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN2_SHIFT)) & ADC_INTEN_ADCMPINTEN2_MASK)
```

ADCMPINTEN2 - Channel 2 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.59 ADC_INTEN_ADCMPINTEN3

```
#define ADC_INTEN_ADCMPINTEN3 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN3_SHIFT)) & ADC_INTEN_ADCMPINTEN3_MASK)
```

ADCMPINTEN3 - Channel 3 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.60 ADC_INTEN_ADCMPINTEN4

```
#define ADC_INTEN_ADCMPINTEN4 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN4_SHIFT)) & ADC_INTEN_ADCMPINTEN4_MASK
```

ADCMPINTEN4 - Channel 4 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.61 ADC_INTEN_ADCMPINTEN5

```
#define ADC_INTEN_ADCMPINTEN5 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN5_SHIFT)) & ADC_INTEN_ADCMPINTEN5_MASK
```

ADCMPINTEN5 - Channel 5 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.62 ADC_INTEN_ADCMPINTEN6

```
#define ADC_INTEN_ADCMPINTEN6 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN6_SHIFT)) & ADC_INTEN_ADCMPINTEN6_MASK
```

ADCMPINTEN6 - Channel 6 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.63 ADC_INTEN_ADCMPINTEN7

```
#define ADC_INTEN_ADCMPINTEN7 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN7_SHIFT)) & ADC_INTEN_ADCMPINTEN7_MASK
```

ADCMPINTEN7 - Channel 7 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.64 ADC_INTEN_ADCMPINTEN8

```
#define ADC_INTEN_ADCMPINTEN8 (x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN8_SHIFT)) & ADC_INTEN_ADCMPINTEN8_MASK
```

ADCMPINTEN8 - Channel 8 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.65 ADC_INTEN_ADCMPINTEN9

```
#define ADC_INTEN_ADCMPINTEN9 (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_ADCMPINTEN9_SHIFT)) & ADC_INTEN_ADCMPINTEN9_MASK)
```

ADCMPINTEN9 - Channel 9 threshold comparison interrupt enable. See description for channel 0.

6.4.13.3.2.66 ADC_INTEN_OVR_INTEN

```
#define ADC_INTEN_OVR_INTEN (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_OVR_INTEN_SHIFT)) & ADC_INTEN_OVR_INTEN_MASK)
```

OVR_INTEN - Overrun interrupt enable. 0b0..Disabled. The overrun interrupt is disabled. 0b1..Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 12 channel data registers will cause an overrun interrupt/DMA trigger. In addition, if the MODE bit for a particular sequence is 0, then an overrun in the global data register for that sequence will also cause this interrupt/DMA trigger to be asserted.

6.4.13.3.2.67 ADC_INTEN_SEQA_INTEN

```
#define ADC_INTEN_SEQA_INTEN (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_SEQA_INTEN_SHIFT)) & ADC_INTEN_SEQA_INTEN_MASK)
```

SEQA_INTEN - Sequence A interrupt enable. 0b0..Disabled. The sequence A interrupt/DMA trigger is disabled. 0b1..Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQA_CTRL register.

6.4.13.3.2.68 ADC_INTEN_SEQB_INTEN

```
#define ADC_INTEN_SEQB_INTEN (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_INTEN_SEQB_INTEN_SHIFT)) & ADC_INTEN_SEQB_INTEN_MASK)
```

SEQB_INTEN - Sequence B interrupt enable. 0b0..Disabled. The sequence B interrupt/DMA trigger is disabled. 0b1..Enabled. The sequence B interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence B, or upon completion of the entire B sequence of conversions, depending on the MODE bit in the SEQB_CTRL register.

6.4.13.3.2.69 ADC_SEQ_CTRL_BURST

```
#define ADC_SEQ_CTRL_BURST (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_BURST_SHIFT) & ADC_SEQ_CTRL_BURST_MASK)
```

BURST - Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence A triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. Note that a new sequence could begin just before BURST is cleared.

6.4.13.3.2.70 ADC_SEQ_CTRL_CHANNELS

```
#define ADC_SEQ_CTRL_CHANNELS (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_CHANNELS_SHIFT) & ADC_SEQ_CTRL_CHANNELS_MASK)
```

CHANNELS - Selects which one or more of the [ADC](#) channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, [ADC](#) conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. This field can ONLY be changed while SEQQA_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.

6.4.13.3.2.71 ADC_SEQ_CTRL_COUNT

```
#define ADC_SEQ_CTRL_COUNT (2U)
```

The count of ADC_SEQ_CTRL

6.4.13.3.2.72 ADC_SEQ_CTRL_LOWPRIOR

```
#define ADC_SEQ_CTRL_LOWPRIOR (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_LOWPRIOR_SHIFT) & ADC_SEQ_CTRL_LOWPRIOR_MASK)
```

LOWPRIOR - Set priority for sequence A. 0b0..Low priority. Any B trigger which occurs while an A conversion sequence is active will be ignored and lost. 0b1..High priority. Setting this bit to a 1 will permit any enabled B sequence trigger (including a B sequence software start) to immediately interrupt sequence A and launch a B sequence in its place. The conversion currently in progress will be terminated. The A sequence that was interrupted will automatically resume after the B sequence completes. The channel whose conversion was terminated will be re-sampled and the conversion sequence will resume from that point.

6.4.13.3.2.73 ADC_SEQ_CTRL_MODE

```
#define ADC_SEQ_CTRL_MODE (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_CTRL_MODE_SHIFT) & ADC_SEQ_CTRL_MODE_MASK)
```

MODE - Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQA_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below. 0b0..End of conversion. The sequence A interrupt/DMA trigger will be set at the end of each individual **ADC** conversion performed under sequence A. This flag will mirror the DATAVALID bit in the SEQA_GDAT register. The OVERRUN bit in the SEQA_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled. 0b1..End of sequence. The sequence A interrupt/DMA trigger will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.

6.4.13.3.2.74 ADC_SEQ_CTRL_SEQ_ENA

```
#define ADC_SEQ_CTRL_SEQ_ENA (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_CTRL_SEQ_ENA_SHIFT) & ADC_SEQ_CTRL_SEQ_ENA_MASK)
```

SEQ_ENA - Sequence Enable. In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQn_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled. In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQn_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled. 0b0..Disabled. Sequence n is disabled. Sequence n triggers are ignored. If this bit is cleared while sequence n is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel. 0b1..Enabled. Sequence n is enabled.

6.4.13.3.2.75 ADC_SEQ_CTRL_SINGLESTEP

```
#define ADC_SEQ_CTRL_SINGLESTEP (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_CTRL_SINGLESTEP_SHIFT) & ADC_SEQ_CTRL_SINGLESTEP_MASK)
```

SINGLESTEP - When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.

6.4.13.3.2.76 ADC_SEQ_CTRL_START

```
#define ADC_SEQ_CTRL_START(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_START_SHIFT) & ADC_SEQ_CTRL_START_MASK)
```

START - Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.

6.4.13.3.2.77 ADC_SEQ_CTRL_SYNCBYPASS

```
#define ADC_SEQ_CTRL_SYNCBYPASS(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_SYNCBYPASS_SHIFT) & ADC_SEQ_CTRL_SYNCBYPASS_MASK)
```

SYNCBYPASS - Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the [ADC](#) clock (regardless of whether the trigger comes from an on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period. 0b0..Enable trigger synchronization. The hardware trigger bypass is not enabled. 0b1..Bypass trigger synchronization. The hardware trigger bypass is enabled.

6.4.13.3.2.78 ADC_SEQ_CTRL_TRIGGER

```
#define ADC_SEQ_CTRL_TRIGGER(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_TRIGGER_SHIFT) & ADC_SEQ_CTRL_TRIGGER_MASK)
```

TRIGGER - Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. See Table 476. In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.

6.4.13.3.2.79 ADC_SEQ_CTRL_TRIGPOL

```
#define ADC_SEQ_CTRL_TRIGPOL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_CTRL_TRIGPOL_SHIFT) & ADC_SEQ_CTRL_TRIGPOL_MASK)
```

TRIGPOL - Select the polarity of the selected input trigger for this conversion sequence. In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write. 0b0..Negative edge. A negative edge launches the conversion sequence on the selected trigger input. 0b1..Positive edge. A positive edge launches the conversion sequence on the selected trigger input.

6.4.13.3.2.80 ADC_SEQ_GDAT_CHN

```
#define ADC_SEQ_GDAT_CHN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_GDAT_CHN_SHIFT)) & ADC_SEQ_GDAT_CHN_MASK)
```

CHN - These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.).

6.4.13.3.2.81 ADC_SEQ_GDAT_COUNT

```
#define ADC_SEQ_GDAT_COUNT (2U)
```

The count of ADC_SEQ_GDAT

6.4.13.3.2.82 ADC_SEQ_GDAT_DATAVALID

```
#define ADC_SEQ_GDAT_DATAVALID(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_GDAT_DATAVALID_SHIFT)) & ADC_SEQ_GDAT_DATAVALID_MASK)
```

DATAVALID - This bit is set to '1' at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQAA_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).

6.4.13.3.2.83 ADC_SEQ_GDAT_OVERRUN

```
#define ADC_SEQ_GDAT_OVERRUN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_GDAT_OVERRUN_SHIFT)) & ADC_SEQ_GDAT_OVERRUN_MASK)
```

OVERRUN - This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQAA_CTRL) for the corresponding sequence is set to '0' (and if the overrun interrupt is enabled).

6.4.13.3.2.84 ADC_SEQ_GDAT_RESULT

```
#define ADC_SEQ_GDAT_RESULT(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << ADC_SEQ_GDAT_RESULT_SHIFT)) & ADC_SEQ_GDAT_RESULT_MASK)
```

RESULT - This field contains the 12-bit [ADC](#) conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP. DATAVALID = 1 indicates that this result has not yet been read.

6.4.13.3.2.85 ADC_SEQ_GDAT_THCMPCROSS

```
#define ADC_SEQ_GDAT_THCMPCROSS(  
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_GDAT_THCMPCROSS_SHIFT) & ADC_SEQ_GDAT_THCMPCROSS_MASK)
```

THCMPCROSS - Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred.

6.4.13.3.2.86 ADC_SEQ_GDAT_THCMPRANGE

```
#define ADC_SEQ_GDAT_THCMPRANGE(  
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_SEQ_GDAT_THCMPRANGE_SHIFT) & ADC_SEQ_GDAT_THCMPRANGE_MASK)
```

THCMPRANGE - Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH).

6.4.13.3.2.87 ADC_THR0_HIGH_THRHIGH

```
#define ADC_THR0_HIGH_THRHIGH(  
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_THR0_HIGH_THRHIGH_SHIFT) & ADC_THR0_HIGH_THRHIGH_MASK)
```

THRHIGH - High threshold value against which **ADC** results will be compared

6.4.13.3.2.88 ADC_THR0_LOW_THRLOW

```
#define ADC_THR0_LOW_THRLOW(  
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_THR0_LOW_THRLOW_SHIFT) & ADC_THR0_LOW_THRLOW_MASK)
```

THRLOW - Low threshold value against which **ADC** results will be compared

6.4.13.3.2.89 ADC_THR1_HIGH_THRHIGH

```
#define ADC_THR1_HIGH_THRHIGH(  
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_THR1_HIGH_THRHIGH_SHIFT) & ADC_THR1_HIGH_THRHIGH_MASK)
```

THRHIGH - High threshold value against which **ADC** results will be compared

6.4.13.3.2.90 ADC_THR1_LOW_THRLOW

```
#define ADC_THR1_LOW_THRLOW(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << ADC_THR1_LOW_THRLOW_SHIFT) & ADC_THR1_LOW_THRLOW_MASK)
```

THRLOW - Low threshold value against which [ADC](#) results will be compared

6.4.13.3.2.91 ADC_TRM_VRANGE

```
#define ADC_TRM_VRANGE(
    x)
```

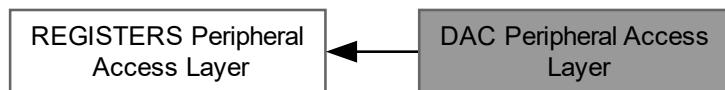
Value:

```
((uint32_t)((uint32_t)(x)) << ADC_TRM_VRANGE_SHIFT) & ADC_TRM_VRANGE_MASK)
```

VRANGE - 1.8V to 3.6V Vdd range: This bit MUST be set to '1' if operation below 2.7V is to be used. Failure to set this bit will result in invalid [ADC](#) results. Note: This bit will not be spec'd on parts that do not support operation below 2.7V

6.4.14 DAC Peripheral Access Layer

Collaboration diagram for DAC Peripheral Access Layer:



Classes

- struct [DAC_t](#)
Structure type to access the Digital Analog Convertor ([DAC](#)).

Macros

- #define [DAC0](#) (([DAC_t](#)*) 0x40014000UL)
- #define [DAC1](#) (([DAC_t](#)*) 0x40018000UL)

6.4.14.1 Detailed Description

6.4.14.2 Macro Definition Documentation

6.4.14.2.1 DAC0

```
#define DAC0 ( (DAC_t*) 0x40014000UL )
```

Register of the DAC0

6.4.14.2.2 DAC1

```
#define DAC1 ( (DAC_t*) 0x40018000UL )
```

Register of the DAC1

6.4.15 I2C Peripheral Access Layer

Collaboration diagram for I2C Peripheral Access Layer:



Topics

- [I2C Register Masks](#)

Classes

- struct [I2C_Type](#)

Macros

- #define [I2C0_BASE](#) (0x40050000u)
- #define [I2C0 \(\(I2C_Type *\)I2C0_BASE\)](#)
- #define [I2C1_BASE](#) (0x40054000u)
- #define [I2C1 \(\(I2C_Type *\)I2C1_BASE\)](#)
- #define [I2C2_BASE](#) (0x40030000u)
- #define [I2C2 \(\(I2C_Type *\)I2C2_BASE\)](#)
- #define [I2C3_BASE](#) (0x40034000u)
- #define [I2C3 \(\(I2C_Type *\)I2C3_BASE\)](#)
- #define [I2C_BASE_ADDRS](#) { [I2C0_BASE](#), [I2C1_BASE](#), [I2C2_BASE](#), [I2C3_BASE](#) }
- #define [I2C_BASE_PTRS](#) { [I2C0](#), [I2C1](#), [I2C2](#), [I2C3](#) }

6.4.15.1 Detailed Description

6.4.15.2 Macro Definition Documentation

6.4.15.2.1 I2C0

```
#define I2C0 ((I2C_Type *)I2C0_BASE)
```

Peripheral I2C0 base pointer

6.4.15.2.2 I2C0_BASE

```
#define I2C0_BASE (0x40050000u)
```

Peripheral I2C0 base address

6.4.15.2.3 I2C1

```
#define I2C1 ((I2C_Type *)I2C1_BASE)
```

Peripheral I2C1 base pointer

6.4.15.2.4 I2C1_BASE

```
#define I2C1_BASE (0x40054000u)
```

Peripheral I2C1 base address

6.4.15.2.5 I2C2

```
#define I2C2 ((I2C_Type *)I2C2_BASE)
```

Peripheral I2C2 base pointer

6.4.15.2.6 I2C2_BASE

```
#define I2C2_BASE (0x40030000u)
```

Peripheral I2C2 base address

6.4.15.2.7 I2C3

```
#define I2C3 ((I2C_Type *)I2C3_BASE)
```

Peripheral I2C3 base pointer

6.4.15.2.8 I2C3_BASE

```
#define I2C3_BASE (0x40034000u)
```

Peripheral I2C3 base address

6.4.15.2.9 I2C_BASE_ADDRS

```
#define I2C_BASE_ADDRS { I2C0_BASE, I2C1_BASE, I2C2_BASE, I2C3_BASE }
```

Array initializer of I2C peripheral base addresses

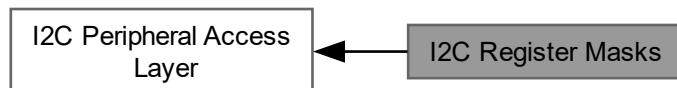
6.4.15.2.10 I2C_BASE_PTRS

```
#define I2C_BASE_PTRS { I2C0, I2C1, I2C2, I2C3 }
```

Array initializer of I2C peripheral base pointers

6.4.15.3 I2C Register Masks

Collaboration diagram for I2C Register Masks:



CFG - Configuration for shared functions.

- #define I2C_CFG_MSTEN(x)
- #define I2C_CFG_SLVEN(x)
- #define I2C_CFG_MONEN(x)
- #define I2C_CFG_TIMEOUTEN(x)
- #define I2C_CFG_MONCLKSTR(x)

STAT - Status register for Master, Slave, and Monitor functions.

- #define I2C_STAT_MSTPENDING(x)
- #define I2C_STAT_MSTSTATE(x)
- #define I2C_STAT_MSTARBLOSS(x)
- #define I2C_STAT_MSTSTSTPERR(x)
- #define I2C_STAT_SLVPENDING(x)
- #define I2C_STAT_SLVSTATE(x)
- #define I2C_STAT_SLVNOTSTR(x)
- #define I2C_STAT_SLVIDX(x)
- #define I2C_STAT_SLVSEL(x)
- #define I2C_STAT_SLVDESEL(x)
- #define I2C_STAT_MONRDY(x)
- #define I2C_STAT_MONOV(x)
- #define I2C_STAT_MONACTIVE(x)
- #define I2C_STAT_MONIDLE(x)
- #define I2C_STAT_EVENTTIMEOUT(x)
- #define I2C_STAT_SCLTIMEOUT(x)

INTENSET - Interrupt Enable Set and read register.

- #define I2C_INTENSET_MSTPENDINGEN(x)
- #define I2C_INTENSET_MSTARBLOSSEN(x)
- #define I2C_INTENSET_MSTSTSTPERREN(x)
- #define I2C_INTENSET_SLVPENDINGEN(x)
- #define I2C_INTENSET_SLVNOTSTREN(x)
- #define I2C_INTENSET_SLVDESELEN(x)
- #define I2C_INTENSET_MONRDYEN(x)
- #define I2C_INTENSET_MONOVEN(x)
- #define I2C_INTENSET_MONIDLEEN(x)
- #define I2C_INTENSET_EVENTTIMEOUTEN(x)
- #define I2C_INTENSET_SCLTIMEOUTEN(x)

MSTCTL - Master control register.

- #define I2C_MSTCTL_MSTCONTINUE(x)
- #define I2C_MSTCTL_MSTSTART(x)
- #define I2C_MSTCTL_MSTSTOP(x)
- #define I2C_MSTCTL_MSTDMA(x)

MSTTIME - Master timing configuration.

- #define I2C_MSTTIME_MSTSCLLOW(x)
- #define I2C_MSTTIME_MSTSCLHIGH(x)

SLVCTL - Slave control register.

- #define I2C_SLVCTL_SLVCONTINUE(x)
- #define I2C_SLVCTL_SLVNACK(x)
- #define I2C_SLVCTL_SLVDMA(x)

SLVADR - Slave address register.

- #define I2C_SLVADR_SADISABLE(x)

SLVQUAL0 - Slave Qualification for address 0.

- #define I2C_SLVQUAL0_QUALMODE0(x)

MONRXDAT - Monitor receiver data register.

- #define I2C_MONRXDAT_MONSTART(x)
- #define I2C_MONRXDAT_MONRESTART(x)
- #define I2C_MONRXDAT_MONNACK(x)

6.4.15.3.1 Detailed Description**6.4.15.3.2 Macro Definition Documentation****6.4.15.3.2.1 I2C_CFG_MONCLKSTR**

```
#define I2C_CFG_MONCLKSTR(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_CFG_MONCLKSTR_SHIFT) & I2C_CFG_MONCLKSTR_MASK)
```

MONCLKSTR - Monitor function Clock Stretching. 0b0..Disabled. The Monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical. 0b1..Enabled. The Monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the Monitor function.

6.4.15.3.2.2 I2C_CFG_MONEN

```
#define I2C_CFG_MONEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_CFG_MONEN_SHIFT) & I2C_CFG_MONEN_MASK)
```

MONEN - Monitor Enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset. 0b0..Disabled. The I2C Monitor function is disabled. 0b1..Enabled. The I2C Monitor function is enabled.

6.4.15.3.2.3 I2C_CFG_MSTEN

```
#define I2C_CFG_MSTEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_CFG_MSTEN_SHIFT) & I2C_CFG_MSTEN_MASK)
```

MSTEN - Master Enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset. 0b0..Disabled. The I2C Master function is disabled. 0b1..Enabled. The I2C Master function is enabled.

6.4.15.3.2.4 I2C_CFG_SLVEN

```
#define I2C_CFG_SLVEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_CFG_SLVEN_SHIFT) & I2C_CFG_SLVEN_MASK)
```

SLVEN - Slave Enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset. 0b0..Disabled. The I2C slave function is disabled. 0b1..Enabled. The I2C slave function is enabled.

6.4.15.3.2.5 I2C_CFG_TIMEOUTEN

```
#define I2C_CFG_TIMEOUTEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_CFG_TIMEOUTEN_SHIFT) & I2C_CFG_TIMEOUTEN_MASK)
```

TIMEOUTEN - I2C bus Time-out Enable. When disabled, the time-out function is internally reset. 0b0..Disabled. Time-out function is disabled. 0b1..Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.

6.4.15.3.2.6 I2C_INTENSET_EVENTTIMEOUTEN

```
#define I2C_INTENSET_EVENTTIMEOUTEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_INTENSET_EVENTTIMEOUTEN_SHIFT) & I2C_INTENSET_EVENTTIMEOUTEN_MASK)
```

EVENTTIMEOUTEN - Event time-out interrupt Enable. 0b0..Disabled. The Event time-out interrupt is disabled. 0b1..Enabled. The Event time-out interrupt is enabled.

6.4.15.3.2.7 I2C_INTENSET_MONIDLEEN

```
#define I2C_INTENSET_MONIDLEEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_INTENSET_MONIDLEEN_SHIFT) & I2C_INTENSET_MONIDLEEN_MASK)
```

MONIDLEEN - Monitor Idle interrupt Enable. 0b0..Disabled. The MonIdle interrupt is disabled. 0b1..Enabled. The Monidle interrupt is enabled.

6.4.15.3.2.8 I2C_INTENSET_MONOVEN

```
#define I2C_INTENSET_MONOVEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_INTENSET_MONOVEN_SHIFT) & I2C_INTENSET_MONOVEN_MASK)
```

MONOVEN - Monitor Overrun interrupt Enable. 0b0..Disabled. The MonOv interrupt is disabled. 0b1..Enabled. The MonOv interrupt is enabled.

6.4.15.3.2.9 I2C_INTENSET_MONRDYEN

```
#define I2C_INTENSET_MONRDYEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_INTENSET_MONRDYEN_SHIFT)) & I2C_INTENSET_MONRDYEN_MASK)
```

MONRDYEN - Monitor data Ready interrupt Enable. 0b0..Disabled. The MonRdy interrupt is disabled. 0b1..Enabled. The MonRdy interrupt is enabled.

6.4.15.3.2.10 I2C_INTENSET_MSTARBLOSSEN

```
#define I2C_INTENSET_MSTARBLOSSEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_INTENSET_MSTARBLOSSEN_SHIFT)) & I2C_INTENSET_MSTARBLOSSEN_MASK)
```

MSTARBLOSSEN - Master Arbitration Loss interrupt Enable. 0b0..Disabled. The MstArbLoss interrupt is disabled. 0b1..Enabled. The MstArbLoss interrupt is enabled.

6.4.15.3.2.11 I2C_INTENSET_MSTPENDINGEN

```
#define I2C_INTENSET_MSTPENDINGEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_INTENSET_MSTPENDINGEN_SHIFT)) & I2C_INTENSET_MSTPENDINGEN_MASK)
```

MSTPENDINGEN - Master Pending interrupt Enable. 0b0..Disabled. The MstPending interrupt is disabled. 0b1..Enabled. The MstPending interrupt is enabled.

6.4.15.3.2.12 I2C_INTENSET_MSTSTSTPERREN

```
#define I2C_INTENSET_MSTSTSTPERREN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_INTENSET_MSTSTSTPERREN_SHIFT)) & I2C_INTENSET_MSTSTSTPERREN_MASK)
```

MSTSTSTPERREN - Master Start/Stop Error interrupt Enable. 0b0..Disabled. The MstStTpErr interrupt is disabled. 0b1..Enabled. The MstStTpErr interrupt is enabled.

6.4.15.3.2.13 I2C_INTENSET_SCLTIMEOUTEN

```
#define I2C_INTENSET_SCLTIMEOUTEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_INTENSET_SCLTIMEOUTEN_SHIFT)) & I2C_INTENSET_SCLTIMEOUTEN_MASK)
```

SCLTIMEOUTEN - SCL time-out interrupt Enable. 0b0..Disabled. The SCL time-out interrupt is disabled. 0b1..Enabled. The SCL time-out interrupt is enabled.

6.4.15.3.2.14 I2C_INTENSET_SLVDESELEN

```
#define I2C_INTENSET_SLVDESELEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_INTENSET_SLVDESELEN_SHIFT) & I2C_INTENSET_SLVDESELEN_MASK)
```

SLVDESELEN - Slave Deselect interrupt Enable. 0b0..Disabled. The SlvDeSel interrupt is disabled. 0b1..Enabled. The SlvDeSel interrupt is enabled.

6.4.15.3.2.15 I2C_INTENSET_SLVNOTSTREN

```
#define I2C_INTENSET_SLVNOTSTREN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_INTENSET_SLVNOTSTREN_SHIFT) & I2C_INTENSET_SLVNOTSTREN_MASK)
```

SLVNOTSTREN - Slave Not Stretching interrupt Enable. 0b0..Disabled. The SlvNotStr interrupt is disabled. 0b1..Enabled. The SlvNotStr interrupt is enabled.

6.4.15.3.2.16 I2C_INTENSET_SLPENDINGEN

```
#define I2C_INTENSET_SLPENDINGEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_INTENSET_SLPENDINGEN_SHIFT) & I2C_INTENSET_SLPENDINGEN_MASK)
```

SLVPENDINGEN - Slave Pending interrupt Enable. 0b0..Disabled. The SlvPending interrupt is disabled. 0b1..Enabled. The SlvPending interrupt is enabled.

6.4.15.3.2.17 I2C_MONRXDAT_MONNACK

```
#define I2C_MONRXDAT_MONNACK(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MONRXDAT_MONNACK_SHIFT) & I2C_MONRXDAT_MONNACK_MASK)
```

MONNACK - Monitor Received NACK. 0b0..Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave receiver. 0b1..Not acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.

6.4.15.3.2.18 I2C_MONRXDAT_MONRESTART

```
#define I2C_MONRXDAT_MONRESTART(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MONRXDAT_MONRESTART_SHIFT) & I2C_MONRXDAT_MONRESTART_MASK)
```

MONRESTART - Monitor Received Repeated Start. 0b0..No repeated start detected. The Monitor function has not detected a Repeated Start event on the I2C bus. 0b1..Repeated start detected. The Monitor function has detected a Repeated Start event on the I2C bus.

6.4.15.3.2.19 I2C_MONRXDAT_MONSTART

```
#define I2C_MONRXDAT_MONSTART (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MONRXDAT_MONSTART_SHIFT) & I2C_MONRXDAT_MONSTART_MASK)
```

MONSTART - Monitor Received Start. 0b0..No start detected. The Monitor function has not detected a Start event on the **I2C** bus. 0b1..Start detected. The Monitor function has detected a Start event on the **I2C** bus.

6.4.15.3.2.20 I2C_MSTCTL_MSTCONTINUE

```
#define I2C_MSTCTL_MSTCONTINUE (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MSTCTL_MSTCONTINUE_SHIFT) & I2C_MSTCTL_MSTCONTINUE_MASK)
```

MSTCONTINUE - Master Continue. 0b0..No effect. 0b1..Informs the Master function to continue to the next operation.

6.4.15.3.2.21 I2C_MSTCTL_MSTDMA

```
#define I2C_MSTCTL_MSTDMA (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MSTCTL_MSTDMA_SHIFT) & I2C_MSTCTL_MSTDMA_MASK)
```

MSTDMA - Master DMA enable. Data operations of the **I2C** can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the **I2C** is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when Automatic Operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop. This bit is read/write. 0b0..Disable. No DMA requests are generated for master operation. 0b1..Enable. A DMA request is generated for **I2C** master data operations. When this **I2C** master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.

6.4.15.3.2.22 I2C_MSTCTL_MSTSTART

```
#define I2C_MSTCTL_MSTSTART (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MSTCTL_MSTSTART_SHIFT) & I2C_MSTCTL_MSTSTART_MASK)
```

MSTSTART - Master Start control. 0b0..No effect. 0b1..Start. A Start will be generated on the **I2C** bus at the next allowed time.

6.4.15.3.2.23 I2C_MSTCTL_MSTSTOP

```
#define I2C_MSTCTL_MSTSTOP (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MSTCTL_MSTSTOP_SHIFT) & I2C_MSTCTL_MSTSTOP_MASK)
```

MSTSTOP - Master Stop control. 0b0..No effect. 0b1..Stop. A Stop will be generated on the **I2C** bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (Master Receiver mode).

6.4.15.3.2.24 I2C_MSTTIME_MSTSCLHIGH

```
#define I2C_MSTTIME_MSTSCLHIGH (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MSTTIME_MSTSCLHIGH_SHIFT) & I2C_MSTTIME_MSTSCLHIGH_MASK)
```

MSTSCLHIGH - Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter tHIGH in the **I2C** bus specification. **I2C** bus specification parameters tSU;STO and tHD;STA have the same values and are also controlled by MSTSCLHIGH. 0b000..2 clocks. Minimum SCL high time is 2 clock of the **I2C** clock pre-divider. 0b001..3 clocks. Minimum SCL high time is 3 clocks of the **I2C** clock pre-divider. 0b010..4 clocks. Minimum SCL high time is 4 clock of the **I2C** clock pre-divider. 0b011..5 clocks. Minimum SCL high time is 5 clock of the **I2C** clock pre-divider. 0b100..6 clocks. Minimum SCL high time is 6 clock of the **I2C** clock pre-divider. 0b101..7 clocks. Minimum SCL high time is 7 clock of the **I2C** clock pre-divider. 0b110..8 clocks. Minimum SCL high time is 8 clock of the **I2C** clock pre-divider. 0b111..9 clocks. Minimum SCL high time is 9 clocks of the **I2C** clock pre-divider.

6.4.15.3.2.25 I2C_MSTTIME_MSTSCLLOW

```
#define I2C_MSTTIME_MSTSCLLOW (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_MSTTIME_MSTSCLLOW_SHIFT) & I2C_MSTTIME_MSTSCLLOW_MASK)
```

MSTSCLLOW - Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t LOW in the **I2C** bus specification. **I2C** bus specification parameters tBUF and tSU;STA have the same values and are also controlled by MSTSCLLOW. 0b000..2 clocks. Minimum SCL low time is 2 clocks of the **I2C** clock pre-divider. 0b001..3 clocks. Minimum SCL low time is 3 clocks of the **I2C** clock pre-divider. 0b010..4 clocks. Minimum SCL low time is 4 clocks of the **I2C** clock pre-divider. 0b011..5 clocks. Minimum SCL low time is 5 clocks of the **I2C** clock pre-divider. 0b100..6 clocks. Minimum SCL low time is 6 clocks of the **I2C** clock pre-divider. 0b101..7 clocks. Minimum SCL low time is 7 clocks of the **I2C** clock pre-divider. 0b110..8 clocks. Minimum SCL low time is 8 clocks of the **I2C** clock pre-divider. 0b111..9 clocks. Minimum SCL low time is 9 clocks of the **I2C** clock pre-divider.

6.4.15.3.2.26 I2C_SLVADR_SADISABLE

```
#define I2C_SLVADR_SADISABLE (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_SLVADR_SADISABLE_SHIFT) & I2C_SLVADR_SADISABLE_MASK)
```

SADISABLE - Slave Address n Disable. 0b0..Enabled. Slave Address n is enabled. 0b1..Ignored Slave Address n is ignored.

6.4.15.3.2.27 I2C_SLVCTL_SLVCONTINUE

```
#define I2C_SLVCTL_SLVCONTINUE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_SLVCTL_SLVCONTINUE_SHIFT) & I2C_SLVCTL_SLVCONTINUE_MASK)
```

SLVCONTINUE - Slave Continue. 0b0..No effect. 0b1..Informs the Slave function to continue to the next operation.

6.4.15.3.2.28 I2C_SLVCTL_SLVDMA

```
#define I2C_SLVCTL_SLVDMA(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_SLVCTL_SLVDMA_SHIFT) & I2C_SLVCTL_SLVDMA_MASK)
```

SLVDMA - Slave DMA enable. 0b0..Disabled. No DMA requests are issued for Slave mode operation. 0b1..Enabled. DMA requests are issued for **I2C** slave data transmission and reception.

6.4.15.3.2.29 I2C_SLVCTL_SLVNACK

```
#define I2C_SLVCTL_SLVNACK(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_SLVCTL_SLVNACK_SHIFT) & I2C_SLVCTL_SLVNACK_MASK)
```

SLVNACK - Slave NACK. 0b0..No effect. 0b1..NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode).

6.4.15.3.2.30 I2C_SLVQUAL0_QUALMODE0

```
#define I2C_SLVQUAL0_QUALMODE0(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_SLVQUAL0_QUALMODE0_SHIFT) & I2C_SLVQUAL0_QUALMODE0_MASK)
```

QUALMODE0 - Qualify mode for slave address 0. 0b0..Mask. The SLVQUAL0 field is used as a logical mask for matching address 0. 0b1..Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.

6.4.15.3.2.31 I2C_STAT_EVENTTIMEOUT

```
#define I2C_STAT_EVENTTIMEOUT(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_STAT_EVENTTIMEOUT_SHIFT) & I2C_STAT_EVENTTIMEOUT_MASK)
```

EVENTTIMEOUT - Event Time-out Interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I2C-bus is idle. 0b0..No time-out. I2C bus events have not caused a time-out. 0b1..Event time-out. The time between I2C bus events has been longer than the time specified by the TIMEOUT register.

6.4.15.3.2.32 I2C_STAT_MONACTIVE

```
#define I2C_STAT_MONACTIVE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_STAT_MONACTIVE_SHIFT) & I2C_STAT_MONACTIVE_MASK)
```

MONACTIVE - Monitor Active flag. Indicates when the Monitor function considers the I2C bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop. 0b0..Inactive. The Monitor function considers the I2C bus to be inactive. 0b1..Active. The Monitor function considers the I2C bus to be active.

6.4.15.3.2.33 I2C_STAT_MONIDLE

```
#define I2C_STAT_MONIDLE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_STAT_MONIDLE_SHIFT) & I2C_STAT_MONIDLE_MASK)
```

MONIDLE - Monitor Idle flag. This flag is set when the Monitor function sees the I2C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit. 0b0..Not idle. The I2C bus is not idle, or this flag has been cleared by software. 0b1..Idle. The I2C bus has gone idle at least once since the last time this flag was cleared by software.

6.4.15.3.2.34 I2C_STAT_MONOV

```
#define I2C_STAT_MONOV(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << I2C_STAT_MONOV_SHIFT) & I2C_STAT_MONOV_MASK)
```

MONOV - Monitor Overflow flag. 0b0..No overrun. Monitor data has not overrun. 0b1..Overrun. A Monitor data overrun has occurred. This can only happen when Monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.

6.4.15.3.2.35 I2C_STAT_MONRDY

```
#define I2C_STAT_MONRDY(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_MONRDY_SHIFT) & I2C_STAT_MONRDY_MASK)
```

MONRDY - Monitor Ready. This flag is cleared when the MONRXDAT register is read. 0b0..No data. The Monitor function does not currently have data available. 0b1..Data waiting. The Monitor function has data waiting to be read.

6.4.15.3.2.36 I2C_STAT_MSTARBLOSS

```
#define I2C_STAT_MSTARBLOSS(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_MSTARBLOSS_SHIFT) & I2C_STAT_MSTARBLOSS_MASK)
```

MSTARBLOSS - Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE. 0b0..No Arbitration Loss has occurred. 0b1..Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle.

6.4.15.3.2.37 I2C_STAT_MSTPENDING

```
#define I2C_STAT_MSTPENDING(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_MSTPENDING_SHIFT) & I2C_STAT_MSTPENDING_MASK)
```

MSTPENDING - Master Pending. Indicates that the Master is waiting to continue communication on the I2C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt. 0b0..In progress. Communication is in progress and the Master function is busy and cannot currently accept a command. 0b1..Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.

6.4.15.3.2.38 I2C_STAT_MSTSTATE

```
#define I2C_STAT_MSTSTATE(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_MSTSTATE_SHIFT) & I2C_STAT_MSTSTATE_MASK)
```

MSTSTATE - Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved. See Table 400 for details of state values and appropriate responses. 0b000..Idle. The Master function is available to be used for a new transaction. 0b001..Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave. 0b010..Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave. 0b011..NACK Address. Slave NACKed address. 0b100..NACK Data. Slave NACKed transmitted data.

6.4.15.3.2.39 I2C_STAT_MSTSTSTPERR

```
#define I2C_STAT_MSTSTSTPERR(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_MSTSTSTPERR_SHIFT) & I2C_STAT_MSTSTSTPERR_MASK)
```

MSTSTSTPERR - Master Start/Stop Error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE. 0b0..No Start/Stop Error has occurred. 0b1..The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the [I2C](#) specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that the bus has not stalled.

6.4.15.3.2.40 I2C_STAT_SCLTIMEOUT

```
#define I2C_STAT_SCLTIMEOUT(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_SCLTIMEOUT_SHIFT) & I2C_STAT_SCLTIMEOUT_MASK)
```

SCLTIMEOUT - SCL Time-out Interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit. 0b0..No time-out. SCL low time has not caused a time-out. 0b1..Time-out. SCL low time has caused a time-out.

6.4.15.3.2.41 I2C_STAT_SLVDESEL

```
#define I2C_STAT_SLVDESEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_SLVDESEL_SHIFT) & I2C_STAT_SLVDESEL_MASK)
```

SLVDESEL - Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit. 0b0..Not deselected. The Slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag. 0b1..Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.

6.4.15.3.2.42 I2C_STAT_SLVIDX

```
#define I2C_STAT_SLVIDX(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_SLVIDX_SHIFT) & I2C_STAT_SLVIDX_MASK)
```

SLVIDX - Slave address match Index. This field is valid when the [I2C](#) slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here. 0b00..Address 0. Slave address 0 was matched. 0b01..Address 1. Slave address 1 was matched. 0b10..Address 2. Slave address 2 was matched. 0b11..Address 3. Slave address 3 was matched.

6.4.15.3.2.43 I2C_STAT_SLVNOTSTR

```
#define I2C_STAT_SLVNOTSTR(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_SLVNOTSTR_SHIFT) & I2C_STAT_SLVNOTSTR_MASK)
```

SLVNOTSTR - Slave Not Stretching. Indicates when the slave function is stretching the **I2C** clock. This is needed in order to gracefully invoke Deep Sleep or Power-down modes during slave operation. This read-only flag reflects the slave function status in real time. 0b0..Stretching. The slave function is currently stretching the **I2C** bus clock. Deep-Sleep or Power-down mode cannot be entered at this time. 0b1..Not stretching. The slave function is not currently stretching the **I2C** bus clock. Deep-sleep or Power-down mode could be entered at this time.

6.4.15.3.2.44 I2C_STAT_SLVPENDING

```
#define I2C_STAT_SLVPENDING(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_SLVPENDING_SHIFT) & I2C_STAT_SLVPENDING_MASK)
```

SLVPENDING - Slave Pending. Indicates that the Slave function is waiting to continue communication on the **I2C**-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the **I2C** interface is in HSCAPABLE mode. See Section 25.7.2.2.2. When the **I2C** interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS **I2C** specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched. 0b0..In progress. The Slave function does not currently need service. 0b1..Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.

6.4.15.3.2.45 I2C_STAT_SLVSEL

```
#define I2C_STAT_SLVSEL(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x)) << I2C_STAT_SLVSEL_SHIFT) & I2C_STAT_SLVSEL_MASK)
```

SLVSEL - Slave selected flag. SLVSEL is set after an address match when software tells the Slave function to acknowledge the address, or when the address has been automatically acknowledged. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, when there is a Stop detected on the bus, when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data. 0b0..Not selected. The Slave function is not currently selected. 0b1..Selected. The Slave function is currently selected.

6.4.15.3.2.46 I2C_STAT_SLVSTATE

```
#define I2C_STAT_SLVSTATE( x)
```

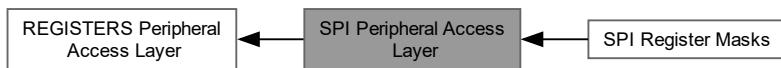
Value:

```
((uint32_t)((uint32_t)(x) << I2C_STAT_SLVSTATE_SHIFT)) & I2C_STAT_SLVSTATE_MASK)
```

SLVSTATE - Slave State code. Each value of this field indicates a specific required service for the Slave function. All other values are reserved. See Table 401 for state values and actions. note that the occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes. 0b00..Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware. 0b01..Slave receive. Received data is available (Slave Receiver mode). 0b10..Slave transmit. Data can be transmitted (Slave Transmitter mode).

6.4.16 SPI Peripheral Access Layer

Collaboration diagram for SPI Peripheral Access Layer:



Topics

- [SPI Register Masks](#)

Classes

- struct [SPI_Type](#)

Macros

- #define [SPI0_BASE](#) (0x40058000u)
- #define [SPI0](#) (([SPI_Type](#) *)[SPI0_BASE](#))
- #define [SPI1_BASE](#) (0x4005C000u)
- #define [SPI1](#) (([SPI_Type](#) *)[SPI1_BASE](#))
- #define [SPI_BASE_ADDRS](#) { [SPI0_BASE](#), [SPI1_BASE](#) }
- #define [SPI_BASE_PTRS](#) { [SPI0](#), [SPI1](#) }

6.4.16.1 Detailed Description

6.4.16.2 Macro Definition Documentation

6.4.16.2.1 SPI0

```
#define SPI0 ((SPI\_Type *)SPI0\_BASE)
```

Peripheral SPI0 base pointer

6.4.16.2.2 SPI0_BASE

```
#define SPI0_BASE (0x40058000u)
```

Peripheral SPI0 base address

6.4.16.2.3 SPI1

```
#define SPI1 ((SPI_Type *)SPI1_BASE)
```

Peripheral SPI1 base pointer

6.4.16.2.4 SPI1_BASE

```
#define SPI1_BASE (0x4005C000u)
```

Peripheral SPI1 base address

6.4.16.2.5 SPI_BASE_ADDRS

```
#define SPI_BASE_ADDRS { SPI0_BASE, SPI1_BASE }
```

Array initializer of SPI peripheral base addresses

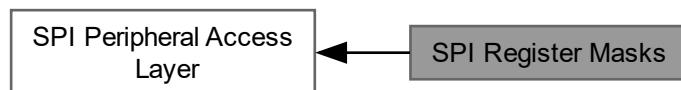
6.4.16.2.6 SPI_BASE_PTRS

```
#define SPI_BASE_PTRS { SPI0, SPI1 }
```

Array initializer of SPI peripheral base pointers

6.4.16.3 SPI Register Masks

Collaboration diagram for SPI Register Masks:



CFG - SPI Configuration register

- #define SPI_CFG_ENABLE(x)
- #define SPI_CFG_MASTER(x)
- #define SPI_CFG_LSBF(x)
- #define SPI_CFG_CPHA(x)
- #define SPI_CFG_CPOL(x)
- #define SPI_CFG_LOOP(x)
- #define SPI_CFG_SPOL0(x)
- #define SPI_CFG_SPOL1(x)
- #define SPI_CFG_SPOL2(x)
- #define SPI_CFG_SPOL3(x)

INTENSET - SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.

- #define SPI_INTENSET_RXRDYEN(x)
- #define SPI_INTENSET_TXRDYEN(x)
- #define SPI_INTENSET_RXOVEN(x)
- #define SPI_INTENSET_TXUREN(x)
- #define SPI_INTENSET_SSAEN(x)
- #define SPI_INTENSET_SSDEN(x)

TXDATCTL - SPI Transmit Data with Control

- #define SPI_TXDATCTL_TXSSEL0_N(x)
- #define SPI_TXDATCTL_TXSSEL1_N(x)
- #define SPI_TXDATCTL_TXSSEL2_N(x)
- #define SPI_TXDATCTL_TXSSEL3_N(x)
- #define SPI_TXDATCTL_EOT(x)
- #define SPI_TXDATCTL_EOF(x)
- #define SPI_TXDATCTL_RXIGNORE(x)
- #define SPI_TXDATCTL_LEN(x)

6.4.16.3.1 Detailed Description

6.4.16.3.2 Macro Definition Documentation

6.4.16.3.2.1 SPI_CFG_CPHA

```
#define SPI_CFG_CPHA(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_CFG_CPHA_SHIFT)) & SPI_CFG_CPHA_MASK)
```

CPHA - Clock Phase select. 0b0..Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge. 0b1..Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.

6.4.16.3.2.2 SPI_CFG_CPOL

```
#define SPI_CFG_CPOL(
    x)
```

Value:

```
((uint32_t) (((uint32_t)(x)) << SPI_CFG_CPOL_SHIFT)) & SPI_CFG_CPOL_MASK)
```

CPOL - Clock Polarity select. 0b0..Low. The rest state of the clock (between transfers) is low. 0b1..High. The rest state of the clock (between transfers) is high.

6.4.16.3.2.3 SPI_CFG_ENABLE

```
#define SPI_CFG_ENABLE(
    x)
```

Value:

```
((uint32_t) (((uint32_t)(x)) << SPI_CFG_ENABLE_SHIFT)) & SPI_CFG_ENABLE_MASK)
```

ENABLE - SPI enable. 0b0..Disabled. The SPI is disabled and the internal state machine and counters are reset. 0b1..Enabled. The SPI is enabled for operation.

6.4.16.3.2.4 SPI_CFG_LOOP

```
#define SPI_CFG_LOOP(
    x)
```

Value:

```
((uint32_t) (((uint32_t)(x)) << SPI_CFG_LOOP_SHIFT)) & SPI_CFG_LOOP_MASK)
```

LOOP - Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing. 0b0..Disabled. 0b1..Enabled.

6.4.16.3.2.5 SPI_CFG_LSBF

```
#define SPI_CFG_LSBF(
    x)
```

Value:

```
((uint32_t) (((uint32_t)(x)) << SPI_CFG_LSBF_SHIFT)) & SPI_CFG_LSBF_MASK)
```

LSBF - LSB First mode enable. 0b0..Standard. Data is transmitted and received in standard MSB first order. 0b1..Reverse. Data is transmitted and received in reverse order (LSB first).

6.4.16.3.2.6 SPI_CFG_MASTER

```
#define SPI_CFG_MASTER(
    x)
```

Value:

```
((uint32_t) (((uint32_t)(x)) << SPI_CFG_MASTER_SHIFT)) & SPI_CFG_MASTER_MASK)
```

MASTER - Master mode select. 0b0..Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output. 0b1..Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.

6.4.16.3.2.7 SPI_CFG_SPOL0

```
#define SPI_CFG_SPOL0 (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_CFG_SPOL0_SHIFT)) & SPI_CFG_SPOL0_MASK)
```

SPOL0 - SSEL0 Polarity select. 0b0..Low. The SSEL0 pin is active low. 0b1..High. The SSEL0 pin is active high.

6.4.16.3.2.8 SPI_CFG_SPOL1

```
#define SPI_CFG_SPOL1 (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_CFG_SPOL1_SHIFT)) & SPI_CFG_SPOL1_MASK)
```

SPOL1 - SSEL1 Polarity select. 0b0..Low. The SSEL1 pin is active low. 0b1..High. The SSEL1 pin is active high.

6.4.16.3.2.9 SPI_CFG_SPOL2

```
#define SPI_CFG_SPOL2 (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_CFG_SPOL2_SHIFT)) & SPI_CFG_SPOL2_MASK)
```

SPOL2 - SSEL2 Polarity select. 0b0..Low. The SSEL2 pin is active low. 0b1..High. The SSEL2 pin is active high.

6.4.16.3.2.10 SPI_CFG_SPOL3

```
#define SPI_CFG_SPOL3 (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_CFG_SPOL3_SHIFT)) & SPI_CFG_SPOL3_MASK)
```

SPOL3 - SSEL3 Polarity select. 0b0..Low. The SSEL3 pin is active low. 0b1..High. The SSEL3 pin is active high.

6.4.16.3.2.11 SPI_INTENSET_RXOVEN

```
#define SPI_INTENSET_RXOVEN (
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_INTENSET_RXOVEN_SHIFT)) & SPI_INTENSET_RXOVEN_MASK)
```

RXOVEN - Determines whether an interrupt occurs when a receiver overrun occurs. This happens in slave mode when there is a need for the receiver to move newly received data to the RXDAT register when it is already in use. The interface prevents receiver overrun in Master mode by not allowing a new transmission to begin when a receiver overrun would otherwise occur. 0b0..No interrupt will be generated when a receiver overrun occurs. 0b1..An interrupt will be generated if a receiver overrun occurs.

6.4.16.3.2.12 SPI_INTENSET_RXRDYEN

```
#define SPI_INTENSET_RXRDYEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_INTENSET_RXRDYEN_SHIFT) & SPI_INTENSET_RXRDYEN_MASK)
```

RXRDYEN - Determines whether an interrupt occurs when receiver data is available. 0b0..No interrupt will be generated when receiver data is available. 0b1..An interrupt will be generated when receiver data is available in the RXDAT register.

6.4.16.3.2.13 SPI_INTENSET_SSAEN

```
#define SPI_INTENSET_SSAEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_INTENSET_SSAEN_SHIFT) & SPI_INTENSET_SSAEN_MASK)
```

SSAEN - Determines whether an interrupt occurs when the Slave Select is asserted. 0b0..No interrupt will be generated when any Slave Select transitions from deasserted to asserted. 0b1..An interrupt will be generated when any Slave Select transitions from deasserted to asserted.

6.4.16.3.2.14 SPI_INTENSET_SSDEN

```
#define SPI_INTENSET_SSDEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_INTENSET_SSDEN_SHIFT) & SPI_INTENSET_SSDEN_MASK)
```

SSDEN - Determines whether an interrupt occurs when the Slave Select is deasserted. 0b0..No interrupt will be generated when all asserted Slave Selects transition to deasserted. 0b1..An interrupt will be generated when all asserted Slave Selects transition to deasserted.

6.4.16.3.2.15 SPI_INTENSET_TXRDYEN

```
#define SPI_INTENSET_TXRDYEN(
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_INTENSET_TXRDYEN_SHIFT) & SPI_INTENSET_TXRDYEN_MASK)
```

TXRDYEN - Determines whether an interrupt occurs when the transmitter holding register is available. 0b0..No interrupt will be generated when the transmitter holding register is available. 0b1..An interrupt will be generated when data may be written to TXDAT.

6.4.16.3.2.16 SPI_INTENSET_TXUREN

```
#define SPI_INTENSET_TXUREN(
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_INTENSET_TXUREN_SHIFT)) & SPI_INTENSET_TXUREN_MASK)
```

TXUREN - Determines whether an interrupt occurs when a transmitter underrun occurs. This happens in slave mode when there is a need to transmit data when none is available. 0b0..No interrupt will be generated when the transmitter underruns. 0b1..An interrupt will be generated if the transmitter underruns.

6.4.16.3.2.17 SPI_TXDATCTL_EOF

```
#define SPI_TXDATCTL_EOF(
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_TXDATCTL_EOF_SHIFT)) & SPI_TXDATCTL_EOF_MASK)
```

EOF - End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits. 0b0..This piece of data transmitted is not treated as the end of a frame. 0b1..This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.

6.4.16.3.2.18 SPI_TXDATCTL_EOT

```
#define SPI_TXDATCTL_EOT(
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_TXDATCTL_EOT_SHIFT)) & SPI_TXDATCTL_EOT_MASK)
```

EOT - End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register. 0b0..This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data. 0b1..This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.

6.4.16.3.2.19 SPI_TXDATCTL_LEN

```
#define SPI_TXDATCTL_LEN(
    x)
```

Value:

```
((uint32_t) (((uint32_t) (x)) << SPI_TXDATCTL_LEN_SHIFT)) & SPI_TXDATCTL_LEN_MASK)
```

LEN - Data Length. Specifies the data length from 1 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0 = Data transfer is 1 bit in length. 0x1 = Data transfer is 2 bits in length. 0x2 = Data transfer is 3 bits in length. ... 0xF = Data transfer is 16 bits in length. 0b0000..0b0001..Data transfer is 1 bit in length. 0b0010..Data transfer is 2 bit in length. 0b0011..Data transfer is 3 bit in length. 0b0100..Data transfer is 4 bit in length. 0b0101..Data transfer is 5 bit in length. 0b0110..Data transfer is 6 bit in length. 0b0111..Data transfer is 7 bit in length. 0b1000..Data transfer is 8 bit in length. 0b1001..Data transfer is 9 bit in length. 0b1010..Data transfer is 10 bit in length. 0b1011..Data transfer is 11 bit in length. 0b1100..Data transfer is 12 bit in length. 0b1101..Data transfer is 13 bit in length. 0b1110..Data transfer is 14 bit in length. 0b1111..Data transfer is 15 bit in length.

6.4.16.3.2.20 SPI_TXDATCTL_RXIGNORE

```
#define SPI_TXDATCTL_RXIGNORE (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_TXDATCTL_RXIGNORE_SHIFT)) & SPI_TXDATCTL_RXIGNORE_MASK)
```

RXIGNORE - Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA. 0b0..Received data must be read in order to allow transmission to progress. In slave mode, an overrun error will occur if received data is not read before new data is received. 0b1..Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.

6.4.16.3.2.21 SPI_TXDATCTL_TXSSEL0_N

```
#define SPI_TXDATCTL_TXSSEL0_N (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_TXDATCTL_TXSSEL0_N_SHIFT)) & SPI_TXDATCTL_TXSSEL0_N_MASK)
```

TXSSEL0_N - Transmit Slave Select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register. 0b0..SSEL0 asserted. 0b1..SSEL0 not asserted.

6.4.16.3.2.22 SPI_TXDATCTL_TXSSEL1_N

```
#define SPI_TXDATCTL_TXSSEL1_N (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_TXDATCTL_TXSSEL1_N_SHIFT)) & SPI_TXDATCTL_TXSSEL1_N_MASK)
```

TXSSEL1_N - Transmit Slave Select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register. 0b0..SSEL1 asserted. 0b1..SSEL1 not asserted.

6.4.16.3.2.23 SPI_TXDATCTL_TXSSEL2_N

```
#define SPI_TXDATCTL_TXSSEL2_N (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_TXDATCTL_TXSSEL2_N_SHIFT)) & SPI_TXDATCTL_TXSSEL2_N_MASK)
```

TXSSEL2_N - Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register. 0b0..SSEL2 asserted. 0b1..SSEL2 not asserted.

6.4.16.3.2.24 SPI_TXDATCTL_TXSSEL3_N

```
#define SPI_TXDATCTL_TXSSEL3_N (
    x)
```

Value:

```
((uint32_t)((uint32_t)(x) << SPI_TXDATCTL_TXSSEL3_N_SHIFT)) & SPI_TXDATCTL_TXSSEL3_N_MASK)
```

TXSSEL3_N - Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register. 0b0..SSEL3 asserted. 0b1..SSEL3 not asserted.

Chapter 7

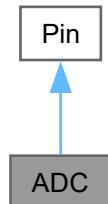
Class Documentation

7.1 ADC Class Reference

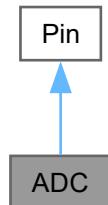
Clase del objeto **ADC** FUNCIONAMIENTO: La clase **ADC** utiliza el **ADCGroup** para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del **ADC** existen. La configuración de conversión se realiza automáticamente con el primer objeto **ADC** creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

```
#include <ADC.h>
```

Inheritance diagram for ADC:



Collaboration diagram for ADC:



Public Member Functions

- `ADC (uint8_t _channel, uint32_t _clk_freq=0, uint32_t _sample_rate=0)`
Default constructor of an ADC.
- `int32_t Get (void)`
Devuelve el valor del ADC.
- `bool IsResultReady (void)`
Indica si la conversión análoga->digital fue terminada.
- `void Trigger (void)`
Ejecuta una conversión analógica->digital.
- `void Initialize (void)`
Inicializador del ADC.

Public Member Functions inherited from Pin

- `Pin (port_t port, uint8_t bit)`
Constructor de clase PIN.

Additional Inherited Members

Public Types inherited from Pin

- enum `port_t`
- enum `max_bits_port_t`
máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Public Attributes inherited from Pin

- const `port_t m_port`
- const `uint8_t m_bit`
- `int8_t m_error`

7.1.1 Detailed Description

Clase del objeto ADC FUNCIONAMIENTO: La clase `ADC` utiliza el `ADCGroup` para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del `ADC` existen. La configuración de conversión se realiza automáticamente con el primer objeto `ADC` creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ADC()

```
ADC::ADC (
    uint8_t _channel,
    uint32_t _clk_freq = 0,
    uint32_t _sample_rate = 0)
```

Default constructor of an ADC.

Crea un `ADC` en el canal correspondiente. La frecuencia y muestreo solo son necesarios de colocar en el primer `ADC` construido ya que el periférico utiliza todos con la misma configuración

Parameters

in	<code>_channel</code>	Canal del ADC .
in	<code>_clk_freq</code>	Frecuencia de clock del periférico ADC .
in	<code>_sample_rate</code>	Frecuencia de muestreo del periférico ADC .

7.1.3 Member Function Documentation**7.1.3.1 Get()**

```
int32_t ADC::Get (
    void )
```

Devuelve el valor del [ADC](#).

Returns

Valor convertido del [ADC](#).

7.1.3.2 Initialize()

```
void ADC::Initialize (
    void )
```

Inicializador del [ADC](#).

Agrega el canal al barrido de lecturas del [ADC](#).

7.1.3.3 IsResultReady()

```
bool ADC::IsResultReady (
    void )
```

Indica si la conversión análoga->digital fue terminada.

Returns

bool: verdadero si el resultado se encuentra, falso sino.

7.1.3.4 Trigger()

```
void ADC::Trigger (
    void )
```

Ejecuta una conversión analógica->digital.

El [ADC](#) empieza la conversión de todos los canales activos.

The documentation for this class was generated from the following files:

- [ADC.h](#)
- [ADC.cpp](#)

7.2 ADC_Group Class Reference

Clase del objeto [ADC_Group](#) FUNCIONAMIENTO: Solo debe crearse UN objeto [ADC](#). Todos los canales y distintos [ADC](#) son manejados por el mismo objeto. Realiza un barrido y guarda a todos los [ADC](#) en un vector de resultados.

```
#include <ADCGroup.h>
```

Public Types

- enum [adc_isr](#)
- enum [irq_source_inten](#)
- enum [error_t](#)
- typedef enum [ADC_Group::adc_isr](#) [adc_isr](#)
- typedef enum [ADC_Group::irq_source_inten](#) [irq_source_inten](#)
- typedef enum [ADC_Group::error_t](#) [error_t](#)

Public Member Functions

- [ADC_Group \(uint32_t clk_freq, uint32_t sample_rate, bool init_channel0=false\)](#)
Constructor de clase [ADC](#).
- virtual [~ADC_Group \(\)=default](#)
- void [Iniciar \(void\)](#)
Inicializa el [ADC](#) en la secuencia A.
- void [SetLowPowerMode \(bool low_power\)](#)
Activa/desactiva el modo de bajo consumo.
- void [SetSampleRate \(void\)](#)
Setea la frecuencia de muestreo.
- void [EnableIrq \(irq_source_inten irq\)](#)
Habilita la interrupcion.
- void [DisableIrq \(irq_source_inten irq\)](#)
Deshabilita la interrupcion.
- [ADC_Group::error_t InitADCChanel \(uint8_t channel\)](#)
Inicializa el canal indicado con la secuencia A.
- [ADC_Group::error_t RemoveADCChanel \(uint8_t channel\)](#)
Desconfigura el canal indicado de la secuencia A del [ADC](#).
- void [TriggerStartSeqA \(void\)](#)
Inicia la conversión analógica - digital de la secuencia A.
- [int32_t GetValue \(uint8_t channel\)](#)
Devuelve el valor de conversión guardado en el buffer. NO EL ACTUAL DEL REGISTRO.
- bool [IsResultReady \(uint8_t channel\) const](#)
Indica si la conversión de channel está terminada o no.
- void [Handler \(adc_isr isr\)](#)
Handler del [ADC](#).

7.2.1 Detailed Description

Clase del objeto [ADC_Group](#) FUNCIONAMIENTO: Solo debe crearse UN objeto [ADC](#). Todos los canales y distintos [ADC](#) son manejados por el mismo objeto. Realiza un barrido y guarda a todos los [ADC](#) en un vector de resultados.

7.2.2 Member Typedef Documentation

7.2.2.1 adc_isr

```
typedef enum ADC_Group::adc_isr ADC_Group::adc_isr
```

Tipo de interrupciones del [ADC](#)

7.2.2.2 error_t

```
typedef enum ADC_Group::error_t ADC_Group::error_t
```

Error en la clase [ADC](#)

7.2.2.3 irq_source_inten

```
typedef enum ADC_Group::irq_source_inten ADC_Group::irq_source_inten
```

Fuente de la interrupcion del [ADC](#)

7.2.3 Member Enumeration Documentation

7.2.3.1 adc_isr

```
enum ADC_Group::adc_isr
```

Tipo de interrupciones del [ADC](#)

7.2.3.2 error_t

```
enum ADC_Group::error_t
```

Error en la clase [ADC](#)

7.2.3.3 irq_source_inten

```
enum ADC_Group::irq_source_inten
```

Fuente de la interrupcion del [ADC](#)

7.2.4 Constructor & Destructor Documentation

7.2.4.1 ADC_Group()

```
ADC_Group::ADC_Group (
    uint32_t clk_freq,
    uint32_t sample_rate,
    bool init_channel0 = false)
```

Constructor de clase [ADC](#).

Crea un [ADC](#) con los parámetros correspondientes

Parameters

in	<i>clk_freq</i>	Frecuencia del periférico ADC .
in	<i>sample_rate</i>	Frecuencia de muestreo del ADC .
in	<i>init_channel0</i>	Si se inicia el canal 0 o no.

7.2.4.2 ~ADC_Group()

```
virtual ADC_Group::~ADC_Group () [virtual], [default]
```

Destructor por defecto

7.2.5 Member Function Documentation**7.2.5.1 DisableIrq()**

```
void ADC_Group::DisableIrq (
    irq_source_inten irq)
```

Deshabilita la interrupcion.

Parameters

in	<i>irq</i>	Interrupcion a deshabilitar
----	------------	-----------------------------

7.2.5.2 EnableIrq()

```
void ADC_Group::EnableIrq (
    irq_source_inten irq)
```

Habilita la interrupcion.

Parameters

in	<i>irq</i>	Interrupcion a habilitar
----	------------	--------------------------

7.2.5.3 GetValue()

```
int32_t ADC_Group::GetValue (
    uint8_t channel)
```

Devuelve el valor de conversión guardado en el buffer. NO EL ACTUAL DEL REGISTRO.

Parameters

in	<i>channel</i>	Canal a obtener la conversión
----	----------------	-------------------------------

Returns

valor de la conversión

7.2.5.4 Handler()

```
void ADC_Group::Handler (
    adc_isr isr)
```

Handler del [ADC](#).

Funcion Handler de todas las interrupciones posibles del [ADC](#).

Parameters

in	<i>isr</i>	Tipo de interrupción
----	------------	----------------------

7.2.5.5 InitADCChanel()

```
ADC_Group::error_t ADC_Group::InitADCChanel (
    uint8_t channel)
```

Inicializa el canal indicado con la secuencia A.

Parameters

in	<i>channel</i>	Canal a inicializar
----	----------------	---------------------

Returns

mensaje de error

7.2.5.6 IsResultReady()

```
bool ADC_Group::IsResultReady (
    uint8_t channel) const
```

Indica si la conversión de channel está terminada o no.

Parameters

in	<i>channel</i>	Canal a preguntar.
----	----------------	--------------------

Returns

verdadero = conversión lista. falso = conversión en proceso

7.2.5.7 RemoveADCChanel()

```
ADC_Group::error_t ADC_Group::RemoveADCChanel (
    uint8_t channel)
```

Desconfigura el canal indicado de la secuencia A del [ADC](#).

Parameters

in	<i>channel</i>	Canal a desconfigurar
----	----------------	-----------------------

Returns

mensaje de error

7.2.5.8 SetLowPowerMode()

```
void ADC_Group::SetLowPowerMode (
    bool low_power)
```

Activa/desactiva el modo de bajo consumo.

Configura el [ADC](#) para no consumir energía al no realizar acciones o sí consumir.

Parameters

in	<i>low_power</i>	Valor de seteo del bajo consumo
----	------------------	---------------------------------

The documentation for this class was generated from the following files:

- [ADCGroup.h](#)
- [ADCGroup.cpp](#)

7.3 ADC_Type Struct Reference

Structure type to access the Analog Digital Convertor ([ADC](#)).

```
#include <LPC845.h>
```

Public Attributes

- [__RW uint32_t CTRL](#)
- [uint8_t RESERVED_0 \[4\]](#)
- [__RW uint32_t SEQ_CTRL \[2\]](#)
- [__R uint32_t SEQ_GDAT \[2\]](#)
- [uint8_t RESERVED_1 \[8\]](#)
- [__R uint32_t DAT \[12\]](#)
- [__RW uint32_t THR0_LOW](#)
- [__RW uint32_t THR1_LOW](#)
- [__RW uint32_t THR0_HIGH](#)
- [__RW uint32_t THR1_HIGH](#)
- [__RW uint32_t CHAN_THRSEL](#)
- [__RW uint32_t INTEN](#)
- [__RW uint32_t FLAGS](#)
- [__RW uint32_t TRM](#)

7.3.1 Detailed Description

Structure type to access the Analog Digital Convertos ([ADC](#)).

[ADC](#) - Register Layout Typedef

7.3.2 Member Data Documentation

7.3.2.1 CHAN_THRSEL

```
__RW uint32_t ADC_Type::CHAN_THRSEL
```

[ADC](#) Channel-Threshold Select register. Specifies which set of threshold compare registers are to be used for each channel, offset: 0x60

7.3.2.2 CTRL

```
__RW uint32_t ADC_Type::CTRL
```

[ADC](#) Control register. Contains the clock divide value, resolution selection, sampling time selection, and mode controls., offset: 0x0

7.3.2.3 DAT

```
__R uint32_t ADC_Type::DAT[12]
```

[ADC](#) Channel N Data register. This register contains the result of the most recent conversion completed on channel N., array offset: 0x20, array step: 0x4

7.3.2.4 FLAGS

```
__RW uint32_t ADC_Type::FLAGS
```

[ADC](#) Flags register. Contains the four interrupt/DMA trigger flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers),, offset: 0x68

7.3.2.5 INTEN

```
__RW uint32_t ADC_Type::INTEN
```

[ADC](#) Interrupt Enable register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated., offset: 0x64

7.3.2.6 RESERVED_0

```
uint8_t ADC_Type::RESERVED_0[4]
```

RESERVED. NOT USED

7.3.2.7 RESERVED_1

`uint8_t ADC_Type::RESERVED_1[8]`

RESERVED. NOT USED

7.3.2.8 SEQ_CTRL

`__RW uint32_t ADC_Type::SEQ_CTRL[2]`

ADC Conversion Sequence-n control register: Controls triggering and channel selection for conversion sequence-n.
Also specifies interrupt mode for sequence-n., array offset: 0x8, array step: 0x4

7.3.2.9 SEQ_GDAT

`__R uint32_t ADC_Type::SEQ_GDAT[2]`

ADC Sequence-n Global Data register. This register contains the result of the most recent **ADC** conversion performed under sequence-n., array offset: 0x10, array step: 0x4

7.3.2.10 THR0_HIGH

`__RW uint32_t ADC_Type::THR0_HIGH`

ADC High Compare Threshold register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0., offset: 0x58

7.3.2.11 THR0_LOW

`__RW uint32_t ADC_Type::THR0_LOW`

ADC Low Compare Threshold register 0: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0., offset: 0x50

7.3.2.12 THR1_HIGH

`__RW uint32_t ADC_Type::THR1_HIGH`

ADC High Compare Threshold register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1., offset: 0x5C

7.3.2.13 THR1_LOW

`__RW uint32_t ADC_Type::THR1_LOW`

ADC Low Compare Threshold register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1., offset: 0x54

7.3.2.14 TRM

`__RW uint32_t ADC_Type::TRM`

`ADC` Startup register., offset: 0x6C

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

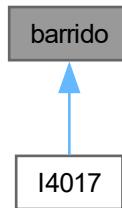
- [LPC845.h](#)

7.4 barrido Class Reference

Clase del objeto barrido Clase abstracta pura para la generación de barridos.

```
#include <Barrido.h>
```

Inheritance diagram for barrido:



Public Member Functions

- virtual void [SetDigito](#) (void)=0
- virtual void [Initialize](#) (void)=0

7.4.1 Detailed Description

Clase del objeto barrido Clase abstracta pura para la generación de barridos.

7.4.2 Member Function Documentation

7.4.2.1 Initialize()

```
virtual void barrido::Initialize (  
    void ) [pure virtual]
```

Funcion de inicializacion

Implemented in [I4017](#).

7.4.2.2 SetDigito()

```
virtual void barrido::SetDigito (
    void ) [pure virtual]
```

constructor por defecto Funcion de encendido del barrido

Implemented in [I4017](#).

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

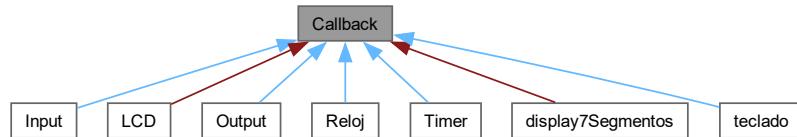
- [Barrido.h](#)

7.5 Callback Class Reference

Clase del objeto [Callback](#).

```
#include <Callback.h>
```

Inheritance diagram for Callback:



Public Member Functions

- void [SetInterrupt \(\)](#)
Activa la interrupción.
- void [UnSetInterrupt \(\)](#)
Desactiva la interrupción.
- virtual void [SWhandler \(void\)=0](#)

7.5.1 Detailed Description

Clase del objeto [Callback](#).

El objeto [Callback](#) debe ser heredado y otorga a los herederos la conexión al systick y a la interrupción temporizada

7.5.2 Member Function Documentation

7.5.2.1 SetInterrupt()

```
void Callback::SetInterrupt (
    void )
```

Activa la interrupción.

Coloca al objeto dentro de la lista del handler systick.

7.5.2.2 SWhandler()

```
virtual void Callback::SWhandler (
    void ) [pure virtual]
```

Este método debe ser implementado por las clases derivadas cada una resolverá que hacer con su irq enganchada al systick del sistema

Implemented in [display7Segmentos](#), [Input](#), [LCD](#), [Output](#), [Reloj](#), [teclado](#), and [Timer](#).

7.5.2.3 UnSetInterrupt()

```
void Callback::UnSetInterrupt (
    void )
```

Desactiva la interrupción.

Borra de la lista handler al objeto actual.

The documentation for this class was generated from the following files:

- [Callback.h](#)
- [Callback.cpp](#)

7.6 CircularBuffer< T > Class Template Reference

Clase del objeto [CircularBuffer](#).

```
#include <CircularBuffer.h>
```

Public Member Functions

- [CircularBuffer \(uint32_t size\)](#)
Constructor del buffer circular.
- [bool pop \(T *item\)](#)
Saca un valor del buffer.
- [void push \(T item\)](#)
Envía un item al buffer.
- [bool isFull \(void\) const](#)
Indica si el buffer esta lleno.
- [virtual ~CircularBuffer \(\)](#)
Elimina el buffer.

7.6.1 Detailed Description

```
template<typename T>
class CircularBuffer< T >
```

Clase del objeto [CircularBuffer](#).

Objeto extra de buffer circular. No posee memoria de la cantidad de datos ingresados.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 CircularBuffer()

```
template<typename T>
CircularBuffer< T >::CircularBuffer (
    uint32_t size) [inline]
```

Constructor del buffer circular.

Crea un buffer circular tipo FIFO con el tamaño indicado.

Parameters

in	size	Tamaño del buffer.
----	------	--------------------

7.6.3 Member Function Documentation

7.6.3.1 isFull()

```
template<typename T>
bool CircularBuffer< T >::isFull (
    void) const [inline]
```

Indica si el buffer está lleno.

Returns

bool: true si está lleno (o totalmente vacío), false si no.

7.6.3.2 pop()

```
template<typename T>
bool CircularBuffer< T >::pop (
    T * item) [inline]
```

Saca un valor del buffer.

Saca el valor más antiguo del buffer. Si no hay valores devuelve false.

Parameters

<code>in, out</code>	<code>item</code>	Puntero al item que se obtendra.
----------------------	-------------------	----------------------------------

Returns

`bool`: true si habia items, false si no.

7.6.3.3 push()

```
template<typename T>
void CircularBuffer< T >::push (
    T item) [inline]
```

Envia un item al buffer.

Envia un item al buffer. No comprueba que se llene.

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

- [CircularBuffer.h](#)

7.7 ComunicacionAsincronica Class Reference

Clase del objeto [ComunicacionAsincronica](#) Clase abstracta pura para la generación de UART.

```
#include <ComunicacionAsincronica.h>
```

Public Member Functions

- [ComunicacionAsincronica \(\)=default](#)
- virtual void [Write \(const char *msg\)=0](#)
- virtual void [Write \(const void *msg, uint32_t n\)=0](#)
- virtual bool [Read \(char *msg, uint32_t n\)=0](#)
- virtual void [UART_IRQHandler \(void\)=0](#)
- virtual [~ComunicacionAsincronica \(\)=default](#)

7.7.1 Detailed Description

Clase del objeto [ComunicacionAsincronica](#) Clase abstracta pura para la generación de UART.

7.7.2 Constructor & Destructor Documentation**7.7.2.1 ComunicacionAsincronica()**

```
ComunicacionAsincronica::ComunicacionAsincronica () [default]
```

Constructor por defecto

7.7.2.2 ~ComunicacionAsincronica()

```
virtual ComunicacionAsincronica::~ComunicacionAsincronica () [virtual], [default]
```

Destructor por defecto

7.7.3 Member Function Documentation

7.7.3.1 Read()

```
virtual bool ComunicacionAsincronica::Read (
    char * msg,
    uint32_t n) [pure virtual]
```

Funcion de lectura

Implemented in [ESP8266](#).

7.7.3.2 UART_IRQHandler()

```
virtual void ComunicacionAsincronica::UART_IRQHandler (
    void) [pure virtual]
```

Funcion de interrupcion

7.7.3.3 Write() [1/2]

```
virtual void ComunicacionAsincronica::Write (
    const char * msg) [pure virtual]
```

Funcion de transmitir

Implemented in [ESP8266](#).

7.7.3.4 Write() [2/2]

```
virtual void ComunicacionAsincronica::Write (
    const void * msg,
    uint32_t n) [pure virtual]
```

Funcion de transmitir

Implemented in [ESP8266](#).

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

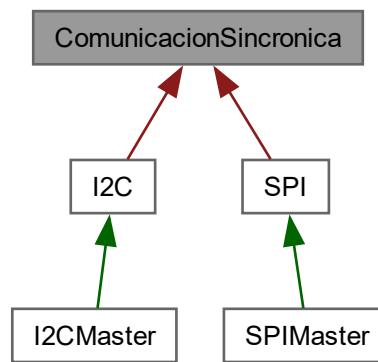
- [ComunicacionAsincronica.h](#)

7.8 ComunicacionSincronica Class Reference

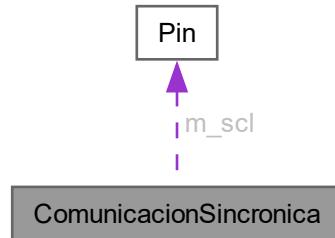
Clase del objeto [ComunicacionAsincronica](#) Clase abstracta pura para la generación de comunicaciones sincrónicas como la [I2C](#) o la [SPI](#).

```
#include <ComunicacionSincronica.h>
```

Inheritance diagram for ComunicacionSincronica:



Collaboration diagram for ComunicacionSincronica:



Public Member Functions

- [ComunicacionSincronica \(\)=default](#)
- [virtual void Write \(uint8_t data\)=0](#)
- [virtual int8_t Read \(uint8_t *data\)=0](#)
- [virtual ~ComunicacionSincronica \(\)=default](#)

Protected Attributes

- const `Pin * m_scl`

7.8.1 Detailed Description

Clase del objeto `ComunicacionAsincronica` Clase abstracta pura para la generación de comunicaciones sincrónicas como la `I2C` o la `SPI`.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 `ComunicacionSincronica()`

```
ComunicacionSincronica::ComunicacionSincronica () [default]
```

Constructor por defecto

7.8.2.2 `~ComunicacionSincronica()`

```
virtual ComunicacionSincronica::~ComunicacionSincronica () [virtual], [default]
```

Destructor por defecto

7.8.3 Member Function Documentation

7.8.3.1 `Read()`

```
virtual int8_t ComunicacionSincronica::Read (
    uint8_t * data) [pure virtual]
```

Funcion de lectura

Implemented in `I2C`, and `SPI`.

7.8.3.2 `Write()`

```
virtual void ComunicacionSincronica::Write (
    uint8_t data) [pure virtual]
```

Funcion de escritura

Implemented in `I2C`, and `SPI`.

7.8.4 Member Data Documentation

7.8.4.1 m_scl

```
const Pin* ComunicacionSincronica::m_scl [protected]
```

Pin de Clock. Debe existir en toda comunicacion sincronica

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

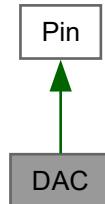
- [ComunicacionSincronica.h](#)

7.9 DAC Class Reference

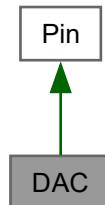
Clase del objeto **DAC** FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

```
#include <DAC.h>
```

Inheritance diagram for DAC:



Collaboration diagram for DAC:



Public Types

- enum `dac_channel`
- enum `dac_error`
- typedef enum `DAC::dac_channel dac_channel`
- typedef enum `DAC::dac_error dac_error`

Public Member Functions

- `DAC (dac_channel channel, uint32_t max_range=MAX_DAC_VALUE)`
Constructor de clase DAC.
- `DAC::dac_error Initialize (void)`
Inicializa el DAC.
- `void Set (uint32_t val)`
Setea el valor del DAC.
- `uint32_t Get (void) const`
Devuelve el valor analógico del DAC.
- `void SetMaxRange (uint32_t max_range)`
Setea el rango máximo.
- `uint32_t GetMaxRange (void) const`
Devuelve el rango máximo que posee el DAC.
- `DAC & operator= (uint32_t val)`
Sobrecarga del operador =.
- `bool operator== (uint32_t val) const`
Sobrecarga del operador ==.
- `bool operator< (uint32_t val) const`
Sobrecarga del operador <.
- `bool operator<= (uint32_t val) const`
Sobrecarga del operador <=.
- `bool operator> (uint32_t val) const`
Sobrecarga del operador >.
- `bool operator>= (uint32_t val) const`
Sobrecarga del operador >=.
- `bool operator!= (uint32_t val) const`
Sobrecarga del operador !=.
- `virtual ~DAC ()`

Additional Inherited Members

Protected Types inherited from Pin

- enum `port_t`
- enum `max_bits_port_t`
máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Protected Member Functions inherited from Pin

- `Pin (port_t port, uint8_t bit)`
Constructor de clase PIN.

Protected Attributes inherited from Pin

- `const port_t m_port`
- `const uint8_t m_bit`
- `int8_t m_error`

7.9.1 Detailed Description

Clase del objeto `DAC` FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta `max_range`. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

7.9.2 Member Typedef Documentation

7.9.2.1 dac_channel

```
typedef enum DAC::dac_channel DAC::dac_channel
```

Canales del `DAC`

7.9.2.2 dac_error

```
typedef enum DAC::dac_error DAC::dac_error
```

Error en la clase `DAC`

7.9.3 Member Enumeration Documentation

7.9.3.1 dac_channel

```
enum DAC::dac_channel
```

Canales del `DAC`

7.9.3.2 dac_error

```
enum DAC::dac_error
```

Error en la clase `DAC`

7.9.4 Constructor & Destructor Documentation

7.9.4.1 DAC()

```
DAC::DAC (
    dac_channel channel,
    uint32_t max_range = MAX_DAC_VALUE)
```

Constructor de clase `DAC`.

Crea un `DAC` con los parámetros correspondientes.

Parameters

in	<i>channel</i>	Canal del DAC a utilizar.
in	<i>max_range</i>	Rango máximo del usuario.

7.9.4.2 ~DAC()

```
DAC::~DAC () [virtual]
```

Destructor por defecto

7.9.5 Member Function Documentation**7.9.5.1 Get()**

```
uint32_t DAC::Get (
    void ) const
```

Devuelve el valor analógico del [DAC](#).

Devuelve el valor de la salida [DAC](#) de acuerdo al rango del usuario.

Returns

valor del [DAC](#).

7.9.5.2 GetMaxRange()

```
uint32_t DAC::GetMaxRange (
    void ) const
```

Devuelve el rango máximo que posee el [DAC](#).

Devuelve el rango que el usuario se asignó para trabajar.

Returns

rango máximo.

7.9.5.3 Initialize()

```
DAC::dac_error DAC::Initialize (
    void )
```

Inicializa el [DAC](#).

Configura todos los registros de dicho [DAC](#).

Returns

Código de error.

7.9.5.4 operator"!=()

```
bool DAC::operator!= (
    uint32_t val) const
```

Sobrecarga del operador !=.

Devuelve verdadero o falso si el [DAC](#) es distinto a val

Parameters

in	val	Valor de comparacion con la salida
----	-----	------------------------------------

Returns

verdadero o falso

7.9.5.5 operator<()

```
bool DAC::operator< (
    uint32_t val) const
```

Sobrecarga del operador <.

Devuelve verdadero o falso si el [DAC](#) es menor a val.

Parameters

in	val	Valor de comparacion con la salida.
----	-----	-------------------------------------

Returns

verdadero o falso.

7.9.5.6 operator<=()

```
bool DAC::operator<= (
    uint32_t val) const
```

Sobrecarga del operador <=.

Devuelve verdadero o falso si el [DAC](#) es menor o igual a val.

Parameters

in	val	Valor de comparacion con la salida.
----	-----	-------------------------------------

Returns

verdadero o falso.

7.9.5.7 operator=()

```
DAC & DAC::operator= (
    uint32_t val)
```

Sobrecarga del operador =.

Fija el valor analógico del [DAC](#) a val.

Parameters

in	val	Valor de seteo del DAC .
----	-----	--

Returns

Referencia a si mismo.

7.9.5.8 operator==()

```
bool DAC::operator== (
    uint32_t val) const
```

Sobrecarga del operador ==.

Devuelve verdadero o falso si el [DAC](#) es igual a val.

Parameters

in	val	Valor de comparacion con la salida.
----	-----	-------------------------------------

Returns

verdadero o falso.

7.9.5.9 operator>()

```
bool DAC::operator> (
    uint32_t val) const
```

Sobrecarga del operador >.

Devuelve verdadero o falso si el [DAC](#) es mayor a val.

Parameters

in	val	Valor de comparacion con la salida.
----	-----	-------------------------------------

Returns

verdadero o falso.

7.9.5.10 operator>=()

```
bool DAC::operator>= (
    uint32_t val) const
```

Sobrecarga del operador >=.

Devuelve verdadero o falso si el [DAC](#) es mayor o igual a val

Parameters

in	val	Valor de comparacion con la salida
----	-----	------------------------------------

Returns

verdadero o falso

7.9.5.11 Set()

```
void DAC::Set (uint32_t val)
```

Setea el valor del [DAC](#).

Enciende la salida analógica con el valor en rango real de voltaje.

Parameters

in	val	Valor de seteo de la salida.
----	-----	------------------------------

7.9.5.12 SetMaxRange()

```
void DAC::SetMaxRange (uint32_t max_range)
```

Setea el rango máximo.

Setea el rango de valores a utilizar por el usuario.

Parameters

in	max_range	Valor del rango.
----	-----------	------------------

The documentation for this class was generated from the following files:

- [DAC.h](#)
- [DAC.cpp](#)

7.10 DAC_t Struct Reference

Structure type to access the Digital Analog Convertor ([DAC](#)).

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t CNTVAL`

7.10.1 Detailed Description

Structure type to access the Digital Analog Convertor ([DAC](#)).

[DAC](#) - Register Layout Typedef

7.10.2 Member Data Documentation

7.10.2.1 BIAS

`__RW uint32_t DAC_t::BIAS`

The settling times noted in the description of the BIAS bit are valid for a capacitance load on the DAC_OUT pin not exceeding 100 pF

7.10.2.2 CNT_ENA

`__RW uint32_t DAC_t::CNT_ENA`

Time-out counter operation. 0=Disable. 1=Enable

7.10.2.3 CNTVAL

`__RW uint32_t DAC_t::CNTVAL`

16-bit reload value for the [DAC](#) interrupt/DMA timer.

7.10.2.4 CR

`__RW uint32_t DAC_t::CR`

This read/write register includes the digital value to be converted to analog, and a bit that trades off performance vs. power.

7.10.2.5 CTRL

`__RW uint32_t DAC_t::CTRL`

This read/write register enables the DMA operation and controls the DMA timer

7.10.2.6 DBLBUF_ENA

`__RW uint32_t DAC_t::DBLBUF_ENA`

Double buffering. 0=Disable. 1=Enable

7.10.2.7 DMA_ENA

`__RW uint32_t DAC_t::DMA_ENA`

Enable. DMA Burst Request [Input 7](#) is enabled for the [DAC](#)

7.10.2.8 INT_DMA_REQ

`__RW uint32_t DAC_t::INT_DMA_REQ`

DMA interrupt request. 0=Clear on any write to the CR register. 1=Set by hardware

7.10.2.9 RESERVED0

`__RW uint32_t DAC_t::RESERVED0`

RESERVED. NOT USED

7.10.2.10 RESERVED1

`__RW uint32_t DAC_t::RESERVED1`

RESERVED. NOT USED

7.10.2.11 RESERVED2

`__RW uint32_t DAC_t::RESERVED2`

RESERVED. NOT USED

7.10.2.12 VALUE

`__RW uint32_t DAC_t::VALUE`

After the selected settling time after this field is written with a new VALUE, the voltage on the DAC_OUT pin (with respect to VSSA) is $\text{VALUE} = ((\text{VREFP})/1024)$

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

- [LPC845.h](#)

7.11 digito Class Reference

Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.

```
#include <Digito.h>
```

Public Types

- enum `codigo_t`
- enum `modo_t`
- enum `SIMBOLOS`

Public Member Functions

- `digito (codigo_t Sistema=BCD, uint8_t Valor=APAGAR)`
Función de escritura del dígito.
- `bool Set (uint16_t valor)`
Devuelve el valor del dígito.
- `uint8_t Get (void)`
Función de limpieza del dígito.
- `void Clear (void)`
Función de limpieza del dígito.

7.11.1 Detailed Description

Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.

7.11.2 Member Enumeration Documentation

7.11.2.1 `codigo_t`

```
enum digito::codigo_t
```

Tipos de dígitos

7.11.2.2 `modo_t`

```
enum digito::modo_t
```

Modos de uso

7.11.2.3 `SIMBOLOS`

```
enum digito::SIMBOLOS
```

Simbolos a escribir

7.11.3 Constructor & Destructor Documentation

7.11.3.1 digito()

```
digito::digito (
    codigo_t Sistema = BCD,
    uint8_t Valor = APAGAR) [inline]
```

constructor por defecto

7.11.4 Member Function Documentation

7.11.4.1 Clear()

```
void digito::Clear (
    void )
```

Función de limpieza del dígito.

Deja el dígito apagado del todo. No confundir con escribir un 0, apagado representa sin energía

7.11.4.2 Get()

```
uint8_t digito::Get (
    void )
```

Devuelve el valor del dígito.

Returns

valor del dígito.

7.11.4.3 Set()

```
bool digito::Set (
    uint16_t valor)
```

Función de escritura del dígito.

Escribe el valor del dígito dependiendo de su sistema.

Parameters

in	valor	valor a escribir.
----	-------	-------------------

Returns

bool. True si tuvo éxito, false sino.

The documentation for this class was generated from the following files:

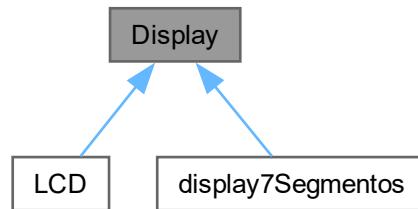
- [Digito.h](#)
- [Digito.cpp](#)

7.12 Display Class Reference

Clase del objeto display Clase abstracta pura para la generación de displays.

```
#include <Display.h>
```

Inheritance diagram for Display:



Public Member Functions

- `Display ()=default`
- `virtual void Write (const int32_t n)=0`
- `virtual void Clear (void)=0`
- `virtual ~Display ()=default`

7.12.1 Detailed Description

Clase del objeto display Clase abstracta pura para la generación de displays.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 `Display()`

```
Display::Display () [default]
```

Constructor por defecto

7.12.2.2 `~Display()`

```
virtual Display::~Display () [virtual], [default]
```

destructor por defecto

7.12.3 Member Function Documentation

7.12.3.1 Clear()

```
virtual void Display::Clear (
    void ) [pure virtual]
```

Funcion de limpieza genérica de un display

Implemented in [display7Segmentos](#), and [LCD](#).

7.12.3.2 Write()

```
virtual void Display::Write (
    const int32_t n) [pure virtual]
```

Funcion de escritura genérica de un display

Implemented in [display7Segmentos](#), and [LCD](#).

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

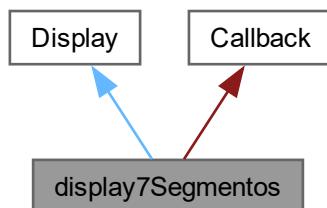
- [Display.h](#)

7.13 display7Segmentos Class Reference

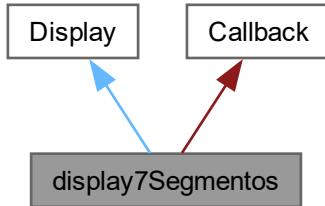
Clase del objeto [display7Segmentos](#) El objeto [display7Segmentos](#) permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor.

```
#include <Display7Segmentos.h>
```

Inheritance diagram for display7Segmentos:



Collaboration diagram for display7Segmentos:



Public Member Functions

- `display7Segmentos (vector< gruposdedigitos * > g, segmentos *s, barrido *b, const uint8_t *Posicion← Relativa, const digito::codigo_t sistema)`
Objeto del tipo display7Segmentos.
- void `SHandler (void) override`
Handler del display de 7 segmentos.
- void `Set (uint32_t valor, uint8_t dsp)`
Función de escritura del display de 7 segmentos.
- void `Write (const int32_t n) override`
Función de escritura del display de 7 segmentos.
- void `Clear (void) override`
Función de limpieza del display de 7 segmentos.
- virtual `~display7Segmentos ()`

Public Member Functions inherited from [Display](#)

- `Display ()=default`
- virtual `~Display ()=default`

7.13.1 Detailed Description

Clase del objeto `display7Segmentos` El objeto `display7Segmentos` permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a `m_ticks`. Para ver mejores resultados modificar dicho valor.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 display7Segmentos()

```
display7Segmentos::display7Segmentos (
    vector< gruposdedigitos * > g,
    segmentos * s,
    barrido * b,
    const uint8\_t * PosicionRelativa,
    const digito::codigo\_t sistema)
```

Objeto del tipo [display7Segmentos](#).

Crea el objeto [display7Segmentos](#) con los parámetros indicados.

Parameters

in	<i>g</i>	Vector con los dígitos a utilizar.
in	<i>s</i>	Segmento a utilizar.
in	<i>b</i>	Dispositivo de barrido a utilizar.
in	<i>PosicionRelativa</i>	Posición relativa del grupo de dígitos.
in	<i>sistema</i>	Sistema a utilizar.

7.13.2.2 ~display7Segmentos()

```
display7Segmentos::~display7Segmentos () [virtual]
```

destructor por defecto

7.13.3 Member Function Documentation

7.13.3.1 Clear()

```
void display7Segmentos::Clear (
    void) [override], [virtual]
```

Función de limpieza del display de 7 segmentos.

Coloca en 0 todos los displays.

Implements [Display](#).

7.13.3.2 Set()

```
void display7Segmentos::Set (
    uint32\_t valor,
    uint8\_t dsp)
```

Función de escritura del display de 7 segmentos.

Parameters

in	<i>valor</i>	valor a escribir.
in	<i>dsp</i>	posición a escribir el valor.

7.13.3.3 SWhandler()

```
void display7Segmentos::SWhandler (
    void ) [override], [virtual]
```

Handler del display de 7 segmentos.

Función interrupción del display de 7 segmentos.

Implements [Callback](#).

7.13.3.4 Write()

```
void display7Segmentos::Write (
    const int32_t n) [override], [virtual]
```

Función de escritura del display de 7 segmentos.

Escribe el display 7 segmentos desde la posición 0.

Parameters

in	<i>n</i>	valor a escribir.
----	----------	-------------------

Implements [Display](#).

The documentation for this class was generated from the following files:

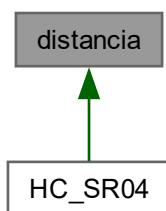
- [Display7Segmentos.h](#)
- [Display7Segmentos.cpp](#)

7.14 distancia Class Reference

Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.

```
#include <distancia.h>
```

Inheritance diagram for distancia:



Public Member Functions

- `distancia ()=default`
- virtual `uint32_t GetDistancia ()=0`
- virtual `bool operator==(uint32_t a)=0`
- virtual `~distancia ()=default`

7.14.1 Detailed Description

Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 `distancia()`

```
distancia::distancia () [default]
```

Constructor por defecto

7.14.2.2 `~distancia()`

```
virtual distancia::~distancia () [virtual], [default]
```

Destructor por defecto

7.14.3 Member Function Documentation

7.14.3.1 `GetDistancia()`

```
virtual uint32_t distancia::GetDistancia () [pure virtual]
```

Obtencion de distancia

Implemented in [HC_SR04](#).

7.14.3.2 `operator==()`

```
virtual bool distancia::operator== (
    uint32_t a) [pure virtual]
```

Consulta sobre la distancia

Implemented in [HC_SR04](#).

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

- `distancia.h`

7.15 ESP8266 Class Reference

Clase del objeto [ESP8266](#) El objeto [ESP8266](#) permite la simple utilización del módulo arduino [ESP8266](#) y el [ESP01](#) mediante comandos AT. El módulo debe estar por defecto en la velocidad [DEFAULT_ESP01_BAUDRATE](#). El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Sí se probó la inicialización y conexión a internet, no se probó la conexión a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver.

```
#include <ESP8266.h>
```

Public Types

- enum [conection_type](#)
- enum [status_type](#)

Public Member Functions

- **[ESP8266](#)** ([Pin::port_t](#) _portTx, [uint8_t](#) _pinTx, [Pin::port_t](#) _portRx, [uint8_t](#) _pinRx, [USART_Type](#) *uart, [uint32_t](#) baudrate)

Default constructor.
- void [Initialize](#) (void)

Inicializa el [ESP8266](#)/ [ESP01](#) con comandos AT en modo cliente.
- [status_type](#) [ConnectToWifi](#) ([const int8_t](#) *wifi_address, [const int8_t](#) *wifi_password, [uint32_t](#) seg_timeout=[SEG_ESP01_TIMEOUT](#))

Se conecta al wifi indicado.
- void [DisconnectToWifi](#) (void)

Se desconecta de la red wifi.
- void [SetIP](#) ([int8_t](#) *ip)

Setea la IP del dispositivo.
- [int8_t](#) * [GetIP](#) (void) const

Devuelve la IP del dispositivo.
- bool [ConnectToServer](#) ([conection_type](#) _mode, [const int8_t](#) *server_ip, [const int8_t](#) *server_port, [uint32_t](#) seg_timeout=[SEG_ESP01_TIMEOUT](#))

Intenta conectarse a un servidor.
- void [DisconnectToServer](#) (void)

Se desconecta del server.
- void [Write](#) ([const char](#) *msg)

Sobrecarga de Transmit de UART.
- void [Write](#) ([const void](#) *msg, [uint32_t](#) n)

Sobrecarga de Transmit de UART.
- bool [Read](#) ([char](#) *msg, [uint32_t](#) n)

Sobrecarga de message de UART.
- [status_type](#) [GetStatus](#) (void) const

Devuelve el estado del dispositivo.
- bool [IsConnectedToWifi](#) (void) const

Pregunta si está conectado a internet o no.
- bool [IsConnectedToServer](#) (void) const

Pregunta si está conectado al server o no.
- virtual ~**[ESP8266](#)** ()

Default destructor.

7.15.1 Detailed Description

Clase del objeto `ESP8266` El objeto `ESP8266` permite la simple utilización del módulo arduino `ESP8266` y el `ESP01` mediante comandos AT. El módulo debe estar por defecto en la velocidad `DEFAULT_ESP01_BAUDRATE`. El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Sí se probó la inicialización y conexión a internet, no se probó la conexión a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver.

7.15.2 Member Enumeration Documentation

7.15.2.1 `conection_type`

```
enum ESP8266::conection_type
```

Tipo de conexión del cliente wifi

7.15.2.2 `status_type`

```
enum ESP8266::status_type
```

Enumeración con todos los estados posibles de la comunicación

7.15.3 Member Function Documentation

7.15.3.1 `ConnectToServer()`

```
bool ESP8266::ConnectToServer (
    conection_type _mode,
    const int8_t * server_ip,
    const int8_t * server_port,
    uint32_t seg_timeout = SEG_ESP01_TIMEOUT)
```

Intenta conectarse a un servidor.

Envía los comandos AT para conectarse a un servidor. Función bloqueante, tiene un time out en caso de fallar.

Parameters

in	<code>_mode</code>	Tipo de conexión UDP/TCP.
in	<code>server_ip</code>	IP/nombre del servidor.
in	<code>server_port</code>	Puerto al que conectarse del servidor.
in	<code>seg_timeout</code>	Tiempo del timeout. Default de 10 segundos.

Returns

`bool`: true si logró conectarse; false si no

7.15.3.2 `ConnectToWifi()`

```
ESP8266::status_type ESP8266::ConnectToWifi (
    const int8_t * wifi_address,
    const int8_t * wifi_password,
    uint32_t seg_timeout = SEG_ESP01_TIMEOUT)
```

Se conecta al wifi indicado.

Explicacion detallada

Parameters

in	<i>wifi_address</i>	Nombre de la red wifi.
in	<i>wifi_password</i>	Nombre de la clave de la red wifi.
in	<i>seg_timeout</i>	Tiempo hasta considerar fallo en la conexion.

Returns

`status_type` = CONNECT_TO_WIFI (3) si funcione, ERROR (0) sino

7.15.3.3 DisconnectToServer()

```
void ESP8266::DisconnectToServer (
    void )
```

Se desconecta del server.

Envía los comandos AT para desconectarse del servidor en caso de estar conectado a uno.

7.15.3.4 DisconnectWifi()

```
void ESP8266::DisconnectWifi (
    void )
```

Se desconecta de la red wifi.

Envía los comandos AT para desconectarse de la red wifi en caso de estar conectada a una.

7.15.3.5 GetIP()

```
int8_t * ESP8266::GetIP (
    void ) const
```

Devuelve la IP del dispositivo.

Returns

`int8_t*`: IP del dispositivo en formato string

7.15.3.6 GetStatus()

```
ESP8266::status_type ESP8266::GetStatus (
    void ) const
```

Devuelve el estado del dispositivo.

Returns

`ESP8266::status_type`: Estado del dispositivo.

7.15.3.7 Initialize()

```
void ESP8266::Initialize (
    void )
```

Inicializa el ESP8266/ ESP01 con comandos AT en modo cliente.

Configura los baudios y modo del dispositivo. FUNCION BLOQUEANTE

7.15.3.8 IsConnectedToServer()

```
bool ESP8266::IsConnectedToServer (
    void ) const
```

Pregunta si está conectado al server o no.

Returns

bool: true está conectado al server, false no lo esta.

7.15.3.9 IsConnectedToWifi()

```
bool ESP8266::IsConnectedToWifi (
    void ) const
```

Pregunta si está conectado a internet o no.

Returns

bool: true está conectado a internet, false no lo esta.

7.15.3.10 Read()

```
bool ESP8266::Read (
    char * msg,
    uint32_t n) [virtual]
```

Sobrecarga de message de UART.

Parameters

in	<i>msg</i>	Mensaje a leer.
in	<i>n</i>	cantidad de caracteres a leer.

Returns

void*: puntero al mensaje a leer.

Implements [ComunicacionAsincronica](#).

7.15.3.11 SetIP()

```
void ESP8266::SetIP (
    int8_t * ip)
```

Setea la IP del dispositivo.

Envía todos los comandos AT para setear una IP y espera su correcta respuesta

Parameters

in	<i>ip</i>	ip a configurar
----	-----------	-----------------

7.15.3.12 Write() [1/2]

```
void ESP8266::Write (
    const char * msg)  [virtual]
```

Sobrecarga de Transmit de UART.

Parameters

in	<i>msg</i>	Mensaje a enviar.
----	------------	-------------------

Implements [ComunicacionAsincronica](#).

7.15.3.13 Write() [2/2]

```
void ESP8266::Write (
    const void * msg,
    uint32_t n)  [virtual]
```

Sobrecarga de Transmit de UART.

Parameters

in	<i>msg</i>	Mensaje a enviar.
in	<i>n</i>	cantidad de caracteres a enviar.

Implements [ComunicacionAsincronica](#).

The documentation for this class was generated from the following files:

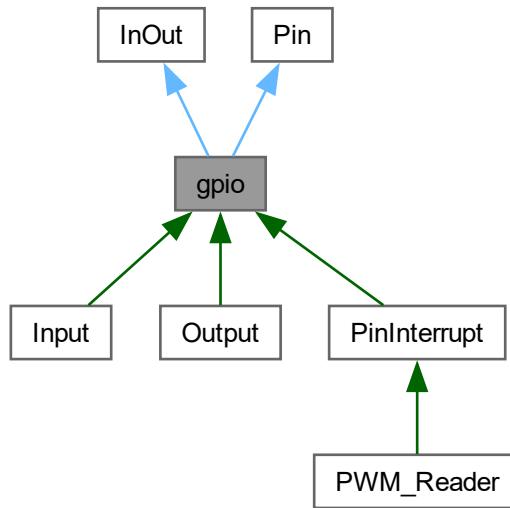
- [ESP8266.h](#)
- [ESP8266.cpp](#)

7.16 gpio Class Reference

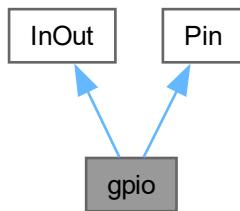
Clase del objeto gpio.

```
#include <gpio.h>
```

Inheritance diagram for gpio:



Collaboration diagram for gpio:



Public Types

- enum [direction_t](#)
Enumeracion de input/output.
- enum [power_t](#)

- enum `mode_t`

Enumeracion de encendido/apagado.
- enum `activity_t`

Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum `interrupt_mode_t`

Enumeracion de activo bajo/alto.
- typedef enum `gpio::direction_t direction_t`

Enumeracion de input/output.
- typedef enum `gpio::mode_t mode_t`

Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum `gpio::activity_t activity_t`

Enumeracion de activo bajo/alto.

Public Types inherited from [Pin](#)

- enum `port_t`
- enum `max_bits_port_t`

máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Public Member Functions

- `gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)`

Constructor de clase GPIO.
- `uint8_t SetPin (void) override`

Enciende la salida.
- `uint8_t ClrPin (void) override`

Limpia el pin.
- `uint8_t SetTogglePin (void) override`

Toggle del pin.
- `uint8_t SetDir (void) override`

Setea la dirección.
- `uint8_t SetToggleDir (void) override`

Toggle de la dirección.
- `uint8_t GetPin (void) override`

Devuelve el valor del pin.
- `uint8_t SetPinMode (void) override`

Configura el modo del pin.
- `uint8_t SetPinResistor (void) override`

Setea la resistencia interna.
- `gpio & operator= (uint8_t a)`

Sobrecarga del operador =.
- virtual `~gpio ()=default`

Public Member Functions inherited from [InOut](#)

- `InOut ()=default`
- virtual `~InOut ()=default`

Public Member Functions inherited from Pin

- `Pin (port_t port, uint8_t bit)`

Constructor de clase PIN.

Protected Attributes

- `const mode_t m_mode`
- `direction_t m_direction`
- `const activity_t m_activity`

Additional Inherited Members

Public Attributes inherited from Pin

- `const port_t m_port`
- `const uint8_t m_bit`
- `int8_t m_error`

7.16.1 Detailed Description

Clase del objeto gpio.

El objeto gpio Permite el manejo de entradas y salidas de uso general

7.16.2 Constructor & Destructor Documentation

7.16.2.1 gpio()

```
gpio::gpio (
    port_t port,
    uint8_t bit,
    mode_t mode,
    direction_t direction,
    activity_t activity = high)
```

Constructor de clase GPIO.

Crea un GPIO con los parámetros correspondientes

Parameters

in	<i>port</i>	Puerto del objeto
in	<i>bit</i>	Bit del objeto
in	<i>mode</i>	Configuración eléctrica del pin
in	<i>direction</i>	Entrada/salida
in	<i>activity</i>	Activo alto/bajo

7.16.2.2 ~gpio()

```
virtual gpio::~gpio () [virtual], [default]
```

Destructor por defecto

7.16.3 Member Function Documentation

7.16.3.1 ClrPin()

```
uint8_t gpio::ClrPin (
    void ) [override], [virtual]
```

Limpia el pin.

Coloca la salida con un 0 lógico

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.2 GetPin()

```
uint8_t gpio::GetPin (
    void ) [override], [virtual]
```

Devuelve el valor del pin.

Entrega el valor lógico de encendido o apagado de la entrada

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.3 operator=()

```
gpio & gpio::operator= (
    uint8_t a)
```

Sobrecarga del operador =.

Enciende o apaga la salida. Solo funciona en modo output

Parameters

in	a	Valor de seteo de la salida
----	---	-----------------------------

Returns

Referencia a si mismo

7.16.3.4 SetDir()

```
uint8_t gpio::SetDir (
    void ) [override], [virtual]
```

Setea la direccion.

Configura la GPIO como entrada o como salida

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.5 SetPin()

```
uint8_t gpio::SetPin (
    void ) [override], [virtual]
```

Enciende la salida.

Coloca la salida con un 1 lógico

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.6 SetPinMode()

```
uint8_t gpio::SetPinMode (
    void ) [override], [virtual]
```

Configura el modo del pin.

Activa el pin en modo PushPull o OpenCollector. SOLO PARA OUTPUTS

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.7 SetPinResistor()

```
uint8_t gpio::SetPinResistor (
    void ) [override], [virtual]
```

Setea la resistencia interna.

Activa el modo Inactive , PullUp , PullDown o Repeater. SOLO PARA INPUTS

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.8 SetToggleDir()

```
uint8_t gpio::SetToggleDir (
    void ) [override], [virtual]
```

Toggle de la dirección.

Pasa de output a input y viceversa

Returns

Devuelve el error

Implements [InOut](#).

7.16.3.9 SetTogglePin()

```
uint8_t gpio::SetTogglePin (
    void ) [override], [virtual]
```

Toggle del pin.

En caso de ser salida. Pasa de encendido a apagado y viceversa.

Returns

Devuelve el error

Implements [InOut](#).

7.16.4 Member Data Documentation

7.16.4.1 m_activity

```
const activity_t gpio::m_activity [protected]
```

Si es activo alto o activo bajo

7.16.4.2 m_direction

```
direction_t gpio::m_direction [protected]
```

Si es input u output

7.16.4.3 m_mode

```
const mode_t gpio::m_mode [protected]
```

La config de resistencia y actividad

The documentation for this class was generated from the following files:

- [gpio.h](#)
- [gpio.cpp](#)

7.17 GPIO_Type Struct Reference

Structure type to access the General Porpose [Input Output](#) (GPIO).

```
#include <LPC845.h>
```

Public Attributes

- [__RW uint8_t B \[2\]\[32\]](#)
- [uint8_t RESERVED_0 \[4032\]](#)
- [__RW uint32_t W \[2\]\[32\]](#)
- [uint8_t RESERVED_1 \[3840\]](#)
- [__RW uint32_t DIR \[2\]](#)
- [uint8_t RESERVED_2 \[120\]](#)
- [__RW uint32_t MASK \[2\]](#)
- [uint8_t RESERVED_3 \[120\]](#)
- [__RW uint32_t PIN \[2\]](#)
- [uint8_t RESERVED_4 \[120\]](#)
- [__RW uint32_t MPIN \[2\]](#)
- [uint8_t RESERVED_5 \[120\]](#)
- [__RW uint32_t SET \[2\]](#)
- [uint8_t RESERVED_6 \[120\]](#)
- [__W uint32_t CLR \[2\]](#)
- [uint8_t RESERVED_7 \[120\]](#)
- [__W uint32_t NOT \[2\]](#)
- [uint8_t RESERVED_8 \[120\]](#)
- [__W uint32_t DIRSET \[2\]](#)
- [uint8_t RESERVED_9 \[120\]](#)
- [__W uint32_t DIRCLR \[2\]](#)
- [uint8_t RESERVED_10 \[120\]](#)
- [__W uint32_t DIRNOT \[2\]](#)

7.17.1 Detailed Description

Structure type to access the General Purpose [Input Output](#) (GPIO).

GPIO - Register Layout Typedef

7.17.2 Member Data Documentation

7.17.2.1 B

```
__RW uint8_t GPIO_Type::B[2][32]
```

Byte pin registers

7.17.2.2 CLR

```
__W uint32_t GPIO_Type::CLR[2]
```

Clear port

7.17.2.3 DIR

```
__RW uint32_t GPIO_Type::DIR[2]
```

Direction registers

7.17.2.4 DIRCLR

```
__W uint32_t GPIO_Type::DIRCLR[2]
```

Clear pin direction bits for port

7.17.2.5 DIRNOT

```
__W uint32_t GPIO_Type::DIRNOT[2]
```

Toggle pin direction bits for port

7.17.2.6 DIRSET

```
__W uint32_t GPIO_Type::DIRSET[2]
```

Set pin direction bits for port

7.17.2.7 MASK

```
__RW uint32_t GPIO_Type::MASK[2]
```

Mask register

7.17.2.8 MPIN

```
__RW uint32_t GPIO_Type::MPIN[2]
```

Masked port register

7.17.2.9 NOT

```
__W uint32_t GPIO_Type::NOT[2]
```

Toggle port

7.17.2.10 PIN

```
__RW uint32_t GPIO_Type::PIN[2]
```

Port pin register

7.17.2.11 RESERVED_0

```
uint8_t GPIO_Type::RESERVED_0[4032]
```

RESERVED. NOT USED

7.17.2.12 RESERVED_1

```
uint8_t GPIO_Type::RESERVED_1[3840]
```

RESERVED. NOT USED

7.17.2.13 RESERVED_10

```
uint8_t GPIO_Type::RESERVED_10[120]
```

RESERVED. NOT USED

7.17.2.14 RESERVED_2

```
uint8_t GPIO_Type::RESERVED_2[120]
```

RESERVED. NOT USED

7.17.2.15 RESERVED_3

```
uint8_t GPIO_Type::RESERVED_3[120]
```

RESERVED. NOT USED

7.17.2.16 RESERVED_4

```
uint8_t GPIO_Type::RESERVED_4[120]
```

RESERVED. NOT USED

7.17.2.17 RESERVED_5

```
uint8_t GPIO_Type::RESERVED_5[120]
```

RESERVED. NOT USED

7.17.2.18 RESERVED_6

```
uint8_t GPIO_Type::RESERVED_6[120]
```

RESERVED. NOT USED

7.17.2.19 RESERVED_7

```
uint8_t GPIO_Type::RESERVED_7[120]
```

RESERVED. NOT USED

7.17.2.20 RESERVED_8

```
uint8_t GPIO_Type::RESERVED_8[120]
```

RESERVED. NOT USED

7.17.2.21 RESERVED_9

```
uint8_t GPIO_Type::RESERVED_9[120]
```

RESERVED. NOT USED

7.17.2.22 SET

```
__RW uint32_t GPIO_Type::SET[2]
```

Write: Set register for port .Read: output bits for port

7.17.2.23 W

```
__RW uint32_t GPIO_Type::W[2][32]
```

Word pin registers

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

- [LPC845.h](#)

7.18 gruposdedigitos Struct Reference

Estructura de grupo de dígitos.

```
#include <GrupoDeDigitos.h>
```

Public Member Functions

- [gruposdedigitos \(uint8_t comienzo, uint8_t cantidad\)](#)

Public Attributes

- const [uint8_t m_comienzo](#)
- const [uint8_t m_cantidad](#)

7.18.1 Detailed Description

Estructura de grupo de dígitos.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 gruposdedigitos()

```
gruposdedigitos::gruposdedigitos (
    uint8_t comienzo,
    uint8_t cantidad) [inline]
```

< Constructor por defecto

7.18.3 Member Data Documentation

7.18.3.1 m_cantidad

```
const uint8\_t gruposdedigitos::m_cantidad
```

Cantidad de dígitos

7.18.3.2 m_comienzo

```
const uint8_t gruposdedigitos::m_comienzo
```

Comienzo del grupo

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

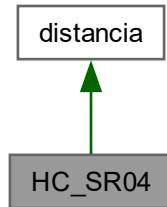
- [GrupoDeDigitos.h](#)

7.19 HC_SR04 Class Reference

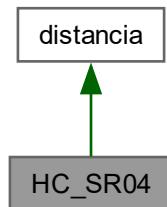
Clase del objeto [HC_SR04](#) El objeto [HC_SR04](#) Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

```
#include <HCSR04.h>
```

Inheritance diagram for HC_SR04:



Collaboration diagram for HC_SR04:



Public Member Functions

- **HC_SR04 (PWM_Reader *&rx, Pwm *&tx)**
Constructor de clase HC_SR04.
- **void Initialize (void)**
Inicializa todas las patas del ultrasónico.
- **uint32_t GetDistancia (void) override**
Devuelve la distancia.
- **void Off (void)**
Apago las patas del ultrasonico.
- **void On (void)**
Enciendo las patas del ultrasonico.
- **bool operator== (const uint32_t a) override**
Operador ==.
- **bool operator<= (const uint32_t a)**
Verifica si la distancia es menor o igual que a.
- **bool operator>= (const uint32_t a)**
Verifica si la distancia es mayor o igual que a.
- **bool operator< (const uint32_t a)**
Verifica si la distancia es menor que a.
- **bool operator> (const uint32_t a)**
Verifica si la distancia es mayor que a.

Additional Inherited Members

Protected Member Functions inherited from **distancia**

- **distancia ()=default**
- **virtual ~distancia ()=default**

7.19.1 Detailed Description

Clase del objeto **HC_SR04** El objeto **HC_SR04** Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

SENSOR ULTRASÓNICO HC-SR04: Se le debe enviar PWM de 60ms o más (recomendamos 80) con un tiempo de encendido de 10us. Recebirá otro pulso variable que debe ser leído por tx. Devuelve el valor de distancia en cm (centímetros)

7.19.2 Constructor & Destructor Documentation

7.19.2.1 **HC_SR04()**

```
HC_SR04::HC_SR04 (
    PWM_Reader *& rx,
    Pwm *& tx)
```

Constructor de clase **HC_SR04**.

Crea un **HC_SR04** con los parámetros correspondientes

Parameters

in	<i>rx</i>	Lector de anchos de pulso utilizado en el pin ECHO.
in	<i>tx</i>	PWM utilizado como pin TRIGG.

7.19.3 Member Function Documentation**7.19.3.1 GetDistancia()**

```
uint32_t HC_SR04::GetDistancia (
    void ) [override], [virtual]
```

Devuelve la distancia.

A menos que esté apagado, devuelve el valor de distancia obtenido. Para evitar overflow del Uint32, se verifica el valor censado.

Returns

m_distancia

Implements [distancia](#).

7.19.3.2 Initialize()

```
void HC_SR04::Initialize (
    void )
```

Inicializa todas las patas del ultrasónico.

Inicializa el PWM con el valor correspondiente y el contador de ancho ancho de pulso.

7.19.3.3 Off()

```
void HC_SR04::Off (
    void )
```

Apago las patas del ultrasonico.

Llamo a las funciones de apagar y guardo en el buffer un valor imposible.

7.19.3.4 On()

```
void HC_SR04::On (
    void )
```

Enciendo las patas del ultrasonico.

Llamo a las funciones de encender de tx y rx.

7.19.3.5 operator<()

```
bool HC_SR04::operator< (
    const uint32_t a)
```

Verifica si la distancia es menor que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	a	Distancia a comparar.
----	---	-----------------------

Returns

bool

7.19.3.6 operator<=()

```
bool HC_SR04::operator<= (
    const uint32_t a)
```

Verifica si la distancia es menor o igual que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	a	Distancia a comparar.
----	---	-----------------------

Returns

bool

7.19.3.7 operator==()

```
bool HC_SR04::operator== (
    const uint32_t a) [override], [virtual]
```

Operador ==.

Consulta si la distancia es igual o no a la entregada.

Parameters

in	a	Valor de distancia.
----	---	---------------------

Returns

bool

Implements [distancia](#).

7.19.3.8 operator>()

```
bool HC_SR04::operator> (
    const uint32_t a)
```

Verifica si la distancia es mayor que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	a	Distancia a comparar.
----	---	-----------------------

Returns

bool

7.19.3.9 operator>=()

```
bool HC_SR04::operator>= (
    const uint32_t a)
```

Verifica si la distancia es mayor o igual que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	a	Distancia a comparar.
----	---	-----------------------

Returns

bool

The documentation for this class was generated from the following files:

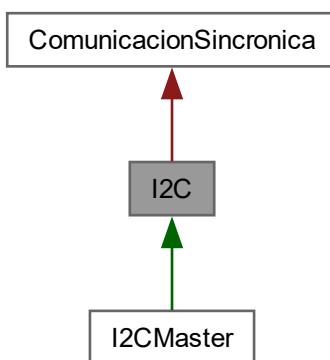
- [HCSR04.h](#)
- [HCSR04.cpp](#)

7.20 I2C Class Reference

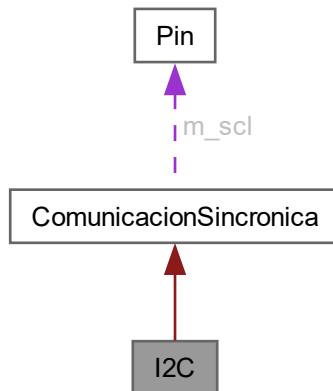
Clase del objeto [I2C](#). El objeto [I2C](#) genera una comunicación sincrónica de tipo [I2C](#). Posee las funciones básicas como start, stop, write y read.

```
#include <I2C.h>
```

Inheritance diagram for I2C:



Collaboration diagram for I2C:



Public Types

- enum `I2C_role_t`
Enumeracion. Modo del I2C. Puede ser master o slave.
- enum `I2C_action_t`
Enumeracion. Acciones del I2C. Pueden ser leer o escribir (write o read).
- enum `I2C_states_t`
Enumeracion. Estados del I2C.

Public Member Functions

- `I2C (I2C_Type *I2C_register, Pin *sda, Pin *scl, I2C_role_t mode=master, uint8_t slv_addr=0)`
Constructor de la clase I2C.
- void `Initialize (uint32_t clk_freq)`
Inicializa el I2C.
- void `EnableInterrupt (void)`
Habilita la interrupcion.
- void `DisableInterrupt (void)`
Deshabilita la interrupcion.
- void `Start (uint8_t addr, I2C_action_t action)`
Inicia una comunicacion I2C.
- void `Stop (void)`
Detiene la comunicacion.
- void `Write (uint8_t data) override`
Escribe un valor a un slave.
- `I2C & operator= (uint8_t data)`
Sobrecarga del operador = (igual).
- `int8_t Read (uint8_t *data, bool continue_reading)`
Lee un valor recibido.
- `int8_t Read (uint8_t *data) override`

- `void Continue (void)`
Lee un valor recibido.
- `void ACK (bool a)`
Envia una señal para continuar.
- `bool ACKaddr (void)`
Envia un Acknowledge.
- `I2C_states_t GetState (void)`
Acknowledge especial del address.
- `I2C_states_t SetTimeOut (uint32_t clk_cycles)`
Regresa el estado del I2C.
- `void SetTimeOut (uint32_t clk_cycles)`
Coloca un valor de TimeOut al I2C.
- `virtual void I2C_IRQHandler (void)`

7.20.1 Detailed Description

Clase del objeto `I2C` El objeto `I2C` genera una comunicación sincrónica de tipo `I2C`. Posee las funciones basicas como start, stop, write y read.

7.20.2 Member Enumeration Documentation

7.20.2.1 I2C_states_t

```
enum I2C::I2C_states_t
```

Enumeracion. Estados del `I2C`.

Un master puede encontrarse listo para transmitir o recibir (`rx_data` y `tx_ready`) o recibiendo un NACK (`NACK_addr` y `NCK_tx`). Un slave puede encontrarse listo para transmitir o recibir (`slvst_tx` y `slvst_rx`) o recibiendo un address para iniciar (`slvst_addr`) Todos los modos pueden estar en idle si no hacen nada o busy si estan en mitad de un procesamiento.

7.20.3 Constructor & Destructor Documentation

7.20.3.1 I2C()

```
I2C::I2C (
    I2C_Type * I2C_register,
    Pin * sda,
    Pin * scl,
    I2C_role_t mode = master,
    uint8_t slv_addr = 0)
```

Constructor de la clase `I2C`.

Genera un `I2C` con los parametros indicados

Parameters

in	<code>I2C_register</code>	Registro del <code>I2C</code> a utilizar. Puede valer I2C0, I2C1, I2C2 o I2C3.
in	<code>sda</code>	<code>Pin</code> de transmision de datos.
in	<code>scl</code>	<code>Pin</code> de clock.
in	<code>mode</code>	Tipo de <code>I2C</code> . Puede ser master o slave.
in	<code>slv_addr</code>	Address propio del <code>I2C</code> . Solo util cuando se lo configura como slave.

7.20.4 Member Function Documentation

7.20.4.1 ACK()

```
void I2C::ACK (
    bool a)
```

Envia un Acknowledge.

Para un [I2C](#) en modo slave, envia un Acknowledge que permite continuar la comunicacion.

Parameters

in	a	bool que indica si la recepcion es exitosa o no. Si es true se envia un ACK. Si es false se envia NACK.
----	---	---

7.20.4.2 ACKAddr()

```
bool I2C::ACKAddr (
    void )
```

Acknowledge especial del address.

Envia un acknowledge solo si el address que se recibio coindide con el propio.

7.20.4.3 Continue()

```
void I2C::Continue (
    void )
```

Envia una señal para continuar.

Carga el registro de continuar. Debe hacerse al usarse la funcion [Read\(\)](#) sin continue_reading.

7.20.4.4 GetState()

```
I2C::I2C_states_t I2C::GetState (
    void )
```

Regresa el estado del [I2C](#).

Regresa el estado actual del [I2C](#), ya sea master o slave. Un master puede estar rx_data si se recibio data, tx_ready si esta listo para transmitir, NACK_addr o NACK_tx si recibio un NACK. Un slave puede estar en slvst_addr si recibio un start, slvst_tx si recibio un pedido a transmitir o slvst_rx si recibio un pedido a leer. Ambos pueden estar en idle si no se encuentran haciendo nada o busy si estan en mitad de un envio de informacion.

7.20.4.5 I2C_IRQHandler()

```
virtual void I2C::I2C_IRQHandler (
    void ) [inline], [virtual]
```

Handler generico de interrupcion [I2C](#). No hace nada, debe heredarse y sobreescibirse.

7.20.4.6 Initialize()

```
void I2C::Initialize (
    uint32_t clk_freq)
```

Inicializa el [I2C](#).

Configura todos los registros para el uso del [I2C](#). Configura los clocks, resetea, configura la SWM, configura los registros y las interrupciones.

Parameters

in	<i>clk_freq</i>	frecuencia del clock de transmision a utilizar. En KHz
----	-----------------	--

7.20.4.7 operator=(*)

```
I2C & I2C::operator= (
    uint8_t data)
```

Sobrecarga del operador = (igual).

Sobrecarga de operador para escritura de un byte.

Parameters

<i>data</i>	byte a escribir.
-------------	------------------

Returns

I2C&: Puntero a si mismo.

7.20.4.8 Read() [1/2]

```
int8_t I2C::Read (
    uint8_t * data) [override], [virtual]
```

Lee un valor recibido.

Lee el valor recibido por un slave. Solo puede ser llamado por un master que previamente haya pedido recibir y o un slave al que le llego un valor.

Parameters

in	<i>data</i>	puntero a char donde se guardara el valor recibido.
----	-------------	---

Returns

int8_t: valor de error. 1 no hubo problemas. 0 no se puede leer aun.

Implements [ComunicacionSincronica](#).

7.20.4.9 Read() [2/2]

```
int8_t I2C::Read (
    uint8_t * data,
    bool continue_reading)
```

Lee un valor recibido.

Lee el valor recibido por un slave. Solo puede ser llamado por un master que previamente haya pedido recibir y o un slave al que le llego un valor.

Parameters

<code>in, out</code>	<code>data</code>	puntero a char donde se guardara el valor recibido.
<code>in</code>	<code>continue_reading</code>	bool que indica si se va a seguir leyendo o no. Si es true, se envia un "continue" y el slave volvera a transmitir informacion. Si es false no se realiza ninguna accion. Si el I2C es slave, continue_reading no tiene proposito.

Returns

`int8_t`: valor de error. 0 no hubo problemas. -1 no se puede leer aun.

7.20.4.10 SetTimeOut()

```
void I2C::SetTimeOut (
    uint32_t clk_cycles)
```

Coloca un valor de TimeOut al I2C.

Configura el TimeOut del I2C. Este siempre sera multiplo de 16 ya que se encuentra fijo en Tomin. Por default el I2C amanece con 65 536 periodos de clock para un timeout (lo cual es el valor maximo).

Parameters

<code>in</code>	<code>clk_cycles</code>	Ciclos de clock para el TimeOut. Se recomienda que sea multiplo de 16.
-----------------	-------------------------	--

7.20.4.11 Start()

```
void I2C::Start (
    uint8_t addr,
    I2C_action_t action)
```

Inicia una comunicacion I2C.

Envia el start a un slave. La funcion solo puede ser llamada por un master si este no se encuentra ocupado.

Parameters

<code>in</code>	<code>addr</code>	address del slave al que se comunicara.
<code>in</code>	<code>action</code>	accion a realizar. Puede ser read o write.

7.20.4.12 Stop()

```
void I2C::Stop (
    void )
```

Detiene la comunicacion.

Envia una señal de stop para detener la comunicacion. Solo puede ser llamada por un master que no se encuentre ocupado.

7.20.4.13 Write()

```
void I2C::Write (
    uint8_t data) [override], [virtual]
```

Escribe un valor a un slave.

Envia un dato. Solo puede ser llamada por un master que previamente haya pedido transmitir informacion o un slave que este en modo recibir transmision.

Parameters

in	data	byte a enviar.
----	------	----------------

Implements [ComunicacionSincronica](#).

The documentation for this class was generated from the following files:

- [I2C.h](#)
- [I2C.cpp](#)

7.21 I2C_Type Struct Reference

```
#include <LPC845.h>
```

Public Attributes

- [__RW uint32_t CFG](#)
- [__RW uint32_t STAT](#)
- [__RW uint32_t INTENSET](#)
- [__W uint32_t INTENCLR](#)
- [__RW uint32_t TIMEOUT](#)
- [__RW uint32_t CLKDIV](#)
- [__R uint32_t INTSTAT](#)
- [uint8_t RESERVED_0 \[4\]](#)
- [__RW uint32_t MSTCTL](#)
- [__RW uint32_t MSTTIME](#)
- [__RW uint32_t MSTDAT](#)
- [uint8_t RESERVED_1 \[20\]](#)
- [__RW uint32_t SLVCTL](#)
- [__RW uint32_t SLVDAT](#)
- [__RW uint32_t SLVADR \[4\]](#)
- [__RW uint32_t SLVQUAL0](#)
- [uint8_t RESERVED_2 \[36\]](#)
- [__R uint32_t MONRXDAT](#)

7.21.1 Detailed Description

[I2C - Register Layout Typedef](#)

7.21.2 Member Data Documentation

7.21.2.1 CFG

```
\_\_RW uint32\_t I2C_Type::CFG
```

Configuration for shared functions., offset: 0x0

7.21.2.2 CLKDIV

```
__RW uint32_t I2C_Type::CLKDIV
```

Clock pre-divider for the entire [I2C](#) interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function., offset: 0x14

7.21.2.3 INTENCLR

```
__W uint32_t I2C_Type::INTENCLR
```

Interrupt Enable Clear register., offset: 0xC

7.21.2.4 INTENSET

```
__RW uint32_t I2C_Type::INTENSET
```

Interrupt Enable Set and read register., offset: 0x8

7.21.2.5 INTSTAT

```
__R uint32_t I2C_Type::INTSTAT
```

Interrupt Status register for Master, Slave, and Monitor functions., offset: 0x18

7.21.2.6 MONRXDAT

```
__R uint32_t I2C_Type::MONRXDAT
```

Monitor receiver data register., offset: 0x80

7.21.2.7 MSTCTL

```
__RW uint32_t I2C_Type::MSTCTL
```

Master control register., offset: 0x20

7.21.2.8 MSTDAT

```
__RW uint32_t I2C_Type::MSTDAT
```

Combined Master receiver and transmitter data register., offset: 0x28

7.21.2.9 MSTTIME

`__RW uint32_t I2C_Type::MSTTIME`

Master timing configuration., offset: 0x24

7.21.2.10 RESERVED_0

`uint8_t I2C_Type::RESERVED_0[4]`

RESERVERD.

7.21.2.11 RESERVED_1

`uint8_t I2C_Type::RESERVED_1[20]`

RESERVERD.

7.21.2.12 RESERVED_2

`uint8_t I2C_Type::RESERVED_2[36]`

RESERVERD.

7.21.2.13 SLVADR

`__RW uint32_t I2C_Type::SLVADR[4]`

Slave address register., array offset: 0x48, array step: 0x4

7.21.2.14 SLVCTL

`__RW uint32_t I2C_Type::SLVCTL`

Slave control register., offset: 0x40

7.21.2.15 SLVDAT

`__RW uint32_t I2C_Type::SLVDAT`

Combined Slave receiver and transmitter data register., offset: 0x44

7.21.2.16 SLVQUAL0

`__RW uint32_t I2C_Type::SLVQUAL0`

Slave Qualification for address 0., offset: 0x58

7.21.2.17 STAT

```
__RW uint32_t I2C_Type::STAT
```

Status register for Master, Slave, and Monitor functions., offset: 0x4

7.21.2.18 TIMEOUT

```
__RW uint32_t I2C_Type::TIMEOUT
```

Time-out value register., offset: 0x10

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

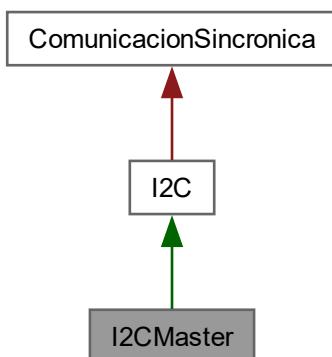
- [LPC845.h](#)

7.22 I2CMaster Class Reference

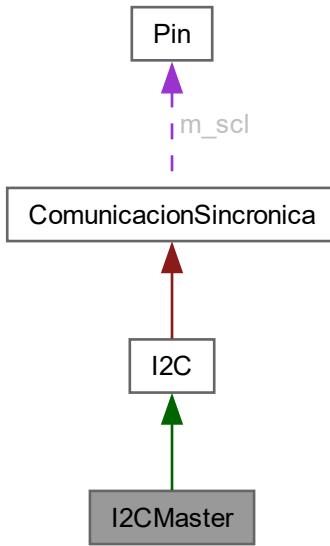
Clase del objeto [I2CMaster](#) El objeto [I2CMaster](#) genera una comunicación tipo master de [I2C](#) utilizando buffers de recepcion y transmision con interrupciones.

```
#include <I2CMaster.h>
```

Inheritance diagram for I2CMaster:



Collaboration diagram for I2CMaster:



Public Member Functions

- `I2CMaster (I2C_Type *I2C_register, Pin *sda, Pin *scl, uint32_t maxRx=15, uint32_t maxTx=15)`
Constructor de la clase `I2CMaster`.
- `void Initialize (uint32_t clk_freq)`
Inicializa el `I2C`.
- `void Write (uint8_t addr, const char *msg)`
Transmite el mensaje indicado.
- `void Write (uint8_t addr, const void *msg, uint32_t n)`
Transmite length cantidad de bytes del mensaje indicado.
- `void RequestRead (uint8_t addr, uint32_t cant_read=1)`
Inicia una comunicacion de lectura.
- `void * Read (void *msg, uint32_t n)`
Lee el buffer de recepcion.
- `bool isIdle (void)`
Indica si el `I2C` esta en reposo o no.

Additional Inherited Members

Protected Types inherited from `I2C`

- enum `I2C_role_t`
Enumeracion. Modo del `I2C`. Puede ser master o slave.
- enum `I2C_action_t`
Enumeracion. Acciones del `I2C`. Pueden ser leer o escribir (write o read).
- enum `I2C_states_t`
Enumeracion. Estados del `I2C`.

Protected Member Functions inherited from I2C

- **I2C (I2C_Type *I2C_register, Pin *sda, Pin *scl, I2C_role_t mode=master, uint8_t slv_addr=0)**
Constructor de la clase I2C.
- void **Initialize (uint32_t clk_freq)**
Inicializa el I2C.
- void **EnableInterrupt (void)**
Habilita la interrupcion.
- void **DisableInterrupt (void)**
Deshabilita la interrupcion.
- void **Start (uint8_t addr, I2C_action_t action)**
Inicia una comunicacion I2C.
- void **Stop (void)**
Detiene la comunicacion.
- void **Write (uint8_t data) override**
Escribe un valor a un slave.
- I2C & **operator= (uint8_t data)**
Sobrecarga del operador = (igual).
- int8_t **Read (uint8_t *data, bool continue_reading)**
Lee un valor recibido.
- int8_t **Read (uint8_t *data) override**
Lee un valor recibido.
- void **Continue (void)**
Envia una señal para continuar.
- void **ACK (bool a)**
Envia un Acknowledge.
- bool **ACKaddr (void)**
Acknowledge especial del address.
- I2C_states_t **GetState (void)**
Regresa el estado del I2C.
- void **SetTimeOut (uint32_t clk_cycles)**
Coloca un valor de TimeOut al I2C.

7.22.1 Detailed Description

Clase del objeto **I2CMaster** El objeto **I2CMaster** genera una comunicación tipo master de **I2C** utilizando buffers de recepcion y transmision con interrupciones.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 I2CMaster()

```
I2CMaster::I2CMaster (
    I2C_Type * I2C_register,
    Pin * sda,
    Pin * scl,
    uint32_t maxRx = 15,
    uint32_t maxTx = 15)
```

Constructor de la clase **I2CMaster**.

Genera un master **I2C** con buffers de transmision y recepcion para envio de strings.

Parameters

in	<i>I2C_register</i>	Registro del I2C a utilizar. Puede valer I2C0, I2C1, I2C2 o I2C3.
in	<i>sda</i>	Pin de transmision de datos.
in	<i>scl</i>	Pin de clock.
in	<i>maxRx</i>	Valor maximo del registro de recepcion. Por default 15.
in	<i>maxTx</i>	Valor maximo del registro de transmision. Por default 15.

7.22.3 Member Function Documentation**7.22.3.1 Initialize()**

```
void I2CMaster::Initialize (
    uint32_t clk_freq)
```

Inicializa el **I2C**.

Configura todos los registros para el uso del **I2C**. Configura los clocks, resetea, configura la SWM, configura los registros y las interrupciones.

Parameters

in	<i>clk_freq</i>	frecuencia del clock de transmision a utilizar.
----	-----------------	---

7.22.3.2 isIdle()

```
bool I2CMaster::isIdle (
    void )
```

Indica si el **I2C** esta en reposo o no.

Indica si el estado del **I2C** es idle. Solo se puede enviar y recibir informacion si se encuentra en idle.

Returns

bool: true si esta en idle, false si no lo esta.

7.22.3.3 Read()

```
void * I2CMaster::Read (
    void * msg,
    uint32_t length)
```

Lee el buffer de recepcion.

Lee una cantidad fija de bytes dentro del buffer de recepcion.

Parameters

in	<i>msg</i>	Puntero a variable donde se enviaran los valores leidos.
in	<i>length</i>	Cantidad de bytes a leer.

Returns

void*: Puntero a los valores leidos. Si hubo error se envia nullptr.

7.22.3.4 RequestRead()

```
void I2CMaster::RequestRead (
    uint8_t addr,
    uint32_t cant_read = 1)
```

Inicia una comunicacion de lectura.

Comienza una comunicacion en modo lectura que no se detiene hasta recibir cant_read bytes. Los bytes leidos se guardaran en el buffer.

Parameters

in	<i>addr</i>	address del slave al que se comunicara.
in	<i>cant_read</i>	cantidad de bytes a transmitir.

7.22.3.5 Write() [1/2]

```
void I2CMaster::Write (
    uint8_t addr,
    const char * msg)
```

Transmite el mensaje indicado.

Coloca el mensaje indicado en el buffer. Importante, el mensaje debe terminar en \0 (String).

Parameters

in	<i>addr</i>	address del slave al que se comunicara.
in	<i>msg</i>	Mensaje a transmitir.

7.22.3.6 Write() [2/2]

```
void I2CMaster::Write (
    uint8_t addr,
    const void * msg,
    uint32_t length)
```

Transmite length cantidad de bytes del mensaje indicado.

Coloca una cantidad fija del mensaje indicado en el buffer.

Parameters

in	<i>addr</i>	address del slave al que se comunicara.
in	<i>msg</i>	Mensaje a transmitir.
in	<i>length</i>	cantidad de bytes a transmitir.

The documentation for this class was generated from the following files:

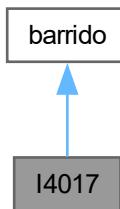
- [I2CMaster.h](#)
- [I2CMaster.cpp](#)

7.23 I4017 Class Reference

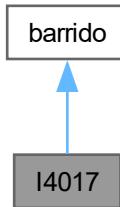
Clase del objeto [I4017](#) El objeto [I4017](#) permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a través de sus patas.

```
#include <I4017.h>
```

Inheritance diagram for I4017:



Collaboration diagram for I4017:



Public Member Functions

- **I4017** (const vector< `gpio` * > &pins4017, `uint8_t` maxsalidas)
- void **SetDigito** (void) override
Función de cambio en el barrido.
- void **SetReset** (void)
Resetea el I4017.
- void **SetClock** (void)
Enciende la pata clock.
- void **Initialize** (void) override
Función de Inicialización del I4017.

7.23.1 Detailed Description

Clase del objeto **I4017** El objeto **I4017** permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a través de sus patas.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 I4017()

```
I4017::I4017 (
    const vector< gpio * > & pins4017,
    uint8_t maxsalidas) [inline]
```

constructor por defecto

7.23.3 Member Function Documentation

7.23.3.1 SetClock()

```
void I4017::SetClock (
    void )
```

Enciende la pata clock.

El **I4017** aumenta en 1 la salida al recibir un pulso de clock.

7.23.3.2 SetDigito()

```
void I4017::SetDigito (
    void ) [override], [virtual]
```

Función de cambio en el barrido.

Se mueve 1 en el barrido de bits.

Implements **barrido**.

7.23.3.3 SetReset()

```
void I4017::SetReset (  
    void )
```

Resetea el [I4017](#).

El [I4017](#) vuelve al valor 0 de salida por enviar un pulso a la pata de Reset.

The documentation for this class was generated from the following files:

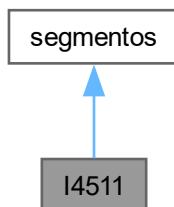
- [I4017.h](#)
- [I4017.cpp](#)

7.24 I4511 Class Reference

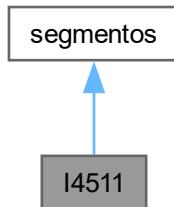
Clase del objeto [I4511](#) El objeto [I4511](#) permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.

```
#include <I4511.h>
```

Inheritance diagram for I4511:



Collaboration diagram for I4511:



Public Member Functions

- [I4511](#) (const vector< [gpio](#) * > &[bcd](#))
- void [SetSegmentos](#) ([uint16_t](#)) override
Función de escritura del segmento.
- void [Initialize](#) (void) override
Función de Inicialización del [I4511](#).

7.24.1 Detailed Description

Clase del objeto [I4511](#) El objeto [I4511](#) permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 I4511()

```
I4511::I4511 (
    const vector< gpio * > &bcd) [inline]
```

constructor por defecto

7.24.3 Member Function Documentation

7.24.3.1 SetSegmentos()

```
void I4511::SetSegmentos (
    uint16\_t valor) [override], [virtual]
```

Función de escritura del segmento.

Escribe el segmento en formato binario.

Parameters

in	<i>valor</i>	valor a escribir.
----	--------------	-------------------

Implements [segmentos](#).

The documentation for this class was generated from the following files:

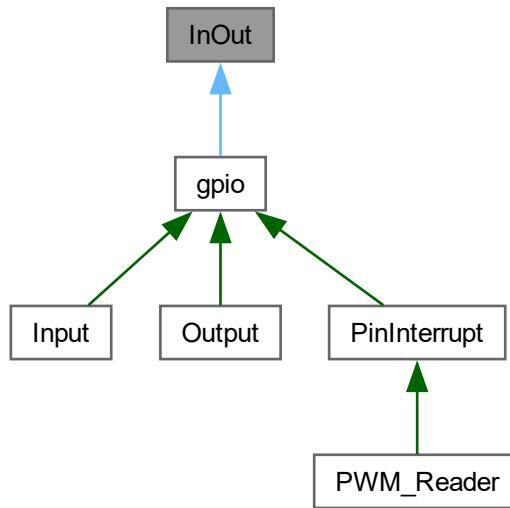
- [I4511.h](#)
- [I4511.cpp](#)

7.25 InOut Class Reference

Clase del objeto [InOut](#).

```
#include <InOut.h>
```

Inheritance diagram for InOut:



Public Member Functions

- `InOut ()=default`
- virtual `uint8_t SetPin (void)=0`
- virtual `uint8_t ClrPin (void)=0`
- virtual `uint8_t SetTogglePin (void)=0`
- virtual `uint8_t SetDir (void)=0`
- virtual `uint8_t SetToggleDir (void)=0`
- virtual `uint8_t GetPin (void)=0`
- virtual `uint8_t SetPinMode (void)=0`
- virtual `uint8_t SetPinResistor (void)=0`
- virtual `~InOut ()=default`

7.25.1 Detailed Description

Clase del objeto [InOut](#).

Clase abstracta pura para la generación de GPIO

7.25.2 Constructor & Destructor Documentation

7.25.2.1 InOut()

```
InOut::InOut () [default]
```

Constructor por defecto

7.25.2.2 ~InOut()

```
virtual InOut::~InOut () [virtual], [default]
```

Destructor por defecto

7.25.3 Member Function Documentation

7.25.3.1 ClrPin()

```
virtual uint8_t InOut::ClrPin (
    void ) [pure virtual]
```

Apaga el pin

Implemented in [gpio](#).

7.25.3.2 GetPin()

```
virtual uint8_t InOut::GetPin (
    void ) [pure virtual]
```

Obtiene valor del pin

Implemented in [gpio](#).

7.25.3.3 SetDir()

```
virtual uint8_t InOut::SetDir (
    void ) [pure virtual]
```

Setea la direccion del pin

Implemented in [gpio](#).

7.25.3.4 SetPin()

```
virtual uint8_t InOut::SetPin (
    void ) [pure virtual]
```

Enciende el pin

Implemented in [gpio](#).

7.25.3.5 SetPinMode()

```
virtual uint8_t InOut::SetPinMode (
    void ) [pure virtual]
```

Setea modo del pin

Implemented in [gpio](#).

7.25.3.6 SetPinResistor()

```
virtual uint8_t InOut::SetPinResistor (
    void ) [pure virtual]
```

Setea resistencia del pin

Implemented in [gpio](#).

7.25.3.7 SetToggleDir()

```
virtual uint8_t InOut::SetToggleDir (
    void ) [pure virtual]
```

Cambia la direccion del pin

Implemented in [gpio](#).

7.25.3.8 SetTogglePin()

```
virtual uint8_t InOut::SetTogglePin (
    void ) [pure virtual]
```

Cambia el valor del pin

Implemented in [gpio](#).

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

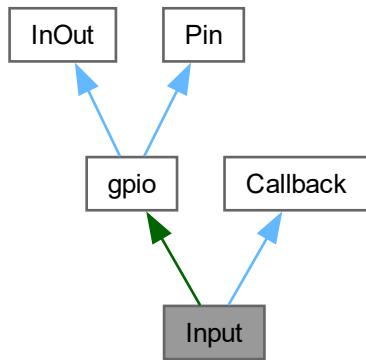
- [InOut.h](#)

7.26 Input Class Reference

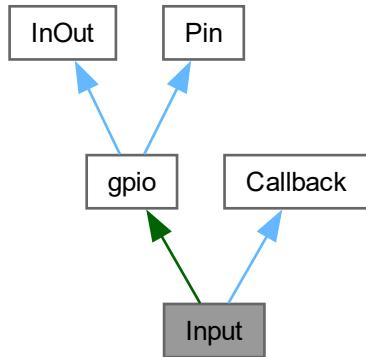
Clase del objeto [Input](#).

```
#include <Input.h>
```

Inheritance diagram for Input:



Collaboration diagram for Input:



Public Member Functions

- [Input \(port_t puerto, uint8_t bit, mode_t modo, activity_t actividad=high, uint8_t MaxBounce=MAX_BOUNCE\)](#)
Constructor de clase Input.
- void [Initialize \(\)](#)

- *Inicializa la entrada.*
- `uint8_t get ()`
Devuelve el valor de la input.
- `void SWhandler () override`
Funcion de interrupcion del systick.
- `bool operator== (uint8_t val)`
Sobrecarga de del operador de igualdad.
- `bool operator!= (uint8_t val)`
Sobrecarga de del operador de desigualdad.
- `virtual ~Input ()`

Public Member Functions inherited from [Callback](#)

- `void SetInterrupt ()`
Activa la interrupcion.
- `void UnSetInterrupt ()`
Desactiva la interrupcion.

Friends

- `bool operator== (uint32_t val, Input &I)`
Sobrecarga de del operador de asignacion.

Additional Inherited Members

Protected Types inherited from [gpio](#)

- enum `direction_t`
Enumeracion de input/output.
- enum `power_t`
Enumeracion de encendido/apagado.
- enum `mode_t`
Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum `activity_t`
Enumeracion de activo bajo/alto.
- enum `interrupt_mode_t`
Enumeracion de interrupciones del pin.
- typedef enum `gpio::direction_t direction_t`
Enumeracion de input/output.
- typedef enum `gpio::mode_t mode_t`
Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum `gpio::activity_t activity_t`
Enumeracion de activo bajo/alto.

Protected Types inherited from [Pin](#)

- enum [port_t](#)
- enum [max_bits_port_t](#)
máximos pines por puerto
- enum [error_t](#)
- typedef enum [Pin::port_t](#) [port_t](#)
- typedef enum [Pin::error_t](#) [error_t](#)

Protected Member Functions inherited from [gpio](#)

- [gpio \(port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high\)](#)
Constructor de clase GPIO.
- [uint8_t SetPin \(void\) override](#)
Enciende la salida.
- [uint8_t ClrPin \(void\) override](#)
Limpia el pin.
- [uint8_t SetTogglePin \(void\) override](#)
Toggle del pin.
- [uint8_t SetDir \(void\) override](#)
Setea la dirección.
- [uint8_t SetToggleDir \(void\) override](#)
Toggle de la dirección.
- [uint8_t GetPin \(void\) override](#)
Devuelve el valor del pin.
- [uint8_t SetPinMode \(void\) override](#)
Configura el modo del pin.
- [uint8_t SetPinResistor \(void\) override](#)
Setea la resistencia interna.
- [gpio & operator= \(uint8_t a\)](#)
Sobrecarga del operador =.
- virtual [~gpio \(\)=default](#)

Protected Member Functions inherited from [InOut](#)

- [InOut \(\)=default](#)
- virtual [~InOut \(\)=default](#)

Protected Member Functions inherited from [Pin](#)

- [Pin \(port_t port, uint8_t bit\)](#)
Constructor de clase PIN.

Protected Member Functions inherited from [Callback](#)

- void [SetInterrupt \(\)](#)
Activa la interrupción.
- void [UnSetInterrupt \(\)](#)
Desactiva la interrupción.

Protected Attributes inherited from gpio

- const `mode_t m_mode`
- `direction_t m_direction`
- const `activity_t m_activity`

Protected Attributes inherited from Pin

- const `port_t m_port`
- const `uint8_t m_bit`
- `int8_t m_error`

7.26.1 Detailed Description

Clase del objeto `Input`.

El objeto `Input` Permite el manejo de entradas digitales con un antirrebote propio por software.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 `Input()`

```
Input::Input (
    port_t puerto,
    uint8_t bit,
    mode_t modo,
    activity_t actividad = high,
    uint8_t MaxBounce = MAX_BOUNCE)
```

Constructor de clase `Input`.

Crea un `Input` con los parámetros correspondientes.

Parameters

in	<i>puerto</i>	Puerto del objeto.
in	<i>bit</i>	Bit del objeto.
in	<i>modo</i>	Configuracion eléctrica del pin.
in	<i>actividad</i>	Activo alto/bajo.
in	<i>MaxBounce</i>	Cantidad de rebotes permitidos.

7.26.2.2 `~Input()`

```
Input::~Input () [virtual]
```

Destructor por defecto

7.26.3 Member Function Documentation

7.26.3.1 get()

```
uint8_t Input::get  
    (void)
```

Devuelve el valor de la input.

Entrega el valor "real" de la entrada sin su rebote.

Returns

`uint32_t`: Valor de la entrada.

7.26.3.2 Initialize()

```
void Input::Initialize  
    (void)
```

Inicializa la entrada.

Configura el GPIO y el buffer.

7.26.3.3 operator"!="()

```
bool Input::operator!=  
    (uint8_t val)
```

Sobrecarga de del operador de desigualdad.

Parameters

in	val	Valor a comparar con el buffer.
----	-----	---------------------------------

Returns

true si la entrada no esta en val.

7.26.3.4 operator==()

```
bool Input::operator==  
    (uint8_t val)
```

Sobrecarga de del operador de igualdad.

Parameters

in	val	Valor a comparar con el buffer.
----	-----	---------------------------------

Returns

true si la entrada esta en val.

7.26.3.5 SWhandler()

```
void Input::SWhandler (
    void ) [override], [virtual]
```

Funcion de interrupcion del systick.

Guarda en el buffer el valor de la entrada en caso de haber pasado el antirrebote.

Implements [Callback](#).

7.26.4 Friends And Related Symbol Documentation**7.26.4.1 operator==**

```
bool operator== (
    uint32_t val,
    Input & I) [friend]
```

Sobrecarga de del operador de asignacion.

Parameters

in	val	Valor a comparar con el buffer
in	I	Entrada a comparar

Returns

true si la entrada esta en val

The documentation for this class was generated from the following files:

- [Input.h](#)
- [Input.cpp](#)

7.27 IOCON_Type Struct Reference

Structure type to access the IOCON.

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t PIO [56]`

7.27.1 Detailed Description

Structure type to access the IOCON.

IOCON - Register Layout Typedef

7.27.2 Member Data Documentation

7.27.2.1 PIO

`__RW uint32_t IOCON_Type::PIO[56]`

I/O configuration for pin [X]

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

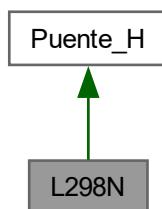
- [LPC845.h](#)

7.28 L298N Class Reference

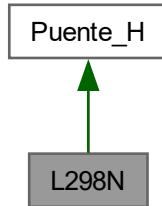
Clase del objeto [L298N](#) El objeto [L298N](#) realiza las acciones de control de dos motores controlados por el correspondiente periférico.

```
#include <L298N.h>
```

Inheritance diagram for L298N:



Collaboration diagram for L298N:



Public Member Functions

- `L298N (gpio *&_motorDer_a, gpio *&_motorDer_b, gpio *&_motorIzq_a, gpio *&_motorIzq_b)`
Constructor de clase L298N.
- void `Initialize` (void) override
Inicializa todas las salidas GPIO.
- void `GirarIzq` (void) override
Gira a la izquierda.
- void `GirarDer` (void) override
Gira a la derecha.
- void `Girar` (const `uint8_t` direccion) override
Gira el motor.
- void `Frenar` (void) override
Frena el motor.
- void `Avanzar` (void) override
Avanza los motores.
- void `Retroceder` (void) override
Pone los motores marcha atrás.
- virtual `~L298N ()`

Additional Inherited Members

Protected Types inherited from [Puente_H](#)

- enum

Protected Member Functions inherited from [Puente_H](#)

- `Puente_H ()=default`
- virtual `~Puente_H ()=default`

7.28.1 Detailed Description

Clase del objeto [L298N](#) El objeto [L298N](#) realiza las acciones de control de dos motores controlados por el correspondiente periférico.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 L298N()

```
L298N::L298N (
    gpio *& _motorDer_a,
    gpio *& _motorDer_b,
    gpio *& _motorIzq_a,
    gpio *& _motorIzq_b)
```

Constructor de clase [L298N](#).

Crea un [L298N](#) con los parámetros correspondientes

Parameters

in	<code>_motorDer_a</code>	Puntero a gpio del motor derecho A.
in	<code>_motorDer_b</code>	Puntero a gpio del motor derecho B.
in	<code>_motorIzq_a</code>	Puntero a gpio del motor izquierdo A.
in	<code>_motorIzq_b</code>	Puntero a gpio del motor izquierdo B.

7.28.2.2 ~L298N()

```
L298N::~L298N () [virtual]
```

Destructor por defecto

7.28.3 Member Function Documentation

7.28.3.1 Avanzar()

```
void L298N::Avanzar (
    void) [override], [virtual]
```

Avanza los motores.

Enciende los 2 motores con tensión en la pata A.

Implements [Puente_H](#).

7.28.3.2 Frenar()

```
void L298N::Frenar (
    void )  [override], [virtual]
```

Frena el motor.

Deja las cuatro salidas en 0.

Implements [Puente_H](#).

7.28.3.3 Girar()

```
void L298N::Girar (
    const uint8_t direccion)  [override], [virtual]
```

Gira el motor.

Utilizando la enumeración de PuenteH realiza el giro correspondiente.

Parameters

in	<i>direccion</i>	Dirección del giro (IZQUIERDA = 0 , DERECHA = 1)
----	------------------	--

Implements [Puente_H](#).

7.28.3.4 GirarDer()

```
void L298N::GirarDer (
    void )  [override], [virtual]
```

Gira a la derecha.

Energiza solamente el motor 2 de forma directa.

Implements [Puente_H](#).

7.28.3.5 GirarIzq()

```
void L298N::GirarIzq (
    void )  [override], [virtual]
```

Gira a la izquierda.

Energiza solamente el motor 1 de forma directa.

Implements [Puente_H](#).

7.28.3.6 Initialize()

```
void L298N::Initialize (
    void )  [override], [virtual]
```

Inicializa todas las salidas GPIO.

Setea la dirección y apaga todas las GPIO que posee.

Implements [Puente_H](#).

7.28.3.7 Retroceder()

```
void L298N::Retroceder (
    void )  [override], [virtual]
```

Pone los motores marcha atrás.

Enciende los motores en sentido contrario a "avanzar".

Implements [Puente_H](#).

The documentation for this class was generated from the following files:

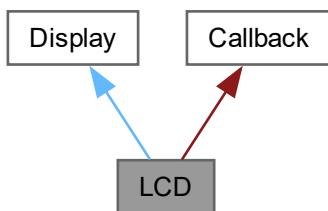
- [L298N.h](#)
- [L298N.cpp](#)

7.29 LCD Class Reference

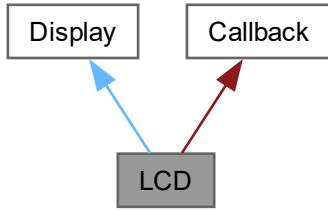
Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

```
#include <LCD.h>
```

Inheritance diagram for LCD:



Collaboration diagram for LCD:



Public Types

- enum

Public Member Functions

- `LCD (gpio *salidas[MAX_PIN_COUNT], const uint8_t filas, const uint8_t columnas)`
Constructor de clase LCD.
- void `Initialize (void)`
Inicializa el LCD.
- void `Write (const char *s)`
Escribe en el LCD.
- void `Write (const int32_t n)`
Escribe en el LCD un número.
- `LCD & operator= (const char *s)`
Escribe en el LCD. Sobre carga del operador =.
- void `WriteAt (const int8_t *a, const uint8_t columna, const uint8_t fila)`
Escribe el LCD.
- void `WriteAt (const int32_t n, const uint8_t columna, const uint8_t fila)`
Escribe el LCD.
- void `Clear (void) override`
Limpia el LCD.

Public Member Functions inherited from [Display](#)

- `Display ()=default`
- virtual `~Display ()=default`

Protected Member Functions

- void `SWhandler (void) override`
Ejecuta la instrucción del LCD periodicamente.

7.29.1 Detailed Description

Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

7.29.2 Member Enumeration Documentation

7.29.2.1 anonymous enum

```
anonymous enum
```

Posiciones del buffer.

7.29.3 Constructor & Destructor Documentation

7.29.3.1 LCD()

```
LCD::LCD (
    gpio * salidas[MAX_PIN_COUNT],
    const uint8_t filas,
    const uint8_t columnas)
```

Constructor de clase [LCD](#).

Crea [LCD](#) PIN con los parámetros correspondientes

Parameters

in	<i>salidas</i>	Vector de GPIO ordenado utilizadas para el LCD .
in	<i>filas</i>	Cantidad de filas del display.
in	<i>columnas</i>	Cantidad de columnas del display.

7.29.4 Member Function Documentation

7.29.4.1 Clear()

```
void LCD::Clear (
    void) [override], [virtual]
```

Limpia el [LCD](#).

Deja el buffer como un string de caracteres ESPACIO que imprimen en blanco.

Implements [Display](#).

7.29.4.2 Initialize()

```
void LCD::Initialize (
    void )
```

Inicializa el [LCD](#).

Crea el buffer y comienza a setear todas las salidas para comenzar a funcionar

7.29.4.3 operator=()

```
LCD & LCD::operator= (
    const char * s)
```

Escribe en el [LCD](#). Sobrecarga del operador =.

Escribe el buffer con el string indicado comenzando en la posición (0,0).

Parameters

in	s	String a escribir.
----	---	--------------------

Returns

void

7.29.4.4 SWhandler()

```
void LCD::SWhandler (
    void ) [override], [protected], [virtual]
```

Ejecuta la instrucción del [LCD](#) periódicamente.

Inicializa y escribe el [LCD](#) periódicamente utilizando el handler del systick.

Implements [Callback](#).

7.29.4.5 Write() [1/2]

```
void LCD::Write (
    const char * s)
```

Escribe en el [LCD](#).

Escribe el buffer con el string indicado comenzando en la posición (0,0).

Parameters

in	s	String a escribir.
----	---	--------------------

7.29.4.6 Write() [2/2]

```
void LCD::Write (
    const int32_t n) [virtual]
```

Escribe en el [LCD](#) un número.

Escribe el buffer con el string indicado comenzando en la posición (0,0). El número tendrá un máximo de 10 dígitos, sin contar el - de signo

Parameters

in	n	Número a escribir.
----	---	--------------------

Implements [Display](#).

7.29.4.7 WriteAt() [1/2]

```
void LCD::WriteAt (
    const int32_t n,
    const uint8_t fila,
    const uint8_t columna)
```

Escribe el [LCD](#).

Escribe en el buffer un número del [LCD](#) desde la posición indicada.

Parameters

in	<i>n</i>	numero a escribir.
in	<i>fila</i>	Fila donde empezar a escribir.
in	<i>columna</i>	Columna donde empezar a escribir.

7.29.4.8 WriteAt() [2/2]

```
void LCD::WriteAt (
    const int8_t * a,
    const uint8_t columna,
    const uint8_t fila)
```

Escribe el [LCD](#).

Escribe en el buffer del [LCD](#) desde la posición indicada.

Parameters

in	<i>a</i>	string a escribir.
in	<i>fila</i>	Fila donde empezar a escribir.
in	<i>columna</i>	Columna donde empezar a escribir.

The documentation for this class was generated from the following files:

- [LCD.h](#)
- [LCD.cpp](#)

7.30 MRT_t Struct Reference

Structure type to access the MRT timer.

```
#include <LPC845.h>
```

7.30.1 Detailed Description

Structure type to access the MRT timer.

MRT - Register Layout Typedef

7.30.2 Member Data Documentation

7.30.2.1 MODE

```
__RW uint32_t MRT_t::MODE
```

Enable timer interrupt

7.30.2.2 RESERVED0

`__R uint32_t MRT_t::RESERVED0`

Actual value of the timer. READ only RESERVED. NOT USED

7.30.2.3 RESERVED1

`__RW uint32_t MRT_t::RESERVED1`

Mode. REPEAT INTERRUPT 0, ONESHOT INTERRUPT 1, ONESHOT BUS STAL 2 RESERVED. NOT USED

7.30.2.4 RESERVED2

`__RW uint32_t MRT_t::RESERVED2`

READ ONLY. indicate the state of the timer. 0=idle, 1=running RESERVED. NOT USED

7.30.2.5 RUN

`__RW uint32_t MRT_t::RUN`

Indicate the interrupt request. 1=pending interrupt

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

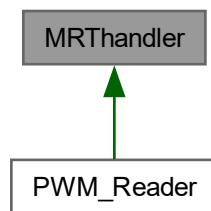
- [LPC845.h](#)

7.31 MRThandler Class Reference

Clase del objeto [MRThandler](#) El objeto [MRThandler](#) debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.

```
#include <MRThandler.h>
```

Inheritance diagram for MRThandler:



Public Member Functions

- **MRThandler** (`MRT_timer_channels _timer_number, MRT_MODES mode`)
Constructor de clase `MRThandler`.
- virtual `~MRThandler ()=default`
- virtual void `MRThandler (void)=0`
- void `EnableInterrupt (void)`
Activa la interrupcion.
- void `DisableInterrupt (void)`
Desactiva la interrupcion.

Public Attributes

- const `MRT_timer_channels m_timer_channel`

Protected Member Functions

- void `MRT_reset_time (void)`
Resetea el timer.
- `uint32_t MRT_get_time (void)`
Devuelve el valor del timer en el momento actual.

7.31.1 Detailed Description

Clase del objeto `MRThandler` El objeto `MRThandler` debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 `MRThandler()`

```
MRThandler::MRThandler (
    MRT_timer_channels _timer_number,
    MRT_MODES mode)
```

Constructor de clase `MRThandler`.

Crea un `MRThandler` con el canal correspondiente

Parameters

in	<code>_timer_number</code>	Canal del <code>MRThandler</code> .
in	<code>mode</code>	Modo del <code>MRThandler</code> .

7.31.2.2 `~MRThandler()`

```
virtual MRThandler::~MRThandler () [virtual], [default]
```

Destructor por defecto

7.31.3 Member Function Documentation

7.31.3.1 MRT_get_time()

```
uint32_t MRThandler::MRT_get_time (
    void ) [protected]
```

Devuelve el valor del timer en el momento actual.

Lee el registro del contador MRT del canal correspondiente.

Returns

devuelve el valor del timer.

7.31.3.2 MRT_reset_time()

```
void MRThandler::MRT_reset_time (
    void ) [protected]
```

Resetea el timer.

Vuelve a cargar el tiempo máximo en el registro.

7.31.3.3 MRTHandler()

```
virtual void MRThandler::MRTHandler (
    void ) [pure virtual]
```

Este método debe ser implementado por las clases derivadas cada una resolverá que hacer con su irq enganchada al systick del sistema

7.31.4 Member Data Documentation

7.31.4.1 m_timer_channel

```
const MRT_timer_channels MRThandler::m_timer_channel
```

Canal del objeto MRT

The documentation for this class was generated from the following files:

- [MRTHandler.h](#)
- [MRTHandler.cpp](#)

7.32 NVIC_Type Struct Reference

Structure type to access the Nested Vectored Interrupt Controller (NVIC).

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t ISER [1U]`
- `uint32_t RESERVED0 [31U]`
- `__RW uint32_t ICER [1U]`
- `uint32_t RSERVED1 [31U]`
- `__RW uint32_t ISPR [1U]`
- `uint32_t RESERVED2 [31U]`
- `__RW uint32_t RESERVED3 [31U]`
- `uint32_t RESERVED4 [64U]`
- `__RW uint32_t IP [8U]`

7.32.1 Detailed Description

Structure type to access the Nested Vectored Interrupt Controller (NVIC).

NVIC - Register Layout Typedef

7.32.2 Member Data Documentation

7.32.2.1 ICER

`__RW uint32_t NVIC_Type::ICER[1U]`

Offset: 0x080 (R/W) Interrupt Clear Enable Register

7.32.2.2 IP

`__RW uint32_t NVIC_Type::IP[8U]`

Offset: 0x300 (R/W) Interrupt Priority Register

7.32.2.3 ISER

`__RW uint32_t NVIC_Type::ISER[1U]`

Offset: 0x000 (R/W) Interrupt Set Enable Register

7.32.2.4 ISPR

`__RW uint32_t NVIC_Type::ISPR[1U]`

Offset: 0x100 (R/W) Interrupt Set Pending Register

7.32.2.5 RESERVED0

```
uint32_t NVIC_Type::RESERVED0[31U]
```

RESERVED. NOT USED

7.32.2.6 RESERVED2

```
uint32_t NVIC_Type::RESERVED2[31U]
```

RESERVED. NOT USED

7.32.2.7 RESERVED3

```
__RW uint32_t NVIC_Type::RESERVED3[31U]
```

RESERVED. NOT USED

7.32.2.8 RESERVED4

```
uint32_t NVIC_Type::RESERVED4[64U]
```

RESERVED. NOT USED

7.32.2.9 RSERVED1

```
uint32_t NVIC_Type::RSERVED1[31U]
```

RESERVED. NOT USED

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

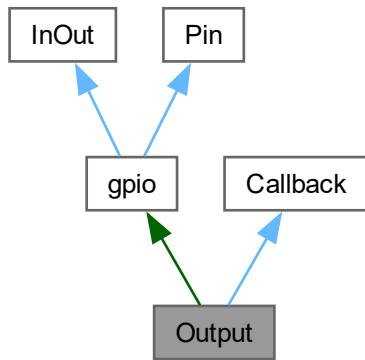
- [LPC845.h](#)

7.33 Output Class Reference

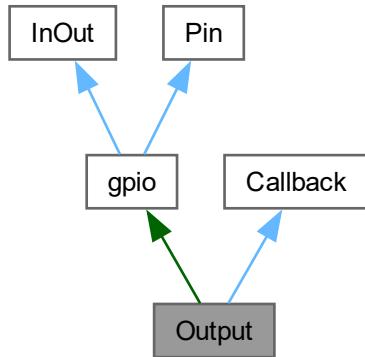
Clase del objeto outputs.

```
#include <Output.h>
```

Inheritance diagram for Output:



Collaboration diagram for Output:



Public Member Functions

- `Output (port_t puerto, uint8_t bit, mode_t modo, activity_t actividad=high, uint8_t estado=on)`
Constructor de clase `Output`.
- `int8_t On (void)`

- `int8_t Off (void)`
Funcion de encendido de la output.
- `int8_t Initialize (void)`
Funcion de configuracion del output.
- `Output & operator= (uint8_t estado)`
Sobrecarga del operador =.
- `bool operator== (uint8_t a)`
Sobrecarga del operador ==.
- `void SWHandler (void) override`
Funcion de interrupcion del systick.
- `virtual ~Output ()`

Public Member Functions inherited from [Callback](#)

- `void SetInterrupt ()`
Activa la interrupcion.
- `void UnSetInterrupt ()`
Desactiva la interrupcion.

Additional Inherited Members

Protected Types inherited from [gpio](#)

- enum `direction_t`
Enumeracion de input/output.
- enum `power_t`
Enumeracion de encendido/apagado.
- enum `mode_t`
Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum `activity_t`
Enumeracion de activo bajo/alto.
- enum `interrupt_mode_t`
Enumeracion de interrupciones del pin.
- typedef enum `gpio::direction_t direction_t`
Enumeracion de input/output.
- typedef enum `gpio::mode_t mode_t`
Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum `gpio::activity_t activity_t`
Enumeracion de activo bajo/alto.

Protected Types inherited from [Pin](#)

- enum `port_t`
- enum `max_bits_port_t`
máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Protected Member Functions inherited from [gpio](#)

- [gpio \(port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high\)](#)
Constructor de clase GPIO.
- [uint8_t SetPin \(void\) override](#)
Enciende la salida.
- [uint8_t ClrPin \(void\) override](#)
Limpia el pin.
- [uint8_t SetTogglePin \(void\) override](#)
Toggle del pin.
- [uint8_t SetDir \(void\) override](#)
Setea la dirección.
- [uint8_t SetToggleDir \(void\) override](#)
Toggle de la dirección.
- [uint8_t GetPin \(void\) override](#)
Devuelve el valor del pin.
- [uint8_t SetPinMode \(void\) override](#)
Configura el modo del pin.
- [uint8_t SetPinResistor \(void\) override](#)
Setea la resistencia interna.
- [gpio & operator= \(uint8_t a\)](#)
Sobrecarga del operador =.
- [virtual ~gpio \(\)=default](#)

Protected Member Functions inherited from [InOut](#)

- [InOut \(\)=default](#)
- [virtual ~InOut \(\)=default](#)

Protected Member Functions inherited from [Pin](#)

- [Pin \(port_t port, uint8_t bit\)](#)
Constructor de clase PIN.

Protected Member Functions inherited from [Callback](#)

- [void SetInterrupt \(\)](#)
Activa la interrupción.
- [void UnSetInterrupt \(\)](#)
Desactiva la interrupción.

Protected Attributes inherited from [gpio](#)

- [const mode_t m_mode](#)
- [direction_t m_direction](#)
- [const activity_t m_activity](#)

Protected Attributes inherited from Pin

- const `port_t m_port`
- const `uint8_t m_bit`
- `int8_t m_error`

7.33.1 Detailed Description

Clase del objeto outputs.

El objeto outputs Permite el manejo de salidas de forma controlada con el systick y un buffer.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 Output()

```
Output::Output (
    port_t puerto,
    uint8_t bit,
    mode_t modo,
    activity_t actividad = high,
    uint8_t estado = on)
```

Constructor de clase `Output`.

Crea un `Output` con los parámetros correspondientes.

Parameters

in	<i>puerto</i>	Puerto del objeto.
in	<i>bit</i>	Bit del objeto.
in	<i>modo</i>	Configuración eléctrica del pin.
in	<i>actividad</i>	Activo alto/bajo.
in	<i>estado</i>	Estado inicial.

7.33.2.2 ~Output()

```
Output::~Output () [virtual]
```

Destructor por defecto

7.33.3 Member Function Documentation

7.33.3.1 Initialize()

```
int8_t Output::Initialize (
    void )
```

Funcion de configuracion del output.

Setea la direccion y resistencia de la salida.

Returns

devuelve el error.

7.33.3.2 Off()

```
int8_t Output::Off (
    void )
```

Funcion de apagado de la output.

Guarda en el buffer el estado de apagado.

Returns

devuelve el error.

7.33.3.3 On()

```
int8_t Output::On (
    void )
```

Funcion de encendido de la output.

Guarda en el buffer el estado de encendido.

Returns

devuelve el error.

7.33.3.4 operator=()

```
Output & Output::operator= (
    uint8_t estado)
```

Sobrecarga del operador =.

Enciende la salida si se iguala a 1, apaga con 0.

Parameters

in	estado	Valor de igualacion.
----	--------	----------------------

Returns

Referencia a si mismo.

7.33.3.5 operator==()

```
bool Output::operator== (
    uint8_t a)
```

Sobrecarga del operador ==.

Indica si la salida se encuentra en el estado "a" o no.

Parameters

in	a	Valor a comparar el buffer.
----	---	-----------------------------

Returns

verdadero o falso.

7.33.3.6 SWhandler()

```
void Output::SWhandler (
    void ) [override], [virtual]
```

Funcion de interrupcion del systick.

Coloca la salida con el valor del buffer.

Implements [Callback](#).

The documentation for this class was generated from the following files:

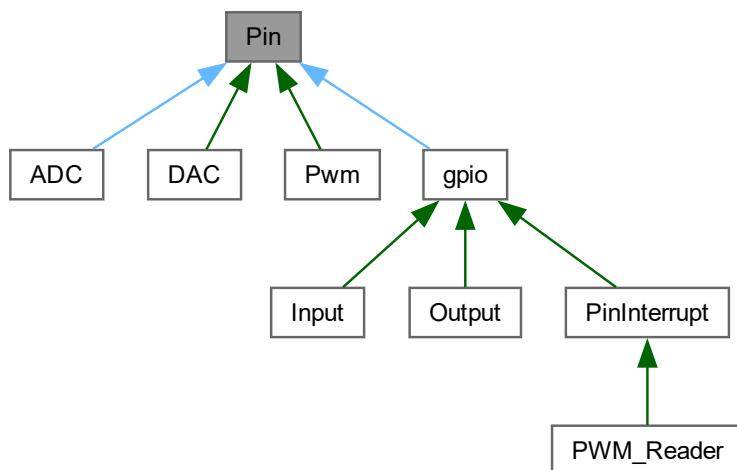
- [Output.h](#)
- [Output.cpp](#)

7.34 Pin Class Reference

Clase del objeto [Pin](#).

```
#include <Pin.h>
```

Inheritance diagram for Pin:



Public Types

- enum `port_t`
- enum `max_bits_port_t`
máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Public Member Functions

- `Pin (port_t port, uint8_t bit)`
Constructor de clase PIN.

Public Attributes

- const `port_t m_port`
- const `uint8_t m_bit`
- `int8_t m_error`

7.34.1 Detailed Description

Clase del objeto `Pin`.

El objeto `Pin` debe ser heredado y posee el puerto y bit del periférico en cuestión

7.34.2 Member Typedef Documentation

7.34.2.1 `error_t`

```
typedef enum Pin::error_t Pin::error_t
```

Enumeración de error

7.34.2.2 `port_t`

```
typedef enum Pin::port_t Pin::port_t
```

Enumeración con los puertos

7.34.3 Member Enumeration Documentation

7.34.3.1 `error_t`

```
enum Pin::error_t
```

Enumeración de error

7.34.3.2 port_t

```
enum Pin::port_t
```

Enumeracion con los puertos

7.34.4 Constructor & Destructor Documentation

7.34.4.1 Pin()

```
Pin::Pin (
    port_t port,
    uint8_t bit)
```

Constructor de clase PIN.

Crea un PIN con los parámetros correspondientes

Parameters

in	port	Puerto del objeto
in	bit	Bit del objeto

7.34.5 Member Data Documentation

7.34.5.1 m_bit

```
const uint8_t Pin::m_bit
```

Guarda el bit del pin

7.34.5.2 m_error

```
int8_t Pin::m_error
```

Error en la clase pin

7.34.5.3 m_port

```
const port_t Pin::m_port
```

Guarda el puerto del pin

The documentation for this class was generated from the following files:

- [Pin.h](#)
- [Pin.cpp](#)

7.35 PIN_INTERRUPT_t Struct Reference

Structure type to access the Pin Interrupt.

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t ISEL`
- `__RW uint32_t IENR`
- `__W uint32_t SIENR`
- `__W uint32_t CIENR`
- `__RW uint32_t IENF`
- `__W uint32_t SIENF`
- `__W uint32_t CIENF`
- `__RW uint32_t RISE`
- `__RW uint32_t FALL`
- `__RW uint32_t IST`
- `__RW uint32_t PMCTRL`
- `__RW uint32_t PMSRC`
- `__RW uint32_t PMCFG`

7.35.1 Detailed Description

Structure type to access the Pin Interrupt.

PIN_INTERRUPT - Register Layout Typedef

7.35.2 Member Data Documentation

7.35.2.1 CIENF

```
__W uint32_t PIN_INTERRUPT_t::CIENF
```

Write to disable falling-edge interrupts

7.35.2.2 CIENR

```
__W uint32_t PIN_INTERRUPT_t::CIENR
```

Write to disable rising-edge interrupts.

7.35.2.3 FALL

```
__RW uint32_t PIN_INTERRUPT_t::FALL
```

GIVES WHITCH PORTS HAVE DETECTED A FALLING EDGE

7.35.2.4 IENF

`__RW uint32_t PIN_INTERRUPT_t::IENF`

Enables falling-edge interrupts

7.35.2.5 IENR

`__RW uint32_t PIN_INTERRUPT_t::IENR`

Enables rising-edge interrupts.

7.35.2.6 ISEL

`__RW uint32_t PIN_INTERRUPT_t::ISEL`

ASSIGN LEVEL SENSITIVE OR EDGE SENSITIVE

7.35.2.7 IST

`__RW uint32_t PIN_INTERRUPT_t::IST`

GIVES WITCH PINES HAVE A INTERRUPT REQUEST

7.35.2.8 PMCFG

`__RW uint32_t PIN_INTERRUPT_t::PMCFG`

Pattern match interrupt bit slice configuration register

7.35.2.9 PMCTRL

`__RW uint32_t PIN_INTERRUPT_t::PMCTRL`

Pattern match interrupt control register

7.35.2.10 PMSRC

`__RW uint32_t PIN_INTERRUPT_t::PMSRC`

Pattern match interrupt bit-slice source register

7.35.2.11 RISE

`__RW uint32_t PIN_INTERRUPT_t::RISE`

GIVES WHITCH PORTS HAVE DETECTED A RISING EDGE

7.35.2.12 SIENF

```
__W uint32_t PIN_INTERRUPT_t::SIENF
```

Write to enable falling-edge interrupts.

7.35.2.13 SIENR

```
__W uint32_t PIN_INTERRUPT_t::SIENR
```

Write to enable rising-edge interrupts

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

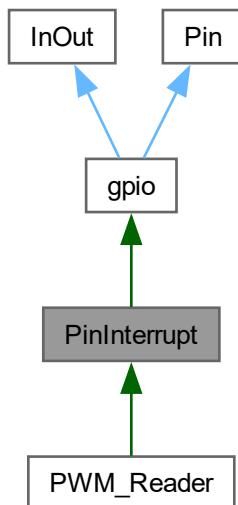
- [LPC845.h](#)

7.36 PinInterrupt Class Reference

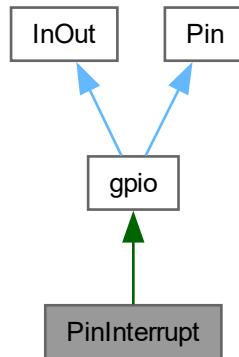
Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interrupciones por pin.

```
#include <Pininterrupt.h>
```

Inheritance diagram for PinInterrupt:



Collaboration diagram for PinInterrupt:



Public Member Functions

- `PinInterrupt (port_t port, uint8_t bit, mode_t gpio_mode, activity_t activity, uint8_t intrp_mode)`
Constructor de clase Pin_interrupt.
- `void EnableInterrupt (void)`
Habilito la interrupción.
- `void DisableInterrupt (void)`
Deshabilito la interrupción.
- `void PinInterrupt_Inicializar (void)`
Inicializo la interrupción por pin.
- `virtual void GpioHandler (void)=0`
- `virtual ~PinInterrupt ()`

Public Attributes

- `const uint8_t m_interrupt_number`
- `const uint8_t m_interrupt_mode`

Static Public Attributes

- `static uint8_t m_cant = 0`

Additional Inherited Members

Protected Types inherited from gpio

- enum `direction_t`
Enumeracion de input/output.
- enum `power_t`

- enum `mode_t`

Enumeracion de encendido/apagado.
- enum `activity_t`

Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum `interrupt_mode_t`

Enumeracion de activo bajo/alto.
- enum `interrupt_mode_t direction_t`

Enumeracion de input/output.
- typedef enum `gpio::direction_t direction_t`

Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum `gpio::mode_t mode_t`

Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum `gpio::activity_t activity_t`

Enumeracion de activo bajo/alto.

Protected Types inherited from Pin

- enum `port_t`
- enum `max_bits_port_t`

máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Protected Member Functions inherited from gpio

- `gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)`

Constructor de clase GPIO.
- `uint8_t SetPin (void) override`

Enciende la salida.
- `uint8_t ClrPin (void) override`

Limpia el pin.
- `uint8_t SetTogglePin (void) override`

Toggle del pin.
- `uint8_t SetDir (void) override`

Setea la dirección.
- `uint8_t SetToggleDir (void) override`

Toggle de la dirección.
- `uint8_t GetPin (void) override`

Devuelve el valor del pin.
- `uint8_t SetPinMode (void) override`

Configura el modo del pin.
- `uint8_t SetPinResistor (void) override`

Setea la resistencia interna.
- `gpio & operator= (uint8_t a)`

Sobrecarga del operador =.
- virtual `~gpio ()=default`

Protected Member Functions inherited from [InOut](#)

- [InOut \(\)](#)=default
- virtual [~InOut \(\)](#)=default

Protected Member Functions inherited from [Pin](#)

- [Pin \(port_t port, uint8_t bit\)](#)
Constructor de clase PIN.

Protected Attributes inherited from [gpio](#)

- const [mode_t m_mode](#)
- [direction_t m_direction](#)
- const [activity_t m_activity](#)

Protected Attributes inherited from [Pin](#)

- const [port_t m_port](#)
- const [uint8_t m_bit](#)
- [int8_t m_error](#)

7.36.1 Detailed Description

Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interrupciones por pin.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 PinInterrupt()

```
PinInterrupt::PinInterrupt (
    port_t port,
    uint8_t bit,
    mode_t gpio_mode,
    activity_t activity,
    uint8_t intrp_mode)
```

Constructor de clase Pin_interrupt.

Crea un Pin_interrupt con los parámetros correspondientes

Parameters

in	<i>port</i>	Puerto del pin interrupt.
in	<i>bit</i>	bit del pin interrupt.
in	<i>gpio_mode</i>	Configuración eléctrica del pin.
in	<i>activity</i>	Activo alto/bajo.
in	<i>intrp_mode</i>	Tipo de interrupción.

7.36.2.2 ~PinInterrupt()

```
PinInterrupt::~PinInterrupt () [virtual]
```

Destructor por defecto

7.36.3 Member Function Documentation

7.36.3.1 DisableInterrupt()

```
void PinInterrupt::DisableInterrupt (
    void )
```

Deshabilito la interrupción.

Deshabilito la interrupción de la pata específica que estoy utilizando.

7.36.3.2 EnableInterrupt()

```
void PinInterrupt::EnableInterrupt (
    void )
```

Habilito la interrupción.

Habilito la interrupción de la pata específica que estoy utilizando.

7.36.3.3 GpioHandler()

```
virtual void PinInterrupt::GpioHandler (
    void ) [pure virtual]
```

Funcion Handler que se ejecuta al realizarse la interrupcion

Implemented in [PWM_Reader](#).

7.36.3.4 PinInterrupt_Inicializar()

```
void PinInterrupt::PinInterrupt_Inicializar (
    void )
```

Inicializo la interrupción por pin.

Modifico todos los registros para que la interrupción por pin esté configurada.

7.36.4 Member Data Documentation

7.36.4.1 m_cant

```
uint8_t PinInterrupt::m_cant = 0 [static]
```

Cantidad de PIN INTERRUPT creados globalmente

7.36.4.2 m_interrupt_mode

```
const uint8_t PinInterrupt::m_interrupt_mode
```

Tipo de interrupción del pin

7.36.4.3 m_interrupt_number

```
const uint8_t PinInterrupt::m_interrupt_number
```

Número de PIN INTERRUPT

The documentation for this class was generated from the following files:

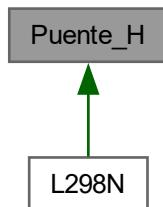
- [Pininterrupt.h](#)
- [Pininterrupt.cpp](#)

7.37 Puente_H Class Reference

Clase del objeto [Puente_H](#) El objeto [Puente_H](#) es la interfaz abstracta pura de cualquier puente H que se desee realizar.

```
#include <PuenteH.h>
```

Inheritance diagram for Puente_H:



Public Types

- enum

Public Member Functions

- `Puente_H ()=default`
- `virtual void Initialize (void)=0`
- `virtual void GirarIzq (void)=0`
- `virtual void GirarDer (void)=0`
- `virtual void Girar (uint8_t direccion)=0`
- `virtual void Frenar (void)=0`
- `virtual void Avanzar (void)=0`
- `virtual void Retroceder (void)=0`
- `virtual ~Puente_H ()=default`

7.37.1 Detailed Description

Clase del objeto `Puente_H` El objeto `Puente_H` es la interfaz abstracta pura de cualquier puente H que se desee realizar.

7.37.2 Member Enumeration Documentation

7.37.2.1 anonymous enum

anonymous enum

Enumeracion de sentidos de giro

7.37.3 Constructor & Destructor Documentation

7.37.3.1 `Puente_H()`

`Puente_H::Puente_H () [default]`

Constructor por defecto

7.37.3.2 `~Puente_H()`

`virtual Puente_H::~Puente_H () [virtual], [default]`

Destructor por defecto

7.37.4 Member Function Documentation

7.37.4.1 `Avanzar()`

```
virtual void Puente_H::Avanzar (
    void) [pure virtual]
```

Avanza los motores

Implemented in [L298N](#).

7.37.4.2 Frenar()

```
virtual void Puente_H::Frenar (
    void ) [pure virtual]
```

Frena los motores

Implemented in [L298N](#).

7.37.4.3 Girar()

```
virtual void Puente_H::Girar (
    uint8_t direccion) [pure virtual]
```

Gira los motores

Implemented in [L298N](#).

7.37.4.4 GirarDer()

```
virtual void Puente_H::GirarDer (
    void ) [pure virtual]
```

Gira a la derecha

Implemented in [L298N](#).

7.37.4.5 GirarIzq()

```
virtual void Puente_H::GirarIzq (
    void ) [pure virtual]
```

Gira a la izquierda

Implemented in [L298N](#).

7.37.4.6 Initialize()

```
virtual void Puente_H::Initialize (
    void ) [pure virtual]
```

Inicializa el Puente H

Implemented in [L298N](#).

7.37.4.7 Retroceder()

```
virtual void Puente_H::Retroceder (
    void ) [pure virtual]
```

Retrocede los motores

Implemented in [L298N](#).

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

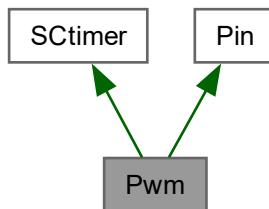
- [PuenteH.h](#)

7.38 Pwm Class Reference

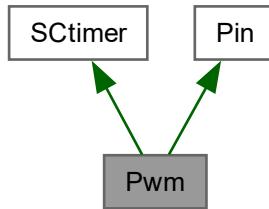
Clase del objeto [Pwm](#).

```
#include <Pwm.h>
```

Inheritance diagram for Pwm:



Collaboration diagram for Pwm:



Public Types

- enum `pwm_time_unit_t`
- enum `pwm_channel_t`
- enum `activity_t`

Public Member Functions

- `Pwm (port_t puerto, uint8_t bit, uint8_t actividad, pwm_channel_t number)`
Constructor de clase Pwm.
- void `Initialize (uint32_t ton, uint32_t toff, pwm_time_unit_t t=MICRO_SEG)`
Inicializo el PWM.
- void `SetTon (uint32_t time, pwm_time_unit_t t=MICRO_SEG)`
Seteo el tiempo de encendido del PWM.
- void `SetPeriod (uint32_t time, pwm_time_unit_t t=MICRO_SEG)`
Seteo el periodo del PWM.
- void `On ()`
Enciende el PWM.
- void `Off ()`
Apaga el PWM.
- virtual `~Pwm ()`

Protected Attributes

- const `uint8_t m_activity`
- `uint32_t m_ton`
- `uint32_t m_toff`
- const `uint8_t m_pwm_channel`

Protected Attributes inherited from Pin

- const `port_t m_port`
- const `uint8_t m_bit`
- `int8_t m_error`

Additional Inherited Members

Protected Types inherited from Pin

- enum `port_t`
- enum `max_bits_port_t`
máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Protected Member Functions inherited from [SCTimer](#)

- [SCTimer \(\)](#)
Constructor de clase SCTimer.
- void [SetTime \(uint32_t time, uint32_t channel\)](#)
Setea el tiempo de un canal del contador.
- void [SetUnify \(bool a\)](#)
Unifica los dos registros del SCT.
- void [SetAutoLimit \(bool a\)](#)
Setea el autolímite.
- void [SetSwitchMatrizSCTOUT \(uint8_t bit, uint8_t port, uint8_t out_number\)](#)
Configura la switch Matrix.
- void [StartTimer \(void\)](#)
Habilita el timer.
- void [StopTimer \(void\)](#)
Deshabilita el timer.
- virtual [~SCTimer \(\)](#)

Protected Member Functions inherited from [Pin](#)

- [Pin \(port_t port, uint8_t bit\)](#)
Constructor de clase PIN.

7.38.1 Detailed Description

Clase del objeto [Pwm](#).

FUNCIONAMIENTO: Genera un PWM realizando un Match entre distintos eventos y señales del SCTimer. Por cómo está configurado, todos los objetos PWM tendrán el mismo periodo. Esto se debe a que todos funcionan con el Evento 0 que los resetea. Debido a esto, solo se puede tener hasta 6 PWM al mismo tiempo y todos tendrán el mismo PERIODO.

7.38.2 Member Enumeration Documentation

7.38.2.1 activity_t

```
enum Pwm::activity_t
```

Enum de activo alto/bajo

7.38.2.2 pwm_channel_t

```
enum Pwm::pwm_channel_t
```

Canales del PWM

7.38.2.3 `pwm_time_unit_t`

```
enum Pwm::pwm_time_unit_t
```

Unidad de tiempo del PWM

7.38.3 Constructor & Destructor Documentation

7.38.3.1 `Pwm()`

```
Pwm::Pwm (
    port_t puerto,
    uint8_t bit,
    uint8_t actividad,
    pwm_channel_t number)
```

Constructor de clase `Pwm`.

Crea un `Pwm` con los parámetros correspondientes

Parameters

in	<i>puerto</i>	Puerto del PWM.
in	<i>bit</i>	Bit del PWM.
in	<i>actividad</i>	Activo alto/bajo.
in	<i>number</i>	Canal del <code>SCtimer</code> a utilizar.

7.38.3.2 `~Pwm()`

```
Pwm::~Pwm () [virtual]
```

Destructor por defecto

7.38.4 Member Function Documentation

7.38.4.1 `Initialize()`

```
void Pwm::Initialize (
    uint32_t ton,
    uint32_t toff,
    pwm_time_unit_t t = MICRO_SEG)
```

Inicializo el PWM.

Utilizando los registros configuro todo para la utilizacion del PWM.

Parameters

in	<i>ton</i>	Tiempo de encendido.
in	<i>toff</i>	Tiempo de apagado (no es el periodo. El periodo es la suma de ambos).
in	<i>t</i>	Unidad de medida de los tiempos de encendido y apagado.

7.38.4.2 Off()

```
void Pwm::Off (
    void )
```

Apaga el PWM.

Utiliza los registros con StopTimer para deshabilitar la salida.

7.38.4.3 On()

```
void Pwm::On (
    void )
```

Enciende el PWM.

Utiliza los registros con StarTimer para habilitar la salida.

7.38.4.4 SetPeriod()

```
void Pwm::SetPeriod (
    uint32_t time,
    pwm_time_unit_t t = MICRO_SEG)
```

Seteo el periodo del PWM.

Utilizando los registros con SetTime creo el tiempo de apagado de mi PWM.

Parameters

in	<i>time</i>	Tiempo del periodo.
in	<i>t</i>	Unidad de medida del tiempo de periodo.

7.38.4.5 SetTon()

```
void Pwm::SetTon (
    uint32_t time,
    pwm_time_unit_t t = MICRO_SEG)
```

Seteo el tiempo de encendido del PWM.

Utilizando los registros con SetTime creo el tiempo de encendido de mi PWM.

Parameters

in	<i>time</i>	Tiempo de encendido.
in	<i>t</i>	Unidad de medida del tiempo de encendido.

7.38.5 Member Data Documentation

7.38.5.1 m_activity

```
const uint8_t Pwm::m_activity [protected]
```

Activo alto/bajo del PWM

7.38.5.2 m_pwm_channel

```
const uint8_t Pwm::m_pwm_channel [protected]
```

Canal del objeto PWM

7.38.5.3 m_toff

```
uint32_t Pwm::m_toff [protected]
```

Tiempo de apagado del PWM en micro segundos

7.38.5.4 m_ton

```
uint32_t Pwm::m_ton [protected]
```

Tiempo de encendido del PWM en micro segundos

The documentation for this class was generated from the following files:

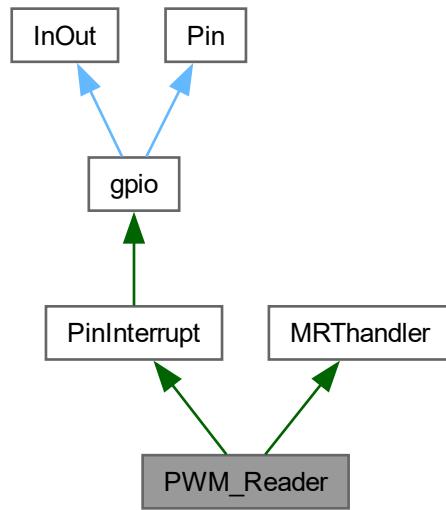
- [Pwm.h](#)
- [Pwm.cpp](#)

7.39 PWM_Reader Class Reference

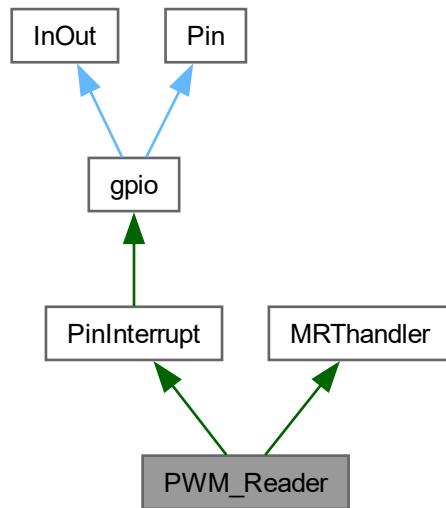
Clase del objeto [PWM_Reader](#).

```
#include <PWMReader.h>
```

Inheritance diagram for PWM_Reader:



Collaboration diagram for PWM_Reader:



Public Member Functions

- `PWM_Reader (port_t puerto, uint8_t bit, mode_t modo, activity_t activity, MRT_timer_channels timer_← channel)`

- *Constructor de clase PWM_Reader.*
- void **Initialize** (void)
Inicializo el PWM_In.
- **uint32_t GetPulseOn** (void) const
Obtengo el largo del pulso.
- void **Off** (void)
Apago el PWM_In.
- void **On** (void)
Enciendo el PWM_In.

Protected Member Functions

- void **GpioHandler** (void) override
Handler de la interrupción por flanco.

Protected Member Functions inherited from PinInterrupt

- **PinInterrupt** (*port_t* port, *uint8_t* bit, *mode_t* gpio_mode, *activity_t* activity, *uint8_t* intrp_mode)
Constructor de clase Pin_interrupt.
- void **EnableInterrupt** (void)
Habilito la interrupción.
- void **DisableInterrupt** (void)
Deshabilito la interrupción.
- void **PinInterrupt_Inicializar** (void)
Inicializo la interrupción por pin.
- virtual **~PinInterrupt** ()

Protected Member Functions inherited from gpio

- **gpio** (*port_t* port, *uint8_t* bit, *mode_t* mode, *direction_t* direction, *activity_t* activity=high)
Constructor de clase GPIO.
- **uint8_t SetPin** (void) override
Enciende la salida.
- **uint8_t ClrPin** (void) override
Limpia el pin.
- **uint8_t SetTogglePin** (void) override
Toggle del pin.
- **uint8_t SetDir** (void) override
Setea la dirección.
- **uint8_t SetToggleDir** (void) override
Toggle de la dirección.
- **uint8_t GetPin** (void) override
Devuelve el valor del pin.
- **uint8_t SetPinMode** (void) override
Configura el modo del pin.
- **uint8_t SetPinResistor** (void) override
Setea la resistencia interna.
- **gpio & operator=** (*uint8_t* a)
Sobrecarga del operador =.
- virtual **~gpio** ()=default

Protected Member Functions inherited from [InOut](#)

- [InOut \(\)=default](#)
- virtual [~InOut \(\)=default](#)

Protected Member Functions inherited from [Pin](#)

- [Pin \(port_t port, uint8_t bit\)](#)

Constructor de clase PIN.

Protected Member Functions inherited from [MRThandler](#)

- void [MRT_reset_time \(void\)](#)
Resetea el timer.
- [uint32_t MRT_get_time \(void\)](#)
Devuelve el valor del timer en el momento actual.
- [MRThandler \(MRT_timer_channels _timer_number, MRT_MODES mode\)](#)
Constructor de clase MRThandler.
- virtual [~MRThandler \(\)=default](#)
- virtual void [MRTHandler \(void\)=0](#)
- void [EnableInterrupt \(void\)](#)
Activa la interrupcion.
- void [DisableInterrupt \(void\)](#)
Desactiva la interrupcion.

Additional Inherited Members

Protected Types inherited from [gpio](#)

- enum [direction_t](#)
Enumeracion de input/output.
- enum [power_t](#)
Enumeracion de encendido/apagado.
- enum [mode_t](#)
Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum [activity_t](#)
Enumeracion de activo bajo/alto.
- enum [interrupt_mode_t](#)
Enumeracion de interrupciones del pin.
- typedef enum [gpio::direction_t direction_t](#)
Enumeracion de input/output.
- typedef enum [gpio::mode_t mode_t](#)
Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum [gpio::activity_t activity_t](#)
Enumeracion de activo bajo/alto.

Protected Types inherited from Pin

- enum `port_t`
- enum `max_bits_port_t`
máximos pines por puerto
- enum `error_t`
- typedef enum `Pin::port_t port_t`
- typedef enum `Pin::error_t error_t`

Protected Attributes inherited from PinInterrupt

- const `uint8_t m_interrupt_number`
- const `uint8_t m_interrupt_mode`

Protected Attributes inherited from gpio

- const `mode_t m_mode`
- `direction_t m_direction`
- const `activity_t m_activity`

Protected Attributes inherited from Pin

- const `port_t m_port`
- const `uint8_t m_bit`
- `int8_t m_error`

Protected Attributes inherited from MRTHandler

- const `MRT_timer_channels m_timer_channel`

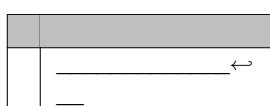
Static Protected Attributes inherited from PinInterrupt

- static `uint8_t m_cant = 0`

7.39.1 Detailed Description

Clase del objeto `PWM_Reader`.

Lee ancho de señal de una entrada en milisegundos. Solo lee pulsos completos. Si se queda en 0, no lo leerá hasta volver a 1. Lectura:



|—m_pulse_on—| —m_pulse_on-----

7.39.2 Constructor & Destructor Documentation

7.39.2.1 PWM_Reader()

```
PWM_Reader::PWM_Reader (
    port_t puerto,
    uint8_t bit,
    mode_t modo,
    activity_t activity,
    MRT_timer_channels timer_channel)
```

Constructor de clase [PWM_Reader](#).

Crea un [PWM_Reader](#) con los parámetros correspondientes

Parameters

in	<i>puerto</i>	Puerto del PWM_Reader .
in	<i>bit</i>	Bit del PWM_Reader .
in	<i>modo</i>	Configuracion eléctrica de la entrada.
in	<i>activity</i>	activo alto/bajo.
in	<i>timer_channel</i>	Canal del MRTtimer a utilizar.

7.39.3 Member Function Documentation**7.39.3.1 GetPulseOn()**

```
uint32_t PWM_Reader::GetPulseOn (
    void ) const
```

Obtengo el largo del pulso.

Realizo la cuenta del tiempo transcurrido entre el último flanco y este.

Returns

largo del pulso

7.39.3.2 GpioHandler()

```
void PWM_Reader::GpioHandler (
    void ) [override], [protected], [virtual]
```

Handler de la interrupción por flanco.

Destructor por defecto

Si el flanco es ascendente reseteo el contador. Si es descendente guado el valor del contador. La cuenta se realiza en otra función para ahorrar tiempo acá

Implements [PinInterrupt](#).

7.39.3.3 Initialize()

```
void PWM_Reader::Initialize (
    void )
```

Inicializo el PWM_In.

Seteo dirección resistencia y habilito la interrupción por flancos.

7.39.3.4 Off()

```
void PWM_Reader::Off (
    void )
```

Apago el PWM_In.

Deshabilito la interrupción y seteo el pulso a un valor gigante que no genere overflow.

7.39.3.5 On()

```
void PWM_Reader::On (
    void )
```

Enciendo el PWM_In.

Habilito la interrupción.

The documentation for this class was generated from the following files:

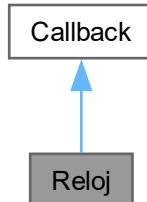
- [PWMReader.h](#)
- [PWMReader.cpp](#)

7.40 Reloj Class Reference

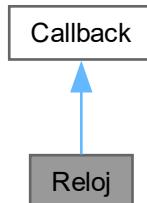
Clase del objeto [Reloj](#).

```
#include <Reloj.h>
```

Inheritance diagram for Reloj:



Collaboration diagram for Reloj:



Public Member Functions

- [Reloj \(\)](#)
Constructor de clase Reloj.
- [int32_t GetHour \(\) const](#)
Devuelve el valor de hora.
- [int32_t GetMin \(\) const](#)
Devuelve el valor de minutos.
- [int32_t GetSeg \(\) const](#)
Devuelve el valor de segundos.
- [void Reset \(\)](#)
Resetea el reloj.
- [void SetTime \(const int32_t _hour, const int32_t _min=-1, const int32_t _seg=-1\)](#)
Setea el reloj.

Public Member Functions inherited from [Callback](#)

- [void SetInterrupt \(\)](#)
Activa la interrupción.

- void [UnSetInterrupt \(\)](#)
Desactiva la interrupción.

Protected Member Functions

- void [SWhandler \(void\) override](#)
Actualiza el reloj.

7.40.1 Detailed Description

Clase del objeto [Reloj](#).

El objeto [Reloj](#) se comportará como un reloj que iniciará a contar desde 0hs o desde el valor asignado por el usuario.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 Reloj()

```
Reloj::Reloj ()  

Constructor de clase Reloj.  

Crea un Reloj por defecto
```

7.40.3 Member Function Documentation

7.40.3.1 GetHour()

```
int32_t Reloj::GetHour (  

    void ) const  

Devuelve el valor de hora.  

Entrega la variable m_hora.
```

Returns

m_hora

7.40.3.2 GetMin()

```
int32_t Reloj::GetMin (   

    void ) const  

Devuelve el valor de minutos.  

Entrega la variable m_minutos.
```

Returns

m_minutos

7.40.3.3 GetSeg()

```
int32_t Reloj::GetSeg (   

    void ) const  

Devuelve el valor de segundos.  

Entrega la variable m_segundos.
```

Returns

m_segundos

7.40.3.4 Reset()

```
void Reloj::Reset (
    void )
```

Resetea el reloj.

Coloca todas las variables en 0.

7.40.3.5 SetTime()

```
void Reloj::SetTime (
    const int32_t _hour,
    const int32_t _min = -1,
    const int32_t _seg = -1)
```

Setea el reloj.

Parameters

in	<i>_hour</i>	horas a setear.
in	<i>_min</i>	minutos a setear.
in	<i>_seg</i>	segundos a setear.

Configura el valor actual de horas, minutos y segundos.

7.40.3.6 SWhandler()

```
void Reloj::SWhandler (
    void ) [override], [protected], [virtual]
```

Actualiza el reloj.

Cada 1000 ticks pasa un segundo. Lo suma a la cuenta.

Implements [Callback](#).

The documentation for this class was generated from the following files:

- [Reloj.h](#)
- [Reloj.cpp](#)

7.41 SCT_t Struct Reference

Structure type to access the SCT.

```
#include <LPC845.h>
```

Public Attributes

- [RW uint32_t CONFIG](#)
- [RW uint32_t CTRL](#)
- [RW uint32_t LIMIT](#)
- [RW uint32_t HALT](#)
- [RW uint32_t STOP](#)
- [RW uint32_t START](#)
- [uint8_t RESERVED_0 \[40\]](#)
- [RW uint32_t COUNT](#)
- [RW uint32_t STATE](#)
- [R uint32_t INPUT](#)
- [RW uint32_t REGMODE](#)
- [RW uint32_t OUTPUT](#)
- [RW uint32_t OUTPUTDIRCTRL](#)
- [RW uint32_t RES](#)
- [RW uint32_t DMAREQ0](#)

- `__RW uint32_t DMAREQ1`
- `uint8_t RESERVED_1 [140]`
- `__RW uint32_t EVEN`
- `__RW uint32_t EVFLAG`
- `__RW uint32_t CONEN`
- `__RW uint32_t CONFLAG`
- `uint8_t RESERVED_2 [224]`
- `uint8_t RESERVED_3 [224]`
- struct {
 - `__RW uint32_t STATE`
 - `__RW uint32_t CTRL`
} `EV` [8]
- `uint8_t RESERVED_4 [448]`
- struct {
 - `__RW uint32_t SET`
 - `__RW uint32_t CLR`
} `OUT` [7]

7.41.1 Detailed Description

Structure type to access the SCT.

SCT - Register Layout Typedef

7.41.2 Member Data Documentation

7.41.2.1 CAP

`__RW uint32_t SCT_t::CAP [8]`

SCT capture register of capture channel, array offset: 0x100, array step: 0x4

7.41.2.2 CAPCTRL

`__RW uint32_t SCT_t::CAPCTRL [8]`

SCT capture control register, array offset: 0x200, array step: 0x4

7.41.2.3 CLR

`__RW uint32_t SCT_t::CLR`

SCT output 0 clear register, array offset: 0x504, array step: 0x8

7.41.2.4 CONEN

`__RW uint32_t SCT_t::CONEN`

SCT conflict interrupt enable register, offset: 0xF8

7.41.2.5 CONFIG

`__RW uint32_t SCT_t::CONFIG`

SCT configuration register, offset: 0x0

7.41.2.6 CONFLAG

`__RW uint32_t SCT_t::CONFLAG`

SCT conflict flag register, offset: 0xFC

7.41.2.7 COUNT

`__RW uint32_t` `SCT_t::COUNT`
SCT counter register, offset: 0x40

7.41.2.8 CTRL

`__RW uint32_t` `SCT_t::CTRL`
SCT control register, offset: 0x4
SCT event control register 0, array offset: 0x304, array step: 0x8

7.41.2.9 DMAREQ0

`__RW uint32_t` `SCT_t::DMAREQ0`
SCT DMA request 0 register, offset: 0x5C

7.41.2.10 DMAREQ1

`__RW uint32_t` `SCT_t::DMAREQ1`
SCT DMA request 1 register, offset: 0x60

7.41.2.11 [struct]

struct { ... } `SCT_t::EV[8]`
SCT event state register

7.41.2.12 EVEN

`__RW uint32_t` `SCT_t::EVEN`
SCT event interrupt enable register, offset: 0xF0

7.41.2.13 EVFLAG

`__RW uint32_t` `SCT_t::EVFLAG`
SCT event flag register, offset: 0xF4

7.41.2.14 HALT

`__RW uint32_t` `SCT_t::HALT`
SCT halt event select register, offset: 0xC

7.41.2.15 INPUT

`__R uint32_t` `SCT_t::INPUT`
SCT input register, offset: 0x48

7.41.2.16 LIMIT

`__RW uint32_t` `SCT_t::LIMIT`
SCT limit event select register, offset: 0x8

7.41.2.17 MATCH

`__RW uint32_t` `SCT_t::MATCH[8]`
SCT match value register of match channels, array offset: 0x100, array step: 0x4

7.41.2.18 MATCHREL

`__RW uint32_t` `SCT_t::MATCHREL[8]`
SCT match reload value register, array offset: 0x200, array step: 0x4

7.41.2.19 [struct]

```
struct { ... } SCT_t::OUT[7]  
SCT output 0 set register
```

7.41.2.20 OUTPUT

```
__RW uint32_t SCT_t::OUTPUT  
SCT output register, offset: 0x50
```

7.41.2.21 OUTPUTDIRCTRL

```
__RW uint32_t SCT_t::OUTPUTDIRCTRL  
SCT output counter direction control register, offset: 0x54
```

7.41.2.22 REGMODE

```
__RW uint32_t SCT_t::REGMODE  
SCT match/capture mode register, offset: 0x4C
```

7.41.2.23 RES

```
__RW uint32_t SCT_t::RES  
SCT conflict resolution register, offset: 0x58
```

7.41.2.24 RESERVED_0

```
uint8_t SCT_t::RESERVED_0[40]  
RESERVED. NOT USED
```

7.41.2.25 RESERVED_1

```
uint8_t SCT_t::RESERVED_1[140]  
RESERVED. NOT USED
```

7.41.2.26 RESERVED_2

```
uint8_t SCT_t::RESERVED_2[224]  
RESERVED. NOT USED
```

7.41.2.27 RESERVED_3

```
uint8_t SCT_t::RESERVED_3[224]  
RESERVED. NOT USED
```

7.41.2.28 RESERVED_4

```
uint8_t SCT_t::RESERVED_4[448]  
RESERVED. NOT USED
```

7.41.2.29 SET

```
__RW uint32_t SCT_t::SET  
SCT output 0 set register, array offset: 0x500, array step: 0x8
```

7.41.2.30 START

```
__RW uint32_t SCT_t::START  
SCT start event select register, offset: 0x14
```

7.41.2.31 STATE

```
__RW uint32_t SCT_t::STATE
SCT state register, offset: 0x44
SCT event state register 0, array offset: 0x300, array step: 0x8
```

7.41.2.32 STOP

```
__RW uint32_t SCT_t::STOP
SCT stop event select register, offset: 0x10
The documentation for this struct was generated from the following file:
```

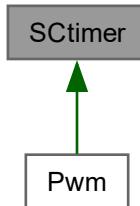
- [LPC845.h](#)

7.42 SCtimer Class Reference

Clase del objeto [SCtimer](#) El objeto [SCtimer](#) debe ser heredado por quienes desen utilizar las interrupciones o funcionalidades del [SCtimer](#).

```
#include <SCtimer.h>
```

Inheritance diagram for SCtimer:



Public Member Functions

- [SCtimer \(\)](#)
Constructor de clase [SCtimer](#).
- void [SetTime \(uint32_t time, uint32_t channel\)](#)
Setea el tiempo de un canal del contador.
- void [SetUnify \(bool a\)](#)
Unifica los dos registros del SCT.
- void [SetAutoLimit \(bool a\)](#)
Setea el autolímite.
- void [SetSwitchMatrizSCTOUT \(uint8_t bit, uint8_t port, uint8_t out_number\)](#)
Configura la switch Matrix.
- void [StartTimer \(void\)](#)
Habilita el timer.
- void [StopTimer \(void\)](#)
Deshabilita el timer.
- virtual [~SCtimer \(\)](#)

7.42.1 Detailed Description

Clase del objeto [SCtimer](#) El objeto [SCtimer](#) debe ser heredado por quienes desen utilizar las interrupciones o funcionalidades del [SCtimer](#).

7.42.2 Constructor & Destructor Documentation

7.42.2.1 ~SCTimer()

```
SCTimer::~SCTimer () [virtual]
```

Destructor por defecto

7.42.3 Member Function Documentation

7.42.3.1 SetAutoLimit()

```
void SCTimer::SetAutoLimit (
    bool a)
```

Setea el autolímite.

Determina si se resetea todos los canales al activar el canal 0 o no.

Parameters

in	<i>a</i>	bool que indica si el autolímite es cierto o falso.
----	----------	---

7.42.3.2 SetSwitchMatrixSCTOUT()

```
void SCTimer::SetSwitchMatrixSCTOUT (
    uint8_t bit,
    uint8_t port,
    uint8_t out_number)
```

Configura la switch Matrix.

Configura en la switch matrix la salida del evento *out_number* al puerto y pin indicados.

Parameters

in	<i>bit</i>	bit de la salida a programar.
in	<i>port</i>	puerto de la salida a programar.
in	<i>out_number</i>	número de evento al que corresponderán el bit y puerto.

7.42.3.3 SetTime()

```
void SCTimer::SetTime (
    uint32_t time,
    uint32_t channel)
```

Setea el tiempo de un canal del contador.

Configura un canal para que active dentro de determinado tiempo.

Parameters

in	<i>time</i>	tiempo de la acción.
in	<i>channel</i>	canal a configurar.

7.42.3.4 SetUnify()

```
void SCTimer::SetUnify (
    bool a)
```

Unifica los dos registros del SCT.

Elije si trabajar con registros high y low o con uno solo.

Parameters

in	a	bool que indica si la unificación es cierta o falsa.
----	---	--

7.42.3.5 StartTimer()

```
void SCTimer::StartTimer (
    void )
```

Habilita el timer.

Enciende el clock del timer.

7.42.3.6 StopTimer()

```
void SCTimer::StopTimer (
    void )
```

Deshabilita el timer.

Apaga el clock del timer.

The documentation for this class was generated from the following files:

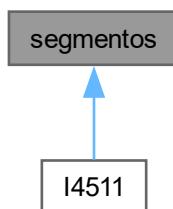
- [SCTimer.h](#)
- [SCTimer.cpp](#)

7.43 segmentos Class Reference

Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.

```
#include <Segmentos.h>
```

Inheritance diagram for segmentos:

**Public Member Functions**

- virtual void [SetSegmentos \(uint16_t\)=0](#)
- virtual void [Initialize \(void\)=0](#)

7.43.1 Detailed Description

Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.

7.43.2 Member Function Documentation**7.43.2.1 Initialize()**

```
virtual void segmentos::Initialize (
    void ) [pure virtual]
```

Funcion de inicializacion
Implemented in [I4511](#).

7.43.2.2 SetSegmentos()

```
virtual void segmentos::SetSegmentos (
    uint16_t ) [pure virtual]
```

constructor por defecto Funcion set del segmento
Implemented in [I4511](#).

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

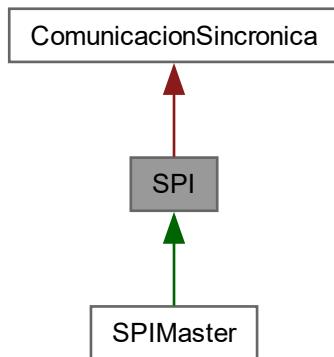
- [Segmentos.h](#)

7.44 SPI Class Reference

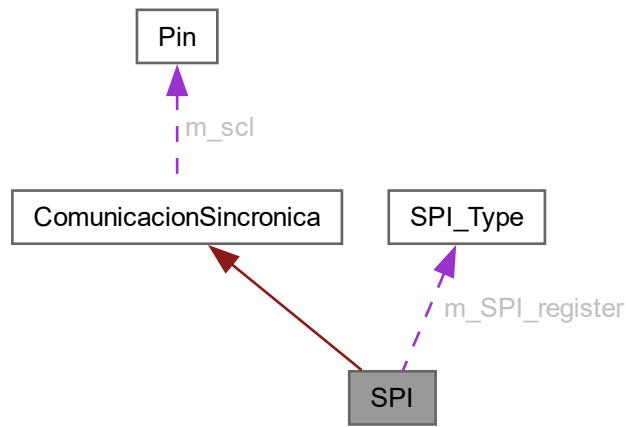
Clase del objeto [SPI](#).

```
#include <SPI.h>
```

Inheritance diagram for SPI:



Collaboration diagram for SPI:



Public Types

- enum `SPI_role_t`
- enum `SPI_mode_t` { `SPI_MODE_0` = 0 , `SPI_MODE_1` , `SPI_MODE_2` , `SPI_MODE_3` }

Public Member Functions

- `SPI (SPI_Type *SPI_register, Pin *miso, Pin *mosi, Pin *clk, SPI_role_t role=master)`
Constructor de la clase SPI.
- void `Initialize (uint32_t clk_freq, SPI_mode_t mode=SPI_MODE_0)`
Inicializa el SPI.
- void `EnableInterrupt (void)`
Habilita la interrupcion.
- void `DisableInterrupt (void)`
Deshabilita la interrupcion de transmision.
- `int8_t AddSSEL (Pin *ssel, uint8_t slave_number)`
Agrega un pin al Chip select.
- void `RemoveSSEL (uint8_t slave_number)`
Elimina un pin del Chip select.
- void `SetSSEL (uint8_t slave_number)`
Activa el Chip Select correspondiente.
- void `ClearSSEL (uint8_t slave_number)`
Desactiva el Chip Select correspondiente.
- void `ClearEOT (void)`
Limpia el End Of Transmition.
- void `SetEOT (void)`
Setea el End Of Transmition.
- void `ClearEOF (void)`
Limpia el End Of Frame.
- void `forceFinish ()`
Fuerza el End of Transmition.

- void `Write (uint8_t data)`
Escribe un valor.
- `int8_t Read (uint8_t *data)`
Lee un valor recibido.
- bool `isTxReady (void)`
Avisa si se puede transmitir informacion.
- bool `isRxReady (void)`
Avisa si se recibio informacion.
- virtual `~SPI ()=default`

Protected Attributes

- `SPI_Type * m_SPI_register`

7.44.1 Detailed Description

Clase del objeto `SPI`.

Crea un objeto `SPI`. Maneja los registros correspondientes del `SPI`.

7.44.2 Member Enumeration Documentation

7.44.2.1 SPI_mode_t

`enum SPI::SPI_mode_t`

Modo de comunicacion del `SPI`. Puede ser MODE0, MODE1, MODE2 o MODE3.

Enumerator

<code>SPI_MODE_0</code>	CPOL=0 y CPHA = 0. Change + Low clock rest.
<code>SPI_MODE_1</code>	CPOL=0 y CPHA = 1 Capture + Low clock rest.
<code>SPI_MODE_2</code>	CPOL=1 y CPHA = 0 Change + High clock rest.
<code>SPI_MODE_3</code>	CPOL=1 y CPHA = 1 Capture + High clock rest.

7.44.2.2 SPI_role_t

`enum SPI::SPI_role_t`

Enumeracion. Modo del `SPI`. Puede ser master o slave.

7.44.3 Constructor & Destructor Documentation

7.44.3.1 SPI()

```
SPI::SPI (
    SPI_Type * SPI_register,
    Pin * miso,
    Pin * mosi,
    Pin * clk,
    SPI_role_t mode = master)
```

Constructor de la clase `SPI`.

Genera un `SPI` con los parametros indicados.

Parameters

in	<i>SPI_register</i>	Registro del SPI a utilizar. Puede valer SPI0 o SPI1.
in	<i>miso</i>	Pin de recepcion de datos del master.
in	<i>mosi</i>	Pin de transmision de datos del master.
in	<i>clk</i>	Pin de clock del SPI .
in	<i>mode</i>	Rol de SPI . Puede ser master o slave.

7.44.3.2 ~SPI()

```
virtual SPI::~SPI () [virtual], [default]
Destructor por default.
```

7.44.4 Member Function Documentation**7.44.4.1 AddSSEL()**

```
int8_t SPI::AddSSEL (
    Pin * ssel,
    uint8_t slave_number)
```

Agrega un pin al Chip select.

Agrega un **Pin** de Chip Select a la comunicacion. El SPI0 admite hasta 4, El SPI1 admite 2.

Parameters

in	<i>ssel</i>	Pin a agregar.
in	<i>slave_number</i>	Numero de Chip Select correspondiente al pin.

Returns

int8_t: Mensaje de error. 0 no hay error. -1 hubo error.

7.44.4.2 ClearEOF()

```
void SPI::ClearEOF (
    void )
```

Limpia el End Of Frame.

El End of Frame indica que la comunicacion aun no termino. Debe setearse si se desea escribir de corrido.

7.44.4.3 ClearEOT()

```
void SPI::ClearEOT (
    void )
```

Limpia el End Of Transmition.

El End of Transmition indica que este es el ultimo byte de la comunicacion. Debe setearse si es el ultimo byte de la transmision iniciada.

7.44.4.4 ClearSSEL()

```
void SPI::ClearSSEL (
    uint8_t slave_number)
```

Desactiva el Chip Select correspondiente.

Apaga el Chip Select del numero indicado. Solo debe ser llamado por el master.

Parameters

in	<i>slave_number</i>	Numero de Chip Select a desactivar.
----	---------------------	-------------------------------------

7.44.4.5 forceFinish()

```
void SPI::forceFinish (
    void )
```

Fuerza el End of Transmission.

Se utiliza cuando una comunicacion debe terminar inmediatamente ya que se desconoce de antemano el bit final a transmitir.

7.44.4.6 Initialize()

```
void SPI::Initialize (
    uint32_t clk_freq,
    SPI_mode_t mode = SPI_MODE_0)
```

Inicializa el [SPI](#).

Configura todos los registros para el uso del [SPI](#). Configura los clocks, resetea, configura la SWM, configura los registros.

Parameters

in	<i>clk_freq</i>	frecuencia del clock de transmision a utilizar. En KHz.
in	<i>mode</i>	Modo del SPI . Configura la polaridad y fase. Puede ser SPI_MODE_0, SPI_MODE_1, SPI_MODE_2 o SPI_MODE_3.

7.44.4.7 isRxReady()

```
bool SPI::isRxReady (
    void )
```

Avisa si se recibio informacion.

Indica si se recibio informacion o no. Debe verificarse antes de llamarse a read.

Returns

bool: true si hay informacion, false si no.

7.44.4.8 isTxReady()

```
bool SPI::isTxReady (
    void )
```

Avisa si se puede transmitir informacion.

Indica si se puede transmitir o no. Debe verificarse antes de llamarse a write.

Returns

bool: true si se puede transmitir, false si no.

7.44.4.9 Read()

```
int8_t SPI::Read (
    uint8_t * data) [virtual]
```

Lee un valor recibido.

Lee el valor recibido por un slave. Solo puede ser llamado por un master.

Parameters

in,out	<i>data</i>	puntero a char donde se guardara el valor recibido.
--------	-------------	---

Returns

`int8_t`: Valor de error. Proviene de herencia, siempre devuelve 0.

Warning

Debe consultarse `isRxReady()` antes de hacer la lectura.

Implements `ComunicacionSincronica`.

7.44.4.10 RemoveSSEL()

```
void SPI::RemoveSSEL (
    uint8_t slave_number)
```

Elimina un pin del Chip select.

Elimina un Pin del Chip Select de la comunicacion. El SPI0 admite hasta 4, El SPI1 admite 2.

Parameters

in	<code>slave_number</code>	Numero de Chip Select a eliminar.
----	---------------------------	-----------------------------------

7.44.4.11 SetEOT()

```
void SPI::SetEOT (
    void )
```

Setea el End Of Transmition.

Setea el End Of Frame.

El End of Transmition indica que este es el ultimo byte de la comunicacion. Debe setearse si es el ultimo byte de la transmision iniciada.

El End of Frame indica que la comunicacion aun no termino. Debe setearse si se desea escribir de corrido.

7.44.4.12 SetSSEL()

```
void SPI::SetSSEL (
    uint8_t slave_number)
```

Activa el Chip Select correspondiente.

Enciende el Chip Select del numero indicado. Solo debe ser llamado por el master.

Parameters

in	<code>slave_number</code>	Numero de Chip Select a activar.
----	---------------------------	----------------------------------

7.44.4.13 Write()

```
void SPI::Write (
    uint8_t data) [virtual]
```

Escribe un valor.

Envia un dato de 8 bits (1 byte) por la comunicacion.

Parameters

in	<code>data</code>	byte a enviar.
----	-------------------	----------------

Note

El `SPI` comienza a transmitir cuando se escribe este registro.

Warning

Debe consultarse `isTxReady()` antes de hacer la escritura.

Implements `ComunicacionSincronica`.

7.44.5 Member Data Documentation

7.44.5.1 m_SPI_register

`SPI_Type* SPI::m_SPI_register [protected]`

Registro a leer para configurar I2C. Protegido ya que se puede acceder por herencia.

The documentation for this class was generated from the following files:

- [SPI.h](#)
- [SPI.cpp](#)

7.45 SPI_Type Struct Reference

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t CFG`
- `__RW uint32_t DLY`
- `__RW uint32_t STAT`
- `__RW uint32_t INTENSET`
- `__W uint32_t INTENCLR`
- `__R uint32_t RXDAT`
- `__RW uint32_t TXDATCTL`
- `__RW uint32_t TXDAT`
- `__RW uint32_t TXCTL`
- `__RW uint32_t DIV`
- `__R uint32_t INTSTAT`

7.45.1 Detailed Description

`SPI` - Register Layout Typedef

7.45.2 Member Data Documentation

7.45.2.1 CFG

`__RW uint32_t SPI_Type::CFG`

`SPI` Configuration register, offset: 0x0

7.45.2.2 DIV

`__RW uint32_t SPI_Type::DIV`

`SPI` clock Divider, offset: 0x24

7.45.2.3 DLY

`__RW uint32_t SPI_Type::DLY`

`SPI` Delay register, offset: 0x4

7.45.2.4 INTENCLR

`__W uint32_t SPI_Type::INTENCLR`

`SPI` Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared., offset: 0x10

7.45.2.5 INTENSET

`__RW uint32_t SPI_Type::INTENSET`

SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set., offset: 0xC

7.45.2.6 INTSTAT

`__R uint32_t SPI_Type::INTSTAT`

SPI Interrupt Status, offset: 0x28

7.45.2.7 RXDAT

`__R uint32_t SPI_Type::RXDAT`

SPI Receive Data, offset: 0x14

7.45.2.8 STAT

`__RW uint32_t SPI_Type::STAT`

SPI Status. Some status flags can be cleared by writing a 1 to that bit position, offset: 0x8

7.45.2.9 TXCTL

`__RW uint32_t SPI_Type::TXCTL`

SPI Transmit Control, offset: 0x20

7.45.2.10 TXDAT

`__RW uint32_t SPI_Type::TXDAT`

SPI Transmit Data., offset: 0x1C

7.45.2.11 TXDATCTL

`__RW uint32_t SPI_Type::TXDATCTL`

SPI Transmit Data with Control, offset: 0x18

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

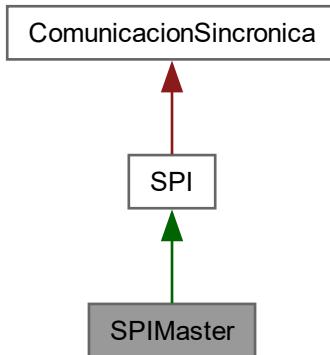
- [LPC845.h](#)

7.46 SPIMaster Class Reference

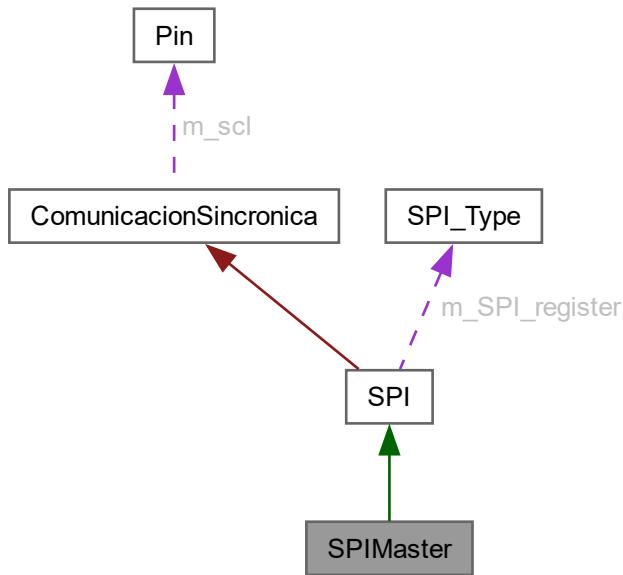
Clase del objeto [SPIMaster](#).

```
#include <SPIMaster.h>
```

Inheritance diagram for SPIMaster:



Collaboration diagram for SPIMaster:



Public Member Functions

- **SPIMaster (SPI_Type *SPI_register, Pin *miso, Pin *mosi, Pin *clk, uint32_t maxRx=15, uint32_t maxTx=15)**
Constructor de la clase SPIMaster.
- void **Initialize (uint32_t clk_freq, SPI_mode_t mode=SPI_MODE_0)**
Inicializa el SPI.
- int8_t **AddSSEL (Pin *ssel, uint8_t slave_number)**
Agrega un pin al Chip select.

- void **RemoveSSEL** (`uint8_t` slave_number)
Elimina un pin del Chip select.
- void **Write** (`uint8_t` slv, const char *msg)
Transmite el mensaje indicado.
- void **Write** (`uint8_t` slv, const void *msg, `uint32_t` length)
Transmite length cantidad de bytes del mensaje indicado.
- void * **Read** (void *msg, `uint32_t` length)
Lee el buffer de recepcion.
- void **RequestRead** (`uint8_t` slv, `uint32_t` cant_read=1)
Inicia una comunicacion de lectura.
- bool **hasData** (void)
Indica si posee data sin leer.
- bool **IsIdle** (void)
Indica si ya termino de comunicar o no.

Additional Inherited Members

Protected Types inherited from SPI

- enum `SPI_role_t`
- enum `SPI_mode_t` { `SPI_MODE_0` = 0, `SPI_MODE_1`, `SPI_MODE_2`, `SPI_MODE_3` }

Protected Member Functions inherited from SPI

- **SPI** (`SPI_Type` *SPI_register, `Pin` *miso, `Pin` *mosi, `Pin` *clk, `SPI_role_t` role=master)
Constructor de la clase SPI.
- void **Initialize** (`uint32_t` clk_freq, `SPI_mode_t` mode=`SPI_MODE_0`)
Inicializa el SPI.
- void **EnableInterrupt** (void)
Habilita la interrupcion.
- void **DisableInterrupt** (void)
Deshabilita la interrupcion de transmision.
- `int8_t` **AddSSEL** (`Pin` *ssel, `uint8_t` slave_number)
Agrega un pin al Chip select.
- void **RemoveSSEL** (`uint8_t` slave_number)
Elimina un pin del Chip select.
- void **SetSSEL** (`uint8_t` slave_number)
Activa el Chip Select correspondiente.
- void **ClearSSEL** (`uint8_t` slave_number)
Desactiva el Chip Select correspondiente.
- void **ClearEOT** (void)
Limpia el End Of Transmition.
- void **SetEOT** (void)
Setea el End Of Transmition.
- void **ClearEOF** (void)
Limpia el End Of Frame.
- void **forceFinish** ()
Fuerza el End of Transmition.
- void **Write** (`uint8_t` data)
Escribe un valor.
- `int8_t` **Read** (`uint8_t` *data)
Lee un valor recibido.

- bool `isTxReady` (void)
Avisa si se puede transmitir informacion.
- bool `isRxReady` (void)
Avisa si serecio informacion.
- virtual `~SPI` ()=default

Protected Attributes inherited from SPI

- `SPI_Type * m_SPI_register`

7.46.1 Detailed Description

Clase del objeto `SPIMaster`.

Crea un objeto `SPI` configurado como maestro. Posee buffers de recepcion y transmision.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 SPIMaster()

```
SPIMaster::SPIMaster (
    SPI_Type * SPI_register,
    Pin * miso,
    Pin * mosi,
    Pin * clk,
    uint32_t maxRx = 15,
    uint32_t maxTx = 15)
```

Constructor de la clase `SPIMaster`.

Genera un master spi con buffers de transmision y recepcion para envio de strings.

Parameters

in	<code>SPI_register</code>	Registro del <code>SPI</code> a utilizar. Puede valer SPI0 o SPI1.
in	<code>miso</code>	<code>Pin</code> de recepcion de datos del master.
in	<code>mosi</code>	<code>Pin</code> de transmision de datos del master.
in	<code>clk</code>	<code>Pin</code> de clock.
in	<code>maxRx</code>	Valor maximo del registro de recepcion. Por default 15.
in	<code>maxTx</code>	Valor maximo del registro de transmision. Por default 15.

7.46.3 Member Function Documentation

7.46.3.1 AddSSEL()

```
int8_t SPIMaster::AddSSEL (
    Pin * ssel,
    uint8_t slave_number)
```

Agrega un pin al Chip select.

Agrega un `Pin` de Chip Select a la comunicacion. El SPI0 admite hasta 4, El SPI1 admite 2.

Parameters

in	<code>ssel</code>	<code>Pin</code> a agregar.
in	<code>slave_number</code>	Numero de Chip Select correspondiente al pin.

Returns

`int8_t`: Mensaje de error. 0 no hay error. -1 hubo error.

7.46.3.2 hasData()

```
bool SPIMaster::hasData (
    void )
```

Indica si posee data sin leer.

Indica si hay algo en el buffer. Read solo funcionara si hay data. No es obligatorio su uso.

Returns

bool: true si hay valores, false sino.

7.46.3.3 Initialize()

```
void SPIMaster::Initialize (
    uint32_t clk_freq,
    SPI_mode_t mode = SPI_MODE_0)
```

Inicializa el [SPI](#).

Configura todos los registros para el uso del [SPI](#). Configura los clocks, resetea, configura la SWM, configura los registros.

Parameters

in	<i>clk_freq</i>	frecuencia del clock de transmision a utilizar. En KHz.
in	<i>mode</i>	Modo del SPI . Configura la polaridad y fase. Puede ser SPI_MODE_0, SPI_MODE_1, SPI_MODE_2 o SPI_MODE_3.

7.46.3.4 IsIdle()

```
bool SPIMaster::IsIdle (
    void )
```

Indica si ya termino de comunicar o no.

Determina si toda la informacion ya fue escrita o leida. Se recomienda verificar para no realizar multiples lecturas seguidas y llenar el buffer.

7.46.3.5 Read()

```
void * SPIMaster::Read (
    void * msg,
    uint32_t length)
```

Lee el buffer de recepcion.

Lee una cantidad fija de bytes dentro del buffer de recepcion.

Parameters

in	<i>msg</i>	Puntero a variable donde se enviarian los valores leidos.
in	<i>length</i>	Cantidad de bytes a leer.

Returns

void*: Puntero a los valores leidos. Si hubo error se envia nullptr.

7.46.3.6 RemoveSSEL()

```
void SPIMaster::RemoveSSEL (
    uint8_t slave_number)
```

Elimina un pin del Chip select.

Elimina un [Pin](#) del Chip Select de la comunicacion. El SPI0 admite hasta 4, El SPI1 admite 2.

Parameters

in	<i>slave_number</i>	Numero de Chip Select a eliminar.
----	---------------------	-----------------------------------

7.46.3.7 RequestRead()

```
void SPIMaster::RequestRead (
    uint8_t slv,
    uint32_t cant_read = 1)
```

Inicia una comunicacion de lectura.

Transmite cant_read bytes para que se inicie una comunicacion. Se transmiten bytes de valor NO_DATA.

Parameters

in	<i>slv</i>	Numero de Slave o Chip Select a encender.
in	<i>cant_read</i>	cantidad de bytes a leer.

Note

Mientras se inicie una escritura con [Write\(\)](#) es innecesario el [RequestRead\(\)](#). Solo debe llamarse si se necesitan leer datos y no enviar nada. Como solo se puede leer cuando el master escribe, es posible que cant_read deba tener un valor +1 del deseado a leer.

7.46.3.8 Write() [1/2]

```
void SPIMaster::Write (
    uint8_t slv,
    const char * msg)
```

Transmite el mensaje indicado.

Coloca el mensaje indicado en el buffer. Importante, el mensaje debe terminar en \0 (String).

Parameters

in	<i>slv</i>	Numero de Slave o Chip Select a encender.
in	<i>msg</i>	Mensaje a transmitir.

7.46.3.9 Write() [2/2]

```
void SPIMaster::Write (
    uint8_t slv,
    const void * msg,
    uint32_t length)
```

Transmite length cantidad de bytes del mensaje indicado.

Coloca una cantidad fija del mensaje indicado en el buffer.

Parameters

in	<i>slv</i>	Numero de Slave o Chip Select a encender.
in	<i>msg</i>	Mensaje a transmitir.
in	<i>length</i>	cantidad de bytes a transmitir.

The documentation for this class was generated from the following files:

- [SPIMaster.h](#)
- [SPIMaster.cpp](#)

7.47 SWM_t Struct Reference

Structure type to access the Switch Matriz (SWM).

```
#include <LPC845.h>
```

Public Attributes

- `uint8_t RESERVED_0 [388]`
- `__RW uint32_t PINENABLE0`
- `__RW uint32_t PINENABLE1`

7.47.1 Detailed Description

Structure type to access the Switch Matriz (SWM).

SWM - Register Layout Typedef

7.47.2 Member Data Documentation

7.47.2.1 PINASSIGN0

```
__RW uint32_t SWM_t::PINASSIGN0
```

Pin assign register 0. Assign movable functions U0_TXD, U0_RXD, U0_RTS, U0_CTS., offset: 0x0

7.47.2.2 PINASSIGN1

```
__RW uint32_t SWM_t::PINASSIGN1
```

Pin assign register 1. Assign movable functions U0_SCLK, U1_TXD, U1_RXD, U1_RTS., offset: 0x4

7.47.2.3 PINASSIGN10

```
__RW uint32_t SWM_t::PINASSIGN10
```

Pin assign register 10. Assign movable functions I2C2_SDA, I2C2_SCL, I2C3_SDA, I2C3_SCL., offset: 0x28

7.47.2.4 PINASSIGN11

```
__RW uint32_t SWM_t::PINASSIGN11
```

Pin assign register 11. Assign movable functions COMP0_OUT, CLKOUT, GPIOINT_BMATCH, UART3_TXD, offset: 0x2C

7.47.2.5 PINASSIGN12

```
__RW uint32_t SWM_t::PINASSIGN12
```

Pin assign register 12. Assign movable functions UART3_RXD, UART3_SCLK, UART4_TXD, UART4_RXD., offset: 0x30

7.47.2.6 PINASSIGN13

```
__RW uint32_t SWM_t::PINASSIGN13
```

Pin assign register 13. Assign movable functions UART4_SCLK, T0_MAT0, T0_MAT1, T0_MAT2., offset: 0x34

7.47.2.7 PINASSIGN14

```
__RW uint32_t SWM_t::PINASSIGN14
```

Pin assign register 14. Assign movable functions T0_MAT3, T0_CAP0, T0_CAP1, T0_CAP2., offset: 0x38

7.47.2.8 PINASSIGN2

```
__RW uint32_t SWM_t::PINASSIGN2
```

Pin assign register 2. Assign movable functions U1_CTS, U1_SCLK, U2_TXD, U2_RXD., offset: 0x8

7.47.2.9 PINASSIGN3

`__RW uint32_t SWM_t::PINASSIGN3`

Pin assign register 3. Assign movable function U2_RTS, U2_CTS, U2_SCLK, SPI0_SCK., offset: 0xC

7.47.2.10 PINASSIGN4

`__RW uint32_t SWM_t::PINASSIGN4`

Pin assign register 4. Assign movable functions SPI0_MOSI, SPI0_MISO, SPI0_SSEL0, SPI0_SSEL1., offset: 0x10

7.47.2.11 PINASSIGN5

`__RW uint32_t SWM_t::PINASSIGN5`

Pin assign register 5. Assign movable functions SPI0_SSEL2, SPI0_SSEL3, SPI1_SCK, SPI1_MOSI, offset: 0x14

7.47.2.12 PINASSIGN6

`__RW uint32_t SWM_t::PINASSIGN6`

Pin assign register 6. Assign movable functions SPI1_MISO, SPI1_SSEL0, SPI1_SSEL1, SCT0_IN0., offset: 0x18

7.47.2.13 PINASSIGN7

`__RW uint32_t SWM_t::PINASSIGN7`

Pin assign register 7. Assign movable functions SCT_IN1, SCT_IN2, SCT_IN3, SCT_OUT0., offset: 0x1C

7.47.2.14 PINASSIGN8

`__RW uint32_t SWM_t::PINASSIGN8`

Pin assign register 8. Assign movable functions SCT_OUT1, SCT_OUT2, SCT_OUT3, SCT_OUT4., offset: 0x20

7.47.2.15 PINASSIGN9

`__RW uint32_t SWM_t::PINASSIGN9`

Pin assign register 9. Assign movable functions SCT_OUT5, SCT_OUT6, I2C1_SDA, I2C1_SCL., offset: 0x24

7.47.2.16 PINASSIGN_DATA

`__RW uint32_t SWM_t::PINASSIGN_DATA[15]`

Pin assign register, array offset: 0x0, array step: 0x4

7.47.2.17 PINENABLE0

`__RW uint32_t SWM_t::PINENABLE0`

Pin enable register 0. Enables fixed-pin functions ACMP_I0, ACMP_I1, SWCLK, SWDIO, XTALIN, XTALOUT, RESET, CLKIN, VDDCMP and so on., offset: 0x1C0

7.47.2.18 PINENABLE1

`__RW uint32_t SWM_t::PINENABLE1`

Pin enable register 1. Enables fixed-pin functions CAPT_X4, CAPT_X5, CAPT_X6, CAPT_X7, CAPT_X8, CAPT_X4, CAPT_YL and CAPT_YH., offset: 0x1C4

7.47.2.19 RESERVED_0

`uint8_t SWM_t::RESERVED_0[388]`

RESERVED. NOT USED

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

- [LPC845.h](#)

7.48 SYSCON_Type Struct Reference

Structure type to access the SYSCON.

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t SYSMEMREMAP`
- `uint8_t RESERVED_0 [4]`
- `__RW uint32_t SYSPLLCTRL`
- `__R uint32_t SYSPLLSTAT`
- `uint8_t RESERVED_1 [16]`
- `__RW uint32_t SYSOSCCTRL`
- `__RW uint32_t WDTOSCCTRL`
- `__RW uint32_t FROOSCCTRL`
- `uint8_t RESERVED_2 [4]`
- `__RW uint32_t FRODIRECTCLKUEN`
- `uint8_t RESERVED_3 [4]`
- `__RW uint32_t SYSRSTSTAT`
- `uint8_t RESERVED_4 [4]`
- `__RW uint32_t SYSPLLCLKSEL`
- `__RW uint32_t SYSPLLCLKUEN`
- `__RW uint32_t MAINCLKPLLSEL`
- `__RW uint32_t MAINCLKPLLUEN`
- `__RW uint32_t MAINCLKSEL`
- `__RW uint32_t MAINCLKUEN`
- `__RW uint32_t SYSAHBCLKDIV`
- `uint8_t RESERVED_5 [4]`
- `__RW uint32_t CAPTCLKSEL`
- `__RW uint32_t ADCCLKSEL`
- `__RW uint32_t ADCCLKDIV`
- `__RW uint32_t SCTCLKSEL`
- `__RW uint32_t SCTCLKDIV`
- `__RW uint32_t EXTCLKSEL`
- `uint8_t RESERVED_6 [8]`
- `__RW uint32_t SYSAHBCLKCTRL0`
- `__RW uint32_t SYSAHBCLKCTRL1`
- `__RW uint32_t PRESETCTRL0`
- `__RW uint32_t PRESETCTRL1`
- `__RW uint32_t FCLKSEL [11]`
- `uint8_t RESERVED_7 [20]`
- `struct {`
 - `__RW uint32_t FRGDIV`
 - `__RW uint32_t FRGMULT`
 - `__RW uint32_t FRGCLKSEL`
 - `uint8_t RESERVED_0 [4]`
- `} FRG [2]`
- `__RW uint32_t CLKOUTSEL`
- `__RW uint32_t CLKOUTDIV`
- `uint8_t RESERVED_8 [4]`
- `__RW uint32_t EXTRACECMD`
- `__R uint32_t PIOPORCAP [2]`

- `uint8_t RESERVED_9` [44]
- `__RW uint32_t IOCONCLKDIV6`
- `__RW uint32_t IOCONCLKDIV5`
- `__RW uint32_t IOCONCLKDIV4`
- `__RW uint32_t IOCONCLKDIV3`
- `__RW uint32_t IOCONCLKDIV2`
- `__RW uint32_t IOCONCLKDIV1`
- `__RW uint32_t IOCONCLKDIV0`
- `__RW uint32_t BODCTRL`
- `__RW uint32_t SYSTCKCAL`
- `uint8_t RESERVED_10` [24]
- `__RW uint32_t IRQLATENCY`
- `__RW uint32_t NMISRC`
- `__RW uint32_t PINTSEL` [8]
- `uint8_t RESERVED_11` [108]
- `__RW uint32_t STARTERP0`
- `uint8_t RESERVED_12` [12]
- `__RW uint32_t STARTERP1`
- `uint8_t RESERVED_13` [24]
- `__RW uint32_t PDSLEEPcfg`
- `__RW uint32_t PDAWAKEcfg`
- `__RW uint32_t PDRUNcfg`
- `uint8_t RESERVED_14` [444]
- `__R uint32_t DEVICE_ID`

7.48.1 Detailed Description

Structure type to access the SYSCON.
SYSCON - Register Layout Typedef

7.48.2 Member Data Documentation

7.48.2.1 ADCCLKDIV

`__RW uint32_t SYSCON_Type::ADCCLKDIV`
ADC clock divider register, offset: 0x68

7.48.2.2 ADCCLKSEL

`__RW uint32_t SYSCON_Type::ADCCLKSEL`
ADC clock source select register, offset: 0x64

7.48.2.3 BODCTRL

`__RW uint32_t SYSCON_Type::BODCTRL`
BOD control register, offset: 0x150

7.48.2.4 CAPTCLKSEL

`__RW uint32_t SYSCON_Type::CAPTCLKSEL`
CAPT clock source select register, offset: 0x60

7.48.2.5 CLKOUTDIV

`__RW uint32_t SYSCON_Type::CLKOUTDIV`
CLKOUT clock divider registers, offset: 0xF4

7.48.2.6 CLKOUTSEL

`__RW uint32_t SYSCON_Type::CLKOUTSEL`
 CLKOUT clock source select register, offset: 0xF0

7.48.2.7 DEVICE_ID

`__R uint32_t SYSCON_Type::DEVICE_ID`
 Part ID register, offset: 0x3F8

7.48.2.8 EXTCLKSEL

`__RW uint32_t SYSCON_Type::EXTCLKSEL`
 external clock source select register, offset: 0x74

7.48.2.9 EXTTRACECMD

`__RW uint32_t SYSCON_Type::EXTTRACECMD`
 External trace buffer command register, offset: 0xFC

7.48.2.10 FCLKSEL

`__RW uint32_t SYSCON_Type::FCLKSEL[11]`
 peripheral clock source select register. FCLK0SEL~FCLK4SEL are for UART0~UART4 clock source select register. FCLK5SEL~FCLK8SEL are for I2C0~I2C3 clock source select register. FCLK9SEL~FCLK10SEL are for SPI0~SPI1 clock source select register., array offset: 0x90, array step: 0x4

7.48.2.11 [struct]

```
struct { ... } SYSCON_Type::FRG[2]
Fractional generator divider
```

7.48.2.12 FRGCLKSEL

`__RW uint32_t SYSCON_Type::FRGCLKSEL`
 FRG N clock source select register, array offset: 0xD8, array step: 0x10

7.48.2.13 FRGDIV

`__RW uint32_t SYSCON_Type::FRGDIV`
 offset: 0xD0, array step: 0x10 fractional generator N divider value register, array offset: 0xD0, array step: 0x10

7.48.2.14 FRGMULT

`__RW uint32_t SYSCON_Type::FRGMULT`
 fractional generator N multiplier value register, array offset: 0xD4, array step: 0x10

7.48.2.15 FRODIRECTCLKUEN

`__RW uint32_t SYSCON_Type::FRODIRECTCLKUEN`
 FRO direct clock source update enable register, offset: 0x30

7.48.2.16 FROOSCCTRL

`__RW uint32_t SYSCON_Type::FROOSCCTRL`
 FRO oscillator control, offset: 0x28

7.48.2.17 IOCONCLKDIV0

`__RW uint32_t SYSCON_Type::IOCONCLKDIV0`
 Peripheral clock 0 to the IOCON block for programmable glitch filter, offset: 0x14C

7.48.2.18 IOCONCLKDIV1

`__RW uint32_t SYSCON_Type::IOCONCLKDIV1`
Peripheral clock 1 to the IOCON block for programmable glitch filter, offset: 0x148

7.48.2.19 IOCONCLKDIV2

`__RW uint32_t SYSCON_Type::IOCONCLKDIV2`
Peripheral clock 2 to the IOCON block for programmable glitch filter, offset: 0x144

7.48.2.20 IOCONCLKDIV3

`__RW uint32_t SYSCON_Type::IOCONCLKDIV3`
Peripheral clock 3 to the IOCON block for programmable glitch filter, offset: 0x140

7.48.2.21 IOCONCLKDIV4

`__RW uint32_t SYSCON_Type::IOCONCLKDIV4`
Peripheral clock 4 to the IOCON block for programmable glitch filter, offset: 0x13C

7.48.2.22 IOCONCLKDIV5

`__RW uint32_t SYSCON_Type::IOCONCLKDIV5`
Peripheral clock 6 to the IOCON block for programmable glitch filter, offset: 0x138

7.48.2.23 IOCONCLKDIV6

`__RW uint32_t SYSCON_Type::IOCONCLKDIV6`
Peripheral clock 6 to the IOCON block for programmable glitch filter, offset: 0x134

7.48.2.24 IRQLATENCY

`__RW uint32_t SYSCON_Type::IRQLATENCY`
IRQ latency register, offset: 0x170

7.48.2.25 MAINCLKPLLSEL

`__RW uint32_t SYSCON_Type::MAINCLKPLLSEL`
Main clock source select register, offset: 0x48

7.48.2.26 MAINCLKPLLUEN

`__RW uint32_t SYSCON_Type::MAINCLKPLLUEN`
Main clock source update enable register, offset: 0x4C

7.48.2.27 MAINCLKSEL

`__RW uint32_t SYSCON_Type::MAINCLKSEL`
Main clock source select register, offset: 0x50

7.48.2.28 MAINCLKUEN

`__RW uint32_t SYSCON_Type::MAINCLKUEN`
Main clock source update enable register, offset: 0x54

7.48.2.29 NMISRC

`__RW uint32_t SYSCON_Type::NMISRC`
NMI source selection register, offset: 0x174

7.48.2.30 PDAWAKECFG

`__RW uint32_t SYSCON_Type::PDAWAKECFG`
Wake-up configuration register, offset: 0x234

7.48.2.31 PDRUNCFG

`__RW uint32_t SYSCON_Type::PDRUNCFG`
Power configuration register, offset: 0x238

7.48.2.32 PDSLEEPcfg

`__RW uint32_t SYSCON_Type::PDSLEEPcfg`
Deep-sleep configuration register, offset: 0x230

7.48.2.33 PINTSEL

`__RW uint32_t SYSCON_Type::PINTSEL[8]`
Pin interrupt select registers N, array offset: 0x178, array step: 0x4

7.48.2.34 PIOPORCAP

`__R uint32_t SYSCON_Type::PIOPORCAP[2]`
POR captured PIO N status register(PIO0 has 32 PIOs, PIO1 has 22 PIOs), array offset: 0x100, array step: 0x4

7.48.2.35 PRESETCTRL0

`__RW uint32_t SYSCON_Type::PRESETCTRL0`
Peripheral reset group 0 control register, offset: 0x88

7.48.2.36 PRESETCTRL1

`__RW uint32_t SYSCON_Type::PRESETCTRL1`
Peripheral reset group 1 control register, offset: 0x8C

7.48.2.37 RESERVED_0

`uint8_t SYSCON_Type::RESERVED_0[4]`
RESERVED. NOT USED

7.48.2.38 RESERVED_1

`uint8_t SYSCON_Type::RESERVED_1[16]`
RESERVED. NOT USED

7.48.2.39 RESERVED_10

`uint8_t SYSCON_Type::RESERVED_10[24]`
RESERVED. NOT USED

7.48.2.40 RESERVED_11

`uint8_t SYSCON_Type::RESERVED_11[108]`
RESERVED. NOT USED

7.48.2.41 RESERVED_12

`uint8_t SYSCON_Type::RESERVED_12[12]`
RESERVED. NOT USED

7.48.2.42 RESERVED_13

`uint8_t` `SYSCON_Type::RESERVED_13[24]`
RESERVED. NOT USED

7.48.2.43 RESERVED_14

`uint8_t` `SYSCON_Type::RESERVED_14[444]`
RESERVED. NOT USED

7.48.2.44 RESERVED_2

`uint8_t` `SYSCON_Type::RESERVED_2[4]`
RESERVED. NOT USED

7.48.2.45 RESERVED_3

`uint8_t` `SYSCON_Type::RESERVED_3[4]`
RESERVED. NOT USED

7.48.2.46 RESERVED_4

`uint8_t` `SYSCON_Type::RESERVED_4[4]`
RESERVED. NOT USED

7.48.2.47 RESERVED_5

`uint8_t` `SYSCON_Type::RESERVED_5[4]`
RESERVED. NOT USED

7.48.2.48 RESERVED_6

`uint8_t` `SYSCON_Type::RESERVED_6[8]`
RESERVED. NOT USED

7.48.2.49 RESERVED_7

`uint8_t` `SYSCON_Type::RESERVED_7[20]`
RESERVED. NOT USED

7.48.2.50 RESERVED_8

`uint8_t` `SYSCON_Type::RESERVED_8[4]`
RESERVED. NOT USED

7.48.2.51 RESERVED_9

`uint8_t` `SYSCON_Type::RESERVED_9[44]`
RESERVED. NOT USED

7.48.2.52 SCTCLKDIV

`__RW uint32_t` `SYSCON_Type::SCTCLKDIV`
SCT clock divider register, offset: 0x70

7.48.2.53 SCTCLKSEL

`__RW uint32_t` `SYSCON_Type::SCTCLKSEL`
SCT clock source select register, offset: 0x6C

7.48.2.54 STARTERPO

`__RW uint32_t SYSCON_Type::STARTERPO`
Start logic 0 pin wake-up enable register 0, offset: 0x204

7.48.2.55 STARTERP1

`__RW uint32_t SYSCON_Type::STARTERP1`
Start logic 0 pin wake-up enable register 1, offset: 0x214

7.48.2.56 SYSAHCLKCTRL0

`__RW uint32_t SYSCON_Type::SYSAHCLKCTRL0`
System clock group 0 control register, offset: 0x80

7.48.2.57 SYSAHCLKCTRL1

`__RW uint32_t SYSCON_Type::SYSAHCLKCTRL1`
System clock group 1 control register, offset: 0x84

7.48.2.58 SYSAHCLKDIV

`__RW uint32_t SYSCON_Type::SYSAHCLKDIV`
System clock divider register, offset: 0x58

7.48.2.59 SYSMEMREMAP

`__RW uint32_t SYSCON_Type::SYSMEMREMAP`
System Remap register, offset: 0x0

7.48.2.60 SYSOSCCTRL

`__RW uint32_t SYSCON_Type::SYSOSCCTRL`
system oscillator control, offset: 0x20

7.48.2.61 SYSPLLCLKSEL

`__RW uint32_t SYSCON_Type::SYSPLLCLKSEL`
System PLL clock source select register, offset: 0x40

7.48.2.62 SYSPLLCLKUEN

`__RW uint32_t SYSCON_Type::SYSPLLCLKUEN`
System PLL clock source update enable register, offset: 0x44

7.48.2.63 SYSPLLCTRL

`__RW uint32_t SYSCON_Type::SYSPLLCTRL`
PLL control, offset: 0x8

7.48.2.64 SYSPLLSTAT

`__R uint32_t SYSCON_Type::SYSPLLSTAT`
PLL status, offset: 0xC

7.48.2.65 SYSRSTSTAT

`__RW uint32_t SYSCON_Type::SYSRSTSTAT`
System reset status register, offset: 0x38

7.48.2.66 SYSTCKCAL

`__RW uint32_t SYSCon_Type::SYSTCKCAL`
 System tick timer calibration register, offset: 0x154

7.48.2.67 WDTOSCCTRL

`__RW uint32_t SYSCon_Type::WDTOSCCTRL`
 Watchdog oscillator control, offset: 0x24
 The documentation for this struct was generated from the following file:

- [LPC845.h](#)

7.49 SysTick_t Struct Reference

Structure type to access the System [Timer](#) (SysTick).

```
#include <LPC845.h>
```

Public Attributes

- `__RW uint32_t CTRL`
- `__RW uint32_t RELOAD`
- `__RW uint32_t CURR`
- `__R uint32_t CALIB`

7.49.1 Detailed Description

Structure type to access the System [Timer](#) (SysTick).

SYSTICK - Register Layout Typedef

7.49.2 Member Data Documentation

7.49.2.1 CALIB

`__R uint32_t SysTick_t::CALIB`
 Offset: 0x00C (R/) SysTick Calibration Register

7.49.2.2 CTRL

`__RW uint32_t SysTick_t::CTRL`
 Offset: 0x000 (R/W) SysTick Control and Status Register

7.49.2.3 CURR

`__RW uint32_t SysTick_t::CURR`
 Offset: 0x008 (R/W) SysTick Current Value Register

7.49.2.4 RELOAD

`__RW uint32_t SysTick_t::RELOAD`
 Offset: 0x004 (R/W) SysTick Reload Value Register
 The documentation for this struct was generated from the following file:

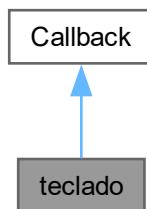
- [LPC845.h](#)

7.50 teclado Class Reference

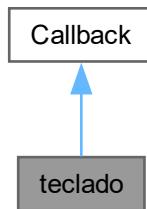
Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.

```
#include <teclado.h>
```

Inheritance diagram for teclado:



Collaboration diagram for teclado:



Public Member Functions

- **teclado** (vector< `gpio *` > &s, vector< `gpio *` > &r)
Constructor de un teclado.
- void **SWhandler** (void) override
Función handler/interrupcion del teclado.
- void **Initialize** (void)
Inicializa el teclado.
- `uint8_t Get` (void)
Devuelve el valor presionado en el teclado.
- virtual `~teclado` ()

Public Member Functions inherited from [Callback](#)

- void **SetInterrupt** ()
Activa la interrupción.
- void **UnSetInterrupt** ()
Desactiva la interrupción.

7.50.1 Detailed Description

Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 teclado()

```
teclado::teclado (
    vector< gpio * > & s,
    vector< gpio * > & r)
```

Constructor de un teclado.

Construye un objeto teclado con los parámetros indicados

Parameters

in	<i>s</i>	vector de filas
in	<i>r</i>	Vector de columnas

7.50.2.2 ~teclado()

```
teclado::~teclado () [virtual]
```

Destructor por defecto

7.50.3 Member Function Documentation

7.50.3.1 Get()

```
uint8_t teclado::Get (
    void )
```

Devuelve el valor presionado en el teclado.

Entrega el valor del buffer o NO_KEY en caso de no entregar nada. La tecla va desde 0 hasta el (filas*columnas)

Returns

Tecla presionada

7.50.3.2 Initialize()

```
void teclado::Initialize (
    void )
```

Inicializa el teclado.

Setea las direcciones y resistencias de las distintas entradas y salidas GPIO

7.50.3.3 SWhandler()

```
void teclado::SWhandler (
    void ) [override], [virtual]
```

Funcion handler/interrupcion del teclado.

Ejecuta el barrido y la lectura del antirrebote

Implements [Callback](#).

The documentation for this class was generated from the following files:

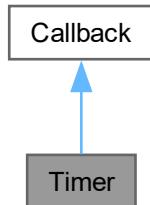
- [teclado.h](#)
- [teclado.cpp](#)

7.51 Timer Class Reference

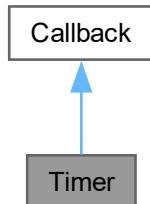
Clase del objeto timer.

```
#include <Timer.h>
```

Inheritance diagram for Timer:



Collaboration diagram for Timer:



Public Types

- enum `bases_t`
- enum `erroresTimers_t`

Enumeración de error del timer.
- enum `standby_t`

Enumeracion de stand By.
- typedef enum `Timer::bases_t bases_t`

Public Member Functions

- `Timer ()`

Constructor de clase timer.
- `Timer (const bases_t base, const Timer_Handler handler=nullptr)`

Constructor de clase timer.
- `void TimerStart (uint32_t time, const Timer_Handler handler, const bases_t base)`

Inicia un timer.
- `void SetTimer (uint32_t time)`

Inicia un timer.

- `uint32_t GetTimer (void) const`
Toma el valor al vuelo del timer en cuestion.
- `void StandByTimer (const uint8_t accion)`
Detiene/Arranca el timer, NO lo resetea.
- `void SetTimerBase (const bases_t base)`
Setea la base de tiempo.
- `void TimerStop (void)`
Detiene el timer.
- `uint32_t GetTmrRun (void)`
Obtiene el valor del timer.
- `void SetTmrEvent (void)`
Setea el evento del timer.
- `void ClrTmrEvent (void)`
Limpia el evento del timer.
- `bool GetTmrEvent (void)`
Indica si el timer venció o no.
- `bool GetmrStandBy (void)`
Informa si el timer está detenido o no.
- `void SetmrStandBy (uint8_t accion)`
Detiene el timer sin apagarlo.
- `void SetTmrHandler (void)`
Ejecuta el timer Handler.
- `void TimerStart (uint32_t time)`
Inicia un timer.
- `Timer & operator= (uint32_t t)`
Sobrecarga de del operador de asignacion.
- `bool operator! ()`
Sobrecarga de del operador de negacion.
- `operator bool ()`
Sobrecarga de del operador de contenido.
- `bool operator== (uint32_t t)`
Sobrecarga de del operador de comparacion.
- `void SWhandler (void) override`
Decremento periodico del timer. Debe ser llamada periodicamente con la base de tiempos.
- `int8_t TmrEvent (void)`
Función encargada de revisar si los timer vencieron y ejecuta automáticamente su función asignada.

Public Member Functions inherited from [Callback](#)

- `void SetInterrupt ()`
Activa la interrupción.
- `void UnSetInterrupt ()`
Desactiva la interrupción.

Protected Attributes

- `volatile uint32_t m_TmrRun`
- `volatile bool m_TmrEvent`
- `void(* m_TmrHandler)(void)`
- `volatile bool m_TmrStandBy`
- `volatile bases_t m_TmrBase`

Friends

- bool `operator== (uint32_t t, Timer &T)`
Sobrecarga de del operador de comparacion.

7.51.1 Detailed Description

Clase del objeto timer.

El objeto timer se conecta al systick y ejecuta una "alarma" al terminar el tiempo. Se comporta como un cronómetro o contador de microondas.

7.51.2 Member Typedef Documentation**7.51.2.1 bases_t**

```
typedef enum Timer::bases_t Timer::bases_t
```

Base de tiempo del timer

7.51.3 Member Enumeration Documentation**7.51.3.1 bases_t**

```
enum Timer::bases_t
```

Base de tiempo del timer

7.51.4 Constructor & Destructor Documentation**7.51.4.1 Timer() [1/2]**

```
Timer::Timer ()
```

Constructor de clase timer.
Crea un timer por defecto

7.51.4.2 Timer() [2/2]

```
Timer::Timer (
    const bases_t base,
    const Timer_Handler handler = nullptr)
```

Constructor de clase timer.
Crea un timer con los parámetros correspondientes.

Parameters

in	<i>handler</i>	Funcion a ejecutar en caso de expirar.
in	<i>base</i>	Base de tiempo del timer.

7.51.5 Member Function Documentation**7.51.5.1 GetmrStandBy()**

```
bool Timer::GetmrStandBy (
    void )
```

Informa si el timer está detenido o no.

Returns

True si el timer esta detenido. False si no.

7.51.5.2 GetTimer()

```
uint32_t Timer::GetTimer (
    void ) const
```

Toma el valor al vuelo del timer en cuestion.
Lee el timer al vuelo.

Returns

`uint32_t`: valor del timer.

7.51.5.3 GetTmrEvent()

```
bool Timer::GetTmrEvent (
    void )
```

Indica si el timer venció o no.

Returns

True si venció. False si no.

7.51.5.4 GetTmrRun()

```
uint32_t Timer::GetTmrRun (
    void )
```

Obtiene el valor del timer.

Returns

Valor del timer.

7.51.5.5 operator bool()

```
Timer::operator bool () [explicit]
```

Sobrecarga de del operador de contenido.

Returns

true por timer vencido y false por no vencido.

7.51.5.6 operator"!"()

```
bool Timer::operator! ()
```

Sobrecarga de del operador de negacion.

Returns

true por timer no vencido y false por vencido.

7.51.5.7 operator=()

```
Timer & Timer::operator= (
    uint32_t time)
```

Sobrecarga de del operador de asignacion.

Parameters

in	<code>time</code>	Valor a asignar a la variable de temporizacion.
----	-------------------	---

Returns

una referencia al propio objeto.

7.51.5.8 operator==()

```
bool Timer::operator== (
```

```
    uint32_t ev)
```

Sobrecarga de del operador de comparacion.
compara un valor numerico contra el flag de finalizacion del timer.

Parameters

in	ev	valor de comparacion (para verificar si vencio el timer).
----	----	---

Returns

bool: true por coincidencia, false por no coincidencia.

7.51.5.9 SetmrStandBy()

```
void Timer::SetmrStandBy (
```

```
    uint8_t accion)
```

Detiene el timer sin apagarlo.

Parameters

in	accion	si lo detiene o no.
----	--------	---------------------

7.51.5.10 SetTimer()

```
void Timer::SetTimer (
```

```
    uint32_t time)
```

Inicia un timer.

Reinicia el timer con el valor t (no lo resetea).

Parameters

in	time	time Tiempo del evento. Dependiente de la base de tiempos.
----	------	--

7.51.5.11 SetTimerBase()

```
void Timer::SetTimerBase (
```

```
    const bases_t base)
```

Setea la base de tiempo.

Fija la base de tiempo entre todos los posibles de [bases_t](#).

Parameters

in	base	Base de tiempo a utilizar.
----	------	----------------------------

7.51.5.12 SetTmrHandler()

```
void Timer::SetTmrHandler (
```

```
    void )
```

Ejecuta el timer Handler.

Llama a la funcion handler entregada por el constructor (siempre que no sea nullptr).

7.51.5.13 StandByTimer()

```
void Timer::StandByTimer (
    const uint8_t accion)
```

Detiene/Arranca el timer, NO lo resetea.
lo pone o lo saca de stand-by.

Parameters

in	<i>accion</i>	RUN lo arranca, PAUSE lo pone en stand-by.
----	---------------	--

7.51.5.14 TimerStart() [1/2]

```
void Timer::TimerStart (
    uint32_t time)
```

Inicia un timer.
Inicia el timer y al transcurrir el tiempo especificado se llama a la funcion apuntada por handler.

Parameters

in	<i>time</i>	Tiempo del evento. Dependiente de la base de tiempos.
----	-------------	---

7.51.5.15 TimerStart() [2/2]

```
void Timer::TimerStart (
    uint32_t time,
    const Timer_Handler handler,
    const bases_t base)
```

Inicia un timer.
Inicia el timer y al transcurrir el tiempo especificado por *time* se llama a la funcion apuntada por *handler*.

Parameters

in	<i>time</i>	Tiempo del evento. Dependiente de la base de tiempos.
in	<i>handler</i>	Callback del evento.
in	<i>base</i>	de tiempo.

7.51.5.16 TmrEvent()

```
int8_t Timer::TmrEvent (
    void)
```

Función encargada de revisar si los timer vencieron y ejecuta automáticamente su función asignada.
Si el timer tiene una funcion asignada, debe colocarse en el while(1) para que se ejecuten automáticamente.

Returns

OKtimers si el timer venció. errorTimer si el timer no venció.

7.51.6 Friends And Related Symbol Documentation

7.51.6.1 operator==

```
bool operator== (
    uint32_t t,
    Timer & T) [friend]
```

Sobrecarga del operador de comparacion.
compara un valor numerico contra el flag de finalizacion del timer

Returns

bool: true por coincidencia, false por no coincidencia

7.51.7 Member Data Documentation

7.51.7.1 m_TmrBase

volatile [bases_t](#) Timer::m_TmrBase [protected]

Base de tiempo del timer

7.51.7.2 m_TmrEvent

volatile bool Timer::m_TmrEvent [protected]

True si el timer venció. False si el timer no venció

7.51.7.3 m_TmrHandler

void(* Timer::m_TmrHandler) (void) [protected]

Función a ejecutar al terminar

7.51.7.4 m_TmrRun

volatile [uint32_t](#) Timer::m_TmrRun [protected]

Tiempo de encendido del timer

7.51.7.5 m_TmrStandBy

volatile bool Timer::m_TmrStandBy [protected]

True si el timer está detenido. False si el timer no lo está

The documentation for this class was generated from the following files:

- [Timer.h](#)
- [Timer.cpp](#)

7.52 timers Class Reference

Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.

```
#include <Timers.h>
```

Public Member Functions

- [timers \(\)](#)
Constructor de clase timers.
- [timers & operator<< \(Timer *t\)](#)
Sobrecarga de del operador <<.
- [void TmrEvent \(void\)](#)
Funcion que revisa si los timer vencieron y los enciende.
- [virtual ~timers \(\)=default](#)

7.52.1 Detailed Description

Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 timers()

`timers::timers ()`

Constructor de clase timers.

Crea un timers con los parámetros correspondientes.

7.52.2.2 ~timers()

`virtual timers::~timers () [virtual], [default]`

Destructor por defecto

7.52.3 Member Function Documentation

7.52.3.1 operator<<()

`timers & timers::operator<< (`
`Timer * t)`

Sobrecarga de del operador <<.

Agrega a la lista de timers el timer pasado.

Parameters

in	<code>t</code>	timer a agregar a la lista.
----	----------------	-----------------------------

Returns

Referencia a si mismo.

7.52.3.2 TmrEvent()

`void timers::TmrEvent (`
`void)`

Funcion que revisa si los timer vencieron y los enciende.

Esta función debe colocarse en el while(1) para que permita revisar los timers vencidos y ejecutar sus funciones asignadas. SOLO USAR SI TIENEN FUNCIONES ASIGNADAS.

The documentation for this class was generated from the following files:

- [Timers.h](#)
- [Timers.cpp](#)

7.53 Uart Class Reference

Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.

`#include <UART.h>`

7.53.1 Detailed Description

Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.

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

- [UART.h](#)

7.54 USART_Type Struct Reference

Structure type to access the Universal Serial Asincronical Reciver Transmitter (USART).

`#include <LPC845.h>`

Public Attributes

- `__RW uint32_t CFG`
- `__RW uint32_t CTL`
- `__RW uint32_t STAT`
- `__RW uint32_t INTENSET`
- `__W uint32_t INTENCLR`
- `__R uint32_t RXDAT`
- `__R uint32_t RXDATSTAT`
- `__RW uint32_t TXDAT`
- `__RW uint32_t BRG`
- `__R uint32_t INTSTAT`
- `__RW uint32_t OSR`
- `__RW uint32_t ADDR`

7.54.1 Detailed Description

Structure type to access the Universal Serial Asincronical Reciver Transmitter (USART).
USART - Register Layout Typedef

7.54.2 Member Data Documentation

7.54.2.1 ADDR

`__RW uint32_t USART_Type::ADDR`

Address register for automatic address matching., offset: 0x2C

7.54.2.2 BRG

`__RW uint32_t USART_Type::BRG`

Baud Rate Generator register. 16-bit integer baud rate divisor value., offset: 0x20

7.54.2.3 CFG

`__RW uint32_t USART_Type::CFG`

USART Configuration register. Basic USART configuration settings that typically are not changed during operation., offset: 0x0

7.54.2.4 CTL

`__RW uint32_t USART_Type::CTL`

USART Control register. USART control settings that are more likely to change during operation., offset: 0x4

7.54.2.5 INTENCLR

`__W uint32_t USART_Type::INTENCLR`

Interrupt Enable Clear register. Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared., offset: 0x10

7.54.2.6 INTENSET

`__RW uint32_t USART_Type::INTENSET`

Interrupt Enable read and Set register. Contains an individual interrupt enable bit for each potential USART interrupt. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set., offset: 0xC

7.54.2.7 INTSTAT

`__R uint32_t USART_Type::INTSTAT`

Interrupt status register. Reflects interrupts that are currently enabled., offset: 0x24

7.54.2.8 OSR

`__RW uint32_t USART_Type::OSR`
Oversample selection register for asynchronous communication., offset: 0x28

7.54.2.9 RXDAT

`__R uint32_t USART_Type::RXDAT`
Receiver Data register. Contains the last character received., offset: 0x14

7.54.2.10 RXDATSTAT

`__R uint32_t USART_Type::RXDATSTAT`
Receiver Data with Status register. Combines the last character received with the current USART receive status.
Allows DMA or software to recover incoming data and status together., offset: 0x18

7.54.2.11 STAT

`__RW uint32_t USART_Type::STAT`
USART Status register. The complete status value can be read here. Writing ones clears some bits in the register.
Some bits can be cleared by writing a 1 to them., offset: 0x8

7.54.2.12 TXDAT

`__RW uint32_t USART_Type::TXDAT`
Transmit Data register. Data to be transmitted is written here., offset: 0x1C
The documentation for this struct was generated from the following file:

- [LPC845.h](#)

Chapter 8

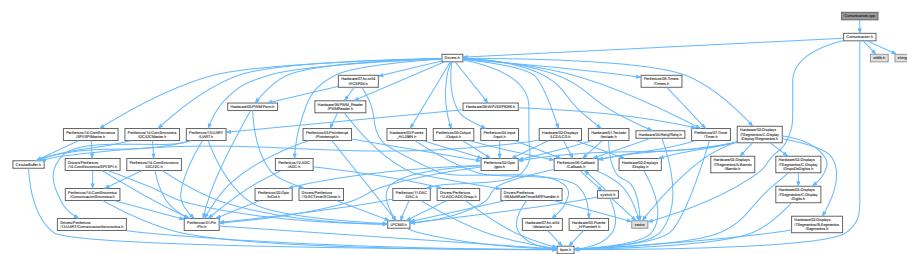
File Documentation

8.1 Comunicacion.cpp File Reference

Clase máquina de estados de comunicación del proyecto.

```
#include "Comunicacion.h"
```

Include dependency graph for Comunicacion.cpp:



8.1.1 Detailed Description

Clase máquina de estados de comunicación del proyecto.

Date

23 nov. 2022

Author

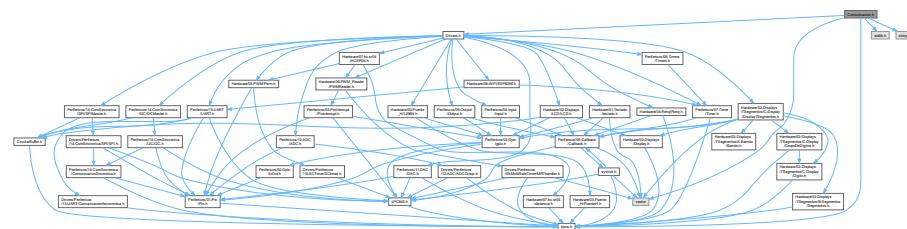
Grupo 4

8.2 Comunicacion.h File Reference

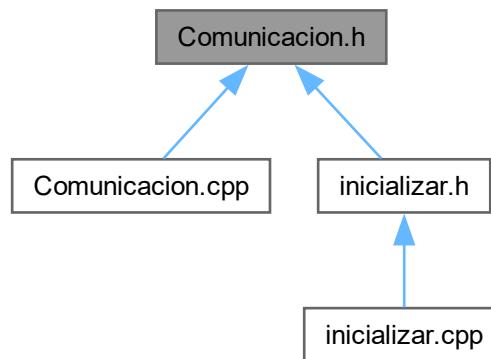
Clase máquina de estados de comunicación del proyecto.

```
#include "tipos.h"
#include "Drivers.h"
#include <vector>
#include <stdlib.h>
#include "string.h"
```

Include dependency graph for Comunicacion.h:



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



8.2.1 Detailed Description

Clase máquina de estados de comunicación del proyecto.

Date

23 nov. 2022

Author

Grupo 4

8.3 Comunicacion.h

[Go to the documentation of this file.](#)

00001

```
*****
```

00009

00010

```
*****
```

00011 *** MODULO

00012

```
*****
```

00013 #ifndef COMUNICACION_H

00014 #define COMUNICACION_H

00015

00016

```
*****
```

00017 *** INCLUDES GLOBALES

```

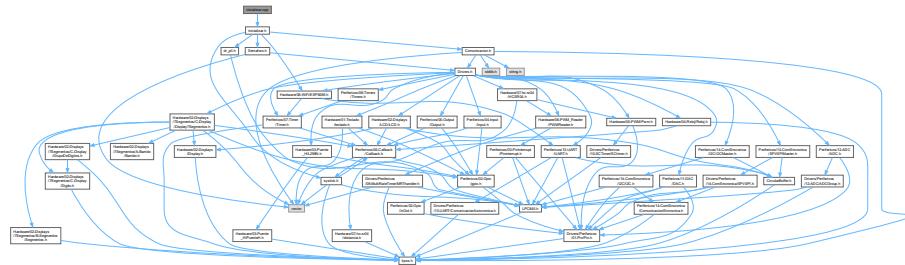
00018
00019 #include "tipos.h"
00020 #include "Drivers.h"
00021 #include <vector>
00022 #include <stdlib.h>      /* atoi */
00023 #include "string.h"       /* strlen , strcmp , strchr */
00024
00025 /*****
00026 *** DEFINES GLOBALES
00027
00028 /*****
00029 *** MACROS GLOBALES
00030
00031
00032 /*****
00033 *** TIPO DE DATOS GLOBALES
00034
00035
00036 /*****
00037 *** VARIABLES GLOBALES
00038
00039
00040 /*****
00041 *** IMPLANTACION DE UNA CLASE
00042
00043
00044 class Comunicacion
00045 {
00046     private:
00047         enum estados { E_VERDE , E_AMARILLO , E_ROJO , E_NADA};
00048
00049     private:
00050         UART * m_com;
00051         Reloj m_reloj;
00052         vector <uint32_t > &m_msg;           //Vector compartido con las demás máquinas de
00053         estado (y por donde se mandan mensajes)
00054
00055         enum estados m_Estado ;
00056         void (Comunicacion::*m_maquina[4]) (void);
00057
00058         uint8_t m_ultimo_msg;
00059         int8_t m_hora[15];
00060
00061     public:
00062         Comunicacion( UART * _com , vector <uint32_t > &msg );
00063         ~Comunicacion();
00064         void Transmitir ( char * text );
00065         void Transmitir ( char * text , int32_t n );
00066         void MaquinaDeEstados ( void );
00067
00068     private: //Maquina de estado
00069         void Verde( void );
00070         void Amarillo( void );
00071         void Rojo( void );
00072         void Nada( void );
00073         void revisarRx( void );
00074         void intToStr(int32_t num, int8_t* str);
00075 };
00076 #endif /* COMUNICACION_H_ */

```

8.4 inicializar.cpp File Reference

Archivo inicializador de la placa.

```
#include "inicializar.h"
Include dependency graph for inicializar.cpp:
```



Variables

- Semaforo * **Semaforo1**
 < *Semaforo*
 - **Output** * **led_roj**
 COMUNICACION.

8.4.1 Detailed Description

Archivo inicializador de la placa.

Date

26 ene. 2023

Author

Técnico Martinez Agustín

Version

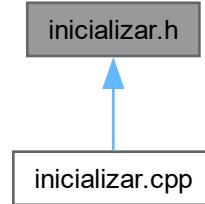
v1.0

8.5 inicializar.h File Reference

Archivo inicializador de la placa.

```
#include <Hardware/08-WiFi/ESP8266.h>
#include "dr_pll.h"
#include "systick.h"
#include "Semaforo.h"
#include "Comunicacion.h"
Include dependency graph for inicializar.h:
```

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



Variables

- Semaforo * **Semaforo1**
< Semaforo

8.5.1 Detailed Description

Archivo inicializador de la placa.

Date

26 ene. 2023

Author

Técnico Martínez Agustín

Version

v1.0

8.6 inicializar.h

[Go to the documentation of this file.](#)

```
00001 /*****
00010
00011 **** MODULO
00012 *** INCLUIDOS GLOBALES
00013 **** INCLUDES GLOBALES
00014 #ifndef INICIALIZAR_H_
00015 #define INICIALIZAR_H_
00016
00017 **** VARIABLES GLOBALES
00018 *** VARIABLES GLOBALES
00019 **** VARIABLES GLOBALES
00020 #include <Hardware/08-WiFi/ESP8266.h>
00021 #include "dr_pll.h"
00022 #include "systick.h"
00023 #include "Semaforo.h"
00024 #include "Comunicacion.h"
00025 **** VARIABLES GLOBALES
00026 *** VARIABLES GLOBALES
00027 **** VARIABLES GLOBALES
```

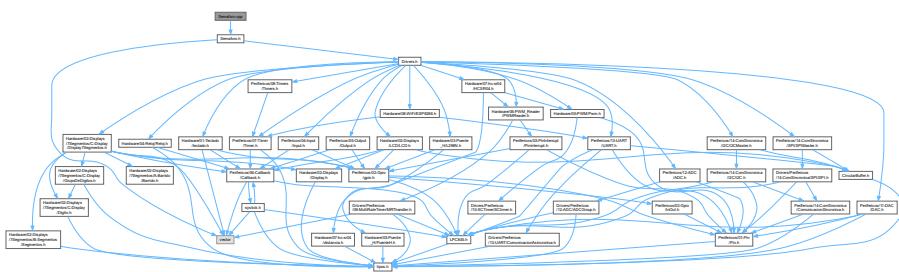
```
00028 extern Semaforo *           Semaforol;
00029 extern Comunicacion *       maquina_com;
00030 extern ESP8266 * wifi;
00031
00032 /*****
00032 *** IMPLANTACION DE UNA CLASE
00033 ****
00034 void InicializarFirmware ( void );
00035 void inicializarApp ( void );
00036
00037 #endif /* INICIALIZAR_H */
```

8.7 Semaforo.cpp File Reference

Clases semáforo de ejemplo utilizando led tricolor propio de la placa.

```
#include "Semaforo.h"
```

Include dependency graph for Semaforo.cpp:



8.7.1 Detailed Description

Clases semáforo de ejemplo utilizando led tricolor propio de la placa.

Date

26 ene. 2023

Author

Técnico Martínez Agustín

Version

v1.0

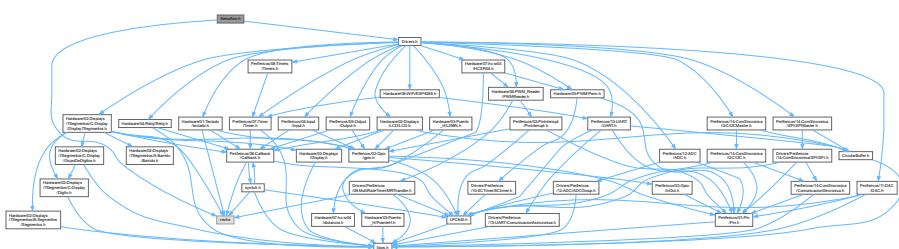
8.8 Semaforo.h File Reference

Clases semáforo de ejemplo utilizando led tricolor propio de la placa.

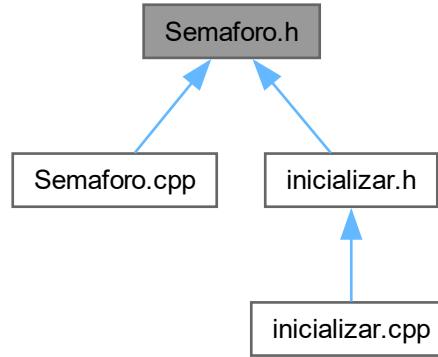
```
#include "Drivers.h"
```

```
#include <vector>
```

Include dependency graph for Semaforo.h:



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



8.8.1 Detailed Description

Clases semaforo de ejemplo utilizando led tricolor propio de la placa.

Date

26 ene. 2023

Author

Técnico Martinez Agustín

Version

v1.0

8.9 Semaforo.h

[Go to the documentation of this file.](#)

```
00001 /*****
00010
00011 ***
00012 *** MODULO
00013 ****
00014 #ifndef SEMAFORO_H_
00015 #define SEMAFORO_H_
00016
00017 /**
00018 *** INCLUDES GLOBALES
00019 ****
00020 #include "Drivers.h"
00021 #include <vector>
00022 /**
00023 *** DEFINES GLOBALES
00024 ****
00025
00026 /**
00027 *** MACROS GLOBALES
```

```

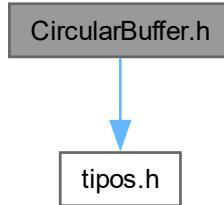
00028
***** ****
00029
00030
00031     *** TIPO DE DATOS GLOBALES
00032
***** ****
00033
00034
00035     /**** VARIABLES GLOBALES
00036
***** ****
00037
00038
00039     /**** IMPLANTACION DE LA CLASE
00040
***** ****
00041 class Semaforo {
00042     private:
00043         enum Estados { RESET = 0 , VERDE , AMARILLO , ROJO };           //Posiciones en el vector de los
00044         CASES.
00045         private:
00046             Output*      &m_verde;          //Periférico a usar
00047             Output*      &m_amarillo;        //Periférico a usar
00048             Output*      &m_rojo;           //Periférico a usar
00049             vector <uint32_t >  &m_msg;       //Vector por referencia donde se envian las comunicaciones
00050             entre distintas máquinas de estado
00051             Timer        m_timer_verde;
00052             Timer        m_timer_amarillo;
00053             Timer        m_timer_rojo;
00054
00055             uint8_t      m_tiempo_verde;
00056             uint8_t      m_tiempo_amarillo;
00057             uint8_t      m_tiempo_rojo;
00058
00059             bool         m_encendido;
00060
00061             uint8_t      m_estado;           //Variable de la máquina de estado
00062             void (Semaforo::*m_maquina[4]) (void); //Vector con todos los CASES de la máquina de estado
00063
00064         public:
00065             Semaforo( Output* &_verde , Output* &_amarillo , Output* &_rojo , vector <uint32_t > &_msg);
00066             void MaquinaDeEstados ( void );
00067
00068             void SetTimeVerde ( uint8_t time );
00069             void SetTimeAmarillo ( uint8_t time );
00070             void SetTimeRojo ( uint8_t time );
00071
00072             virtual ~Semaforo() {};
00073
00074         private:
00075             void Reset ( void );
00076
00077             void Verde ( void );
00078             void Amarillo ( void );
00079             void Rojo ( void );
00080     };
00081
00082 #endif /* SEMAFORO_H_ */

```

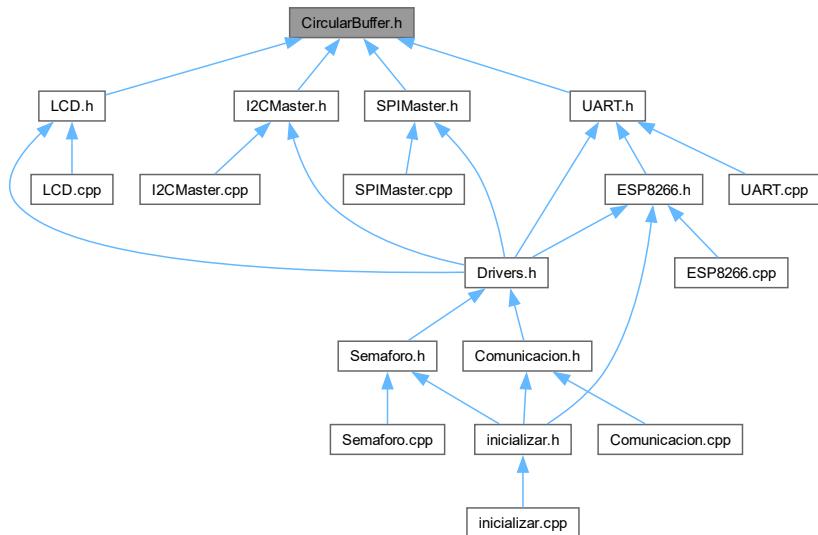
8.10 CircularBuffer.h File Reference

Modulo de comunicacion [CircularBuffer](#).

```
#include <tipos.h>
Include dependency graph for CircularBuffer.h:
```



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



Classes

- class [CircularBuffer< T >](#)

Clase del objeto CircularBuffer.

8.10.1 Detailed Description

Modulo de comunicacion [CircularBuffer](#).

Date

8 ene. 2025

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.11 CircularBuffer.h

[Go to the documentation of this file.](#)

```

00001  /*****
00012  ****
00013  *** MODULO
00014  ****
00015  #ifndef CIRCULARBUFFER_H_
00016  #define CIRCULARBUFFER_H_
00020  ****
00021  *** INCLUDES GLOBALES
00022  ****
00023  #include <tipos.h>
00024  ****
00025  *** DEFINES GLOBALES
00026  ****
00027  ****
00028  ****
00029  *** MACROS GLOBALES
00030  ****
00031  ****
00032  ****
00033  *** TIPO DE DATOS GLOBALES
00034  ****
00035  ****
00036  ****
00037  *** VARIABLES GLOBALES
00038  ****
00039  ****
00040  ****
00041  *** IMPLANTACION DE UNA CLASE
00042  ****
00048 template<typename T>
00049 class CircularBuffer {
00050 public:
00051 private:
00052     T*          m_buffer;
00053     uint32_t    m_idxIn, m_idxOut, m_size;
00054 public:
00061     CircularBuffer(uint32_t size) :
00062         m_buffer(new T[size]), m_idxIn(0), m_idxOut(0), m_size(size) {
00063     }
00071     bool pop(T *item) {
00072         if (m_idxIn == m_idxOut)
00073             return false;
00074         *item = m_buffer[m_idxOut++];
00075         m_idxOut %= m_size;
00076         return true;
00077     }
00083     void push(T item) {
00084         m_buffer[m_idxIn++] = item;
00085         m_idxIn %= m_size;
00086     }
00087     bool isFull(void) const {
00095         return (m_idxIn == m_idxOut);
00096     }
00102     virtual ~CircularBuffer() {
00103         delete[] m_buffer;

```

```

00104      }
00105  };
00106
00107 #endif /* CIRCULARBUFFER_H_ */

```

8.12 Drivers.h File Reference

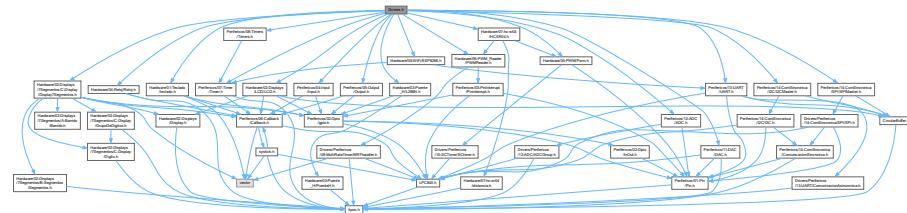
Archivo Master con todos los includes de perifericos del Kit.

```

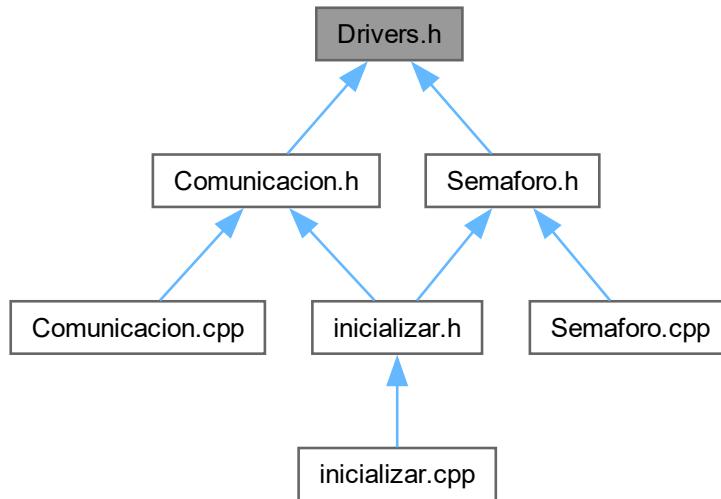
#include <Hardware/01-Teclado/teclado.h>
#include <Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h>
#include <Hardware/02-Displays/LCD/LCD.h>
#include <Hardware/03-Puente_H/L298N.h>
#include <Hardware/04-Reloj/Reloj.h>
#include <Hardware/05-PWM/Pwm.h>
#include <Hardware/06-PWM_Reader/PWMReader.h>
#include <Hardware/07-hc-sr04/HCSR04.h>
#include <Hardware/08-WiFi/ESP8266.h>
#include <Perifericos/04-Input/Input.h>
#include <Perifericos/05-Output/Output.h>
#include <Perifericos/07-Timer/Timer.h>
#include <Perifericos/08-Timers/Timers.h>
#include <Perifericos/11-DAC/DAC.h>
#include <Perifericos/12-ADC/ADC.h>
#include <Perifericos/13-UART/UART.h>
#include <Perifericos/14-ComSincronica/I2C/I2CMaster.h>
#include <Perifericos/14-ComSincronica/SPI/SPIMaster.h>

```

Include dependency graph for Drivers.h:



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



8.12.1 Detailed Description

Archivo Master con todos los includes de perifericos del Kit.

Date

29 dic. 2024

Version

2.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.13 Drivers.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef DRIVERS_H_
00013 #define DRIVERS_H_
00014
00015 // Lista de todos los include de Hardware creados. Comentar los que no serán utilizados.
00016 #include <Hardware/01-Teclado/teclado.h>
00017 #include <Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h>
00018 #include <Hardware/02-Displays/LCD/LCD.h>
00019 #include <Hardware/03-Puente_H/L298N.h>
00020 #include <Hardware/04-Reloj/Reloj.h>
00021 #include <Hardware/05-PWM/Pwm.h>
00022 #include <Hardware/06-PWM_Reader/PWMReader.h>
00023 #include <Hardware/07-hc-sr04/HCSR04.h>
00024 #include <Hardware/08-WiFi/ESP8266.h>
00025 #include <Perifericos/04-Input/Input.h>
00026 #include <Perifericos/05-Output/Output.h>
00027 #include <Perifericos/07-Timer/Timer.h>
00028 #include <Perifericos/08-Timers/Timers.h>
00029 #include <Perifericos/11-DAC/DAC.h>
00030 #include <Perifericos/12-ADC/ADC.h>
  
```

```

00031 #include <Perifericos/13-UART/UART.h>
00032 #include <Perifericos/14-ComSincronica/I2C/I2CMaster.h>
00033 #include <Perifericos/14-ComSincronica/SPI/SPIMaster.h>
00034
00035
00036 #endif /* DRIVERS_H_ */

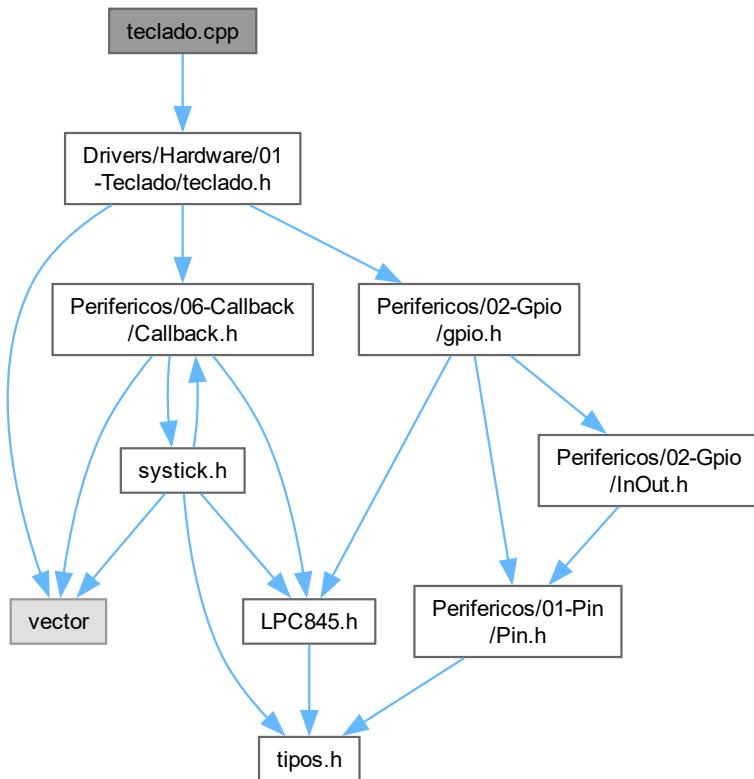
```

8.14 teclado.cpp File Reference

Modulo de teclado matricial.

```
#include <Drivers/Hardware/01-Teclado/teclado.h>
```

Include dependency graph for teclado.cpp:



8.14.1 Detailed Description

Modulo de teclado matricial.

Date

24 jul. 2022

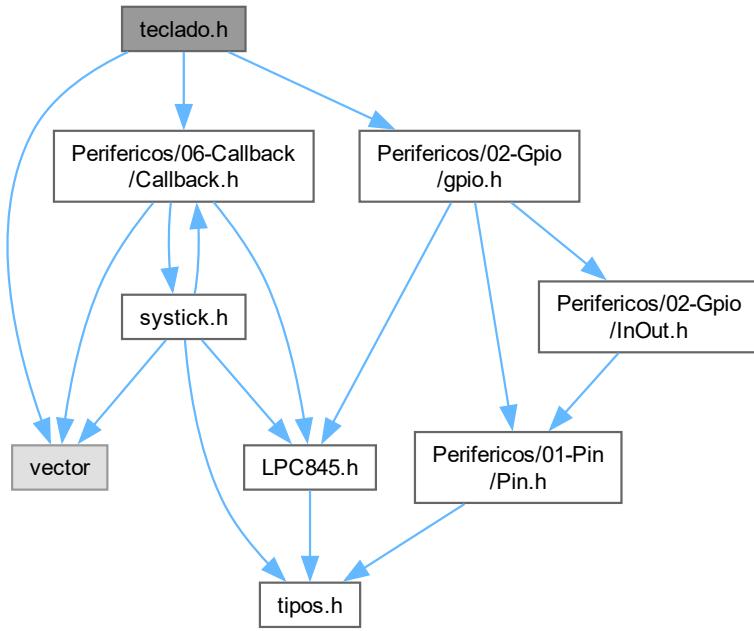
Author

Ing. Marcelo Trujillo

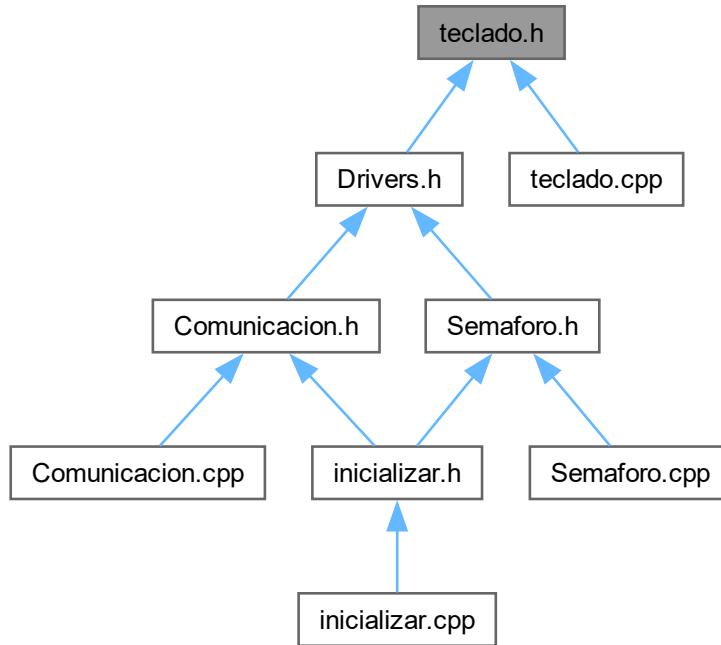
8.15 teclado.h File Reference

Modulo de teclado matricial.

```
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
#include <vector>
Include dependency graph for teclado.h:
```



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



Classes

- class [teclado](#)

Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.

Macros

- `#define NO_KEY 0xff`

8.15.1 Detailed Description

Modulo de teclado matricial.

Date

24 jul. 2022

Author

Ing. Marcelo Trujillo

8.15.2 Macro Definition Documentation

8.15.2.1 NO_KEY

```
#define NO_KEY 0xff
Número representativo a "ninguna tecla se a presionado"
```

8.16 teclado.h

[Go to the documentation of this file.](#)

```

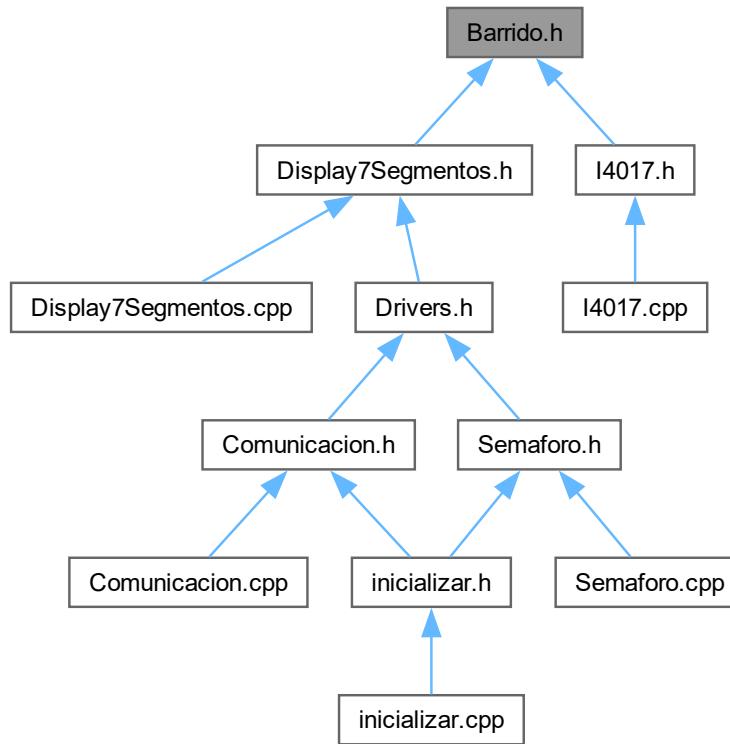
00001  /*
00009
0010
0011  *** MODULO
0012
0013
0014 #ifndef TECLADO_H_
0015 #define TECLADO_H_
0019
0020  *** INCLUDES GLOBALES
0021
0022 #include <Perifericos/02-Gpio/gpio.h>
0023 #include <Perifericos/06-Callback/Callback.h>
0024 #include <vector>
0025
0026
0027  *** DEFINES GLOBALES
0028
0029
0030
0031  *** MACROS GLOBALES
0032
0033
0034
0035  *** TIPO DE DATOS GLOBALES
0036
0037
0038
0039  *** VARIABLES GLOBALES
0040
0041
0042
0043  *** IMPLANTACION DE UNA CLASE
0044
0052 class teclado : public Callback
0053 {
0054     public:
0055         #define NO_KEY      0xff
0056     private:
0057         const vector<gpio *> &m_scn;
0058         const vector<gpio *> &m_ret;
0059
0060         uint8_t      m_TeclaEstadoInicial ;
0061         uint32_t     m_TeclaCantidadDeRebotes ;
0062         uint8_t      m_BufferTeclado ;
0063         const uint8_t   m_MaxRebotes;
0064         const uint32_t  m_RebotesHold ;
0065
0066
0067     public:
0068         teclado ( vector<gpio *> &s , vector<gpio *> &r );
0069         void    SWhandler ( void ) override;
0070         void    Initialize ( void );
0071         uint8_t  Get( void );
0072         virtual ~teclado();
0073
0074     private:
0075         uint8_t TecladoHW ( void );
0076         void    TecladoSW ( uint8_t TeclaEstadoActual );
0077 };
0078
0079 #endif /* TECLADO_H_ */

```

8.17 Barrido.h File Reference

Clase abstracta de manejo barridos de datos.

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



Classes

- class [barrido](#)

Clase del objeto barrido Clase abstracta pura para la generación de barridos.

8.17.1 Detailed Description

Clase abstracta de manejo barridos de datos.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

8.18 Barrido.h

[Go to the documentation of this file.](#)

00001

00009

00010

00011

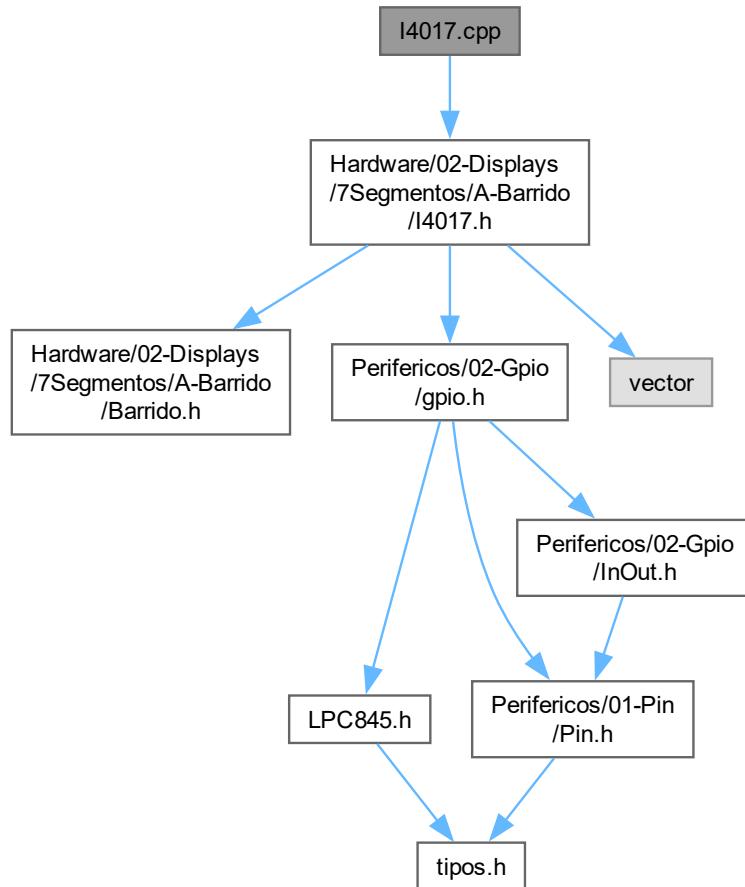
00012

```
00013 #ifndef BARRIDO_H_
00014 #define BARRIDO_H_
00018 /*****
00019 *** INCLUDES GLOBALES
00020 ****
00021
00022 /*****
00023 *** DEFINES GLOBALES
00024 ****
00025
00026 /*****
00027 *** MACROS GLOBALES
00028 ****
00029
00030 /*****
00031 *** TIPO DE DATOS GLOBALES
00032 ****
00033
00034 /*****
00035 *** VARIABLES GLOBALES
00036 ****
00037 /*****
00038 *** IMPLANTACION DE UNA CLASE
00039 ****
00045 class barrido
00046 {
00047     public:
00048         barrido();
00049         virtual void SetDigito ( void ) = 0;
00050         virtual void Initialize ( void ) = 0;
00051         virtual ~barrido();
00052 };
00053
00054 #endif /* BARRIDO_H_ */
```

8.19 I4017.cpp File Reference

Objeto de control del integrado [I4017](#).

```
#include <Hardware/02-Displays/7Segmentos/A-Barrido/I4017.h>
Include dependency graph for I4017.cpp:
```



8.19.1 Detailed Description

Objeto de control del integrado [I4017](#).

Date

27 jul. 2022

Author

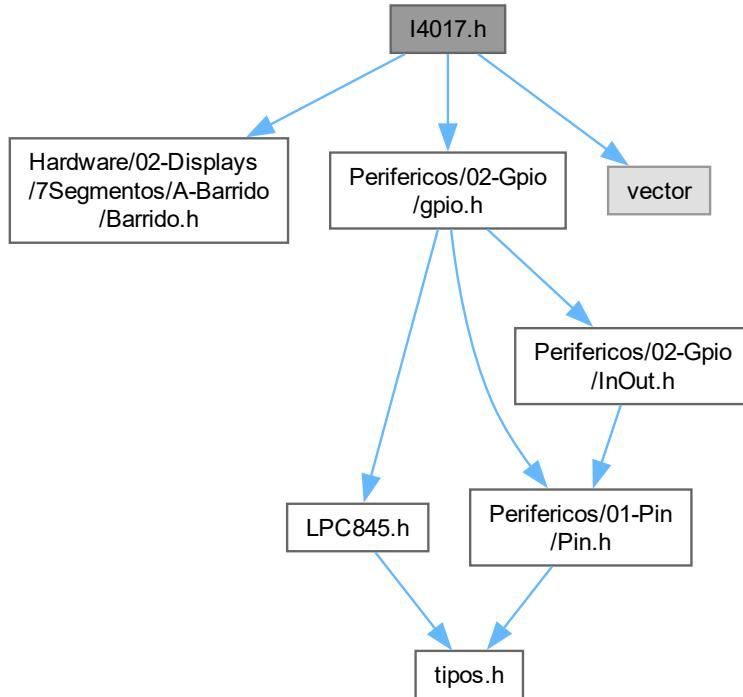
Ing. Marcelo Trujillo

8.20 I4017.h File Reference

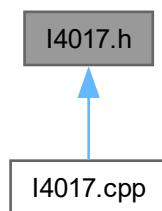
Objeto de control del integrado [I4017](#).

```
#include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <vector>
```

Include dependency graph for l4017.h:



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



Classes

- class [l4017](#)

Clase del objeto l4017 El objeto l4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a través de sus patas.

8.20.1 Detailed Description

Objeto de control del integrado [l4017](#).

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

8.21 I4017.h

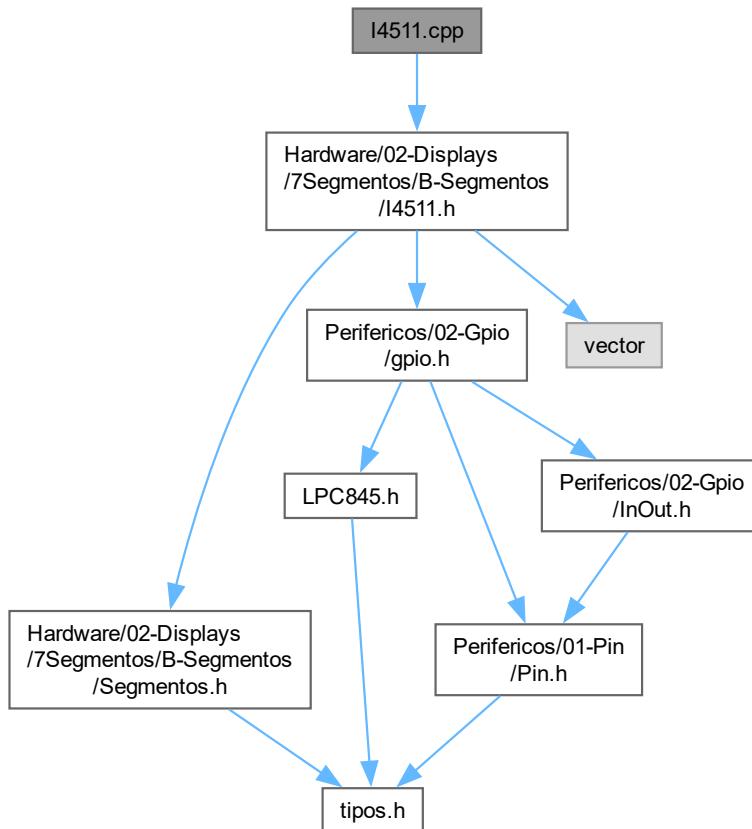
[Go to the documentation of this file.](#)

```
00001  /*****
00002  ****
00003  **** MODULO
00004  ****
00005  **** INCLUDES GLOBALES
00006  ****
00007  **** DEFINES GLOBALES
00008  ****
00009  **** MACROS GLOBALES
00010  ****
00011  **** TIPO DE DATOS GLOBALES
00012  ****
00013  **** VARIABLES GLOBALES
00014  ****
00015  **** IMPLANTACION DE UNA CLASE
00016  ****
00017  class I4017 : public barrido
00018  {
00019      uint8_t m_indice ;
00020      const vector <gpio * > &m_pins4017;
00021      const uint8_t m_maxsalidas ;
00022
00023      public:
00024          // tiene que recibir los gpio clk y reset
00025          I4017( const vector <gpio * > &pins4017 , uint8_t maxsalidas ) : m_indice (0) , m_pins4017
00026          (pins4017 ) , m_maxsalidas ( (maxsalidas <= 10 ) ? maxsalidas:10) {}
00027
00028          void SetDigito ( void ) override;
00029          void SetReset( void );
00030          void SetClock( void );
00031          void Initialize ( void ) override;
00032          virtual ~I4017() {};
00033
00034
00035  #endif /* I4017_H_ */
```

8.22 I4511.cpp File Reference

Objeto de control del integrado [I4511](#).

```
#include <Hardware/02-Displays/7Segmentos/B-Segmentos/I4511.h>
Include dependency graph for I4511.cpp:
```



8.22.1 Detailed Description

Objeto de control del integrado [I4511](#).

Date

27 jul. 2022

Author

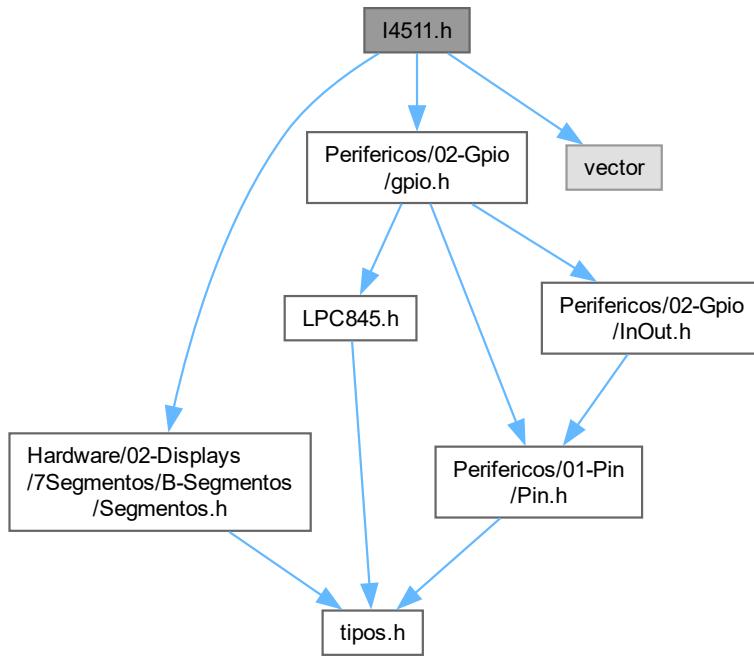
Ing. Marcelo Trujillo

8.23 I4511.h File Reference

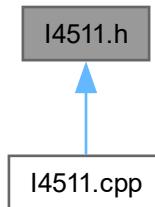
Objeto de control del integrado [I4511](#).

```
#include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <vector>
```

Include dependency graph for I4511.h:



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



Classes

- class [I4511](#)

Clase del objeto I4511. El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.

8.23.1 Detailed Description

Objeto de control del integrado [I4511](#).

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

8.24 I4511.h

[Go to the documentation of this file.](#)

```

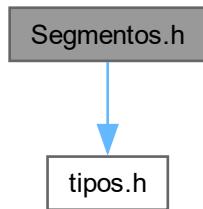
00001  /*****
00002   ****
00003   ****
00004   ****
00005   ****
00006   ****
00007   ****
00008   ****
00009   ****
00010   ****
00011   *** MODULO
00012   ****
00013 #ifndef I4511_H_
00014 #define I4511_H_
00015   ***
00016   ****
00017   *** INCLUDES GLOBALES
00018   ****
00019   *** INCLUDES GLOBALES
00020   ****
00021 #include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
00022 #include <Perifericos/02-Gpio/gpio.h>
00023 #include <vector>
00024
00025 using namespace std;
00026
00027
00028   ***
00029   *** DEFINES GLOBALES
00030   ****
00031
00032   ***
00033   *** MACROS GLOBALES
00034
00035
00036   ***
00037   *** TIPO DE DATOS GLOBALES
00038
00039
00040   ***
00041   *** VARIABLES GLOBALES
00042
00043
00044   ***
00045   *** IMPLANTACION DE UNA CLASE
00046
00047
00048 class I4511 : public segmentos
00049 {
00050     const vector<gpio * > & m_bcd ;
00051
00052     public:
00053         // tiene que recibir los gpio BCD
00054         I4511( const vector<gpio * > &bcd) : m_bcd (bcd){}
00055         void SetSegmentos ( uint16_t ) override;
00056         void Initialize ( void ) override;
00057         virtual ~I4511() {};
00058     };
00059
00060 #endif /* I4511_H_ */

```

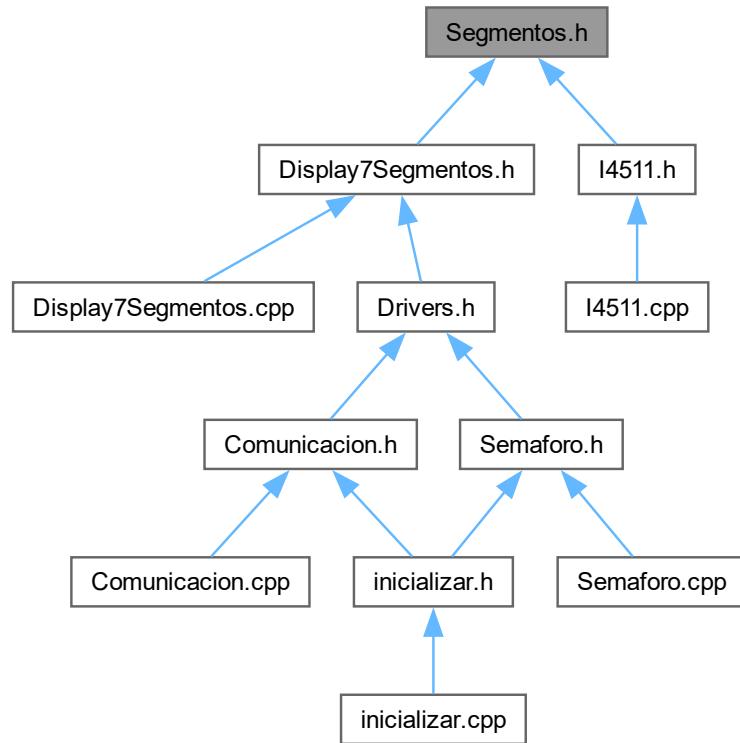
8.25 Segmentos.h File Reference

Clase abstracta de manejo de segmentos binarios.

```
#include "tipos.h"
Include dependency graph for Segmentos.h:
```



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



Classes

- class [segmentos](#)

Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.

8.25.1 Detailed Description

Clase abstracta de manejo de segmentos binarios.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

8.26 Segmentos.h

[Go to the documentation of this file.](#)

```

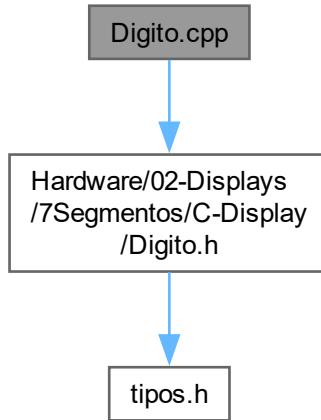
00001  /*****
00002   **** MODULO
00003   **** MACROS GLOBALES
00004   **** TIPO DE DATOS GLOBALES
00005   **** VARIABLES GLOBALES
00006   **** IMPLANTACION DE UNA CLASE
00007   ****
00008   ****
00009   ****
00010   ****
00011   *** MODULO
00012   ****
00013   ****
00014 #ifndef SEGMENTOS_H_
00015 #define SEGMENTOS_H_
00016
00017   ****
00018   *** INCLUDES GLOBALES
00019
00020   ****
00021   *** DEFINES GLOBALES
00022
00023   ****
00024   ****
00025   *** MACROS GLOBALES
00026
00027   ****
00028   ****
00029   *** TIPO DE DATOS GLOBALES
00030
00031   ****
00032   ****
00033   *** VARIABLES GLOBALES
00034
00035   ****
00036   ****
00037   *** IMPLANTACION DE UNA CLASE
00038
00039   ****
00040   ****
00041   ***
00042   ****
00043   class segmentos
00044 {
00045     public:
00046     segmentos() {};
00047     virtual void SetSegmentos ( uint16_t ) = 0;
00048     virtual void Initialize ( void ) = 0;
00049     virtual ~segmentos() {};
00050 };
00051
00052 #endif /* SEGMENTOS_H_ */

```

8.27 Dígito.cpp File Reference

Objeto dígito genérico para implementaciones posteriores.

```
#include <Hardware/02-Displays/7Segmentos/C-Display/Dígito.h>
Include dependency graph for Dígito.cpp:
```



Variables

- const `uint8_t Tabla_Digitos_BCD_7seg [] = {CERO, UNO, DOS, TRES, CUATRO, CINCO, SEIS, SIETE, OCHO, NUEVE}`

8.27.1 Detailed Description

Objeto dígito genérico para implementaciones posteriores.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

8.27.2 Variable Documentation

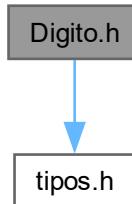
8.27.2.1 Tabla_Digitos_BCD_7seg

```
const uint8_t Tabla_Digitos_BCD_7seg [ ] = {CERO, UNO, DOS, TRES, CUATRO, CINCO, SEIS, SIETE, OCHO, NUEVE}
Tabla de dígitos BCD en binario
```

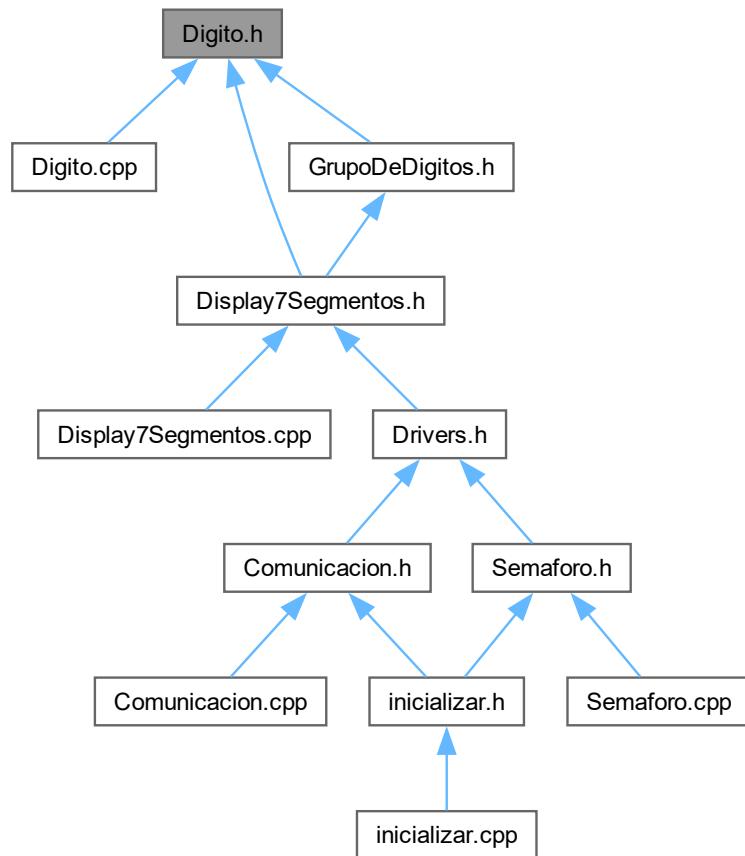
8.28 Dígito.h File Reference

Objeto dígito genérico para implementaciones posteriores.

```
#include "tipos.h"
Include dependency graph for Digito.h:
```



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



Classes

- class [digito](#)

Clase del objeto dígito El objeto dígito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.

8.28.1 Detailed Description

Objeto dígito genérico para implementaciones posteriores.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

8.29 Dígito.h

[Go to the documentation of this file.](#)

```

00001
00002  ****
00003
00004
00005  ***
00006  *** MODULO
00007
00008
00009
00010  ****
00011  *** INCLUDES GLOBALES
00012
00013 #ifndef DIGITO_H_
00014 #define DIGITO_H_
00015
00016  ****
00017  *** DEFINES GLOBALES
00018
00019
00020  ****
00021 #include "tipos.h"
00022
00023
00024  ***
00025  *** MACROS GLOBALES
00026
00027
00028  ***
00029
00030
00031
00032  ***
00033  *** TIPO DE DATOS GLOBALES
00034
00035
00036  ***
00037  *** VARIABLES GLOBALES
00038
00039
00040  ***
00041  *** IMPLANTACION DE UNA CLASE
00042
00043
00044
00045 class digito
00046 {
00047     public:
00048         typedef enum { BCD , SEGMENTOS , ASCHII } codigo_t;
00049         typedef enum { APAGAR = 0xff, PARPADEAR = 1} modo_t;
00050         typedef enum { menos = 10 , a , b , c , d , e , f , g , h , n , o , p , r , t , u } SIMBOLOS;
00051
00052     private:
00053         uint16_t m_Vvalor;
00054         const codigo_t m_Sistema;
00055
00056     public:
00057         digito( codigo_t Sistema = BCD , uint8_t Valor = APAGAR ) : m_Vvalor( Valor ) , m_Sistema (
00058             Sistema ) {};
00059
00060
00061
00062
00063
00064
00065

```

```

00066     bool Set(uint16_t valor);
00067     uint8_t Get( void );
00068     void Clear( void );
00069
00070     virtual ~digito() {};
00071 };
00072
00073 #endif /* DIGITO_H_ */

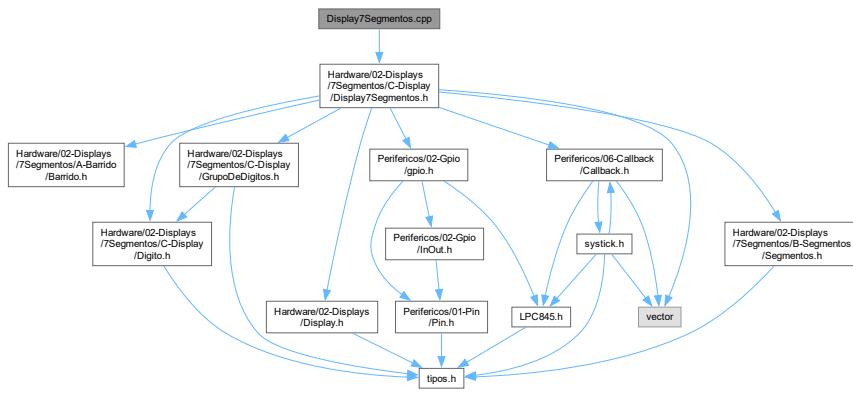
```

8.30 Display7Segmentos.cpp File Reference

Clase para la creación de displays de 7 segmentos.

```
#include <Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h>
```

Include dependency graph for Display7Segmentos.cpp:



8.30.1 Detailed Description

Clase para la creación de displays de 7 segmentos.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

8.31 Display7Segmentos.h File Reference

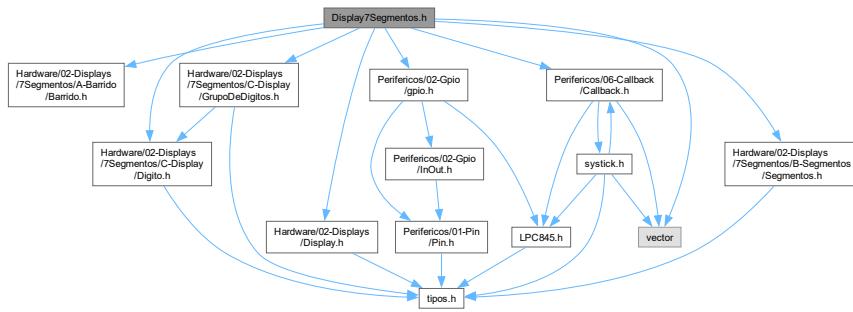
Clase para la creación de displays de 7 segmentos.

```

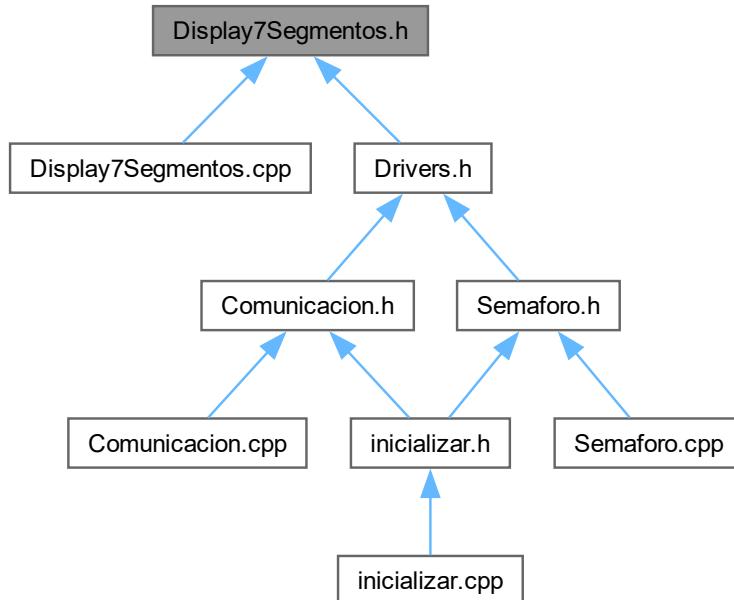
#include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
#include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
#include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
#include <Hardware/02-Displays/7Segmentos/C-Display/GrupoDeDigitos.h>
#include <Hardware/02-Displays/Display.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
#include <vector>

```

Include dependency graph for Display7Segmentos.h:



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



Classes

- class [display7Segmentos](#)

Clase del objeto display7Segmentos El objeto [display7Segmentos](#) permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a `m_ticks`. Para ver mejores resultados modificar dicho valor.

Macros

- `#define UPDATE_TICKS (3)`

8.31.1 Detailed Description

Clase para la creación de displays de 7 segmentos.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

8.31.2 Macro Definition Documentation

8.31.2.1 UPDATE_TICKS

```
#define UPDATE_TICKS (3)
Velocidad de escritura de los leds. MODIFICAR EN CASO DE SER NECESARIO
```

8.32 Display7Segmentos.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002 ****
00003 ****
00004 ****
00005 ****
00006 ****
00007 ****
00008 ****
00009 ****
00010 ****
00011 *** MODULO
00012 ****
00013 ****
00014 #ifndef DISPLAY7SEGMENTOS_H_
00015 #define DISPLAY7SEGMENTOS_H_
00016 ****
00017 *** INCLUDES GLOBALES
00018 ****
00019 ****
00020 #include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
00021 #include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
00022 #include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
00023 #include <Hardware/02-Displays/7Segmentos/C-Display/GrupoDeDigitos.h>
00024 #include <Hardware/02-Displays/Display.h>
00025 ****
00026 ****
00027 ****
00028 #include <Perifericos/02-Gpio/gpio.h>
00029 #include <Perifericos/06-Callback/Callback.h>
00030 ****
00031 #include <vector>
00032 ****
00033 ****
00034 *** DEFINES GLOBALES
00035 ****
00036 ****
00037 ****
00038 *** MACROS GLOBALES
00039 ****
00040 ****
00041 ****
00042 *** TIPO DE DATOS GLOBALES
00043 ****
00044 ****
00045 ****
00046 *** VARIABLES GLOBALES
00047 ****
00048 ****
00049 ****
00050 ****
00051 *** IMPLANTACION DE UNA CLASE
```

```

00052 ****
00053
00054
00055 // Esta funcion posee:
00056 // un barrido,
00057 // un segmentos,
00058 // 6 digitos,
00059 // configuracion,
00060 // heredar swhandler.
00068 class display7Segmentos : public Display, Callback
00069 {
00070     private:
00072         #define UPDATE_TICKS (3)
00073     private:
00074         vector < gruposdedigitos* > m_grupos;
00075             segmentos           *m_seg ;
00076             barrido            *m_dig ;
00077             uint8_t              m_maxdigitos;
00078             uint8_t              m_inx;
00079             uint8_t              m_ticks;
00080
00081         vector < digito *>      m_bufferdisplay ;
00082         const uint8_t*          m_PosicionRelativa;
00083         const digito::codigo_t  m_sistema;
00084
00085     public:
00086         display7Segmentos( vector < gruposdedigitos * > g ,
00087                             segmentos * s , barrido * b ,
00088                             const uint8_t *PosicionRelativa ,
00089                             const digito::codigo_t sistema);
00090         void    SWhandler ( void ) override;
00091         void    Set( uint32_t valor , uint8_t dsp );
00092         void    Write ( const int32_t n ) override;
00093         void    Clear ( void ) override;
00094
00095     virtual ~display7Segmentos();
00096 };
00097
00098 #endif /* DISPLAY7SEGMENTOS_H_ */

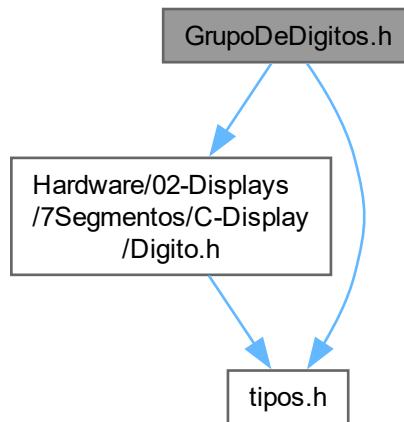
```

8.33 GrupoDeDigitos.h File Reference

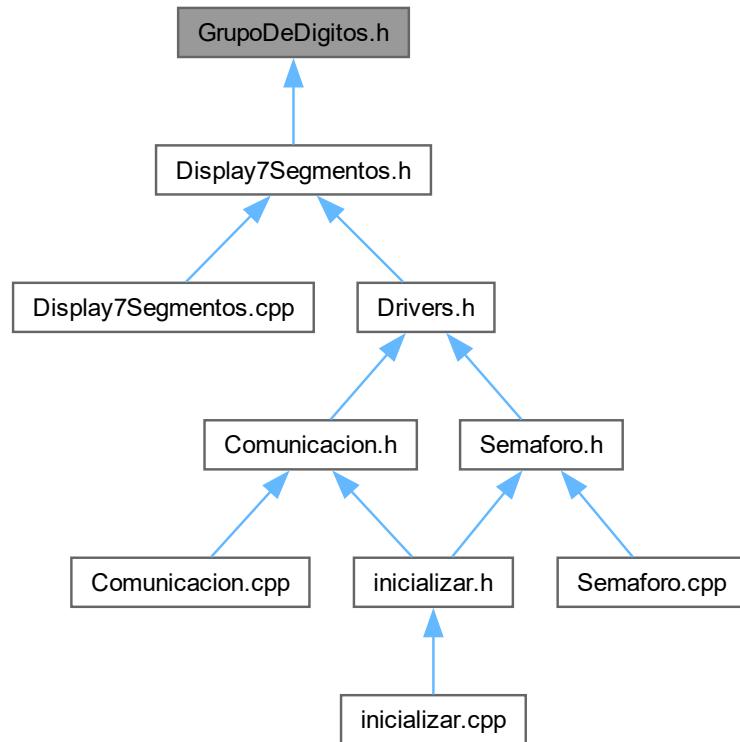
Clase para la agrupación de dígitos.

```
#include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
#include "tipos.h"
```

Include dependency graph for GrupoDeDigitos.h:



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



Classes

- struct **gruposdedigitos**
Estructura de grupo de dígitos.

8.33.1 Detailed Description

Clase para la agrupación de dígitos.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

8.34 GrupoDeDigitos.h

[Go to the documentation of this file.](#)

```
00001 /*****
00009
00010 ****
00011 *** MODULO
00012
```

```

00013 #ifndef GRUPOSDEDIGITOS_H_
00014 #define GRUPOSDEDIGITOS_H_
00018
00019 /* *** INCLUDES GLOBALES
00020
00021 #include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
00022 #include "tipos.h"
00023
00024
00025 /* *** DEFINES GLOBALES
00026
00027
00028
00029 /* *** MACROS GLOBALES
00030
00031
00032
00033 /* *** TIPO DE DATOS GLOBALES
00034
00035
00036
00037 /* *** VARIABLES GLOBALES
00038
00039
00040
00041 /* *** IMPLANTACION DE UNA CLASE
00042
00043
00044 struct gruposdedigitos
00045 {
00046     const uint8_t          m_comienzo;
00047     const uint8_t          m_cantidad;
00048
00049     public:
00050         gruposdedigitos(uint8_t    comienzo , uint8_t    cantidad ) :
00051             m_comienzo(comienzo) , m_cantidad (cantidad) {}
00052         virtual ~gruposdedigitos() {};
00053     };
00054
00055 #endif /* GRUPOSDEDIGITOS_H_ */

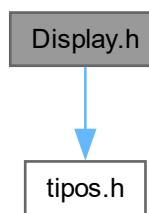
```

8.35 Display.h File Reference

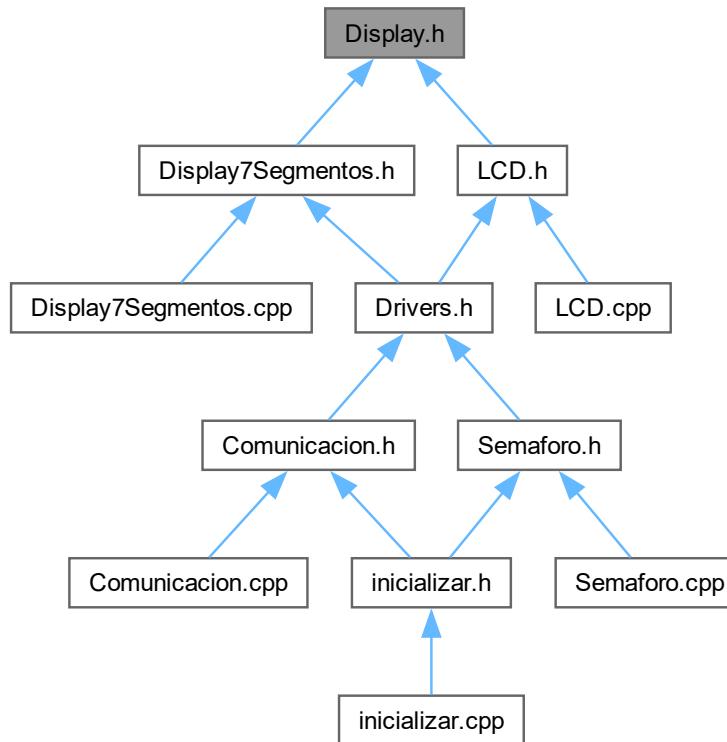
Clase base para objetos del tipo pantallas/displays.

#include "tipos.h"

Include dependency graph for Display.h:



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



Classes

- class [Display](#)

Clase del objeto display Clase abstracta pura para la generación de displays.

8.35.1 Detailed Description

Clase base para objetos del tipo pantallas/displays.

Proporciona métodos para inicializar y enviar datos al [LCD](#). Diseñada para facilitar la integración en proyectos embebidos.

Date

22 jun. 2022

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.36 Display.h

[Go to the documentation of this file.](#)

```

00001 /*****
00011 ****
00012 *** MODULO
  
```

```

00013
00014 #ifndef DISPLAY_H_
00015 #define DISPLAY_H_
00019
00020 *** INCLUDES GLOBALES
00021
00022 #include "tipos.h"
00023
00024
00025 *** IMPLANTACION DE LA CLASE
00026
00027
00028
00029
00030
00031
00032 class Display
00033 {
00034     public:
00035         Display() = default;
00036         virtual void Write ( const int32_t n ) = 0;
00037         virtual void Clear ( void ) = 0;
00038         virtual ~Display() = default;
00039 };
00040
00041 #endif /* DISPLAY_H_ */

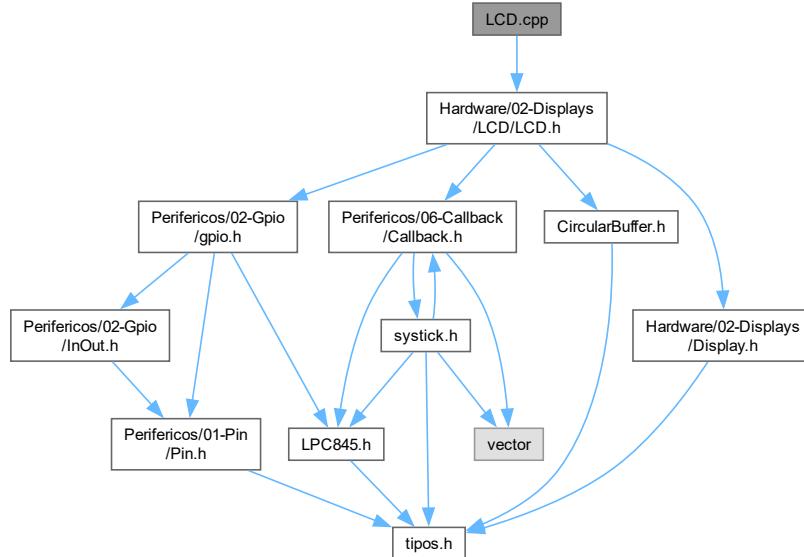
```

8.37 LCD.cpp File Reference

Clase para manejar un [LCD](#) con comunicación de 4 patas (solo escritura).

#include <Hardware/02-Displays/LCD/LCD.h>

Include dependency graph for LCD.cpp:



8.37.1 Detailed Description

Clase para manejar un [LCD](#) con comunicación de 4 patas (solo escritura).

Proporciona métodos para inicializar y enviar datos al [LCD](#). Diseñada para facilitar la integración en proyectos embebidos.

Date

22 jun. 2022

Author

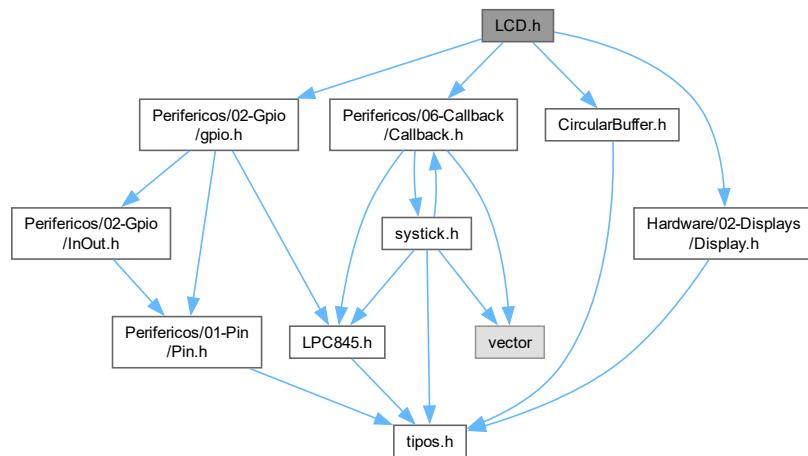
Técnico. Martinez Agustín (masteragus365@gmail.com)

8.38 LCD.h File Reference

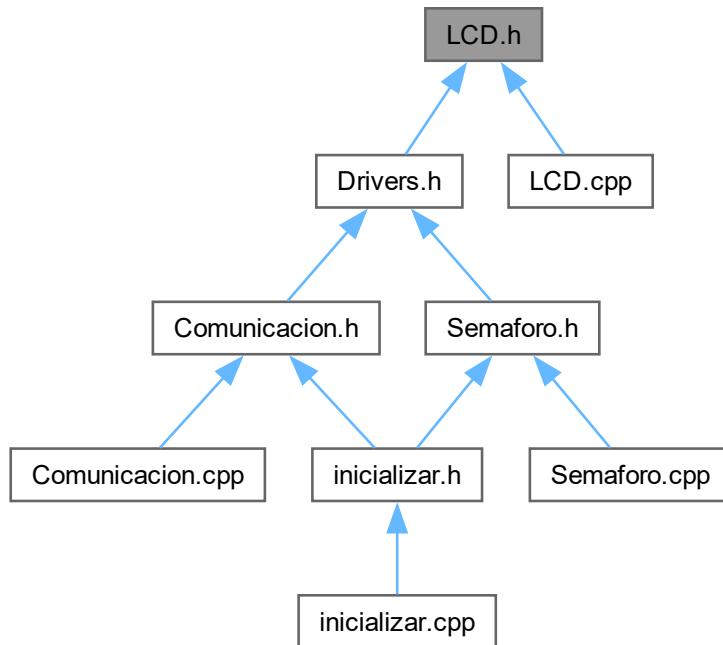
Clase para manejar un [LCD](#) con comunicación de 4 patas (solo escritura).

```
#include <Hardware/02-Displays/Display.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
#include <CircularBuffer.h>
```

Include dependency graph for LCD.h:



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



Classes

- class [LCD](#)

Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

Macros

- `#define MAX_PIN_COUNT 6`
- `#define INSTRUCTION_8BITS 0b00000010`
- `#define INSTRUCTION_FUNC_SET 0b00100000 /* 1 DL N F - -*/`
- `#define INSTRUCTION_DISP_CTRL 0b00001000 /* 1 D C B*/`
- `#define INSTRUCTION_ENTRY_MODE_SET 0b00000100 /* 1 I/D S*/`
- `#define INSTRUCTION_CLEAR_DISP 0b00000001`
- `#define INSTRUCTION_SET_POS 0b10000000 /* 1 ADD ADD ADD ADD ADD ADD ADD*/`
- `#define ROW_OFFSET 0x40`

8.38.1 Detailed Description

Clase para manejar un [LCD](#) con comunicación de 4 patas (solo escritura).

Proporciona métodos para inicializar y enviar datos al [LCD](#). Diseñada para facilitar la integración en proyectos embebidos. El modulo evita utilizar librerías standar de C++ (como vector o string) para disminuir su espacio en memoria del embebido. La comunicación es de 4 bits en paralelo, sin blink ni cursor, ni shifteo.

Date

11 ene. 2025

Version

2.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.38.2 Macro Definition Documentation

8.38.2.1 INSTRUCTION_8BITS

```
#define INSTRUCTION_8BITS 0b00000010
Instrucción de 8 bits inicial para pasar al modo 4 bits
```

8.38.2.2 INSTRUCTION_CLEAR_DISP

```
#define INSTRUCTION_CLEAR_DISP 0b00000001
Instrucción de limpieza del display.
```

8.38.2.3 INSTRUCTION_DISP_CTRL

```
#define INSTRUCTION_DISP_CTRL 0b00001000 /** 1 D C B*/
Instrucción de control del display. 1 seguido de D (display On), C (cursor On), B(blink).
```

8.38.2.4 INSTRUCTION_ENTRY_MODE_SET

```
#define INSTRUCTION_ENTRY_MODE_SET 0b00000100 /** 1 I/D S*/
Instrucción de entrada de escritura del display. 1, seguido de I/D (inc. puntero) y S (shift display)
```

8.38.2.5 INSTRUCTION_FUNC_SET

```
#define INSTRUCTION_FUNC_SET 0b00100000 /** 1 DL N F - -*/
Instrucción de modo de funcionamiento del display. 1 seguido de DL (4 bits) , N (filas) , F (resolución).
```

8.38.2.6 INSTRUCTION_SET_POS

```
#define INSTRUCTION_SET_POS 0b10000000 /** 1 ADD ADD ADD ADD ADD ADD*/
Instrucción de seteo de DDRAM del display. Cambia la posición del display
```

8.38.2.7 MAX_PIN_COUNT

```
#define MAX_PIN_COUNT 6
Cantidad de pines del LCD.
```

8.38.2.8 ROW_OFFSET

```
#define ROW_OFFSET 0x40
Offset de instrucción para indicar un cambio de renglón
```

8.39 LCD.h

[Go to the documentation of this file.](#)

```
00001 /*****
00013 ****
00014 *** MODULO
00015 ****
00016 ****
00017 #ifndef LCD_H_
00018 #define LCD_H_
```

```

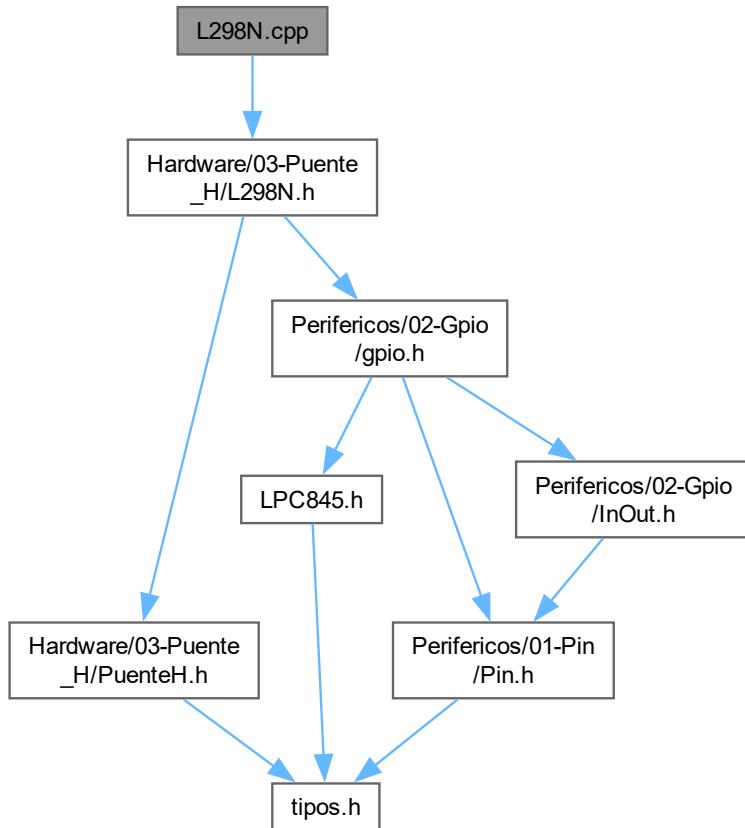
00022
00023     /***** INCLUDES GLOBALES *****/
00024
00025 #include <Hardware/02-Displays/Display.h>
00026 #include <Perifericos/02-Gpio/gpio.h>
00027 #include <Perifericos/06-Callback/Callback.h>
00028 #include <CircularBuffer.h>
00029
00030     /***** DEFINES GLOBALES *****/
00031
00032
00033     /***** MACROS GLOBALES *****/
00034 #define MAX_PIN_COUNT 6
00035
00036
00037     /***** TIPO DE DATOS GLOBALES *****/
00038
00039
00040
00041     /***** VARIABLES GLOBALES *****/
00042
00043
00044
00045     /***** IMPLANTACION DE LA CLASE *****/
00046
00047
00048 class LCD : public Display, Callback
00049 {
00050     public:
00051         enum { D4 = 0 , D5 , D6 , D7 , RS , ENA };
00052         #define MAX_PIN_COUNT 6
00053
00054     private:
00055         typedef enum {instruction , data } msj_type_t;
00056         #define INSTRUCTION_8BITS          0b00000010
00057         #define INSTRUCTION_FUNC_SET       0b00100000
00058         #define INSTRUCTION_DISP_CTRL      0b00001000
00059         #define INSTRUCTION_ENTRY_MODE_SET 0b00000100
00060         #define INSTRUCTION_CLEAR_DISP    0b00000001
00061         #define INSTRUCTION_SET_POS       0b10000000
00062         #define ROW_OFFSET                0x40
00063
00064     private:
00065         gpio*           m_salidas[MAX_PIN_COUNT];
00066         CircularBuffer<uint8_t> m_buffer;
00067         uint8_t          m_Ymax , m_Ypos;
00068         uint8_t          m_Xmax , m_Xpos;
00069         uint32_t         m_ticks;
00070
00071     public:
00072         LCD( gpio* salidas[MAX_PIN_COUNT] , const uint8_t filas , const uint8_t columnas );
00073         void Initialize( void );
00074         void Write ( const char *s );
00075         void Write ( const int32_t n );
00076         LCD& operator= ( const char *s );
00077         void WriteAt( const int8_t *a , const uint8_t columna, const uint8_t fila );
00078         void WriteAt ( const int32_t n , const uint8_t columna, const uint8_t fila );
00079         void Clear( void ) override;
00080         virtual ~LCD();
00081
00082     protected:
00083         void SHandler ( void ) override;
00084     private:
00085         void doublePush( uint8_t a , msj_type_t type );
00086         void intToStr(int32_t num, int8_t* str);
00087
00088 };
00089
00090
00091 #endif /* LCD_H_ */

```

8.40 L298N.cpp File Reference

Clase del módulo de puente H LN298N.

```
#include <Hardware/03-Puente_H/L298N.h>
Include dependency graph for L298N.cpp:
```



8.40.1 Detailed Description

Clase del módulo de puente H LN298N.

Proporciona métodos para inicializar y enviar datos al puente H LN298N.

Date

25 sep. 2022

Version

1.0

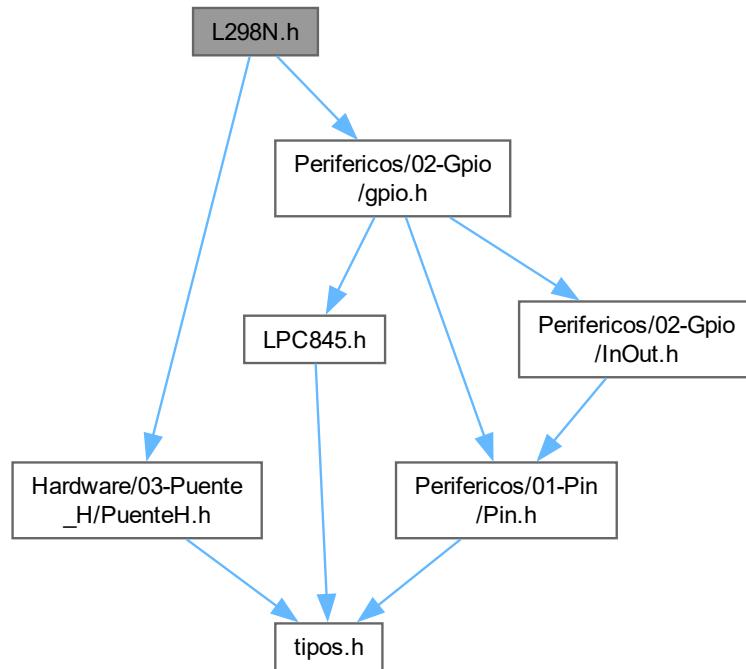
Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

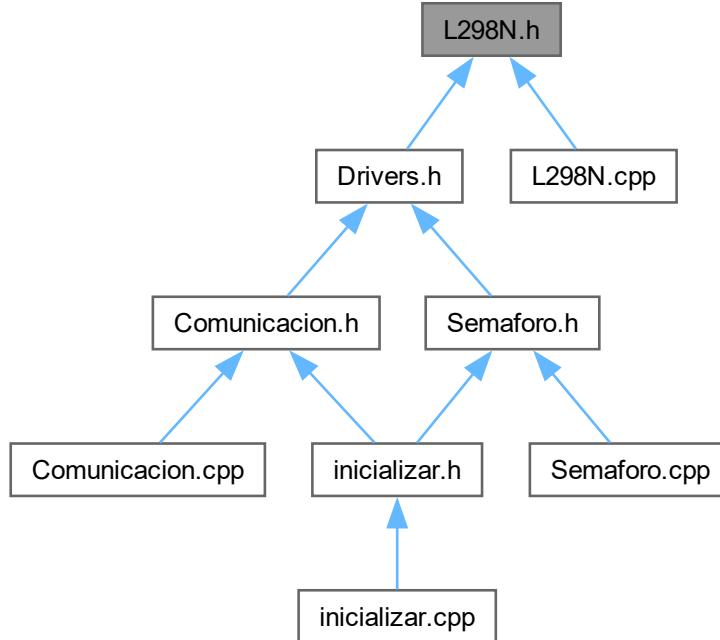
8.41 L298N.h File Reference

Clase del módulo de puente H LN298N.

```
#include <Hardware/03-Puente_H/PuenteH.h>
#include <Perifericos/02-Gpio/gpio.h>
Include dependency graph for L298N.h:
```



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



Clases

- class [L298N](#)

Clase del objeto L298N El objeto L298N realiza las acciones de control de dos motores controlados por el correspondiente periférico.

8.41.1 Detailed Description

Clase del módulo de puente H LN298N.

Proporciona métodos para inicializar y enviar datos al puente H LN298N.

Date

25 sep. 2022

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.42 L298N.h

[Go to the documentation of this file.](#)

00001

/*****

00012

```

00013 #ifndef L298N_H_
00014 #define L298N_H_
00018
00019 /* INCLUDES GLOBALES
00020
00021 #include <Hardware/03-Puente_H/PuenteH.h>
00022 #include <Perifericos/02-Gpio/gpio.h>
00023
00024 *** DEFINES GLOBALES
00025
00026
00027
00028 *** MACROS GLOBALES
00029
00030
00031
00032 *** TIPO DE DATOS GLOBALES
00033
00034
00035
00036 *** IMPLANTACION DE LA CLASE
00037
00038
00039 class L298N : protected Puente_H
00040 {
00041     private:
00042         gpio* &m_motor1_a;
00043         gpio* &m_motor1_b;
00044         gpio* &m_motor2_a;
00045         gpio* &m_motor2_b;
00046     public:
00047         L298N( gpio* &_motorDer_a , gpio* &_motorDer_b , gpio* &_motorIzq_a , gpio*
00048             &_motorIzq_b );
00049         void Initialize( void ) override;
00050         void GirarIzq ( void ) override;
00051         void GirarDer ( void ) override;
00052         void Girar ( const uint8_t direccion ) override;
00053         void Frenar ( void ) override;
00054         void Avanzar ( void ) override;
00055         void Retroceder ( void ) override;
00056         virtual ~L298N();
00057
00058 };
00059
00060 #endif /* L298N_H_ */

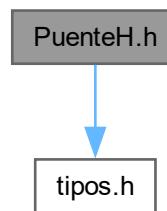
```

8.43 PuenteH.h File Reference

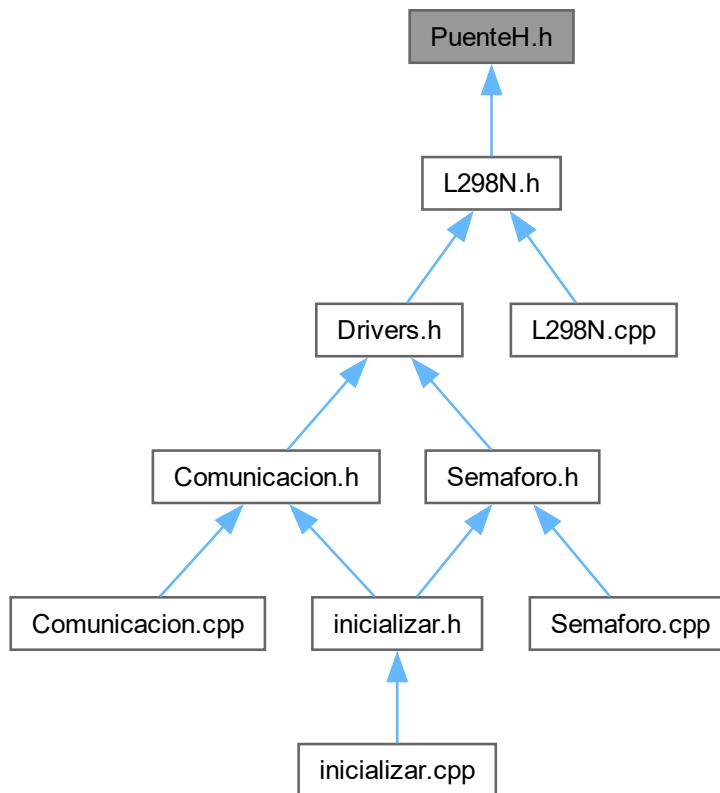
Clase base para objetos del tipo Puente H.

```
#include "tipos.h"
```

Include dependency graph for PuenteH.h:



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



Classes

- class `Puente_H`

Clase del objeto `Puente_H` El objeto `Puente_H` es la interfaz abstracta pura de cualquier puente H que se deseé realizar.

8.43.1 Detailed Description

Clase base para objetos del tipo Puente H.

Date

25 sep. 2022

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.44 PuenteH.h

[Go to the documentation of this file.](#)

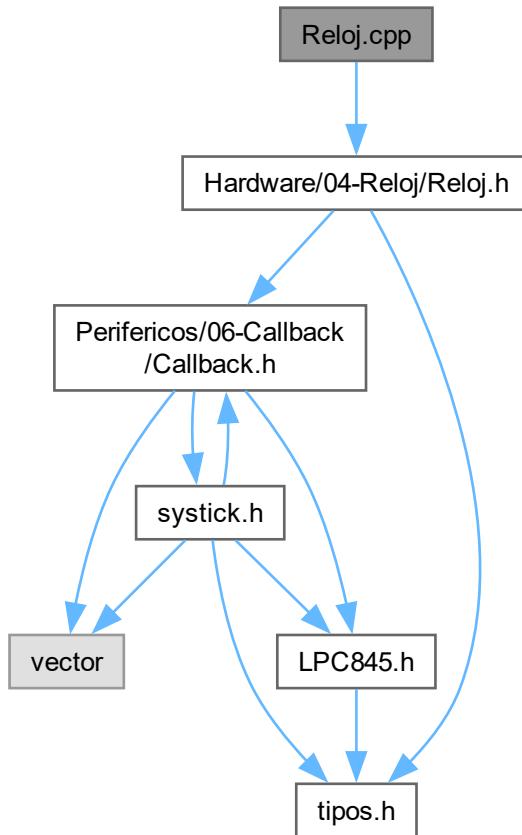
00001

```
00012 /*****  
00013 #ifndef PUENTEH_H_  
00014 #define PUENTEH_H_  
00015  
00016 /*****  
00017 *** INCLUDES GLOBALES  
00018  
00019 /*****  
00020 #include "tipos.h"  
00021  
00022 /*****  
00023 *** DEFINES GLOBALES  
00024  
00025 /*****  
00026 *** MACROS GLOBALES  
00027  
00028 /*****  
00029 *** TIPO DE DATOS GLOBALES  
00030  
00031  
00032 /*****  
00033 *** IMPLANTACION DE LA CLASE  
00034  
00035 /*****  
00036 class Puente_H  
00037 {  
00038     public:  
00039         enum { IZQUIERDA = 0 , DERECHA};  
00040     public:  
00041         Puente_H( ) = default;  
00042         virtual void Initialize(void) = 0;  
00043         virtual void GirarIzq(void) = 0;  
00044         virtual void GirarDer(void) = 0;  
00045         virtual void Girar ( uint8_t direccion ) = 0;  
00046         virtual void Frenar ( void ) = 0;  
00047         virtual void Avanzar( void ) = 0;  
00048         virtual void Retroceder( void ) = 0;  
00049         virtual ~Puente_H() = default;  
00050 };  
00051  
00052  
00053  
00054  
00055 };  
00056  
00057 #endif /* PUENTEH_H_ */
```

8.45 Reloj.cpp File Reference

Objeto que guardará el tiempo desde que se creo.

```
#include <Hardware/04-Reloj/Reloj.h>
Include dependency graph for Reloj.cpp:
```



8.45.1 Detailed Description

Objeto que guardará el tiempo desde que se creo.
Proporciona métodos para inicializar un reloj.

Date

27 nov. 2022

Version

1.0

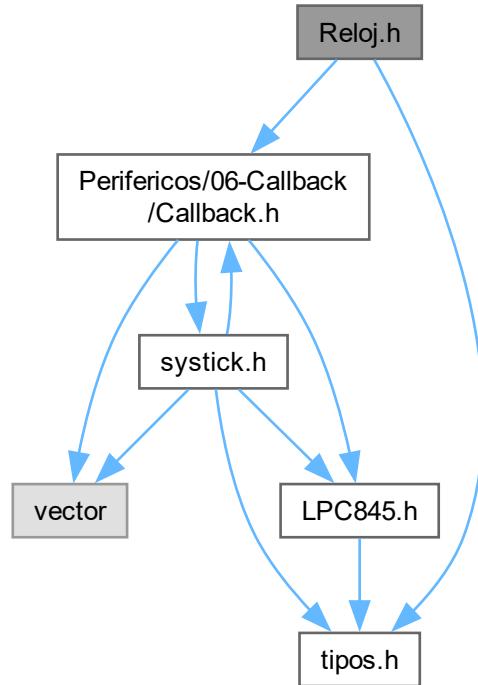
Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

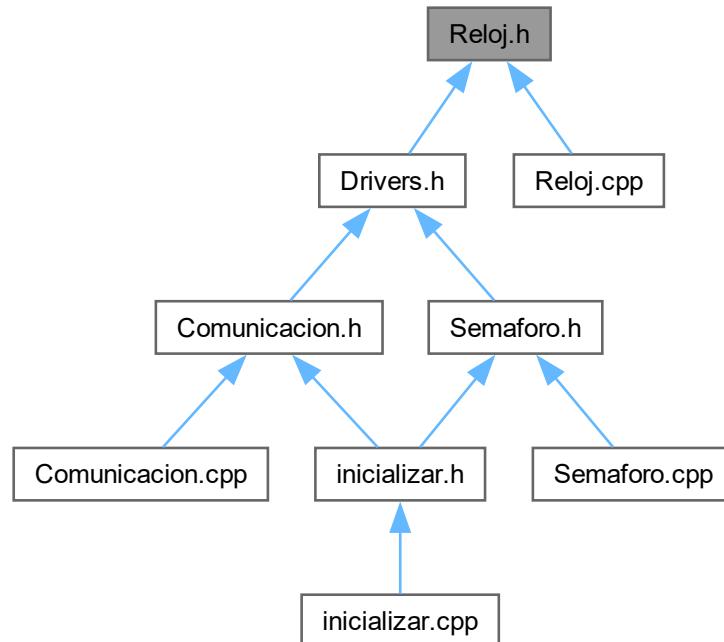
8.46 Reloj.h File Reference

Objeto que guardará el tiempo desde que se creo.

```
#include <Perifericos/06-Callback/Callback.h>
#include "tipos.h"
Include dependency graph for Reloj.h:
```



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



Classes

- class [Reloj](#)

Clase del objeto Reloj.

8.46.1 Detailed Description

Objeto que guardará el tiempo desde que se creo.
Proporciona métodos para inicializar un reloj.

Date

27 nov. 2022

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.47 Reloj.h

[Go to the documentation of this file.](#)

00001

```
 /*****
00012 ****/
00013 ****/
```

```

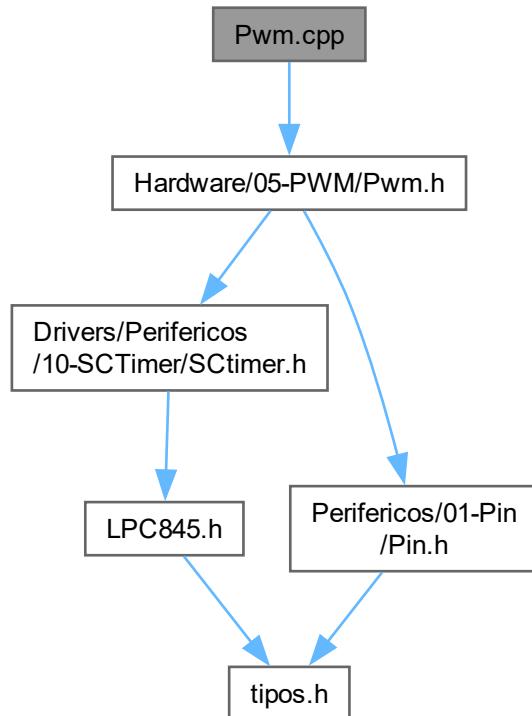
00014 *** MODULO
00015
00016 #ifndef RELOJ_H_
00017 #define RELOJ_H_
00021
00022 *** INCLUDES GLOBALES
00023
00024 #include <Perifericos/06-Callback/Callback.h>
00025 #include "tipos.h"
00026
00027 *** DEFINES GLOBALES
00028
00029
00030
00031 *** MACROS GLOBALES
00032
00033
00034
00035 *** TIPO DE DATOS GLOBALES
00036
00037
00038
00039 *** VARIABLES GLOBALES
00040
00041
00042
00043 *** IMPLANTACION DE LA CLASE
00044
00045
00046 class Reloj : public Callback
00047 {
00048     private:
00049         int32_t      m_hora;
00050         int32_t      m_minutos;
00051         int32_t      m_segundos;
00052         uint32_t     m_cont;
00053
00054     public:
00055         Reloj();
00056         int32_t GetHour( void ) const;
00057         int32_t GetMin( void ) const;
00058         int32_t GetSeg( void ) const;
00059         void    Reset( void );
00060         void    SetTime( const int32_t _hour , const int32_t _min = -1 , const int32_t _seg = -1);
00061
00062         virtual ~Reloj();
00063
00064     private:
00065         void    Update( void );
00066
00067     protected:
00068         void    SWhandler( void ) override;
00069 };
00070
00071
00072
00073
00074
00075
00076 #endif /* RELOJ_H_ */

```

8.48 Pwm.cpp File Reference

Generador de PWM sin interrupción.

```
#include <Hardware/05-PWM/Pwm.h>
Include dependency graph for Pwm.cpp:
```



8.48.1 Detailed Description

Generador de PWM sin interrupción.
Proporciona métodos para inicializar un PWM.

Date

7 oct. 2022

Version

1.0

Author

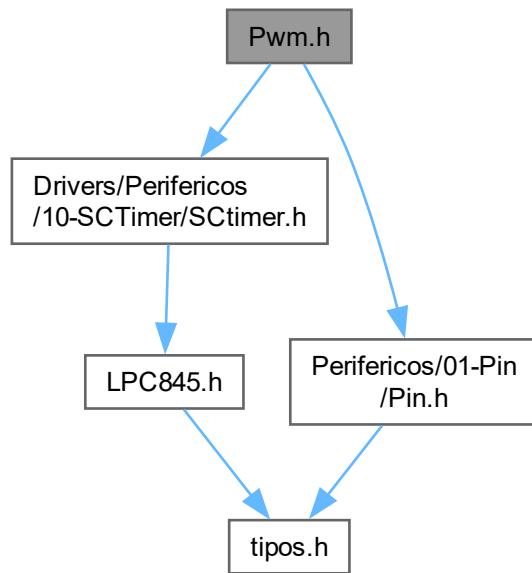
Técnico. Martinez Agustín (masteragus365@gmail.com)

8.49 Pwm.h File Reference

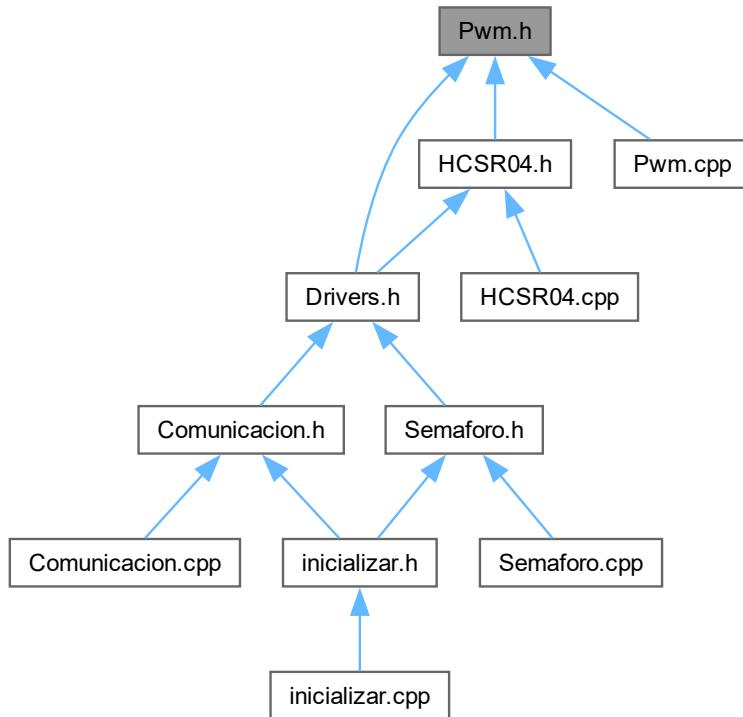
Generador de PWM sin interrupción.

```
#include <Drivers/Perifericos/10-SCTimer/SCTimer.h>
#include <Perifericos/01-Pin/Pin.h>
```

Include dependency graph for Pwm.h:



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



Classes

- class [Pwm](#)

Clase del objeto Pwm.

8.49.1 Detailed Description

Generador de PWM sin interrupción.
Proporciona métodos para inicializar un PWM.

Date

7 oct. 2022

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.50 Pwm.h

[Go to the documentation of this file.](#)

00001

/*****

```

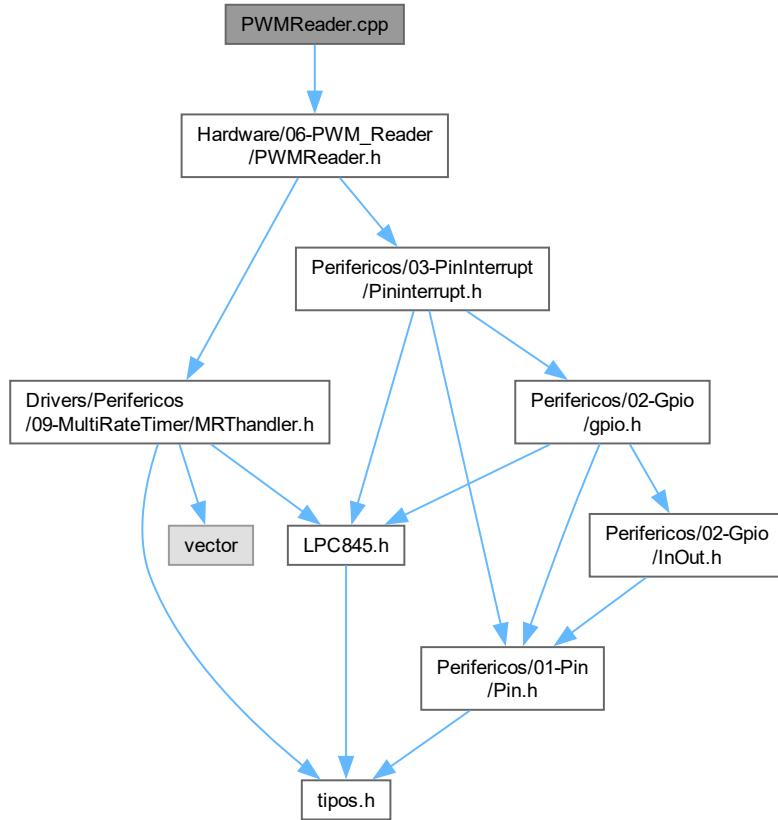
00012
00013 #ifndef PWM_H_
00014 #define PWM_H_
00018
00019 /* INCLUDES GLOBALES
00020
00021 #include <Drivers/Perifericos/10-SCTimer/SCtimer.h>
00022 #include <Perifericos/01-Pin/Pin.h>
00023
00024 /* DEFINES GLOBALES
00025
00026
00027
00028 /* MACROS GLOBALES
00029
00030
00031
00032 /* TIPO DE DATOS GLOBALES
00033
00034
00035
00036 /* IMPLANTACION DE LA CLASE
00037
00038
00039 class Pwm : protected SCtimer , protected Pin
00040 {
00041 public:
00042     typedef enum { SEG = 0, MILI_SEG , MICRO_SEG } pwm_time_unit_t;
00043     typedef enum { CHANNEL_1 = 1 , CHANNEL_2 , CHANNEL_3 , CHANNEL_4 , CHANNEL_5 , CHANNEL_6 }
00044     pwm_channel_t;
00045     enum activity_t      { low , high };
00046
00047 protected:
00048     const uint8_t      m_activity;
00049     uint32_t          m_ton;
00050     uint32_t          m_toff;
00051     const uint8_t      m_pwm_channel;
00052
00053 public:
00054     Pwm( port_t puerto , uint8_t bit , uint8_t actividad , pwm_channel_t number );
00055     void Initialize( uint32_t ton , uint32_t toff , pwm_time_unit_t t = MICRO_SEG );
00056     void SetTon( uint32_t time , pwm_time_unit_t t = MICRO_SEG );
00057     void SetPeriod( uint32_t time , pwm_time_unit_t t = MICRO_SEG );
00058     void On();
00059     void Off();
00060     virtual ~Pwm();
00061
00062 };
00063
00064 #endif /* PWM_H_ */

```

8.51 PWMReader.cpp File Reference

Pata que lee tamaños de pulsos de entrada.

```
#include <Hardware/06-PWM_Reader/PWMReader.h>
Include dependency graph for PWMReader.cpp:
```



8.51.1 Detailed Description

Pata que lee tamaños de pulsos de entrada.

Date

22 jun. 2022

Version

1.0

Author

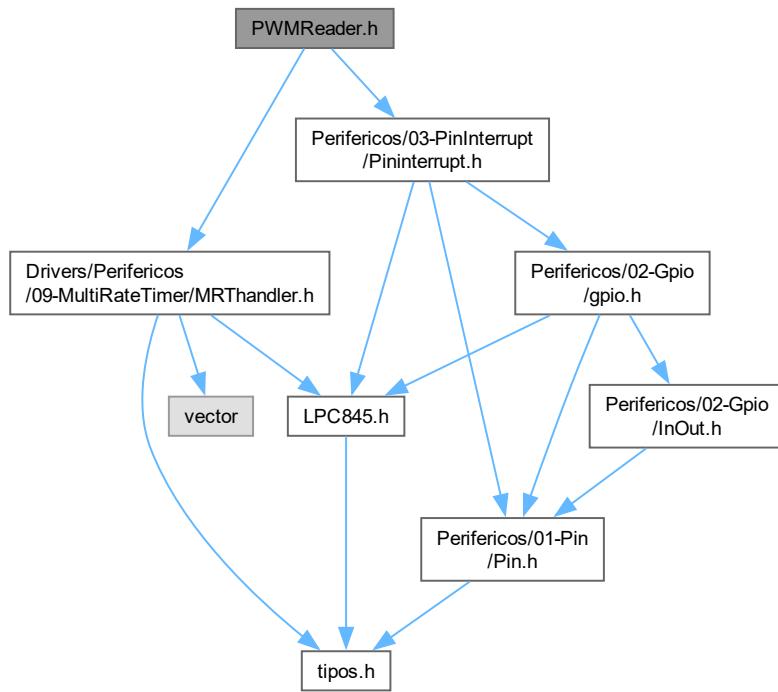
Técnico. Martínez Agustín (masteragus365@gmail.com)

8.52 PWMReader.h File Reference

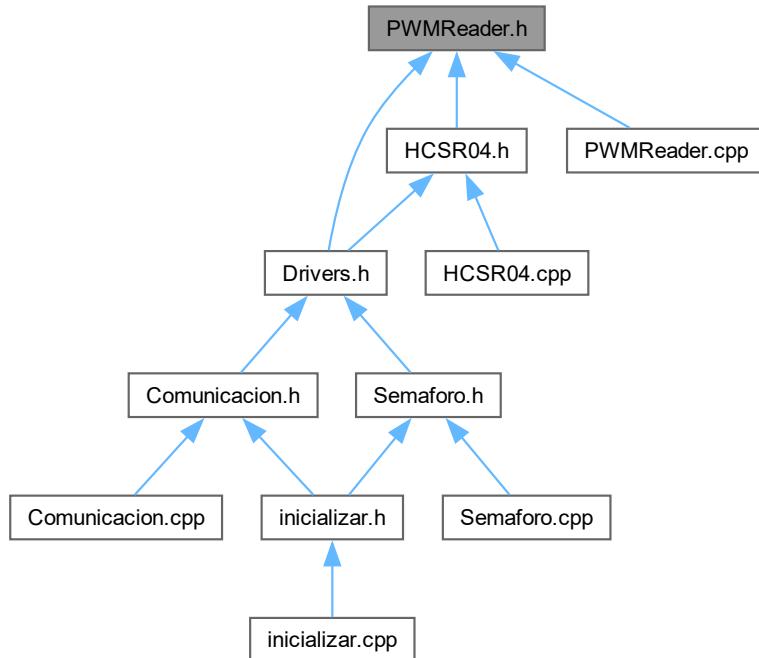
Pata que lee tamaños de pulsos de entrada.

```
#include <Drivers/Perifericos/09-MultiRateTimer/MRThandler.h>
#include <Perifericos/03-PinInterrupt/Pininterrupt.h>
```

Include dependency graph for PWMReader.h:



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



Classes

- class [PWM_Reader](#)
Clase del objeto PWM_Reader.

8.52.1 Detailed Description

Pata que lee tamaños de pulsos de entrada.

Date

22 jun. 2022

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.53 PWMReader.h

[Go to the documentation of this file.](#)

```

00001 /*****
00012 ****/
00013 *** MODULO
00014 *****/

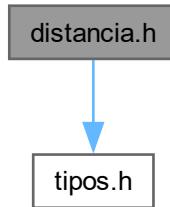
```

```
00015 #ifndef PWM_READER_H_
00016 #define PWM_READER_H_
00020
00021 /*****
00021 *** INCLUDES GLOBALES
00022 ****
00023 #include <Drivers/Perifericos/09-MultiRateTimer/MRThandler.h>
00024 #include <Perifericos/03-PinInterrupt/Pininterrupt.h>
00025
00026 /*****
00026 *** DEFINES GLOBALES
00027 ****
00028
00029
00030 /*****
00030 *** MACROS GLOBALES
00031 ****
00032
00033
00034 /*****
00034 *** TIPO DE DATOS GLOBALES
00035 ****
00036
00037
00038 /*****
00038 *** VARIABLES GLOBALES
00039 ****
00040
00041
00042 /*****
00042 *** IMPLANTACION DE LA CLASE
00043 ****
00058 class PWM_Reader : protected PinInterrupt , protected MRThandler
00059 {
00060     private:
00061         uint32_t m_pulse_on;
00062
00063     public:
00064         PWM_Reader( port_t puerto , uint8_t bit , mode_t modo ,
00065                     activity_t activity , MRT_timer_channels timer_channel );
00066         void Initialize( void );
00067         uint32_t GetPulseOn( void ) const;
00068         void Off( void );
00069         void On( void );
00070         virtual ~PWM_Reader() {};
00071     protected:
00072         void GpioHandler( void ) override;
00073 };
00074
00075 #endif /* PWM_READER_H_ */
```

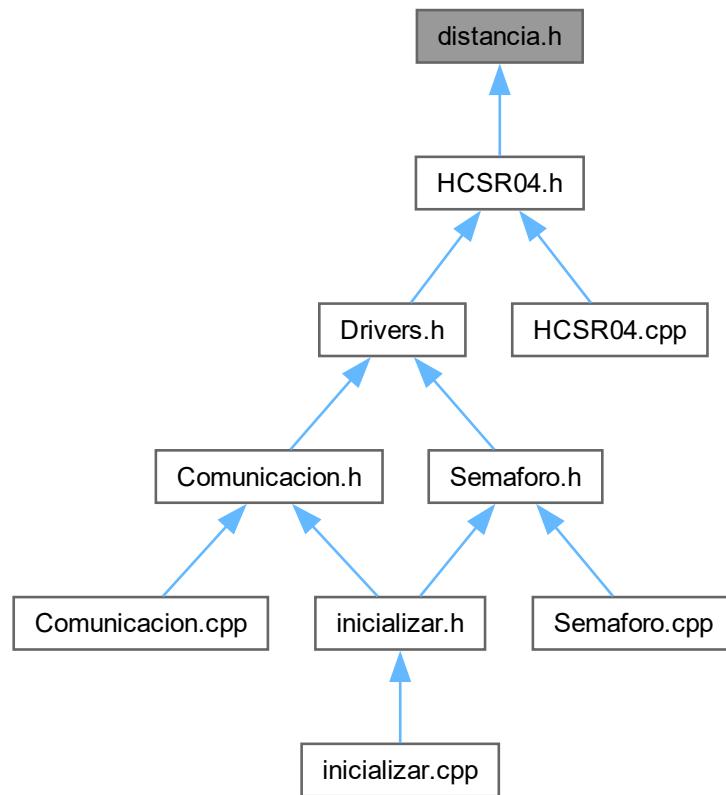
8.54 distancia.h File Reference

Clase base para objetos medidores de distancias.

```
#include "tipos.h"
Include dependency graph for distancia.h:
```



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



Classes

- class [distancia](#)

Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.

8.54.1 Detailed Description

Clase base para objetos medidores de distancias.

Date

22 jun. 2022

Version

1.0

Author

Técnico. Martinez Agustin (masteragus365@gmail.com)

8.55 distancia.h

[Go to the documentation of this file.](#)

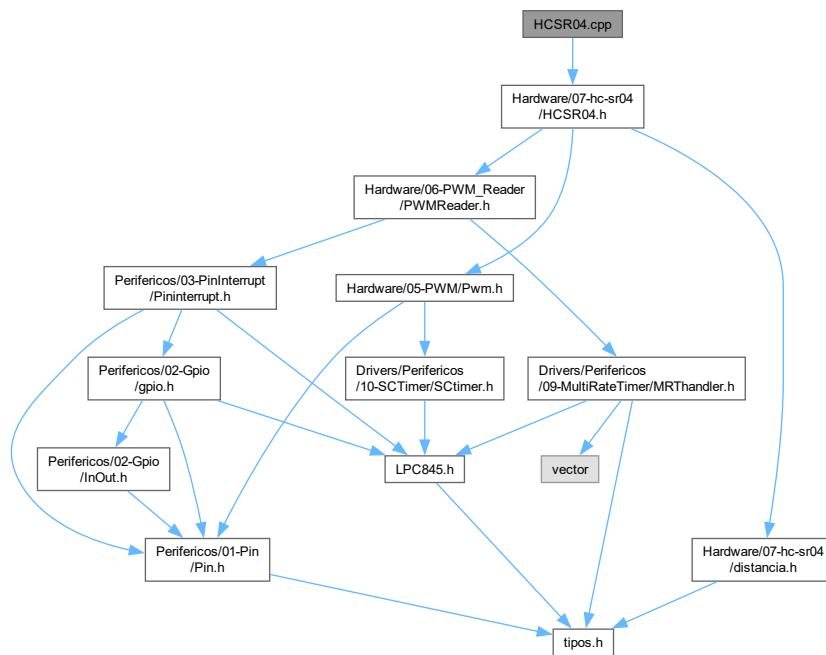
```
00001 /*****
00012 #ifndef DISTANCIA_H_
00013 #define DISTANCIA_H_
00017
00018 *** INCLUDES GLOBALES
00019
00020 #include "tipos.h"
00021
00022 *** DEFINES GLOBALES
00023
00024
00025
00026 *** MACROS GLOBALES
00027
00028
00029
00030 *** TIPO DE DATOS GLOBALES
00031
00032
00033
00034 *** IMPLANTACION DE LA CLASE
00035
00041 class distancia
00042 {
00043 public:
00044     distancia () = default;
00045     virtual uint32_t GetDistancia() = 0;
00046     virtual bool operator==( uint32_t a ) = 0;
00047     virtual ~distancia() = default;
00048 };
00049
00050 #endif /* DISTANCIA_H_ */
```

8.56 HCSR04.cpp File Reference

Clase del sensor ultrasónico HCSR04.

```
#include <Hardware/07-hc-sr04/HCSR04.h>
```

Include dependency graph for HCSR04.cpp:



8.56.1 Detailed Description

Clase del sensor ultrasónico HCSR04.

Date

22 jun. 2022

Version

1.0

Author

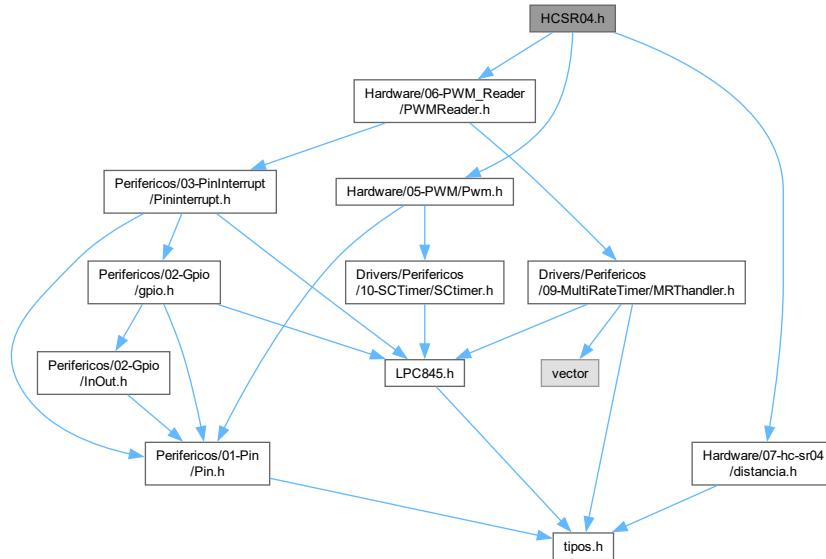
Técnico. Martinez Agustin (masteragus365@gmail.com)

8.57 HCSR04.h File Reference

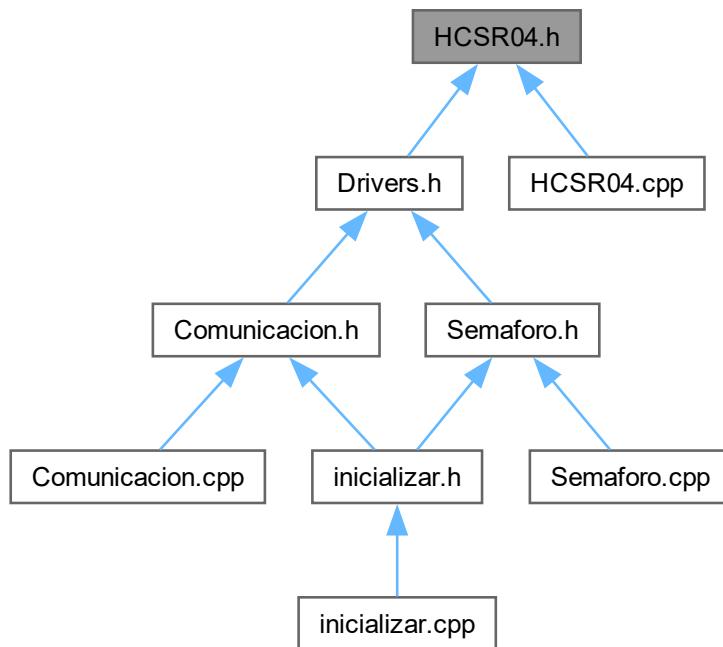
Clase del sensor ultrasónico HCSR04.

```
#include <Hardware/05-PWM/Pwm.h>
#include <Hardware/06-PWM_Reader/PWMReader.h>
#include <Hardware/07-hc-sr04/distancia.h>
```

Include dependency graph for HCSR04.h:



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



Classes

- class [HC_SR04](#)

Clase del objeto HC_SR04 El objeto HC_SR04 Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

Macros

- #define PERIODO 80
- #define CALC_DISTANCIA(a)
- #define DISTANCIA_MAX 400

8.57.1 Detailed Description

Clase del sensor ultrasónico HCSR04.

Date

22 jun. 2022

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.57.2 Macro Definition Documentation

8.57.2.1 CALC_DISTANCIA

```
#define CALC_DISTANCIA(
    a)
```

Value:

```
( ((a)*164) / 10000 )
```

Fórmula de cálculo tiempo a distancia

8.57.2.2 DISTANCIA_MAX

```
#define DISTANCIA_MAX 400
```

Distancia máxima en centímetros

8.57.2.3 PERIODO

```
#define PERIODO 80
```

Período del PWM En milisegundos

8.58 HCSR04.h

[Go to the documentation of this file.](#)

```
00001  ****
00012 #ifndef HCSR04_H_
00013 #define HCSR04_H_
00017
00018 *** INCLUDES GLOBALES
00019
00020 #include <Hardware/05-PWM/Pwm.h>
00021 #include <Hardware/06-PWM_Reader/PWMReader.h>
00022 #include <Hardware/07-hc-sr04/distancia.h>
00023
00024 *** DEFINES GLOBALES
00025
00026
00027
00028 *** MACROS GLOBALES
00029
00030
00031
00032 *** TIPO DE DATOS GLOBALES
00033
00034
00035
00036 *** IMPLANTACION DE LA CLASE
00037
00049 class HC_SR04 : protected distancia
00050 {
00051     private:
00052         #define PERIODO          80
00053         #define CALC_DISTANCIA(a) (( (a)*164) / 10000 )
00056     public:
00058         #define DISTANCIA_MAX    400
00059     private:
00060         PWM_Reader*           &m_rx;
00061         Pwm*                  &m_tx;
00062         uint32_t               m_distancia;
00063         bool                   m_stop;
00064
00065     public:
00066         HC_SR04( PWM_Reader* &rx , Pwm* &tx );
00067         void                  Initialize( void );
00068         uint32_t              GetDistancia( void ) override;
00069         void                  Off( void );
```

```

00070     void      On( void );
00071     bool      operator==( const uint32_t a ) override;
00072     bool      operator<=( const uint32_t a );
00073     bool      operator>=( const uint32_t a );
00074     bool      operator<( const uint32_t a );
00075     bool      operator>( const uint32_t a );
00076
00077     virtual   ~HC_SR04() {};
00078 };
00079
00080 #endif /* HCSR04_H */

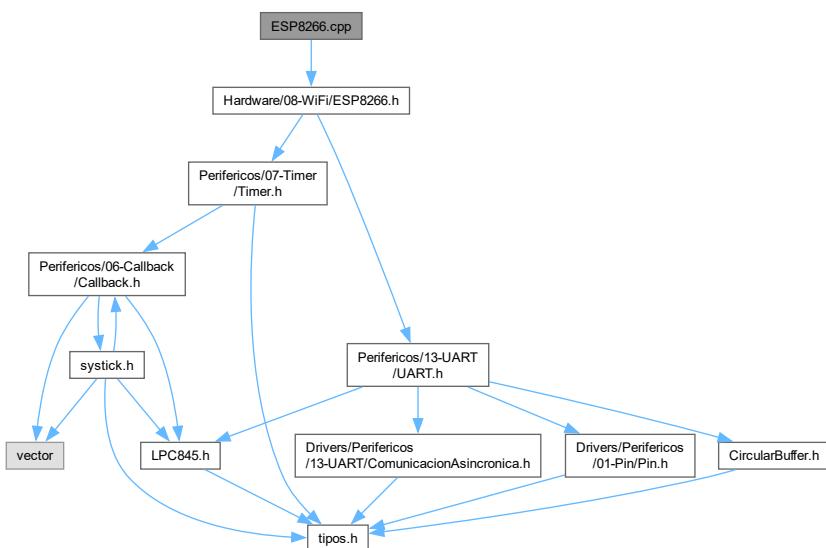
```

8.59 ESP8266.cpp File Reference

Modulo de comunicacion WIFI con [ESP8266](#).

```
#include <Hardware/08-WiFi/ESP8266.h>
```

Include dependency graph for ESP8266.cpp:



8.59.1 Detailed Description

Modulo de comunicacion WIFI con [ESP8266](#).

Modulo de comunicacion WIFI con el [ESP8266](#) utilizando comandos AT por puerto serie.

Date

2 mar. 2023

Version

1.0

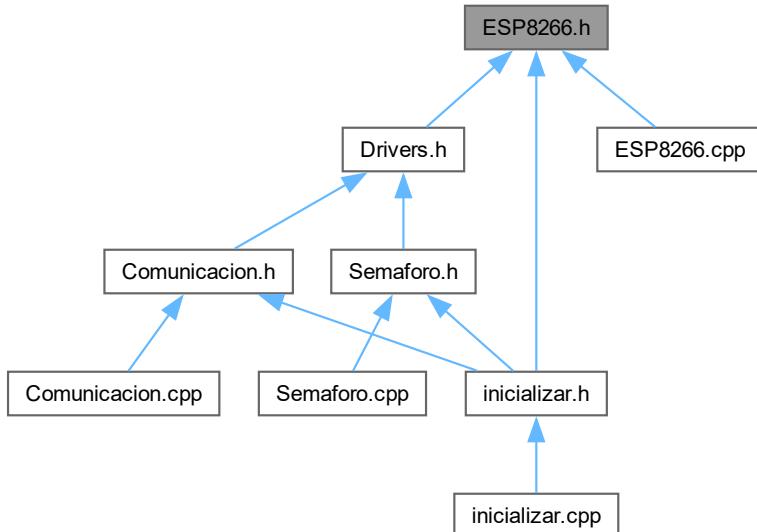
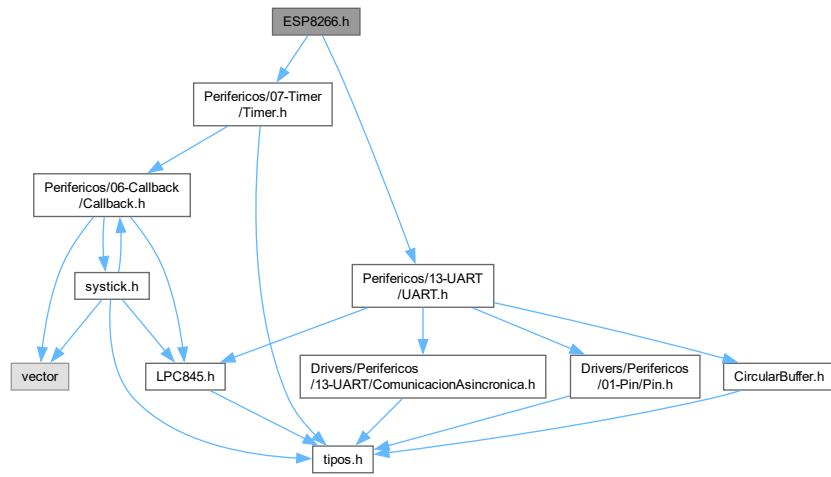
Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.60 ESP8266.h File Reference

Modulo de comunicacion WIFI con [ESP8266](#).

```
#include <Perifericos/07-Timer/Timer.h>
#include <Perifericos/13-UART/UART.h>
Include dependency graph for ESP8266.h:
```



Classes

- class [ESP8266](#)

*Clase del objeto **ESP8266**. El objeto **ESP8266** permite la simple utilización del módulo arduino **ESP8266** y el **ESP01** mediante comandos AT. El módulo debe estar por defecto en la velocidad **DEFAULT_ESP01_BAUDRATE**. El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Si se probó la inicialización y conexión a internet, no se probó la conexión a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver.*

Macros

- #define DEFAULT_ESP01_BAUDRATE 115200
- #define SEG_ESP01_TIMEOUT 20

8.60.1 Detailed Description

Modulo de comunicacion WIFI con [ESP8266](#).

Modulo de comunicacion WIFI con el [ESP8266](#) utilizando comandos AT por puerto serie.

Date

2 mar. 2023

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.60.2 Macro Definition Documentation**8.60.2.1 DEFAULT_ESP01_BAUDRATE**

#define DEFAULT_ESP01_BAUDRATE 115200

Velocidad de transmision del ESP por defecto, antes de haber sido modificado

8.60.2.2 SEG_ESP01_TIMEOUT

#define SEG_ESP01_TIMEOUT 20

Tiempo por default de espera para la realización de los comandos AT

8.61 ESP8266.h

[Go to the documentation of this file.](#)

```

00001  ****
00012  ****
00013  *** MODULO
00014  ****
00015 #ifndef ESP8266_H_
00016 #define ESP8266_H_
00017
00018  ****
00019  *** INCLUDES GLOBALES
00020  ****
00021 #include <Perifericos/07-Timer/Timer.h>
00022 #include <Perifericos/13-UART/UART.h>
00023  ****
00024  *** DEFINES GLOBALES
00025  ****
00026
00027  ****
00028  *** MACROS GLOBALES
00029  ****
00030
00031  ****
00032  *** TIPO DE DATOS GLOBALES
00033  ****

```

```

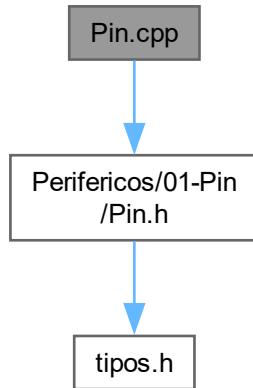
00034
00035
00036     **** VARIABLES GLOBALES
00037
00038
00039     ****
00040     *** IMPLANTACION DE LA CLASE
00041
00042     ****
00043
00044 class ESP8266 : protected UART
00045 {
00046     public:
00047         typedef enum { TCP = 0 , UDP = 1 } conection_type;
00048         typedef enum { ERROR = 0 , NOTHING , INITIALIZED , CONNECT_TO_WIFI , CONNECT_TO_SERVER }
00049             status_type;
00050     private:
00051         #define DEFAULT_ESP01_BAUDRATE      115200
00052         #define SEG_ESP01_TIMEOUT          20
00053
00054     const uint32_t m_baudrate;
00055     int8_t * m_address;
00056     int8_t * m_password;
00057     int8_t * m_IP;
00058     status_type m_status;
00059
00060     uint8_t m_aux;
00061
00062 public:
00063     ESP8266( Pin::port_t _portTx , uint8_t _pinTx , Pin::port_t _portRx , uint8_t _pinRx ,
00064               USART_Type * usart , uint32_t baudrate );
00065     void Initialize( void );
00066     status_type ConnectToWifi ( const int8_t * wifi_address , const int8_t * wifi_password , uint32_t
00067     seg_timeout = SEG_ESP01_TIMEOUT );
00068     void DisconnectToWifi ( void );
00069     void SetIP ( int8_t *ip );
00070     int8_t* GetIP( void ) const;
00071     bool ConnectToServer ( conection_type _mode , const int8_t* server_ip , const int8_t*
00072     server_port , uint32_t seg_timeout = SEG_ESP01_TIMEOUT );
00073     void DisconnectToServer ( void );
00074     void Write ( const char * msg );
00075     void Write ( const void * msg , uint32_t n );
00076     bool Read ( char * msg , uint32_t n );
00077     status_type GetStatus ( void ) const;
00078     bool IsConnectedToWifi( void ) const;
00079     bool IsConnectedToServer( void ) const;
00080     virtual ~ESP8266();
00081
00082 private:
00083     bool LeerOk ( void );
00084     int8_t* toString ( const uint32_t n );
00085     uint32_t Strlen ( const int8_t * a );
00086     uint32_t pow ( uint32_t num , uint32_t exp );
00087 };
00088
00089 #endif /* ESP8266_H_ */

```

8.62 Pin.cpp File Reference

Clase Abstracta de cualquier pin del microcontrolador.

```
#include <Perifericos/01-Pin/Pin.h>
Include dependency graph for Pin.cpp:
```



8.62.1 Detailed Description

Clase Abstracta de cualquier pin del microcontrolador.

Date

10 ene. 2023

Version

1.0

Author

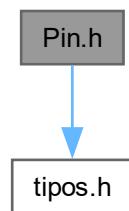
Técnico. Martinez Agustin (masteragus365@gmail.com)

8.63 Pin.h File Reference

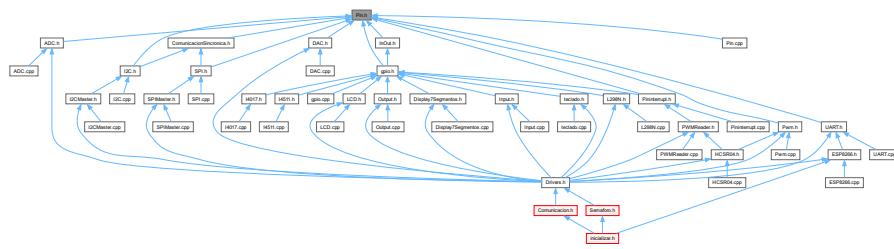
Clase Abstracta de cualquier pin del microcontrolador.

```
#include "tipos.h"
```

Include dependency graph for Pin.h:



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



Classes

- class [Pin](#)

Clase del objeto Pin.

Variables

- const [uint8_t IOCON_INDEX_PIO0](#) [] = { 17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,20,0,0,35 };
- const [uint8_t IOCON_INDEX_PIO1](#) [] = { 36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,0,0,0,50,51 };

8.63.1 Detailed Description

Clase Abstracta de cualquier pin del microcontrolador.

Date

10 ene. 2023

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.64 Pin.h

[Go to the documentation of this file.](#)

```

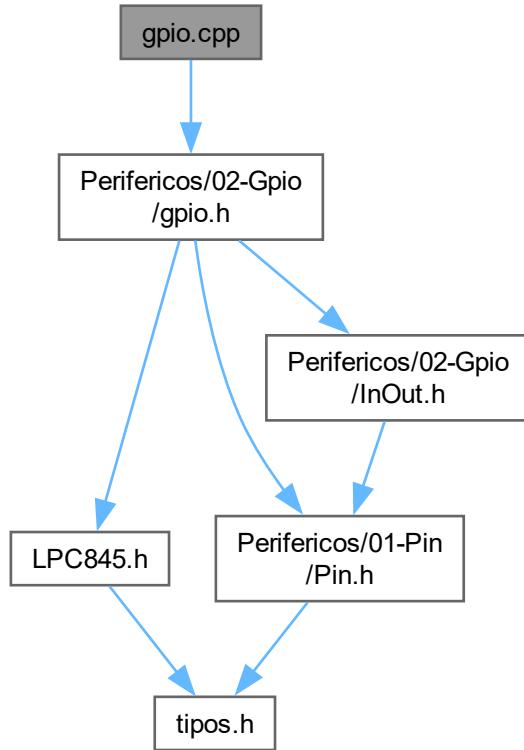
00001  ****
00012  ****
00013  *** MODULO
00014  ****
00015 #ifndef PIN_H_
00016 #define PIN_H_
00020
00021  ****
00022  *** INCLUDES GLOBALES
00023  ****
00024 #include "tipos.h"
00025
00026  ****
00027  *** DEFINES GLOBALES
00028  ****
00029
00030  ****
00031  *** MACROS GLOBALES
  
```

```
00032
*****
00033
00034 /*****
00035 *** TIPO DE DATOS GLOBALES
00036 ****
00037
00038 /*****
00039 *** VARIABLES GLOBALES
00040 ****
00041 const uint8_t IOCON_INDEX_PIO0[] = {
00042     17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,20,0,0,35};
00043 const uint8_t IOCON_INDEX_PIO1[] = {
00044     36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,0,0,50,51};
00045 ****
00046 *** IMPLANTACION DE LA CLASE
00047 ****
00048 class Pin
00049 {
00050     public:
00051         typedef enum port_t           { port0 , port1 } port_t;
00052         enum max_bits_port_t{ b_port0 = 31 , b_port1 = 9 };
00053         typedef enum error_t          { error = 2 , ok } error_t;
00054
00055     public:
00056         const  port_t m_port ;
00057         const  uint8_t m_bit ;
00058         int8_t  m_error ;
00059
00060     public:
00061         Pin( port_t port , uint8_t bit );
00062         virtual ~Pin() {};
00063     };
00064 #endif /* PIN_H_ */
```

8.65 gpiocpp File Reference

Módulo con clase de manejo de GPIO.

```
#include <Perifericos/02-Gpio/gpio.h>
Include dependency graph for gpio.cpp:
```



8.65.1 Detailed Description

Módulo con clase de manejo de GPIO.

Date

22 jun. 2022

Author

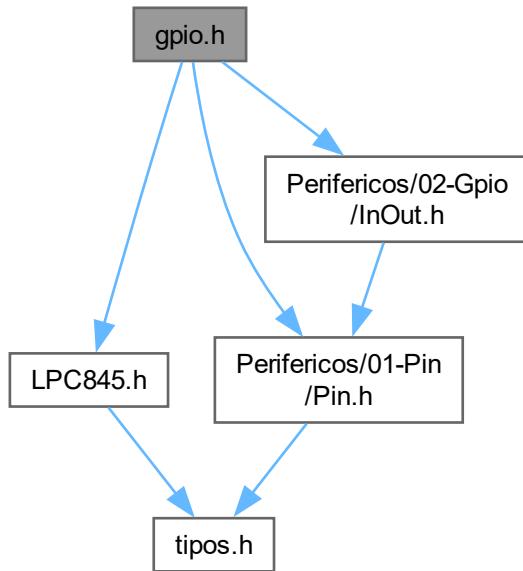
Ing. Marcelo Trujillo

8.66 gpio.h File Reference

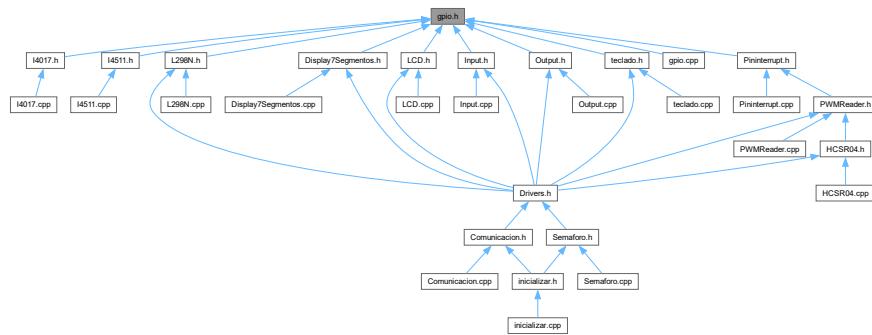
Módulo con clase de manejo de GPIO.

```
#include <LPC845.h>
#include <Perifericos/01-Pin/Pin.h>
#include <Perifericos/02-Gpio/InOut.h>
```

Include dependency graph for gpio.h:



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



Classes

- class `gpio`

Clase del objeto gpio.

8.66.1 Detailed Description

Módulo con clase de manejo de GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

8.67 gpio.h

[Go to the documentation of this file.](#)

```
00001 //*****
00002 **** MODULO
00003 **** INCLUDES GLOBALES
00004 #ifndef GPIO_H_
00005 #define GPIO_H_
00006
00007 **** DEFINES GLOBALES
00008
00009 **** MACROS GLOBALES
00010
00011 **** TIPO DE DATOS GLOBALES
00012
00013 **** VARIABLES GLOBALES
00014
00015 **** IMPLANTACION DE LA CLASE
00016
00017 class gpio : public InOut , public Pin
00018 {
00019     public:
00020         typedef enum direction_t { input , output } direction_t;
00021         enum power_t { off , on };
00022         typedef enum mode_t { pushpull = 0 , opencolector , inactive = 0 , pulldown , pullup ,
00023             repeater } mode_t; // el orden esta atado a las hojas de datos
00024         typedef enum activity_t { low , high } activity_t;
00025         enum interrupt_mode_t { rising_edge = 0 , falling_edge , rising_falling_edge , low_level ,
00026             high_level };
00027
00028     protected:
00029         const mode_t m_mode ;
00030         direction_t m_direction ;
00031         const activity_t m_activity ;
00032
00033     public:
00034         gpio ( port_t port , uint8_t bit , mode_t mode , direction_t direction , activity_t activity =
00035             high );
00036
00037         uint8_t SetPin ( void ) override;
00038         uint8_t ClrPin ( void ) override;
00039         uint8_t SetTogglePin ( void ) override;
00040         uint8_t SetDir ( void ) override;
00041         uint8_t SetToggleDir ( void ) override;
00042         uint8_t GetPin ( void ) override;
00043         uint8_t SetPinMode ( void ) override;
00044         uint8_t SetPinResistor ( void ) override;
```

```

00079     gpio& operator= ( uint8_t a );
00080
00081     virtual ~gpio() = default;
00082 };
00083
00084 #endif /* GPIO_H_ */

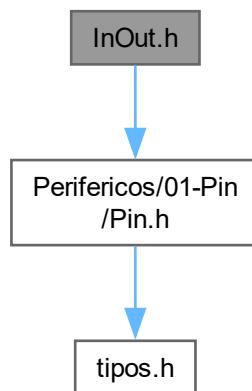
```

8.68 InOut.h File Reference

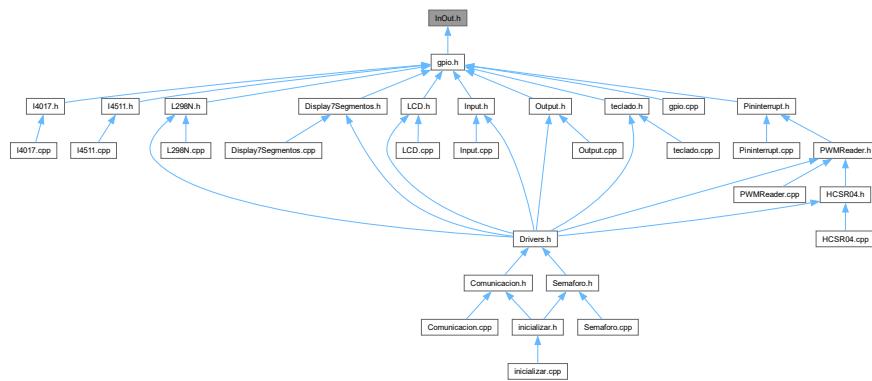
Clase Abstracta Pura de las GPIO.

```
#include <Perifericos/01-Pin/Pin.h>
```

Include dependency graph for InOut.h:



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



Classes

- class [InOut](#)

Clase del objeto [InOut](#).

8.68.1 Detailed Description

Clase Abstracta Pura de las GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

8.69 InOut.h

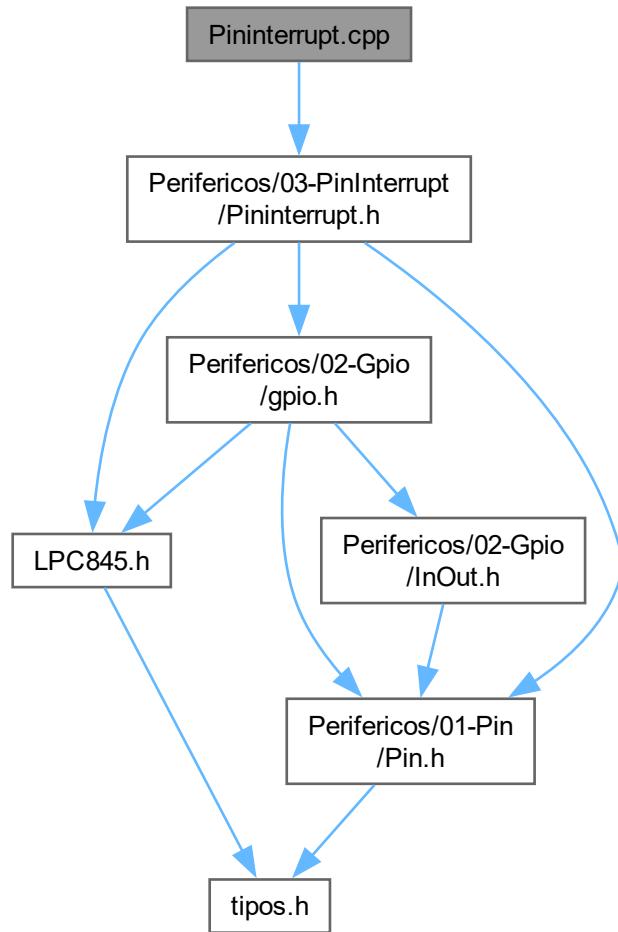
[Go to the documentation of this file.](#)

```
00001  /*****
00002  ****
00003  **** MODULO
00004  ****
00005  **** INCLUDES GLOBALES
00006  ****
00007  **** MACROS GLOBALES
00008  ****
00009  **** TIPO DE DATOS GLOBALES
00010  ****
00011  **** VARIABLES GLOBALES
00012  ****
00013  **** IMPLANTACION DE LA CLASE
00014  ****
00015  class InOut
00016  {
00017      public:
00018          InOut () = default;
00019          virtual uint8_t SetPin ( void )      = 0;
00020          virtual uint8_t ClrPin ( void )      = 0;
00021          virtual uint8_t SetTogglePin ( void ) = 0;
00022          virtual uint8_t SetDir ( void )      = 0;
00023          virtual uint8_t SetToggleDir ( void ) = 0;
00024          virtual uint8_t GetPin ( void )      = 0;
00025          virtual uint8_t SetPinMode ( void )   = 0;
00026          virtual uint8_t SetPinResistor ( void ) = 0;
00027      };
00028  #endif /* IN_OUT_H_ */
```

8.70 Pininterrupt.cpp File Reference

Clase para entradas con interrupciones por flanco.

```
#include <Perifericos/03-PinInterrupt/Pininterrupt.h>
Include dependency graph for Pininterrupt.cpp:
```



Variables

- `PinInterrupt * g_gpiohandler [8]`

8.70.1 Detailed Description

Clase para entradas con interrupciones por flanco.

Date

17 sep. 2022

Version

1.0

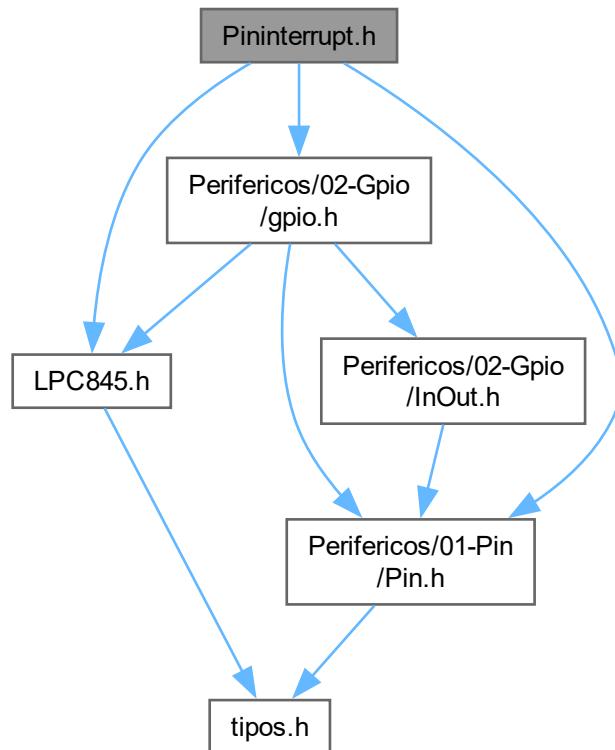
Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

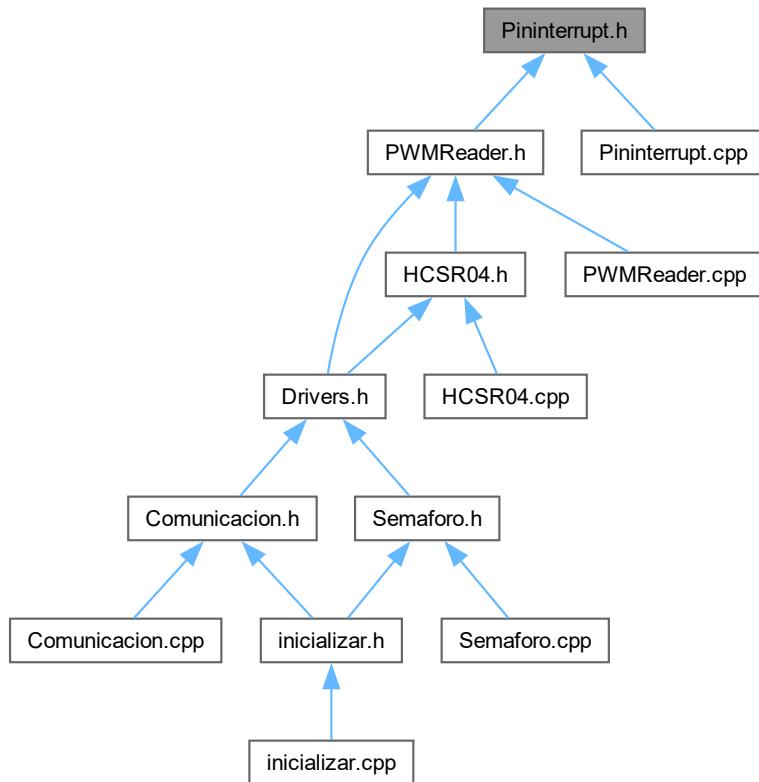
8.71 Pininterrupt.h File Reference

Clase para entradas con interrupciones por flanco.

```
#include <LPC845.h>
#include <Perifericos/01-Pin/Pin.h>
#include <Perifericos/02-Gpio/gpio.h>
Include dependency graph for Pininterrupt.h:
```



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



Classes

- class [PinInterrupt](#)

Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interrupciones por pin.

Macros

- #define [MAX_PININTERRUPT](#) 8

Variables

- [PinInterrupt * g_gpiohandler \[MAX_PININTERRUPT\]](#)

8.71.1 Detailed Description

Clase para entradas con interrupciones por flanco.

Date

17 sep. 2022

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.71.2 Macro Definition Documentation

8.71.2.1 MAX_PININTERRUPT

```
#define MAX_PININTERRUPT 8
```

Cantidad máxima de interrupciones de pin

8.72 Pininterrupt.h

[Go to the documentation of this file.](#)

```
00001
00012 #ifndef PININTERRUPT_H_
00013 #define PININTERRUPT_H_
00017
00018 *** INCLUDES GLOBALES
00019
00020 #include <LPC845.h>
00021 #include <Perifericos/01-Pin/Pin.h>
00022 #include <Perifericos/02-Gpio/gpio.h>
00023
00024
00025 *** DEFINES GLOBALES
00026
00027 #if defined (__cplusplus)
00028     extern "C" {
00029         void PININT0_IRQHandler(void);
00030     }
00031     extern "C" {
00032         void PININT1_IRQHandler(void);
00033     }
00034     extern "C" {
00035         void PININT2_IRQHandler(void);
00036     }
00037     extern "C" {
00038         void PININT3_IRQHandler(void);
00039     }
00040     extern "C" {
00041         void PININT4_IRQHandler(void);
00042     }
00043     extern "C" {
00044         void PININT5_IRQHandler(void);
00045     }
00046     extern "C" {
00047         void PININT6_IRQHandler(void);
00048     }
00049     extern "C" {
00050         void PININT7_IRQHandler(void);
00051     }
00052 #endif
00053
00054 *** MACROS GLOBALES
00055
00056
00057
00058 *** TIPO DE DATOS GLOBALES
00059
00060
00061
00062 *** IMPLANTACION DE LA CLASE
00063
00064
00065
00066
00067
00068
00069 class PinInterrupt : protected gpio
00070 {
00071 public:
00072     static uint8_t m_cant;
00073     const uint8_t m_interrupt_number;
00074     const uint8_t m_interrupt_mode;
```

```

00075
00076 public:
00077     #define MAX_PININTERRUPT      8
00078
00079
00080 public:
00081     PinInterrupt( port_t port , uint8_t bit , mode_t gpio_mode , activity_t activity ,
00082     uint8_t intrp_mode );
00083     void           EnableInterrupt ( void );
00084     void           DisableInterrupt ( void );
00085     void           PinInterrupt_Inicializar( void );
00086     virtual void   GpioHandler(void) = 0;
00087     virtual       ~PinInterrupt();
00088 private:
00089     void           PinInterrupt_Enable_clock( void );
00090 }
00091
00092 extern PinInterrupt * g_gpiohandler[MAX_PININTERRUPT];
00093
00094
00095 #endif /* PININTERRUPT_H_ */

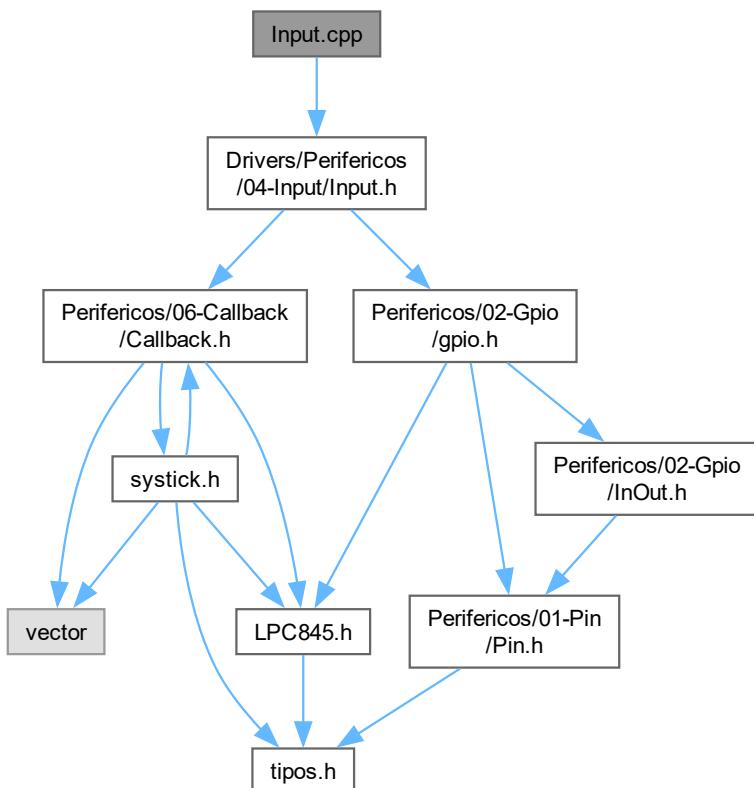
```

8.73 Input.cpp File Reference

funciones miembro de la clase `Input`.

#include <Drivers/Perifericos/04-Input/Input.h>

Include dependency graph for Input.cpp:



Functions

- bool `operator== (uint32_t val, Input &l)`

Sobrecarga de del operador de asignacion.

8.73.1 Detailed Description

funciones miembro de la clase [Input](#).

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

8.73.2 Function Documentation

8.73.2.1 operator==()

```
bool operator== (
    uint32_t val,
    Input & I)
```

Sobrecarga de del operador de asignacion.

Parameters

in	val	Valor a comparar con el buffer.
in	I	Entrada a comparar.

Returns

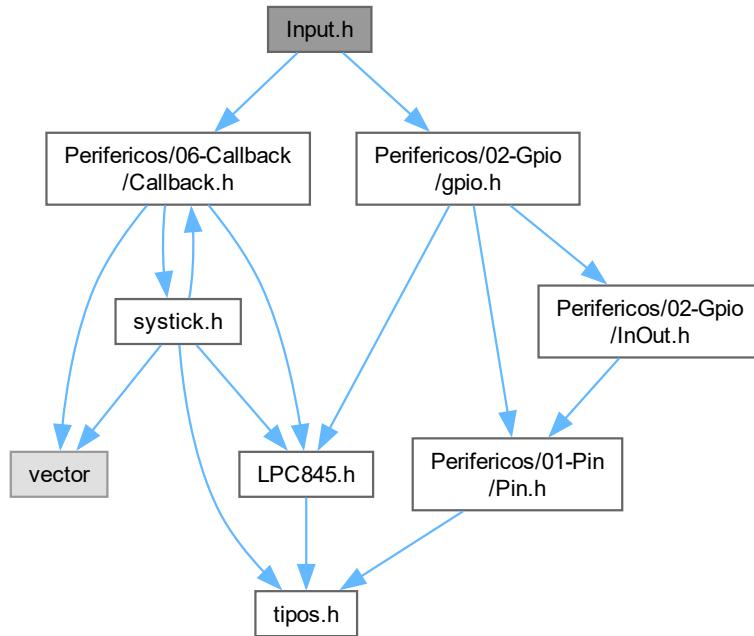
true si la entrada esta en val.

8.74 Input.h File Reference

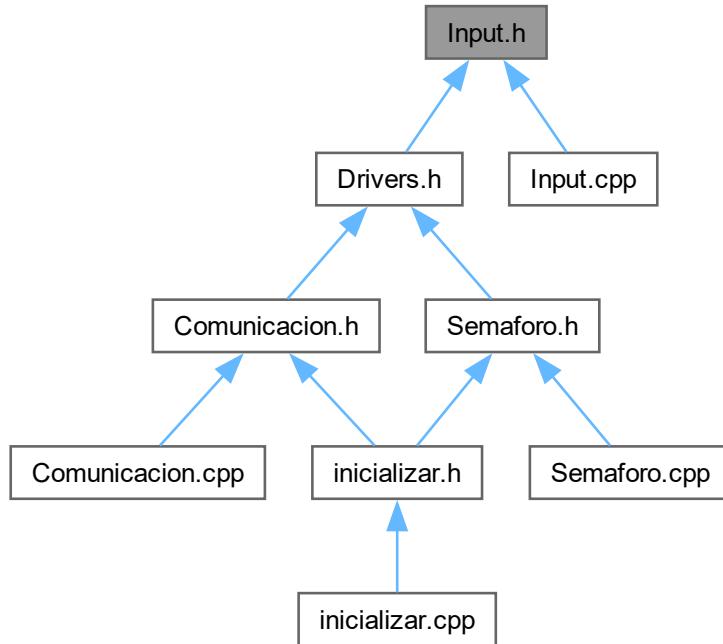
funciones miembro de la clase [Input](#).

```
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
```

Include dependency graph for Input.h:



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



Clases

- class [Input](#)

Clase del objeto [Input](#).

Macros

- `#define MAX_BOUCNE 4`

8.74.1 Detailed Description

funciones miembro de la clase [Input](#).

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

8.74.2 Macro Definition Documentation

8.74.2.1 MAX_BOUCNE

`#define MAX_BOUCNE 4`

Cantidad de rebotes por defecto

8.75 Input.h

[Go to the documentation of this file.](#)

00001

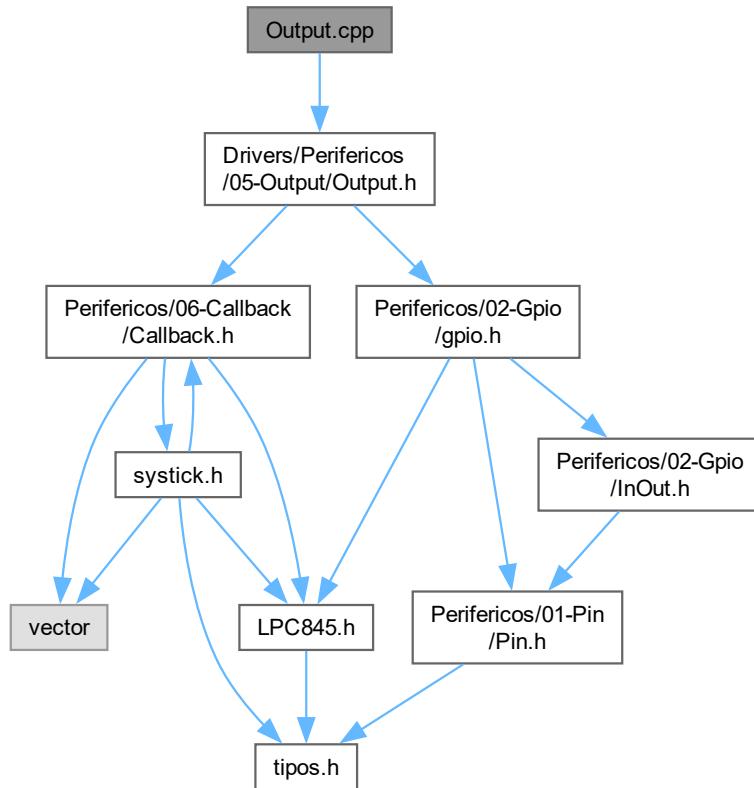
```
00009 /*****
00010 ****
00011 *** MODULO
00012 ****
00013 #ifndef INPUTS_H_
00014 #define INPUTS_H_
00018 ****
00019 *** INCLUDES GLOBALES
00020 ****
00021 #include <Perifericos/02-Gpio/gpio.h>
00022 #include <Perifericos/06-Callback/Callback.h>
00023
00024 ****
00025 *** DEFINES GLOBALES
00026 ****
00027
00028 ****
00029 *** MACROS GLOBALES
00030 ****
00031
00032 ****
00033 *** TIPO DE DATOS GLOBALES
00034 ****
00035
00036 ****
00037 *** VARIABLES GLOBALES
00038 ****
00039 ****
```

```
00040
00041     /***** IMPLANTACION DE LA CLASE *****
00042
00043     Input : protected gpio , public Callback
00044 {
00045     private:
00046         uint8_t m_BufferEntrada ;
00047         uint8_t m_MaxBounce;
00048         uint8_t m_CountBounce ;
00049     private:
00050         #define MAX_BOOUNCE      4
00051
00052     public:
00053         Input( port_t puerto , uint8_t bit , mode_t modo , activity_t actividad = high , uint8_t
00054             MaxBounce = MAX_BOOUNCE );
00055         void Initialize ( void );
00056         uint8_t get ( void );
00057         void SHandler ( void ) override;
00058         bool operator== ( uint8_t val );
00059         bool operator!= ( uint8_t val );
00060         friend bool operator==( uint32_t val , Input &I );
00061
00062         virtual ~Input();
00063     private:
00064         void SetBuffer ( void );
00065 };
00066
00067 #endif /* INPUTS_H_ */
```

8.76 Output.cpp File Reference

Clase del tipo [Output](#) o salida digital.

```
#include <Drivers/Perifericos/05-Output/Output.h>
Include dependency graph for Output.cpp:
```



8.76.1 Detailed Description

Clase del tipo [Output](#) o salida digital.

Date

12 jul. 2022

Author

Ing. Marcelo Trujillo

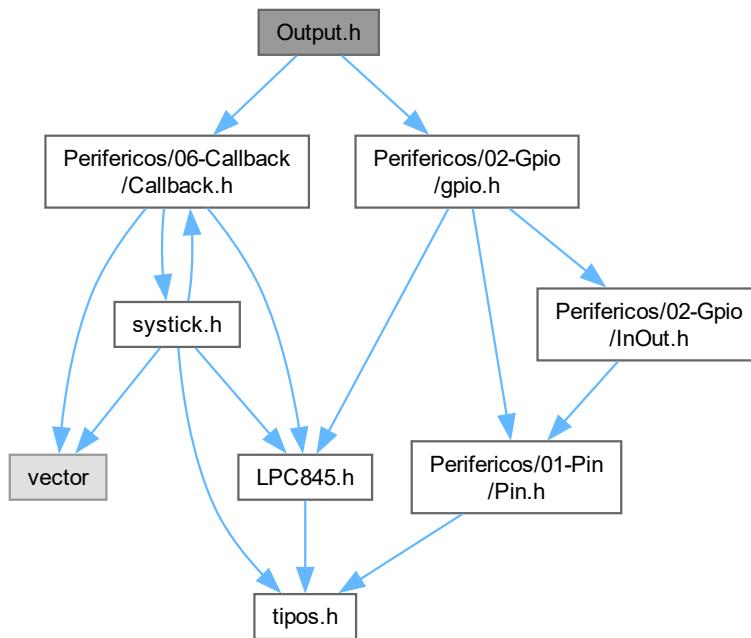
8.77 Output.h File Reference

Clase del tipo [Output](#) o salida digital.

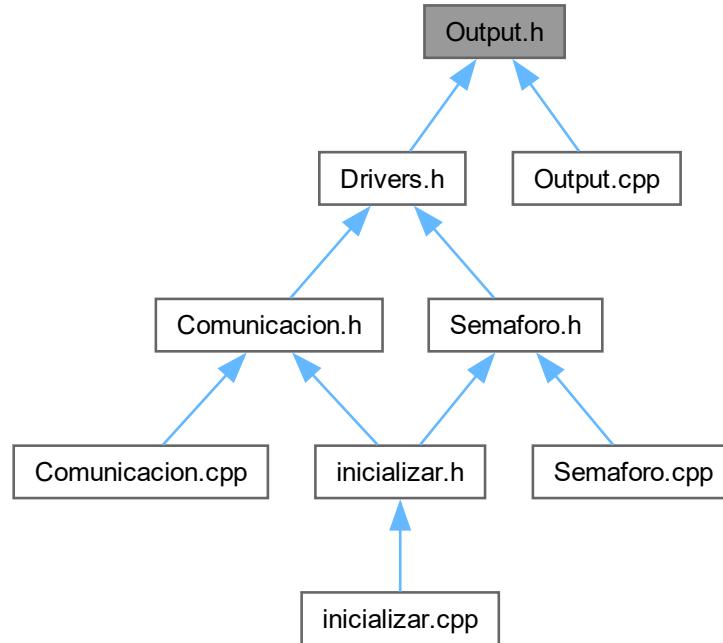
```
#include <Perifericos/02-Gpio/gpio.h>
```

```
#include <Perifericos/06-Callback/Callback.h>
```

Include dependency graph for Output.h:



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



Classes

- class [Output](#)

Clase del objeto outputs.

8.77.1 Detailed Description

Clase del tipo [Output](#) o salida digital.

Date

12 jul. 2022

Author

Ing. Marcelo Trujillo

8.78 Output.h

[Go to the documentation of this file.](#)

00001

```

/*****
00009
0010
0011  *** MODULO
0012
0013 #ifndef OUTPUTS_H_
0014 #define OUTPUTS_H_
0018
*****
```

```

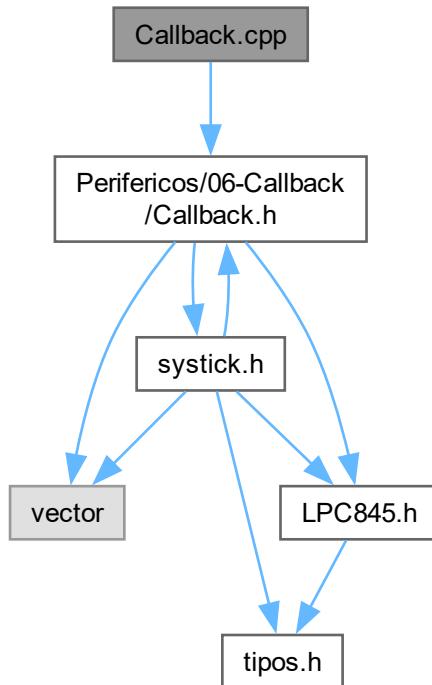
00019  *** INCLUDES GLOBALES
00020
00021 #include <Perifericos/02-Gpio/gpio.h>
00022 #include <Perifericos/06-Callback/Callback.h>
00023
00024
00025 /*****
00026  *** DEFINES GLOBALES
00027
00028
00029 /*****
00030  *** MACROS GLOBALES
00031
00032
00033 /*****
00034  *** TIPO DE DATOS GLOBALES
00035
00036
00037 /*****
00038  *** VARIABLES GLOBALES
00039
00040
00041 /*****
00042  *** IMPLANTACION DE UNA CLASE
00043
00044
00045 class Output : protected gpio , public Callback
00046 {
00047     uint8_t m_buffer;
00048
00049     public:
00050         Output( port_t puerto , uint8_t bit , mode_t modo , activity_t actividad = high , uint8_t
00051         estado = on );
00052             int8_t On ( void );
00053             int8_t Off ( void );
00054             int8_t Initialize( void );
00055
00056             Output& operator= ( uint8_t estado );
00057             bool operator== ( uint8_t a );
00058             void SWhandler ( void ) override;
00059
00060             virtual ~Output();
00061
00062
00063
00064 };
00065
00066 #endif /* OUTPUTS_H_ */

```

8.79 Callback.cpp File Reference

funciones miembro de la clase [Callback](#).

```
#include <Perifericos/06-Callback/Callback.h>
Include dependency graph for Callback.cpp:
```



Variables

- `vector< Callback * > g_Handler`

8.79.1 Detailed Description

funciones miembro de la clase [Callback](#).

Date

04 ene. 2025

Author

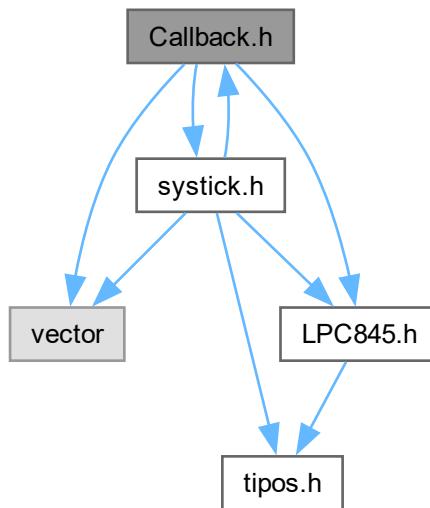
Tecnico Martinez Agustin

8.80 Callback.h File Reference

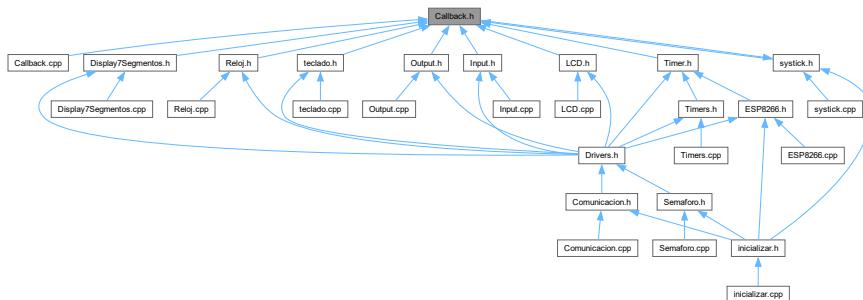
Clase virtual pura. Esta clase debe ser heredada por las clases que se tienen que enganchar del Systick [Timer](#).

```
#include <LPC845.h>
#include <vector>
#include <systick.h>
```

Include dependency graph for Callback.h:



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



Classes

- class `Callback`
Clase del objeto `Callback`.

Macros

- `#define TICK_SECONDS(x)`
- `#define TICK_MILLISECONDS(x)`
- `#define TICK_MICROSECONDS(x)`

Variables

- `vector<Callback * > g_Handler`

8.80.1 Detailed Description

Clase virtual pura. Esta clase debe ser heredada por las clases que se tienen que enganchar del Systick [Timer](#).

Date

4 may. 2022

Author

Ing. Marcelo Trujillo

8.80.2 Macro Definition Documentation

8.80.2.1 TICK_MICROSECONDS

```
#define TICK_MICROSECONDS(
    x)
```

Value:

```
((x) * (g_systick_freq/1000000))
```

MACRO expresion to obtain the corresponding TICKS for "x" microseconds

8.80.2.2 TICK_MILISECONDS

```
#define TICK_MILISECONDS(
    x)
```

Value:

```
((x) * (g_systick_freq/1000))
```

MACRO expresion to obtain the corresponding TICKS for "x" miliseconds

8.80.2.3 TICK_SECONDS

```
#define TICK_SECONDS(
    x)
```

Value:

```
((x) * g_systick_freq)
```

MACRO expresion to obtain the corresponding TICKS for "x" Seconds

8.81 Callback.h

[Go to the documentation of this file.](#)

```
00001
00002 /*****
00003 #ifndef CALLBACK_H_
00004 #define CALLBACK_H_
00005
00006 /*****
00007 *** INCLUDES GLOBALES
00008
00009 ****
00010 #include <LPC845.h>
00011 #include <vector>
00012 #include <systick.h>
00013 using namespace std;
00014
00015 /*****
00016 *** DEFINES GLOBALES
00017
00018 ****
00019 *** MACROS GLOBALES
00020
00021 ****
00022 ***
```

```

00031 *** IMPLANTACION DE LA CLASE
00032 ****
00039 class Callback
00040 {
00041     protected:
00043         #define TICK_SECONDS(x)      ((x) * g_systick_freq)
00045         #define TICK_MILISECONDS(x)  ((x) * (g_systick_freq/1000))
00047         #define TICK_MICROSECONDS(x) ((x) * (g_systick_freq/1000000))
00048     public:
00049         Callback() = default;
00050         void SetInterrupt();
00051         void UnSetInterrupt();
00052
00056         virtual void SWhandler( void ) = 0;
00057
00058         virtual ~Callback() = default;
00059 };
00060
00062 extern vector <Callback*> g_Handler;
00063
00064 #endif /* CALLBACK_H_ */

```

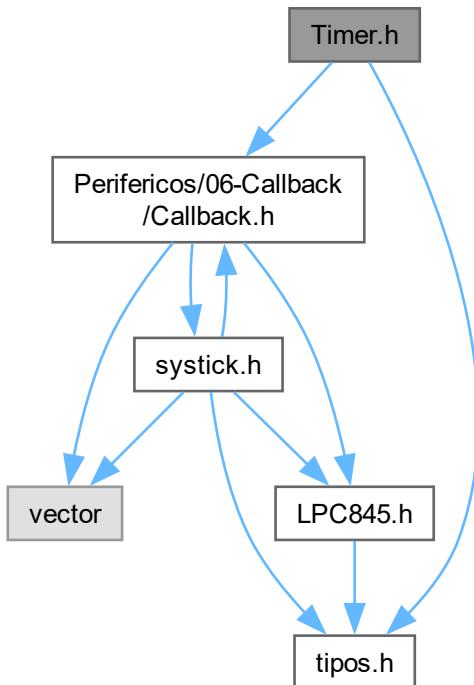
8.82 Timer.h File Reference

Clase para creacion de temporizadores.

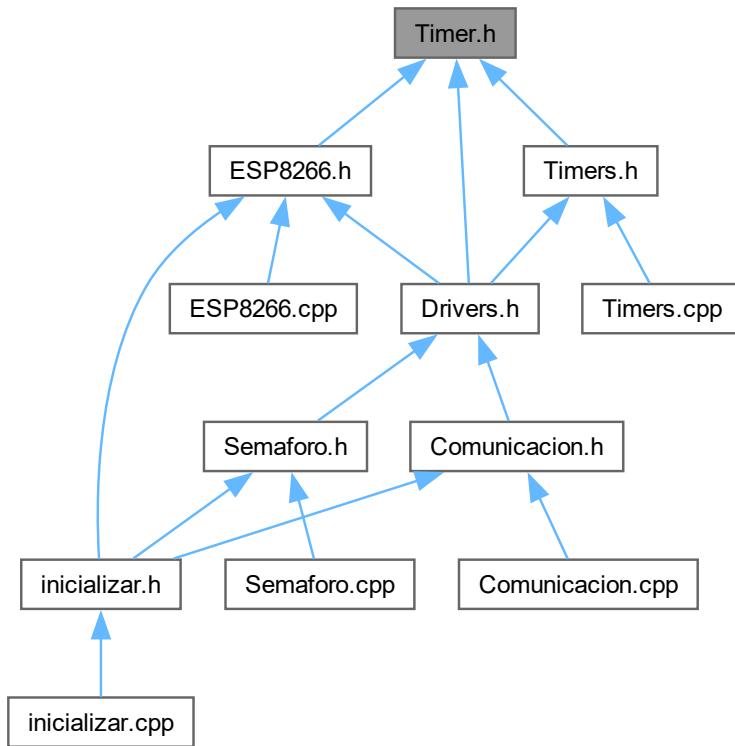
```
#include <Perifericos/06-Callback/Callback.h>
```

```
#include "tipos.h"
```

Include dependency graph for Timer.h:



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



Classes

- class [Timer](#)
Clase del objeto timer.

TypeDefs

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

8.82.1 Detailed Description

Clase para creacion de temporizadores.

Date

4 may. 2022

Author

Ing. Marcelo Trujillo

8.83 Timer.h

[Go to the documentation of this file.](#)

00001

/*****

```

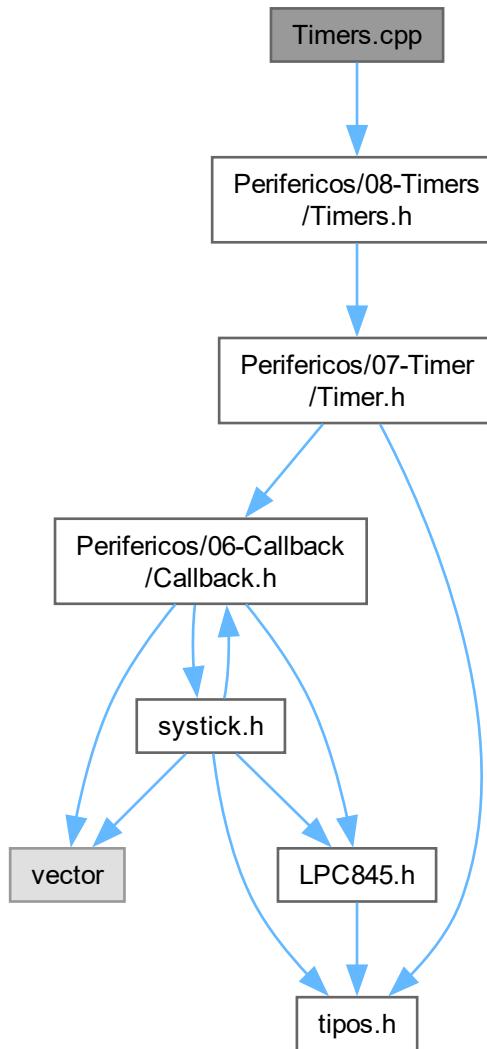
00009 #ifndef TIMER_H
00010 #define TIMER_H
00014
00015 /* **** INCLUDES GLOBALES **** */
00016
00017 #include <Perifericos/06-Callback/Callback.h>
00018 #include "tipos.h"
00019
00020
00021 /* **** DEFINES GLOBALES **** */
00022
00023
00024
00025 /* **** MACROS GLOBALES **** */
00026
00027
00028
00029 /* **** TIPO DE DATOS GLOBALES **** */
00030
00031 typedef void (*Timer_Handler)(void);
00032
00033
00034 /* **** IMPLANTACION DE LA CLASE **** */
00035
00036
00037 class Timer : public Callback
00038 {
00039     private:
00040         enum ticks_t { DECIMAS = 100 , SEGUNDOS = 10 , MINUTOS = 60 };
00041
00042     public:
00043         typedef enum bases_t { DEC , SEG , MIN , HOR } bases_t;
00044         enum errorestimers_t { errorTimer , OKtimers };
00045         enum standby_t { RUN , PAUSE };
00046
00047     protected:
00048         volatile uint32_t m_TmrRun;
00049         volatile bool m_TmrEvent;
00050         void (* m_TmrHandler ) (void);
00051         volatile bool m_TmrStandBy ;
00052         volatile bases_t m_TmrBase ;
00053
00054
00055     public:
00056         Timer( );
00057         Timer ( const bases_t base , const Timer_Handler handler = nullptr );
00058         void TimerStart ( uint32_t time, const Timer_Handler handler , const bases_t base );
00059         void SetTimer( uint32_t time );
00060         uint32_t GetTimer( void ) const;
00061         void StandByTimer( const uint8_t accion );
00062         void SetTimerBase( const bases_t base );
00063         void TimerStop( void );
00064         uint32_t GetTmrRun( void );
00065         void SetTmrEvent( void );
00066         void ClrTmrEvent( void );
00067         bool GetTmrEvent( void );
00068         bool GetmrStandBy( void );
00069         void SetmrStandBy( uint8_t accion );
00070         void SetTmrHandler( void );
00071         void TimerStart( uint32_t time );
00072
00073     Timer& operator=( uint32_t t );
00074     bool operator!();
00075     explicit operator bool () ;
00076     bool operator==( uint32_t t );
00077
00078 // por el hecho de haber convertido el operador bool como explicit
00079 // me obligo a realizar las funciones amigas del operador==
00080 // con sus dos prototipos, porque dejo de aceptar
00081 // la promocion automatica de tipos
00082     friend bool operator==( uint32_t t , Timer &T );
00083 // Implementacion de funcion virtual pura heredada
00084     void SWhandler( void ) override;
00085     int8_t TmrEvent ( void );
00086
00087     virtual ~Timer();
00088 }
00089
00090 #endif /* TIMER_H */

```

8.84 Timers.cpp File Reference

funciones miembro de la clase timers.

```
#include <Perifericos/08-Timers/Timers.h>
Include dependency graph for Timers.cpp:
```



8.84.1 Detailed Description

funciones miembro de la clase timers.

Modulo para crear conjuntos de [Timer](#).

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

Date

10 jul. 2022

Author

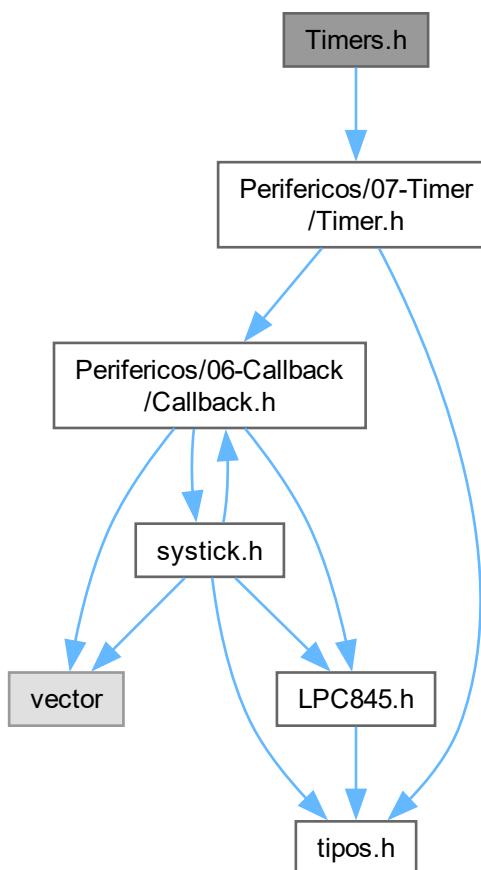
Ing. Marcelo Trujillo

8.85 Timers.h File Reference

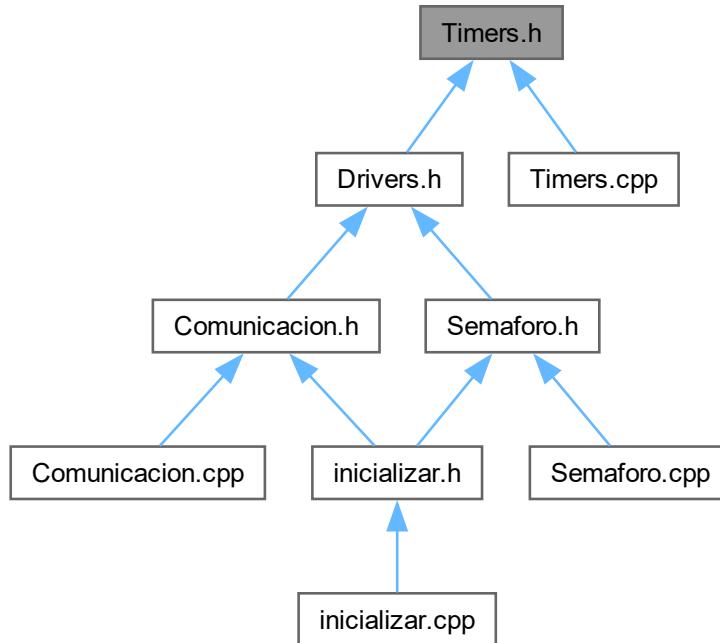
Modulo para crear conjuntos de Timer.

```
#include <Perifericos/07-Timer/Timer.h>
```

Include dependency graph for Timers.h:



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



Classes

- class [timers](#)

Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.

8.85.1 Detailed Description

Modulo para crear conjuntos de [Timer](#).

Date

10 jul. 2022

Author

Ing. Marcelo Trujillo

8.86 Timers.h

[Go to the documentation of this file.](#)

```

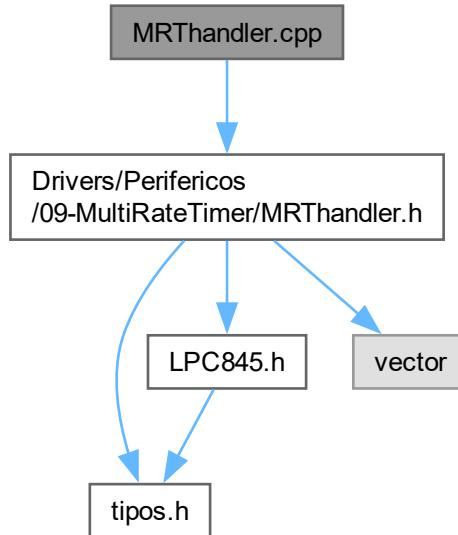
00001  ****
00002  *** MODULO
00003  ****
00004  #ifndef TIMERS_H_
00005  #define TIMERS_H_
  
```

```
00018
00019     /***** INCLUDES GLOBALES *****/
00020
00021 #include <Perifericos/07-Timer/Timer.h>
00022
00023
00024     /***** DEFINES GLOBALES *****/
00025
00026
00027
00028     /***** MACROS GLOBALES *****/
00029
00030
00031
00032     /***** TIPO DE DATOS GLOBALES *****/
00033
00034
00035
00036     /***** VARIABLES GLOBALES *****/
00037
00038
00039
00040     /***** IMPLANTACION DE UNA CLASE *****/
00041
00042
00043
00044
00045
00046
00047 class timers
00048 {
00049     private:
00050         vector <Timer* > m_timers;
00051
00052     public:
00053         timers();
00054         timers & operator<<( Timer *t );
00055         void TmrEvent( void );
00056     virtual ~timers() = default;
00057 }
00058 #endif /* TIMERS_H_ */
```

8.87 MRThandler.cpp File Reference

Handler del timer MRT.

```
#include <Drivers/Perifericos/09-MultiRateTimer/MRThandler.h>
Include dependency graph for MRThandler.cpp:
```



Variables

- std::vector< [MRThandler](#) * > [g_MRThandler](#)

8.87.1 Detailed Description

Handler del timer MRT.

Date

2 sep. 2022

Version

1.0

Author

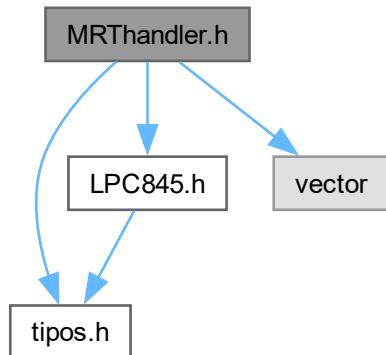
Técnico. Martinez Agustin (masteragus365@gmail.com)

8.88 MRThandler.h File Reference

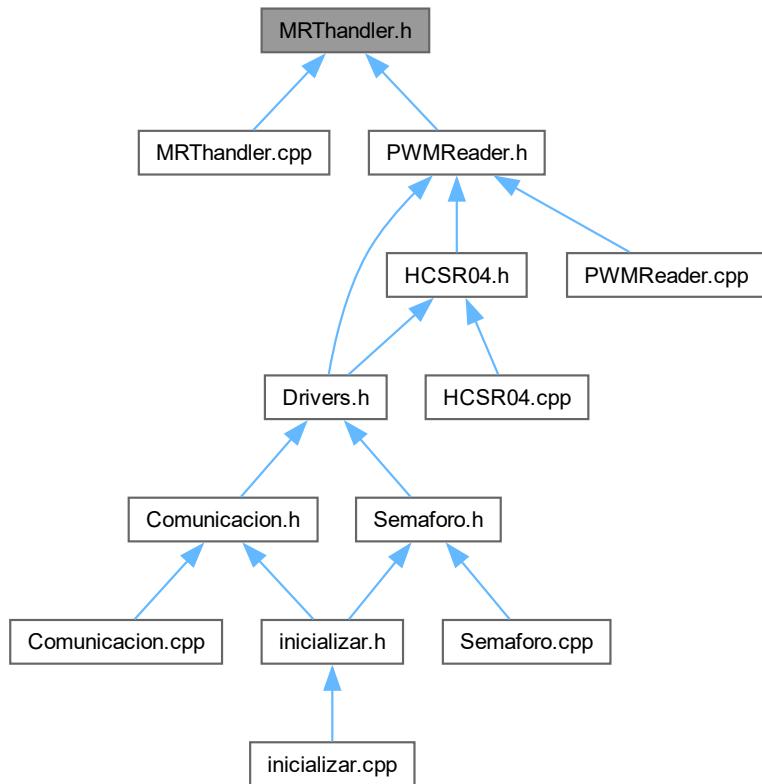
Handler del timer MRT.

```
#include "tipos.h"
#include "LPC845.h"
#include "vector"
```

Include dependency graph for MRThandler.h:



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



Classes

- class [MRThandler](#)

Clase del objeto `MRThandler` El objeto `MRThandler` debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.

Macros

- `#define MAX_MRT_CHANNEL 4`

Enumerations

- enum `type_MRT::MRT_timer_channels`
- enum `type_MRT::MRT_MODES`

Variables

- `std::vector<MRThandler * > g_MRThandler`

8.88.1 Detailed Description

Handler del timer MRT.

Date

2 sep. 2022

Version

1.0

Author

Técnico. Martinez Agustin (masteragus365@gmail.com)

8.88.2 Macro Definition Documentation

8.88.2.1 MAX_MRT_CHANNEL

`#define MAX_MRT_CHANNEL 4`

Cantidad de canales del MRT

8.88.3 Enumeration Type Documentation

8.88.3.1 MRT_MODES

`enum type_MRT::MRT_MODES`

Modos del MRT

8.88.3.2 MRT_timer_channels

`enum type_MRT::MRT_timer_channels`

Canales del MRT

8.89 MRThandler.h

[Go to the documentation of this file.](#)

```
00001 //*****
00012 //*****
00013 *** MODULO
00014 ****
00015 #ifndef MRTHANDLER_H_
00016 #define MRTHANDLER_H_
```

```

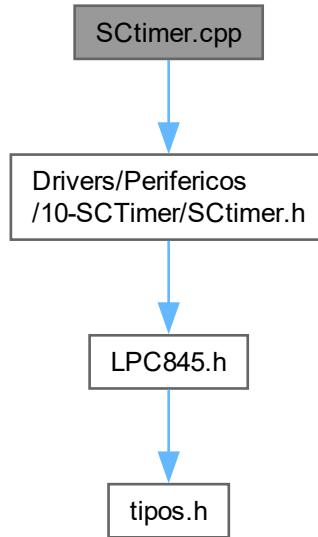
00020
00021  **** INCLUDES
00022
00023 #include "tipos.h"
00024 #include "LPC845.h"
00025 #include "vector"
00026
00027  *** DEFINES GLOBALES
00028
00029 #if defined (__cplusplus)
00030     extern "C" {
00031     void MRT_IRQHandler(void);
00032     }
00033 #endif
00034
00035  *** MACROS GLOBALES
00036
00037 namespace type_MRT
00038 {
00039     typedef enum { CHANNEL_0 = 0, CHANNEL_1, CHANNEL_2, CHANNEL_3 } MRT_timer_channels;
00040     typedef enum { REPEAT = 0, ONE_SHOT, ONE_SHOT_BUS, COUNTER } MRT_MODES ;
00041     #define MAX_MRT_CHANNEL      4
00042 }
00043 using namespace type_MRT;
00044
00045  *** TIPO DE DATOS GLOBALES
00046
00047  *** VARIABLES GLOBALES
00048
00049
00050
00051
00052  *** IMPLANTACION DE LA CLASE
00053
00054
00055
00056
00057
00058 class MRTHandler
00059 {
00060     private:
00061         static bool m_first[MAX_MRT_CHANNEL];
00062     public:
00063         const MRT_timer_channels m_timer_channel;
00064     public:
00065         MRTHandler( MRT_timer_channels _timer_number , MRT_MODES mode );
00066         virtual ~MRTHandler() = default;
00067
00068         virtual void    MRTHandler ( void ) = 0;
00069         void          EneableInterrupt ( void );
00070         void          DisableInterrupt ( void );
00071
00072     protected:
00073         void          MRT_reset_time ( void );
00074         uint32_t     MRT_get_time ( void );
00075     private:
00076         void          MRT_Inicializar ( MRT_timer_channels timer , MRT_MODES mode );
00077         void          MRT_Reset ( void );
00078 }
00079
00080 extern std::vector <MRTHandler *> g_MRTHandler;
00081
00082 #endif /* MRTHANDLER_H_ */

```

8.90 SCTimer.cpp File Reference

Modulo de Salida autonoma temporizada.

```
#include <Drivers/Perifericos/10-SCTimer/SCtimer.h>
Include dependency graph for SCtimer.cpp:
```



8.90.1 Detailed Description

Modulo de Salida autonoma temporizada.

Date

7 oct. 2022

Version

1.0

Author

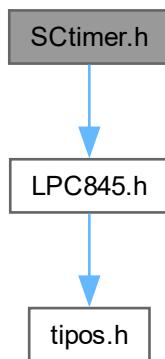
Técnico. Martinez Agustín (masteragus365@gmail.com)

8.91 SCtimer.h File Reference

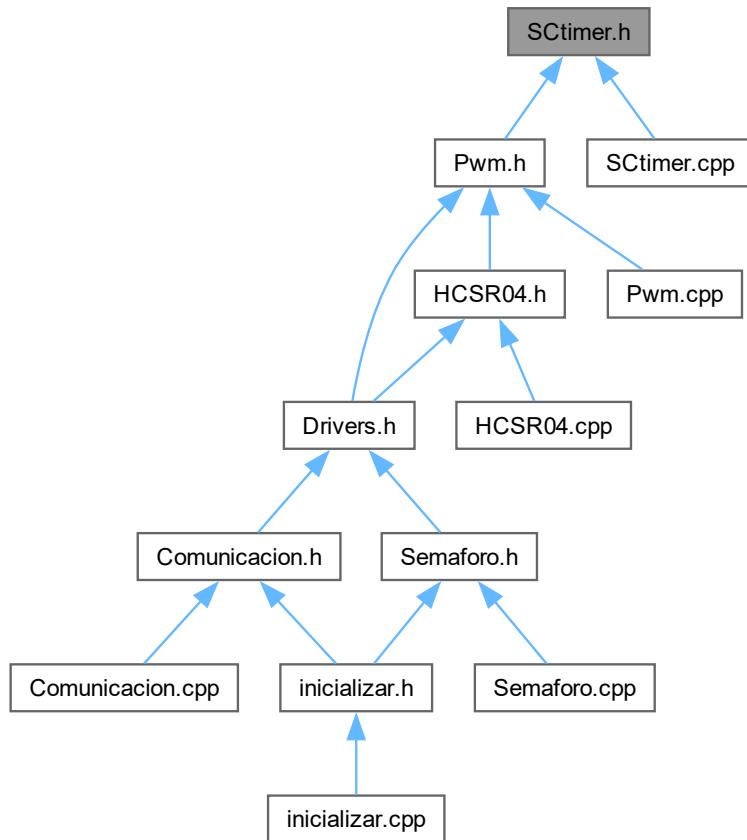
Modulo de Salida autónoma temporizada.

```
#include "LPC845.h"
```

Include dependency graph for SCtimer.h:



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



Classes

- class [SCtimer](#)

Clase del objeto SCtimer. El objeto SCtimer debe ser heredado por quienes deseen utilizar las interrupciones o funcionalidades del SCtimer.

8.91.1 Detailed Description

Modulo de Salida autonoma temporizada.

Date

7 oct. 2022

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.92 SCtimer.h

[Go to the documentation of this file.](#)

```

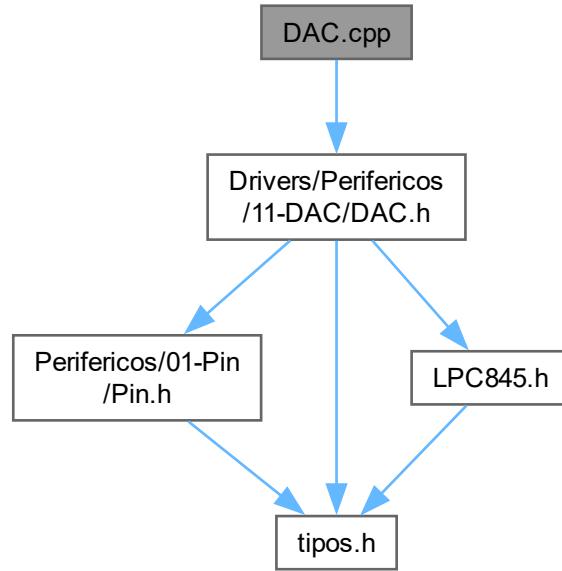
00001  /*****
00012  **** MODULO
00013  ***
00014  ****
00015 #ifndef SCTIMER_H_
00016 #define SCTIMER_H_
00020
00021  ***
00022  **** INCLUDES GLOBALES
00023 #include "LPC845.h"
00024
00025  ***
00026  **** DEFINES GLOBALES
00027
00028
00029  ***
00030  **** MACROS GLOBALES
00031
00032
00033  ***
00034  **** TIPO DE DATOS GLOBALES
00035
00036
00037  ***
00038  **** IMPLANTACION DE LA CLASE
00039
00040
00041
00042
00043
00044 class SCtimer {
00045 public:
00046     SCtimer ();
00047     void SetTime ( uint32_t time , uint32_t channel );
00048     void SetUnify ( bool a );
00049     void SetAutoLimit ( bool a );
00050     void SetSwitchMatrixSCTOUT ( uint8_t bit , uint8_t port , uint8_t out_number);
00051     void StartTimer ( void );
00052     void StopTimer ( void );
00053
00054     virtual ~SCtimer();
00055 };
00056
00057 #endif /* SCTIMER_H_ */

```

8.93 DAC.cpp File Reference

Objeto [DAC](#).

```
#include <Drivers/Perifericos/11-DAC/DAC.h>
Include dependency graph for DAC.cpp:
```



8.93.1 Detailed Description

Objeto [DAC](#).

Date

12 ene. 2023

Version

1.0

Author

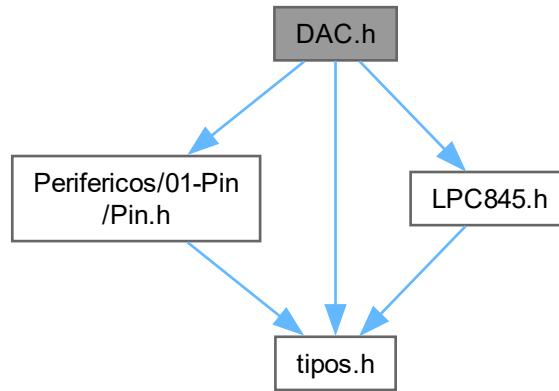
Técnico. Martinez Agustín (masteragus365@gmail.com)

8.94 DAC.h File Reference

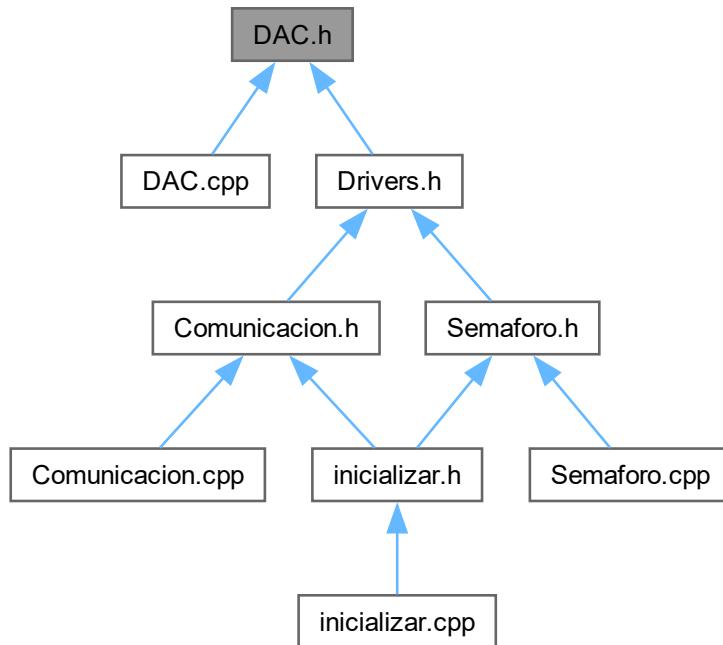
Objeto **DAC**.

```
#include <Perifericos/01-Pin/Pin.h>
#include "LPC845.h"
#include "tipos.h"
```

Include dependency graph for DAC.h:



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



Classes

- class [DAC](#)

Clase del objeto [DAC](#) FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

Macros

- `#define MAX_DAC_CHANNEL (2)`
- `#define MAX_DAC_VALUE (0x3FF)`

8.94.1 Detailed Description

Objeto [DAC](#).

Date

12 ene. 2023

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.94.2 Macro Definition Documentation

8.94.2.1 MAX_DAC_CHANNEL

```
#define MAX_DAC_CHANNEL (2)
```

Máximos canales del DAC

8.94.2.2 MAX_DAC_VALUE

```
#define MAX_DAC_VALUE (0x3FF)
```

Máximo valor del DAC por defecto

8.95 DAC.h

[Go to the documentation of this file.](#)

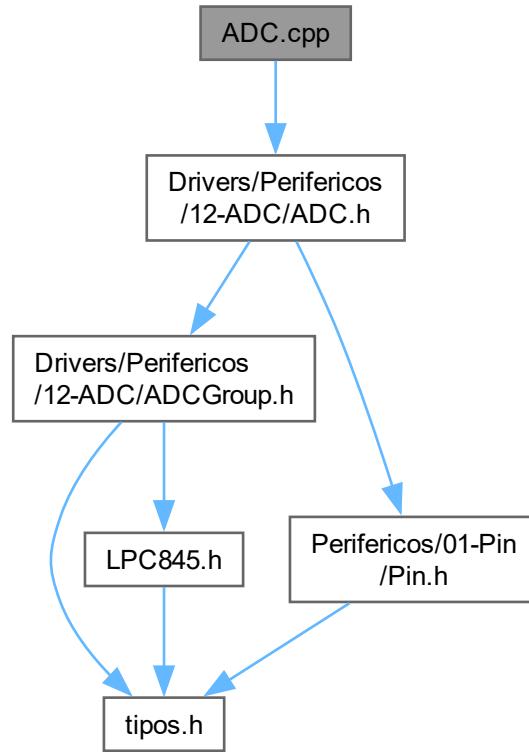
```
00001 /*****
00012 ****
00013 *** MODULO
00014 ****
00015 #ifndef DAC_H_
00016 #define DAC_H_
00020 ****
00021 *** INCLUDES GLOBALES
00022 ****
00023 #include <Perifericos/01-Pin/Pin.h>
00024 #include "LPC845.h"
00025 #include "tipos.h"
00026 ****
00027 *** DEFINES GLOBALES
00028 ****
00029 ****
00030 ****
00031 *** MACROS GLOBALES
00032 ****
00033 ****
00034 ****
00035 *** TIPO DE DATOS GLOBALES
00036 ****
00037 ****
00038 ****
00039 *** VARIABLES GLOBALES
00040 ****
00041 ****
00042 ****
00043 *** IMPLANTACION DE LA CLASE
00044 ****
00054 class DAC : protected Pin
00055 {
00056     public:
00058         #define MAX_DAC_CHANNEL      (2)
00060         #define MAX_DAC_VALUE        (0x3FF)
00062         typedef enum dac_channel { DAC_CHANNEL_0 = 0 , DAC_CHANNEL_1 = 1 } dac_channel;
00064         typedef enum dac_error { ERROR = -1 , OK = 0 } dac_error;
00065     private:
00068         enum OFFSET { SWM_DACOUT0 = 26 , SWM_DACOUT1 = 27 , SYS_DACOUT0 = 27 , SYS_DACOUT1 = 1 ,
00069             IOCON_DAC = 16 };
00070     private:
00071         const dac_channel m_dac_channel;
00072         uint32_t m_buffer;
00073         uint32_t m_max_range;
00074         dac_error m_error;
00075     public:
```

```
00077     DAC( dac_channel channel , uint32_t max_range = MAX_DAC_VALUE );
00078     DAC::dac_error Initialize ( void );
00079     void Set ( uint32_t val );
00080     uint32_t Get ( void ) const;
00081
00082     void SetMaxRange ( uint32_t max_range );
00083     uint32_t GetMaxRange ( void ) const;
00084
00085     DAC& operator= ( uint32_t val );
00086     bool operator== ( uint32_t val ) const;
00087     bool operator< ( uint32_t val ) const;
00088     bool operator<= ( uint32_t val ) const;
00089     bool operator> ( uint32_t val ) const;
00090     bool operator>= ( uint32_t val ) const;
00091     bool operator!= ( uint32_t val ) const;
00092
00093     virtual ~DAC();
00094
00095 private:
00096     void PowerDAC( void );
00097     void EnableClock ( void );
00098     void EnableSWM ( void );
00099     void EnableIOCONDAC( void );
00100
00101     void UnPowerDAC ( void );
00102     void DisableClock ( void );
00103     void DisableSWM ( void );
00104     void DisableIOCONDAC( void );
00105 };
00106
00107 #endif /* DAC_H_ */
```

8.96 ADC.cpp File Reference

Funciones miembro de [ADC](#).

```
#include <Drivers/Perifericos/12-ADC/ADC.h>
Include dependency graph for ADC.cpp:
```



Variables

- `uint8_t pin_index [MAX_ADC_CHANNELS] = { 7 , 6 , 14 , 23 , 22 , 21 , 20 , 19 , 18 , 17 , 13 , 4 }`

8.96.1 Detailed Description

Funciones miembro de [ADC](#).

Date

22 feb. 2023

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.96.2 Variable Documentation

8.96.2.1 pin_index

```
uint8_t pin_index[MAX_ADC_CHANNELS] = { 7 , 6 , 14 , 23 , 22 , 21 , 20 , 19 , 18 , 17 , 13 , 4 }
```

Pin index for [ADC](#) channels

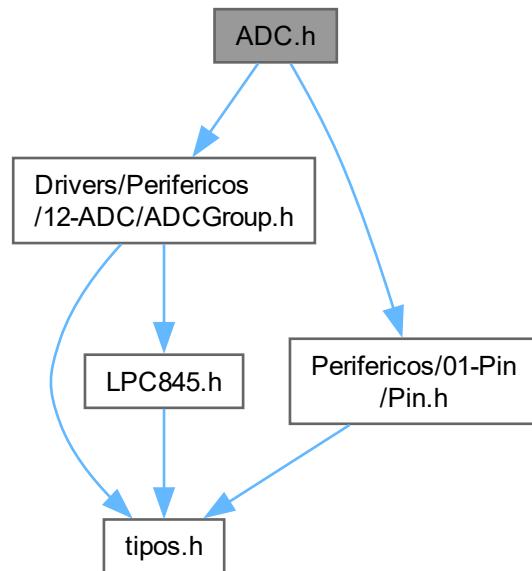
8.97 ADC.h File Reference

Modulo de [ADC](#).

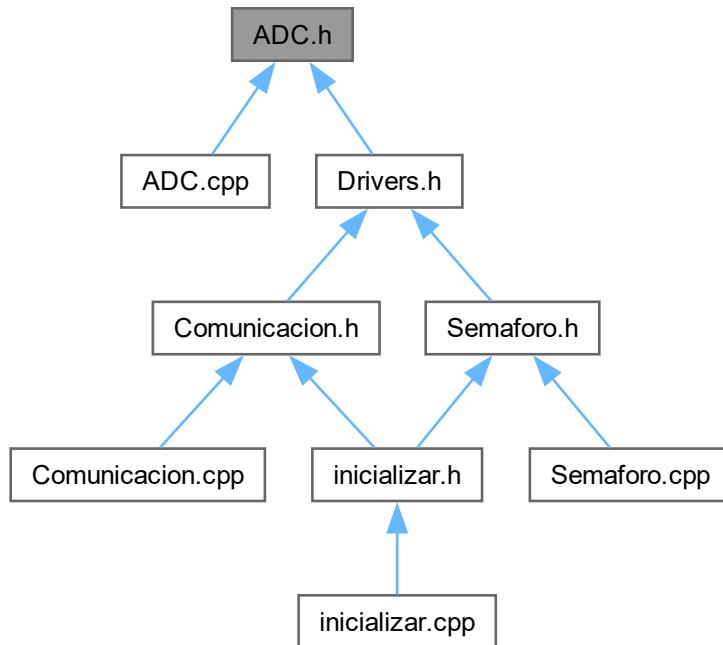
```
#include <Drivers/Perifericos/12-ADC/ADCGroup.h>
```

```
#include <Perifericos/01-Pin/Pin.h>
```

Include dependency graph for ADC.h:



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



Classes

- class [ADC](#)

Clase del objeto ADC FUNCIONAMIENTO: La clase [ADC](#) utiliza el `ADCGroup` para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del [ADC](#) existen. La configuración de conversión se realiza automáticamente con el primer objeto [ADC](#) creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

8.97.1 Detailed Description

Modulo de [ADC](#).

Date

22 feb. 2023

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.98 ADC.h

[Go to the documentation of this file.](#)

00001

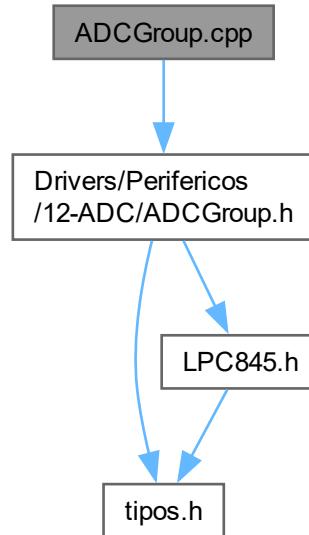
/*****

```
00012
00013     /***** MODULO *****
00014
00015 #ifndef ADC_H_
00016 #define ADC_H_
00020
00021     *** INCLUDES GLOBALES
00022
00023 #include <Drivers/Perifericos/12-ADC/ADCGroup.h>
00024 #include <Perifericos/01-Pin/Pin.h>
00025
00026     *** DEFINES GLOBALES
00027
00028
00029
00030     *** MACROS GLOBALES
00031
00032
00033
00034     *** TIPO DE DATOS GLOBALES
00035
00036
00037
00038     *** VARIABLES GLOBALES
00039
00040
00041
00042     *** IMPLANTACION DE LA CLASE
00043
00052 class ADC : public Pin
00053 {
00054     private:
00055         static ADC_Group* m_global_adc;
00056         uint8_t m_channel;
00057     public:
00058         ADC( uint8_t _channel , uint32_t _clk_freq = 0 , uint32_t _sample_rate = 0 );
00059         int32_t Get ( void );
00060         bool IsResultReady ( void );
00061         void Trigger ( void );
00062         void Initialize ( void );
00063         virtual ~ADC() = default;
00064     };
00065
00066 #endif /* ADC_H_ */
```

8.99 ADCGroup.cpp File Reference

Funciones miembro de ADCGroup.

```
#include <Drivers/Perifericos/12-ADC/ADCGroup.h>
Include dependency graph for ADCGroup.cpp:
```



Variables

- `ADC_Group * g_adc = nullptr`
vector de interrupciones ADC

8.99.1 Detailed Description

Funciones miembro de ADCGroup.
 Modulo de grupo de ADC.

Date

13 nov. 2022

Author

Federico

8.100 ADCGroup.h

```

00001
00002  /*****
00003
00004  ****
00005  *** MODULO
00006
00007  ****
00008
00009  #ifndef ADCGROUP_H_
00010
00011  *** INCLUDES GLOBALES
00012
00013
00014
00015
00016
00017
00018
00019
00020
  ****

```

```

00021 #include "tipos.h"
00022 #include "LPC845.h"
00023
00024 /***** DEFINES GLOBALES *****
00025
00026 #if defined (__cplusplus)
00027     extern "C" {
00028         void ADC_SEQA_IRQHandler ( void );
00029     }
00030     extern "C" {
00031         void ADC_SEQB_IRQHandler ( void );
00032     }
00033     extern "C" {
00034         void ADC_THCMP_IRQHandler ( void );
00035     }
00036     extern "C" {
00037         void ADC_OVR_IRQHandler ( void );
00038     }
00039     extern "C" {
00040         void UART4_IRQHandler ( void );
00041     }
00042 #endif
00043
00044 /***** MACROS GLOBALES *****
00045 *** MACROS GLOBALES
00046
00047 /***** TIPO DE DATOS GLOBALES *****
00048
00049 *** VARIABLES GLOBALES
00050
00051
00052 /***** IMPLANTACION DE LA CLASE *****
00053 *** VARIABLES GLOBALES
00054
00055
00056 /***** IMPLEMENTACION DE LA CLASE *****
00057 *** IMPLEMENTACION DE LA CLASE
00058
00059
00060 class ADC_Group
00061 {
00062     public:
00063         #define MAX_ADC_CHANNELS      (12)
00064
00065         typedef enum adc_isr { SEQA_ISR , SEQB_ISR , THCMP_ISR , OVR_ISR } adc_isr;
00066         typedef enum irq_source_inten{ ADC_SEQA_IRQ_INTEN = 0, ADC_SEQB_IRQ_INTEN = 1, ADC_OVR_IRQ_INTEN =
00067 2, INVALID_IRQ_INTEN = 0xFF} irq_source_inten;
00068         //Todas las de threshold no serán tenidas en cuenta
00069         typedef enum error_t { OK = 0 , ERROR = -1 } error_t;
00070
00071     private:
00072         #define CLOCKS_PER_SAMPLE    (25)
00073         #define CLK_500Khz           (500000)
00074         typedef enum irq_source_nvic{ADC_SEQA_IRQ = 16, ADC_SEQB_IRQ = 17, ADC_THCMP_IRQ = 18, ADC_OVR_IRQ
00075 = 19, INVALID_IRQ = 0xFF } irq_source_nvic;
00076         typedef enum trm_voltage_config { HIGH_VOLTAGE = 0 , LOW_VOLTAGE = 1 } trm_voltage_config;
00077         typedef enum conversion_mode { CONVERSION_INTERRUPT = 0 , SEQUENCE_INTERRUPT = 1} conversion_mode;
00078
00079         uint32_t      m_sample_rate;
00080         uint32_t      m_clk_freq;
00081         uint16_t      m_enabled_channels;
00082         uint32_t      m_result[MAX_ADC_CHANNELS];
00083         bool          m_result_ready[MAX_ADC_CHANNELS];
00084
00085     public:
00086         ADC_Group( uint32_t clk_freq , uint32_t sample_rate , bool init_channel0 = false );
00087         virtual ~ADC_Group() = default;
00088         void     Inicializar( void );
00089
00090         void     SetLowPowerMode   ( bool low_power );
00091         void     SetSampleRate    ( void );
00092
00093         void     EnableIrq     ( irq_source_inten irq );
00094         void     DisableIrq    ( irq_source_inten irq );
00095
00096         ADC_Group::error_t   InitADCChanel ( uint8_t channel ); //Adds a channel to the ADC
00097         ADC_Group::error_t   RemoveADCChanel ( uint8_t channel ); //Removes a channel to the ADC

```

```

00112     void             TriggerStartSeqA( void );
00113
00114     int32_t GetValue    ( uint8_t channel );
00115     bool  IsResultReady ( uint8_t channel ) const;
00116     void   Handler      ( adc_isr isr );
00117
00118 private:
00119     void   InitADC ();
00120     uint32_t CalculateDivisor ( uint32_t sample_rate );
00121     void   SetADCVoltage ( trm_voltage_config config );
00122     void   EnableADCPower ( void );
00123     void   EnableADCClock ( void );
00124     void   ConfigSWM ( uint8_t channel , bool enable );
00125     void   CalibrateADC ();
00126     void   ADCConfig ();
00127     void   AddChannel_to_SequenceA ( uint8_t channel );
00128     void   AddChannel_to_SequenceB ( uint8_t channel );
00129     void   RemoveChannelOfSequenceA ( uint8_t channel );
00130     void   RemoveChannelOfSequenceB ( uint8_t channel );
00131     void   SetSeqAMode ( conversion_mode mode );
00132     void   SetSeqBMode ( conversion_mode mode );
00133     void   EnableSeqA ( void );
00134     void   EnableSeqB ( void );
00135     void   DisableSeqA ( void );
00136     void   DisableSeqB ( void );
00137     void   SetUpSeqA ( void );
00138     void   EnableNvicADCInterrupt ( irq_source_nvic source );
00139     void   DisableNvicADCInterrupt ( irq_source_nvic source );
00140     irq_source_nvic GetNvicIrq ( irq_source_inten irq );
00141     void   handlerSeqA ( void );
00142     uint16_t   GetResult ( uint8_t channel );
00143 };
00144
00145 #endif /* ADCGROUP_H */

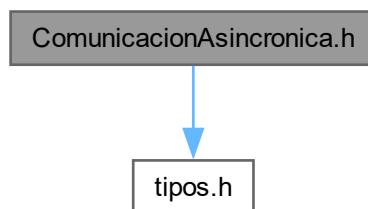
```

8.101 ComunicacionAsincronica.h File Reference

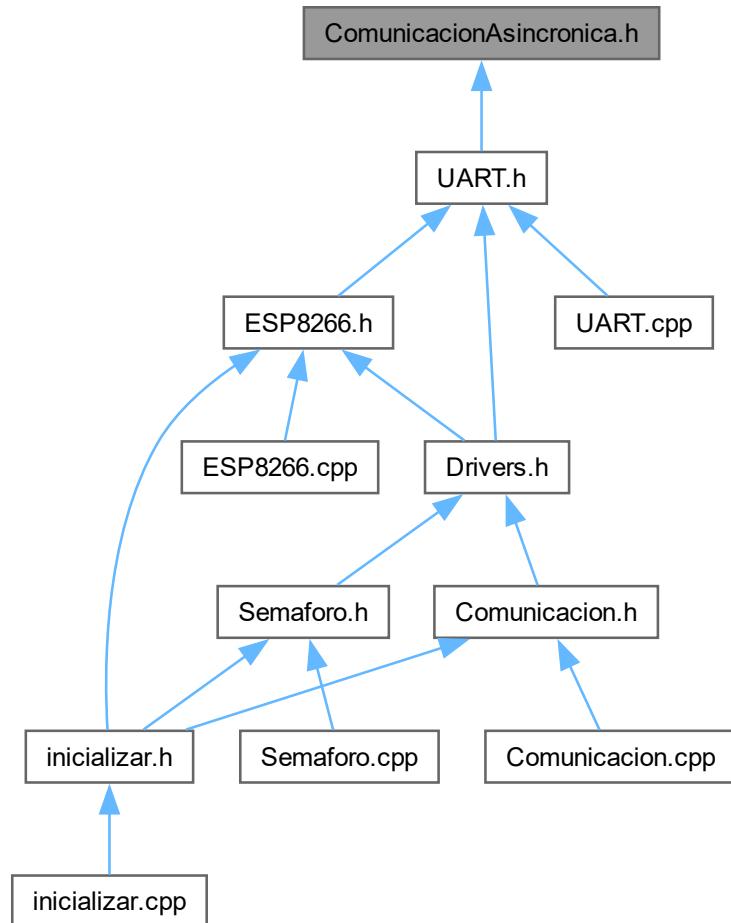
Objeto base para la creacion de comunicaciones asincrónicas.

#include "tipos.h"

Include dependency graph for ComunicacionAsincronica.h:



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



Classes

- class `ComunicacionAsincronica`

Clase del objeto `ComunicacionAsincronica` Clase abstracta pura para la generación de UART.

8.101.1 Detailed Description

Objeto base para la creacion de comunicaciones asincrónicas.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

8.102 ComunicacionAsincronica.h

[Go to the documentation of this file.](#)

```

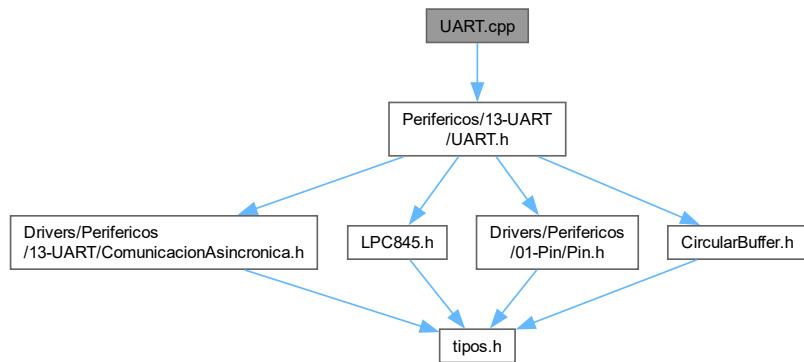
00001
00009 /*****
00010 ***
00011 *** MODULO
00012 ****
00013
00014 #ifndef COMUNICACIONASINCRONICA_H_
00015 #define COMUNICACIONASINCRONICA_H_
00019
00020 *** INCLUDES GLOBALES
00021
00022 #include "tipos.h"
00023
00024
00025 *** DEFINES GLOBALES
00026
00027
00028
00029 *** MACROS GLOBALES
00030
00031
00032
00033 *** TIPO DE DATOS GLOBALES
00034
00035
00036
00037 *** VARIABLES GLOBALES
00038
00039
00040
00041 *** IMPLANTACION DE UNA CLASE
00042
00048 class ComunicacionAsincronica
00049 {
00050     public:
00051         ComunicacionAsincronica() = default;
00052         virtual void Write ( const char * msg ) = 0;
00053         virtual void Write ( const void * msg , uint32_t n ) = 0;
00054         virtual bool Read ( char * msg , uint32_t n ) = 0;
00055         virtual void UART IRQHandler (void) = 0;
00056         virtual ~ComunicacionAsincronica() = default;
00057     //protected:
00058     // virtual void pushRx ( uint8_t dato ) = 0;           /**< Envia recepcion */
00059     // virtual uint8_t popRx (uint8_t * dato ) = 0;          /**< Devuelve recepcion */
00060     // virtual void pushTx ( uint8_t dato ) = 0;           /**< Envia transmision*/
00061     // virtual uint8_t popTx (uint8_t * dato ) = 0;          /**< Devuelve transmision */
00062 };
00063
00064 #endif /* COMUNICACIONASINCRONICA_H_ */

```

8.103 UART.cpp File Reference

Funciones miembro de [Uart](#).

```
#include <Perifericos/13-UART/UART.h>
Include dependency graph for UART.cpp:
```



Variables

- `ComunicacionAsincronica * g_usart [5]`

8.103.1 Detailed Description

Funciones miembro de [Uart](#).

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

8.103.2 Variable Documentation

8.103.2.1 g_usart

`ComunicacionAsincronica* g_usart [5]`

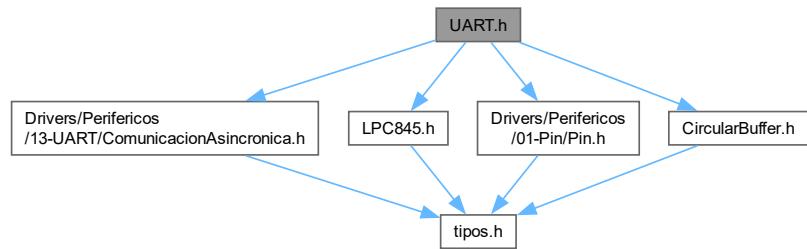
Vector de UART s

8.104 UART.h File Reference

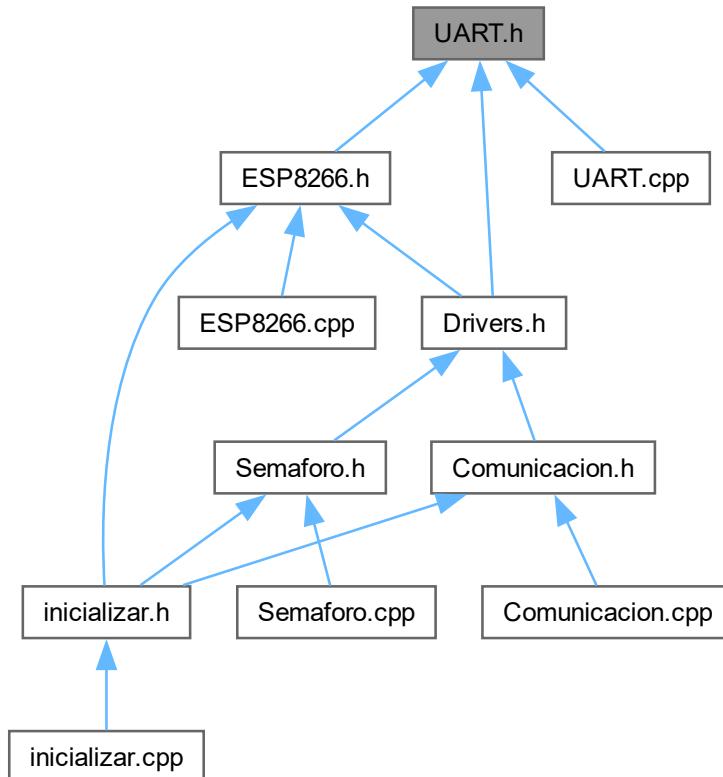
Modulo de Comunicacion [Uart](#).

```
#include <Drivers/Perifericos/13-UART/ComunicacionAsincronica.h>
#include "LPC845.h"
#include <Drivers/Perifericos/01-Pin/Pin.h>
#include <CircularBuffer.h>
```

Include dependency graph for UART.h:



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



8.104.1 Detailed Description

Modulo de Comunicacion [Uart](#).

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

8.105 UART.h

[Go to the documentation of this file.](#)

```
00001 //*****
00002 ****
00003 **** MODULO
00004 ****
00005 ****
00006 #ifndef UART_H_
00007 #define UART_H_
00008 ****
00009 **** INCLUDES GLOBALES
00010 ****
00011 ****
00012 ****
00013 #include <Drivers/Perifericos/13-UART/ComunicacionAsincronica.h>
00014 #include "LPC845.h"
00015 #include <Drivers/Perifericos/01-Pin/Pin.h>
00016 #include <CircularBuffer.h>
00017 ****
00018 **** DEFINES GLOBALES
00019 ****
00020 ****
00021 ****
00022 #if defined (__cplusplus)
00023     extern "C" {
00024         void UART0_IRQHandler ( void );
00025     }
00026     extern "C" {
00027         void UART1_IRQHandler ( void );
00028     }
00029     extern "C" {
00030         void UART2_IRQHandler ( void );
00031     }
00032     extern "C" {
00033         void UART3_IRQHandler ( void );
00034     }
00035     extern "C" {
00036         void UART4_IRQHandler ( void );
00037     }
00038     extern "C" {
00039         void UART5_IRQHandler ( void );
00040     }
00041     extern "C" {
00042         void USART_IRQHandler ( void );
00043     }
00044     extern "C" {
00045         void USART1_IRQHandler ( void );
00046     }
00047     *** MACROS GLOBALES
00048 ****
00049 ****
00050 ****
00051 **** TIPO DE DATOS GLOBALES
00052 ****
00053 ****
00054 ****
00055 **** VARIABLES GLOBALES
00056 ****
00057 ****
00058 ****
00059 **** IMPLANTACION DE UNA CLASE
00060 ****
00061 ****
00062 class UART : public ComunicacionAsincronica
00063 {
00064     public:
00065         typedef enum { NoParidad , par = 2, impar} paridad_t;
00066         typedef enum { siete_bits , ocho_bits } bits_de_datos;
00067         #define PORT_TX_USB      Pin::port0
00068         #define PIN_TX_USB       25
00069         #define PORT_RX_USB      Pin::port0
00070         #define PIN_RX_USB       24
00071         #define USART_USB        USART0
00072     private:
00073 
```

```

00083     const Pin m_tx;
00084     const Pin m_rx;
00085     USART_Type* m_usart;
00086     CircularBuffer<uint8_t> m_bufferRx;
00087     CircularBuffer<uint8_t> m_bufferTx;
00088     bool m_flagTx;
00089
00090     public:
00091         UART( Pin::port_t portTx , uint8_t pinTx , Pin::port_t portRx , uint8_t pinRx ,
00092               USART_Type * usart , uint32_t baudrate , bits_de_datos BitsDeDatos , paridad_t
00093               paridad ,
00094               uint32_t maxRx , uint32_t maxTx );
00095         void Write ( const char * msg) override;
00096         void Write ( const void * msg , uint32_t n ) override;
00097         bool Read ( char * msg , uint32_t n ) override;
00098         void Read (char* n);
00099         void SetBaudRate ( uint32_t baudrate );
00100
00101     private:
00102         void EnableSW ( void );
00103         void EnableClock ( void );
00104         void Config ( uint32_t baudrate , bits_de_datos BitsDeDatos , paridad_t paridad );
00105         void UART_IRQHandler ( void ) override;
00106
00107 //     void pushRx ( uint8_t dato ) override;
00108 //     uint8_t popRx ( uint8_t * dato ) override;
00109 //     void pushTx ( uint8_t dato ) override;
00110 //     uint8_t popTx ( uint8_t * dato ) override;
00111         void EnableInterrupt ( void );
00112         void DisableInterrupt ( void );
00113 };
00114
00115 #endif /* UART_H_ */

```

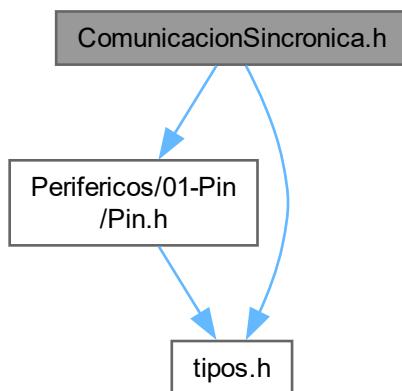
8.106 ComunicacionSincronica.h File Reference

Objeto base para la creacion de comunicaciones sincrónicas.

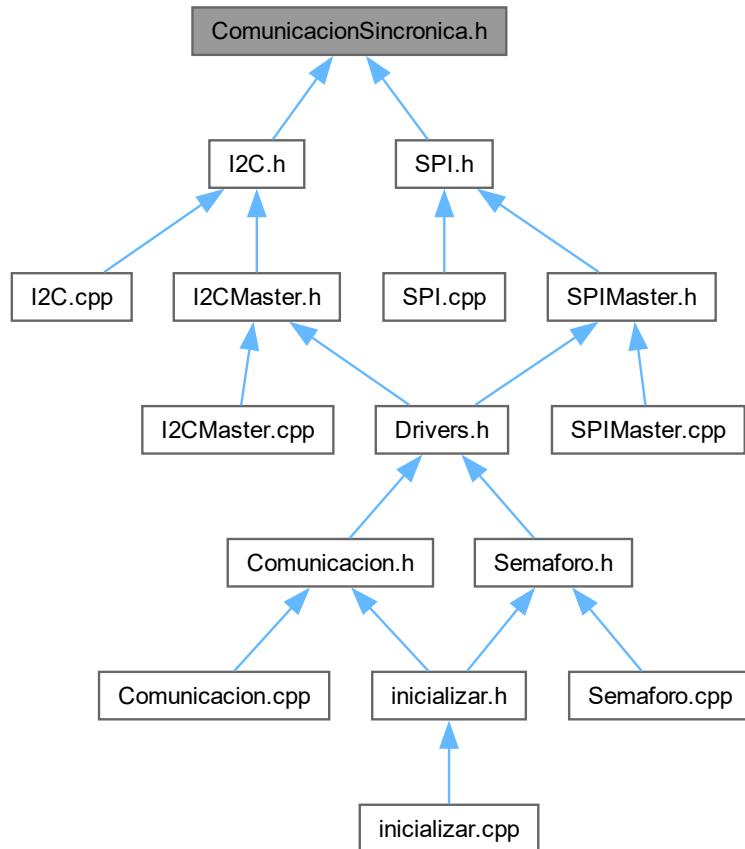
```
#include <Perifericos/01-Pin/Pin.h>
```

```
#include "tipos.h"
```

Include dependency graph for ComunicacionSincronica.h:



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



Classes

- class [ComunicacionSincronica](#)

Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI.

8.106.1 Detailed Description

Objeto base para la creacion de comunicaciones sincrónicas.

Date

3 ene. 2025

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.107 ComunicacionSincronica.h

[Go to the documentation of this file.](#)

```

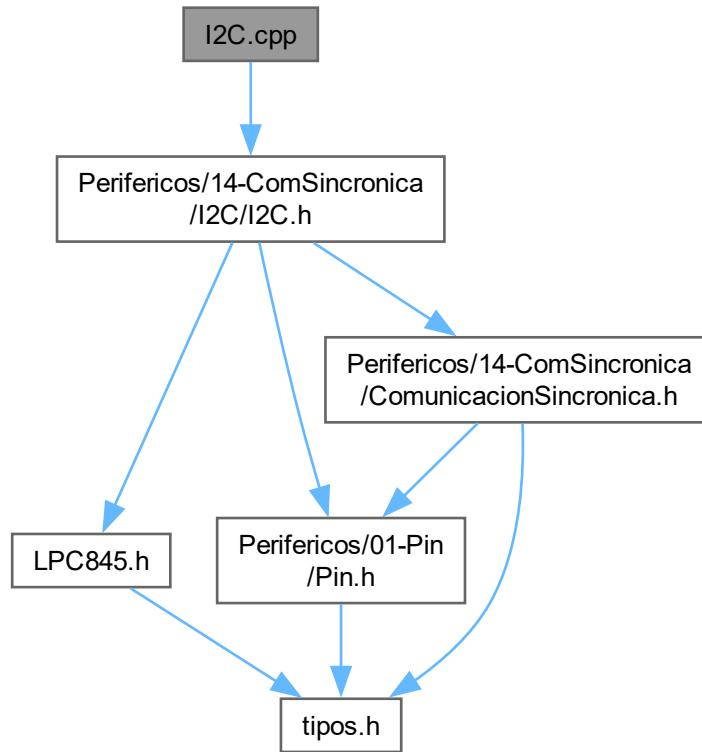
00001
00012 /*****
00013 *** MODULO
00014 ****
00015
00016 #ifndef COMUNICACIONSINCRONICA_H_
00017 #define COMUNICACIONSINCRONICA_H_
00021
00022 *** INCLUDES GLOBALES
00023
00024 #include <Perifericos/01-Pin/Pin.h>
00025 #include "tipos.h"
00026
00027
00028 *** DEFINES GLOBALES
00029
00030
00031
00032 *** MACROS GLOBALES
00033
00034
00035
00036 *** TIPO DE DATOS GLOBALES
00037
00038
00039
00040 *** VARIABLES GLOBALES
00041
00042
00043
00044 *** IMPLANTACION DE UNA CLASE
00045
00051 class ComunicacionSincronica
00052 {
00053     protected:
00054         const Pin* m_scl;
00055
00056     public:
00057         ComunicacionSincronica() = default;
00058         virtual void Write ( uint8_t data ) = 0;
00059         virtual int8_t Read ( uint8_t* data ) = 0;
00060         virtual ~ComunicacionSincronica() = default;
00061 };
00062
00063 #endif /* COMUNICACIONSINCRONICA_H_ */

```

8.108 I2C.cpp File Reference

Funciones miembro de [I2C](#).

```
#include <Perifericos/14-ComSincronica/I2C/I2C.h>
Include dependency graph for I2C.cpp:
```



Macros

- #define MAX_IC2 4

Variables

- I2C * g_i2c [MAX_IC2]

8.108.1 Detailed Description

Funciones miembro de `I2C`.

Date

3 ene. 2025

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.108.2 Macro Definition Documentation

8.108.2.1 MAX_IC2

```
#define MAX_IC2 4
```

Cantidad maxima de I2C que presenta el microcontrolador.

8.108.3 Variable Documentation

8.108.3.1 g_i2c

```
I2C* g_i2c[MAX_IC2]
```

Vector de UART s

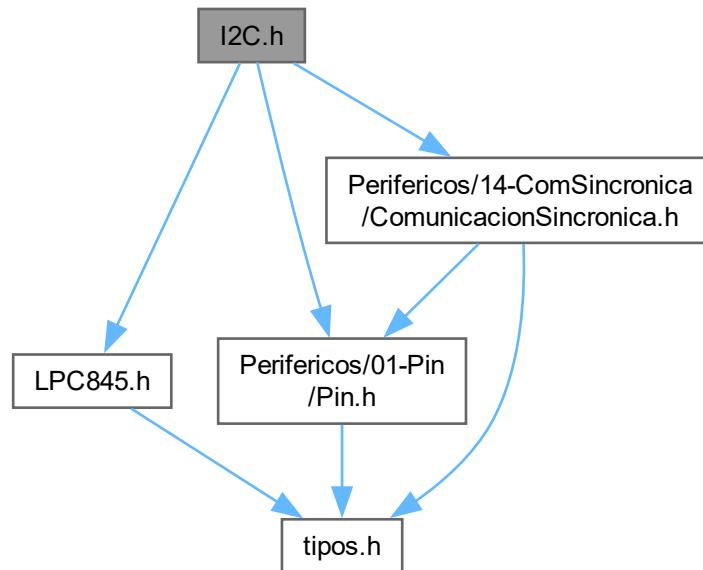
8.109 I2C.h File Reference

Modulo de comunicacion I2C.

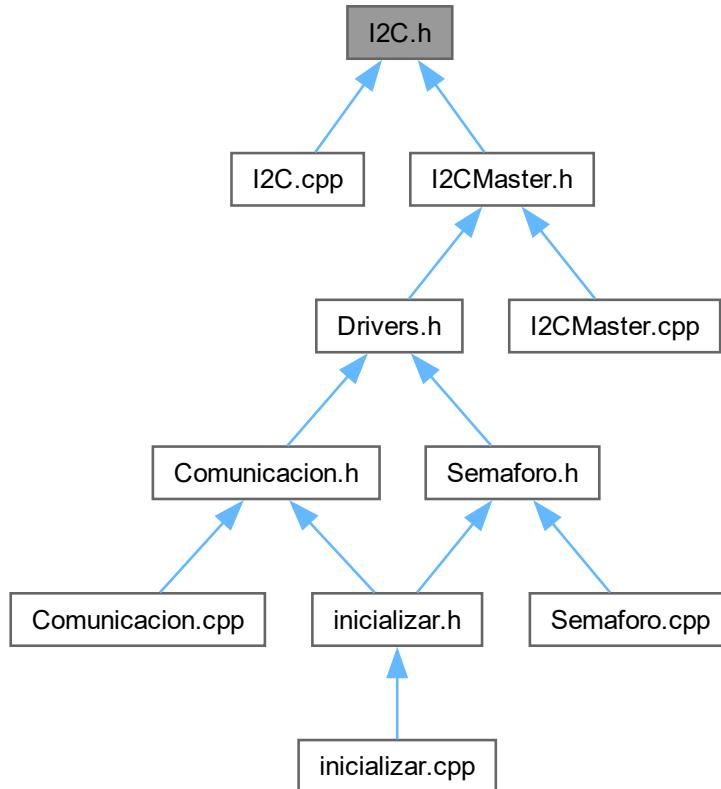
```
#include <Perifericos/01-Pin/Pin.h>
#include "LPC845.h"
```

```
#include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
```

Include dependency graph for I2C.h:



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



Classes

- class [I2C](#)

Clase del objeto I2C. El objeto I2C genera una comunicación sincrónica de tipo I2C. Posee las funciones básicas como start, stop, write y read.

Macros

- #define [I2C_MAX_FREQ](#) 400

8.109.1 Detailed Description

Modulo de comunicacion [I2C](#).

USO: Se crea y se inicializa. Posee 2 modos: Master y Slave. MASTER: Start(): Inicia la comunicacion con Start en accion de read/write. Solo puede llamarse a start si el [I2C](#) está en idle o pendiente de lectura.

Read() y Write(): Se escribe y se lee de a 1 byte con las funciones Read y Write. Previamente se debe hacer el start correspondiente para funcionar. Solo puede llamarse a Read y Write si el [I2C](#) esta en tx_ready o rx_data.

Stop(): Al finalizar se le da a stop y se termina la conversación con el dispositivo. Solo puede llamarse a Stop si el [I2C](#) esta en modos distintos al busy.

SLAVE: El slave nunca inicia la comunicacion. Debe revisar si el mismo se encuentra pendiente (estados slvst_\leftarrow addr, slvst_tx o slvst_rx)

ACKAddr(): Para iniciar la comunicacion, se debe reconocer que el address traído es el correspondiente al dispositivo.

Read(): Si llegó algo al slave, se leera. Solo funcionara si el I2C esta en slvst_rx. Debe realizarse un ACK() para continuar.

Write(): Si llego una peticion de escritura al slave, se escribe. Solo funcionara si el I2C esta en slvst_tx.

ACK(): Cuando se recibe un dato, se debe reconocer la llegada del mismo.

Date

3 ene. 2025

Version

1.0

Author

Técnico. Martinez Agustin (masteragus365@gmail.com)

8.109.2 Macro Definition Documentation

8.109.2.1 I2C_MAX_FREQ

#define I2C_MAX_FREQ 400

Maxima frecuencia del I2C. Solo valida para I2C1 a I2C3

8.110 I2C.h

[Go to the documentation of this file.](#)

```

00001  ****
00034
00035  ***
00036  *** MODULO
00037
00038 #ifndef I2C_H_
00039 #define I2C_H_
00040
00044
00045
00046  ***
00047  *** INCLUDES GLOBALES
00048 #include <Perifericos/01-Pin/Pin.h>
00049 #include "LPC845.h"
00050 #include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
00051
00052  ***
00053  *** DEFINES GLOBALES
00054
00055 #if defined (__cplusplus)
00056     extern "C" {
00057         void I2C0_IRQHandler ( void );
00058     }
00059     extern "C" {
00060         void I2C1_IRQHandler ( void );
00061     }
00062     extern "C" {
00063         void I2C2_IRQHandler ( void );
00064     }
00065     extern "C" {
00066         void I2C3_IRQHandler ( void );
00067     }
00068 #endif
00069  ***
00070  *** MACROS GLOBALES
00071
00072
00073  ***

```

```

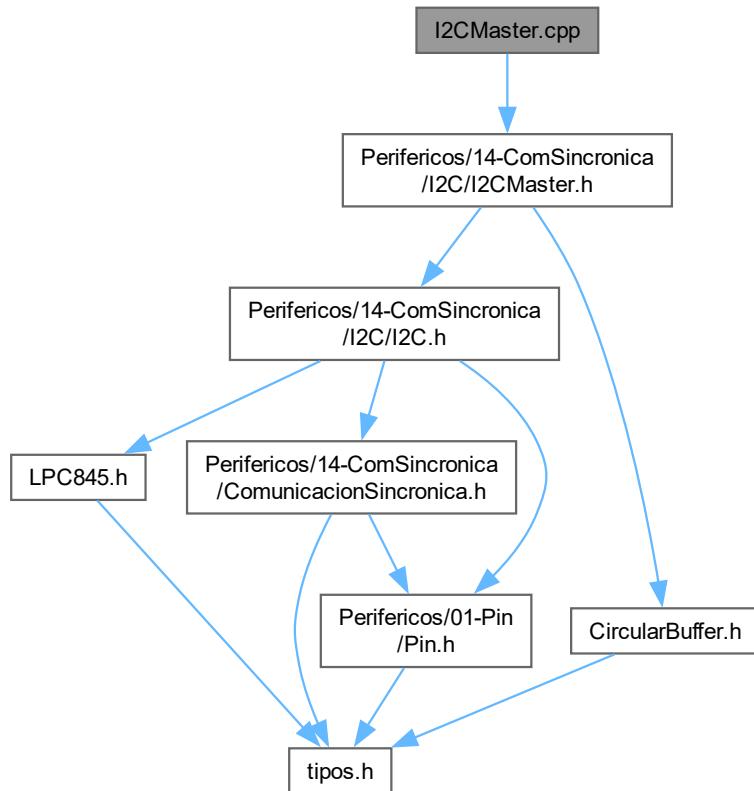
00074
00075
00076
00077 /*****
00078 *** VARIABLES GLOBALES
00079
00080 ****
00081 *** IMPLANTACION DE UNA CLASE
00082
00083 ****
00084
00085 class I2C: ComunicacionSincronica
00086 { //La I2C1 a I2C3 no van a mas de 400khz
00087 public:
00088 #define I2C_MAX_FREQ 400
00089     typedef enum {master = 1 , slave = 2} I2C_role_t;
00090     typedef enum {write = 0 , read = 1 } I2C_action_t;
00091     typedef enum {idle = 0 , rx_data = 1 , tx_ready , NACK_addr , NACK_tx ,
00092                 slvst_addr = 0 , slvst_rx , slvst_tx ,
00093                 busy = 10 , timeout} I2C_states_t;
00094
00095 private:
00096     I2C_Type* m_I2C_register ;
00097     const Pin* m_sda;
00098
00099     I2C_role_t m_role ;
00100     const uint8_t m_slv_addr;
00101
00102     static uint8_t m_cant_created;
00103
00104 public:
00105     I2C ( I2C_Type* I2C_register , Pin* sda , Pin* scl , I2C_role_t mode =
00106           master, uint8_t slv_addr = 0 );
00107     void Initialize ( uint32_t clk_freq );
00108     void EnableInterrupt ( void );
00109     void DisableInterrupt ( void );
00110
00111     void Start ( uint8_t addr , I2C_action_t action );
00112     void Stop ( void );
00113     void Write ( uint8_t data ) override;
00114     I2C& operator= ( uint8_t data ); //Sobrecarga de escritura
00115     int8_t Read ( uint8_t* data , bool continue_reading );
00116     int8_t Read ( uint8_t* data ) override;
00117     void Continue ( void );
00118
00119     void ACK ( bool a );
00120     bool ACKaddr ( void );
00121     I2C_states_t GetState ( void );
00122
00123     void SetTimeOut ( uint32_t clk_cycles );
00124
00125     virtual void I2C_IRQHandler ( void ) { }
00126     virtual ~I2C();
00127
00128 private:
00129     void config ( uint8_t& register_number );
00130     void EnableSWM ( void );
00131     void ConfigClock ( uint8_t& clk_offset , uint8_t& rst_offset );
00132     void configBaudRate ( uint32_t clk_freq );
00133 };
00134
00135 #endif /* I2C_H_ */

```

8.111 I2CMaster.cpp File Reference

Modulo para master I2C.

```
#include <Perifericos/14-ComSincronica/I2C/I2CMaster.h>
Include dependency graph for I2CMaster.cpp:
```



8.111.1 Detailed Description

Modulo para master [I2C](#).

Date

3 ene. 2025

Version

1.0

Author

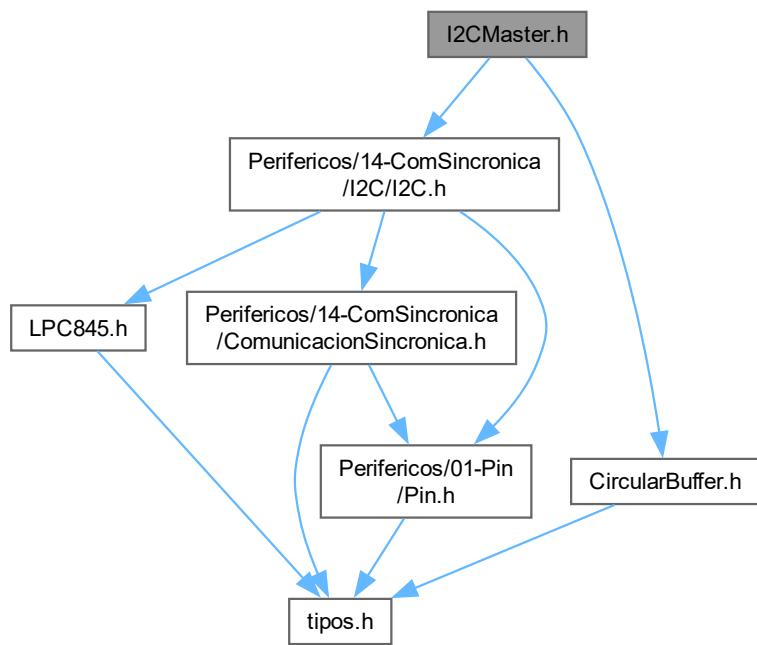
Técnico. Martínez Agustín (masteragus365@gmail.com)

8.112 I2CMaster.h File Reference

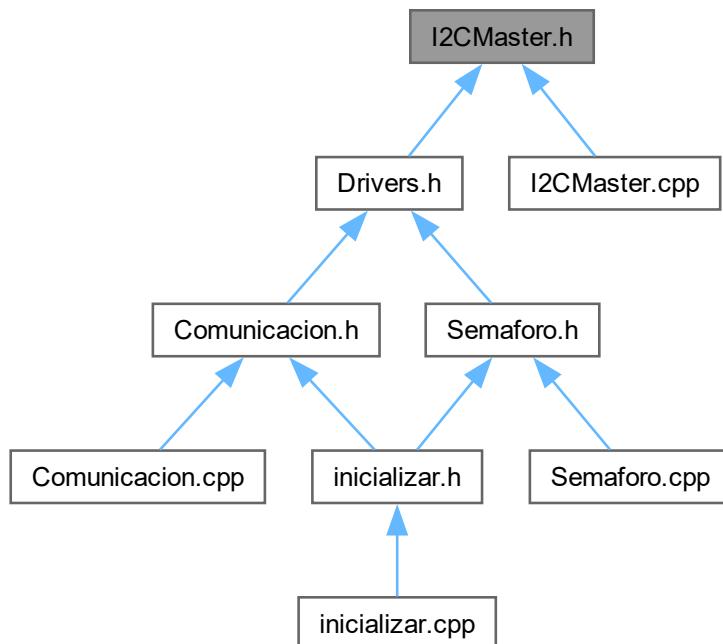
Modulo para master [I2C](#).

```
#include <Perifericos/14-ComSincronica/I2C/I2C.h>
#include <CircularBuffer.h>
```

Include dependency graph for I2CMaster.h:



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



Classes

- class **I2CMaster**

*Clase del objeto **I2CMaster**. El objeto **I2CMaster** genera una comunicación tipo master de **I2C** utilizando buffers de recepción y transmisión con interrupciones.*

8.112.1 Detailed Description

Modulo para master **I2C**.

USO: Se crea y se inicializa. Se envia información con Write() y se pide lectura con RequestRead(). Solo se pueden ejecutar si el **I2C** esta en idle. Write(): Escribe el string en el buffer y lo envia continuamente por interrupcion. RequestRead(): Pide una lectura continua de n bytes que se guardan en un buffer por interrupcion. Read(): Lee el buffer de recepcion. Puede leerse en cualquier momento y por partes. isIdle(): Indica si el **I2C** esta en reposo. Solo cuando esto sea cierto se podra leer y escribir al slave.

Aun no soporta timeout recibidos. **I2C** posee los metodos para agregar dicha función al Handler.

Date

3 ene. 2025

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.113 I2CMaster.h

[Go to the documentation of this file.](#)

```

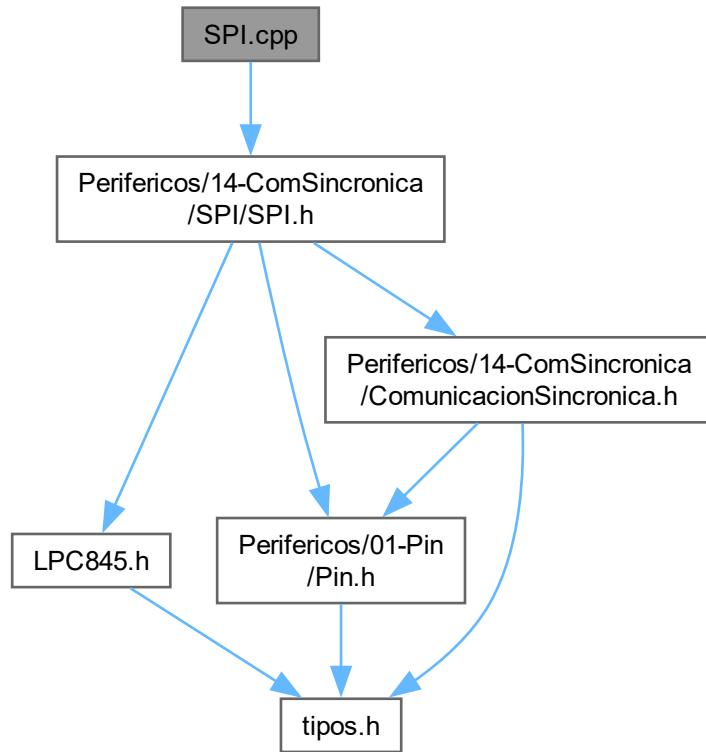
00001  /*
00019
00020  **** MODULO
00022
00023 #ifndef I2CMASTER_H_
00024 #define I2CMASTER_H_
00025
00029
00030
00031  *** INCLUDES GLOBALES
00032
00033 #include <Perifericos/14-ComSincronica/I2C/I2C.h>
00034 #include <CircularBuffer.h>
00035
00036  *** DEFINES GLOBALES
00037
00038
00039
00040  *** MACROS GLOBALES
00041
00042
00043
00044  *** TIPO DE DATOS GLOBALES
00045
00046
00047
00048  *** VARIABLES GLOBALES
00049
00050
00051
00052  *** IMPLANTACION DE UNA CLASE
00053
00054
00055
00056
00057
00058
00059 class I2CMaster : protected I2C
00060 {
00061     private:
00062         CircularBuffer<uint8_t> m_bufferRx;
00063         CircularBuffer<uint8_t> m_bufferTx;
00064         uint32_t m_cant_rw;
00065         I2C_action_t m_action;
00066
00067     public:
00068         I2CMaster (I2C_Type* I2C_register , Pin* sda , Pin* scl , uint32_t maxRx = 15,
00069             uint32_t maxTx = 15);
00070         void Initialize ( uint32_t clk_freq );
00071         void Write ( uint8_t addr, const char * msg );
00072         void Write ( uint8_t addr, const void * msg , uint32_t n );
00073         void* RequestRead ( uint8_t addr, uint32_t cant_read = 1 );
00074         void* Read ( void * msg , uint32_t n );
00075         bool hasData ( void );
00076         bool isIdle ( void );
00077         virtual ~I2CMaster();
00078
00079     private:
00080         void I2C_IRQHandler ( void ) override;
00081
00082 #endif /* I2CMASTER_H_ */

```

8.114 SPI.cpp File Reference

Funciones miembro de [SPI](#).

```
#include <Perifericos/14-ComSincronica/SPI/SPI.h>
Include dependency graph for SPI.cpp:
```



Macros

- #define MAX_SPI 2

Variables

- SPI * g_spi [MAX_SPI]

8.114.1 Detailed Description

Funciones miembro de SPI.

Date

3 ene. 2025

Version

1.0

Author

Técnico. Martinez Agustín (masteragus365@gmail.com)

8.114.2 Macro Definition Documentation

8.114.2.1 MAX_SPI

```
#define MAX_SPI 2
```

Cantidad maxima de I₂C que presenta el microcontrolador.

8.114.3 Variable Documentation

8.114.3.1 g_spi

```
SPI* g_spi[MAX_SPI]
```

Vector de UART s

8.115 SPI.h File Reference

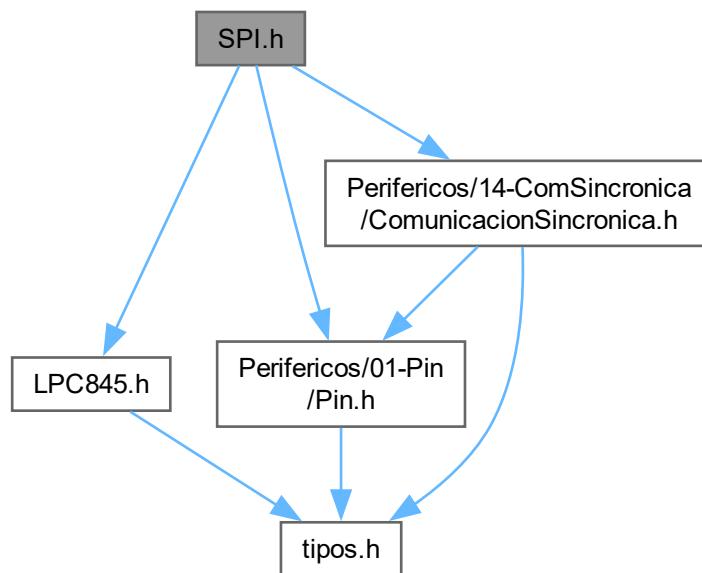
Modulo de comunicacion SPI.

```
#include <Perifericos/01-Pin/Pin.h>
```

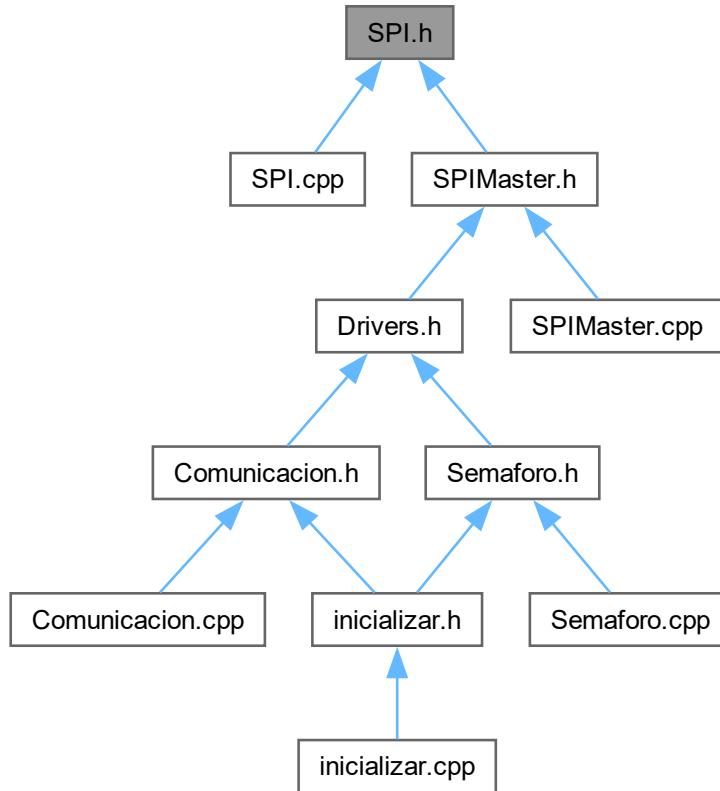
```
#include "LPC845.h"
```

```
#include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
```

Include dependency graph for SPI.h:



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



Classes

- class [SPI](#)

Clase del objeto SPI.

Macros

- `#define MAX_SSEL_SPI0 4`
- `#define MAX_SSEL_SPI1 2`

8.115.1 Detailed Description

Modulo de comunicacion [SPI](#).

USO: Se crea, se inicializa y se agregan los slaves correspondientes `AddSSEL()`. Posee 2 modos: Master y Slave.
 MASTER: `SetSSEL()`: Setea el Slave correspondiente. `SetEOF()` o `SetEOT()`: Configura el dato a transmitir como unico o como el primero de una cadena. Se debe setear si se desean enviar varios de corrido. `isTxReady()`: Verifica que se puede iniciar una transmision. `Write()`: Escribe el byte.

`isRxReady()`: Si luego de escribir llego un dato, se verifica con esta funcion. `Read()`: Luego de una escritura, se lee lo que llego.

SLAVE: Not fully implemented. Necesita `isSSELAsserted(slv)`

Date

3 ene. 2025

Version

1.0

AuthorTécnico. Martinez Agustin (masteragus365@gmail.com)**8.115.2 Macro Definition Documentation****8.115.2.1 MAX_SSEL_SPI0**

```
#define MAX_SSEL_SPI0 4
Maxima cantidad de chip selects del SPI0
```

8.115.2.2 MAX_SSEL_SPI1

```
#define MAX_SSEL_SPI1 2
Maxima cantidad de chip selects del SPI1
```

8.116 SPI.h

[Go to the documentation of this file.](#)

```
00001 /*****
00025 ****
00026 *** MODULO
00027 ****
00028 #ifndef SPI_H_
00029 #define SPI_H_
00030
00034 ****
00035 *** INCLUDES GLOBALES
00036 ****
00037 #include <Perifericos/01-Pin/Pin.h>
00038 #include "LPC845.h"
00039 #include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
00040 ****
00041 *** DEFINES GLOBALES
00042 ****
00043 #if defined (__cplusplus)
00044     extern "C" {
00045         void SPI0_IRQHandler ( void );
00046     }
00047     extern "C" {
00048         void SPI1_IRQHandler ( void );
00049     }
00050 #endif
00051 ****
00052 *** MACROS GLOBALES
00053 ****
00054
00055 ****
00056 *** TIPO DE DATOS GLOBALES
00057 ****
00058
00059 ****
00060 *** VARIABLES GLOBALES
00061 ****
```

```

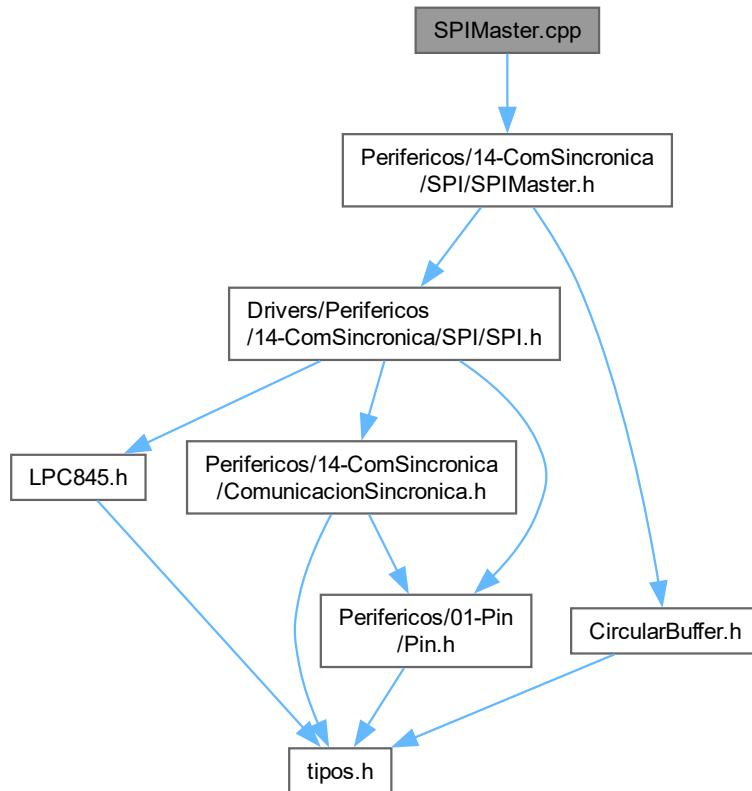
00062
00063
00064     *** IMPLANTACION DE UNA CLASE
00065
00066
00067
00068
00069
00070
00071
00072 class SPI: ComunicacionSincronica
00073 {
00074     public:
00075         typedef enum {master = 1 , slave = 2} SPI_role_t;
00076         typedef enum {
00077             SPI_MODE_0 = 0 ,
00078             SPI_MODE_1 ,
00079             SPI_MODE_2 ,
00080             SPI_MODE_3
00081         } SPI_mode_t;
00082
00083 #define MAX_SSEL_SPIO          4
00084 #define MAX_SSEL_SPI1          2
00085
00086
00087
00088
00089     private:
00090     static const uint8_t SPI0_PINASSIGN_IDX[] ;
00091     static const uint8_t SPI0_SSEL_OFFSET_IDX[] ;
00092     static const uint8_t SPI1_PINASSIGN_IDX[] ;
00093     static const uint8_t SPI1_SSEL_OFFSET_IDX[] ;
00094
00095
00096     protected:
00097         SPI_Type* m_SPI_register;
00098
00099     private:
00100         const Pin* m_miso;
00101         const Pin* m_mosi;
00102         Pin* m_ssel[MAX_SSEL_SPI0] = {nullptr};
00103         SPI_role_t m_role;
00104
00105     public:
00106         SPI           ( SPI_Type* SPI_register , Pin* miso , Pin* mosi , Pin* clk , SPI_role_t
00107             role = master );
00108         void Initialize      ( uint32_t clk_freq , SPI_mode_t mode = SPI_MODE_0 );
00109         void EnableInterrupt ( void );
00110         void DisableInterrupt ( void );
00111
00112         int8_t AddSSEL ( Pin* ssel , uint8_t slave_number );
00113         void RemoveSSEL ( uint8_t slave_number );
00114         void SetsSEL ( uint8_t slave_number );
00115         void ClearSSEL ( uint8_t slave_number );
00116
00117         void ClearEOT ( void );
00118         void SetEOT ( void );
00119         void ClearEOF ( void );
00120         void SetEOF ( void );
00121         void forceFinish();
00122         void Write ( uint8_t data );
00123         int8_t Read ( uint8_t* data );
00124         bool isTxReady( void );
00125         bool isRxReady( void );
00126
00127         virtual void SPI_IRQHandler ( void ) {}
00128         virtual ~SPI() = default;
00129
00130     private:
00131         void config ( uint32_t clk_freq , SPI_mode_t mode );
00132         void EnableSWM ( void );
00133         void ConfigClock ( void );
00134
00135 #endif /* SPI_H_ */

```

8.117 SPIMaster.cpp File Reference

Modulo de comunicacion **SPIMaster**.

```
#include <Perifericos/14-ComSincronica/SPI/SPIMaster.h>
Include dependency graph for SPIMaster.cpp:
```



8.117.1 Detailed Description

Modulo de comunicacion [SPIMaster](#).

Date

9 ene. 2025

Version

1.0

Author

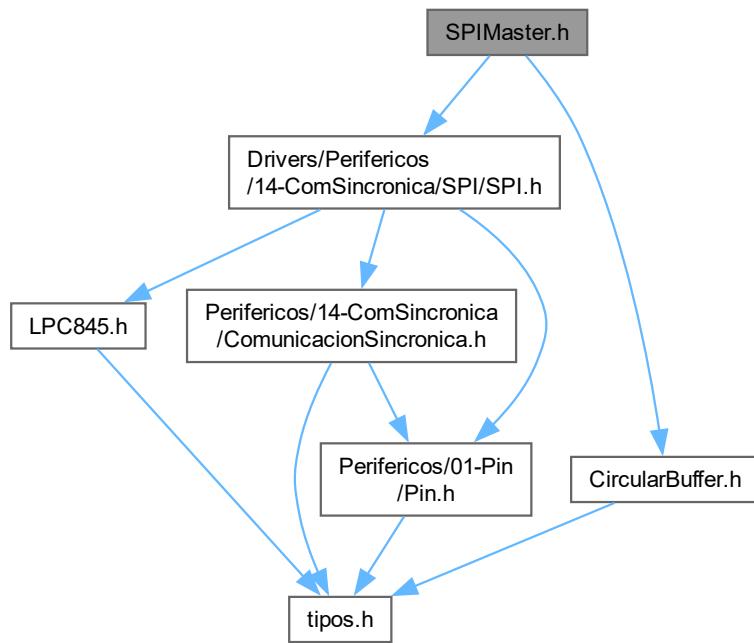
Técnico. Martinez Agustin (masteragus365@gmail.com)

8.118 SPIMaster.h File Reference

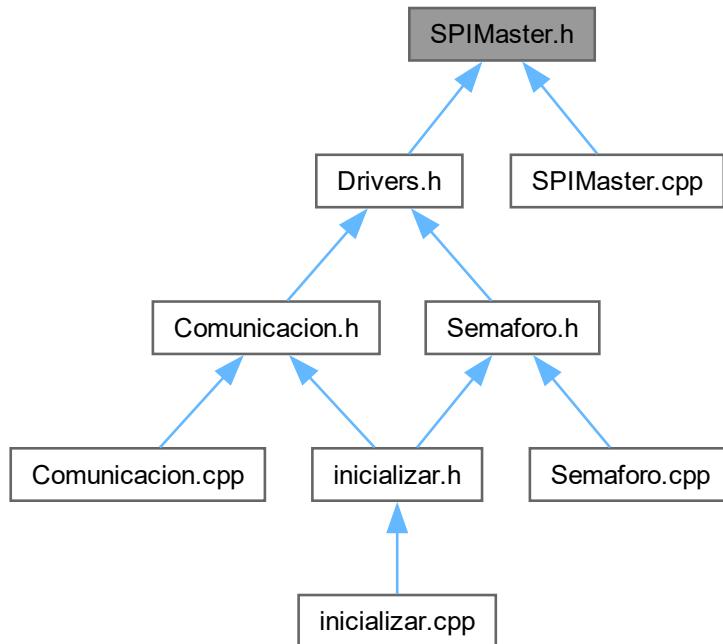
Modulo de comunicacion [SPIMaster](#).

```
#include <Drivers/Perifericos/14-ComSincronica/SPI/SPI.h>
#include <CircularBuffer.h>
```

Include dependency graph for SPIMaster.h:



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



Classes

- class [SPIMaster](#)
Clase del objeto SPIMaster.

Macros

- `#define NO_DATA (uint8_t) 0xFF`

8.118.1 Detailed Description

Modulo de comunicacion [SPIMaster](#).

Objeto [SPI](#) del tipo master con buffers de recepcion y transmision. USO: Se crea y se inicializa. `AddSSEL()`: Se agrega un pin de slave. Ej: `com.addSSEL(pin1, 0);` `Write()`: Se escribe una cantidad de datos al slave correspondiente. Debe tener asignado un pin previamente con `AddSSEL()`. `ReaqueastRead()`: Envia n valores de basura. El [SPI](#) solo activa el clk si el master esta enviando informacion. Solo debe usarse si se desea leer algo sin enviar nada al mismo tiempo. `Read()`: Lee el buffer. El buffer solo puede llenarse si el master esta escribiendo algo. Tendra data si previamente se realizo un `Write()` o un `RequestRead()`. `hasData()`: Indica si tiene algo para leer o no. `isIdle()`: Indica si esta en reposo o no. Si desea que los mensajes no se envien de corrido debe verificar idle antes de hacer un write.

Warning

El modulo no se probó a frecuencias muy altas. La forma de onda parecía correcta, pero el analizador lógico utilizado no llegaba a leer el clock correctamente.

Date

9 ene. 2025

Version

1.0

AuthorTécnico. Martinez Agustín (masteragus365@gmail.com)**8.118.2 Macro Definition Documentation****8.118.2.1 NO_DATA**

#define NO_DATA (uint8_t) 0xFF

Valor enviado cuando solo se desea leer.

8.119 SPIMaster.h[Go to the documentation of this file.](#)

```

00001 //*****
00023 //*****
00024 *** MODULO
00025 ****
00026 #ifndef SPIMASTER_H_
00027 #define SPIMASTER_H_
00028
00032 //*****
00033 *** INCLUDES GLOBALES
00034 ****
00035 #include <Drivers/Perifericos/14-ComSincronica/SPI/SPI.h>
00036 #include <CircularBuffer.h>
00037 ****
00038 *** DEFINES GLOBALES
00039 ****
00040
00041 //*****
00042 *** MACROS GLOBALES
00043 ****
00044
00045 //*****
00046 *** TIPO DE DATOS GLOBALES
00047 ****
00048
00049 //*****
00050 *** VARIABLES GLOBALES
00051 ****
00052
00053 //*****
00054 *** IMPLANTACION DE UNA CLASE
00055 ****
00061 class SPIMaster : protected SPI
00062 {
00063 public:
00065     #define NO_DATA      (uint8_t) 0xFF
00066 private:
00067     CircularBuffer<uint8_t> m_bufferRx ;
00068     CircularBuffer<uint8_t> m_bufferTx ;
00069     bool                  m_isSending;
00070     uint32_t              m_ssSel;
00071 public:
00072     SPIMaster( SPI_Type *SPI_register, Pin *miso, Pin *mosi, Pin *clk, uint32_t maxRx = 15,
00073                uint32_t maxTx = 15 );
00074     void Initialize( uint32_t clk_freq, SPI_mode_t mode = SPI_MODE_0 );
00075     int8_t    AddSSEL( Pin *ssSel, uint8_t slave_number );

```

```

00076     void      RemoveSSEL( uint8_t slave_number );
00077
00078     void      Write( uint8_t slv, const char *msg );
00079     void      Write( uint8_t slv, const void *msg, uint32_t length );
00080     void*     Read( void *msg, uint32_t length );
00081     void      RequestRead ( uint8_t slv, uint32_t cant_read = 1 );
00082     bool      hasData ( void );
00083
00084     bool      IsIdle ( void );
00085
00086     virtual ~SPIMaster();
00087
00088 private:
00089     void      SPI_IRQHandler ( void ) override;
00090 };
00091
00092 #endif /* SPIMASTER_H_ */

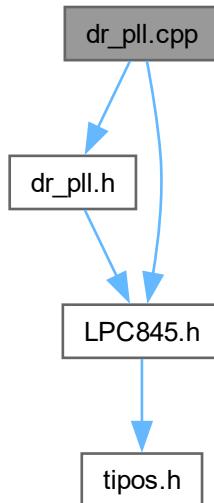
```

8.120 dr_pll.cpp File Reference

Descripcion del modulo.

```
#include "dr_pll.h"
#include <LPC845.h>
```

Include dependency graph for dr_pll.cpp:



Functions

- void [Iniciar_PLL](#) (void)

Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

8.120.1 Detailed Description

Descripcion del modulo.

Date

5 feb. 2020

Version

1.0

Author

Ing. Marcelo Trujillo

8.120.2 Function Documentation**8.120.2.1 Inicializar_PLL()**

```
void Inicializar_PLL (
    void )
```

Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

Author

Ing. Gustavo Fresno

Date

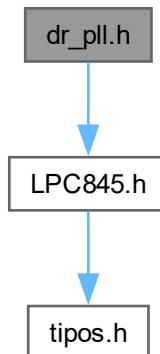
5 feb. 2020

8.121 dr_pll.h File Reference

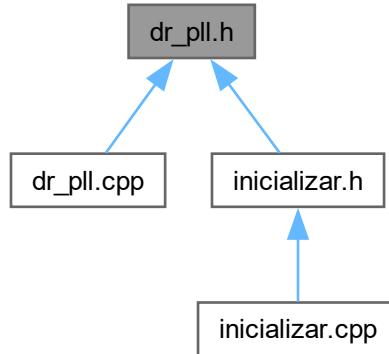
Descripcion del modulo.

```
#include <LPC845.h>
```

Include dependency graph for dr_pll.h:



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



Functions

- void [IniciarPLL](#) (void)
Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

8.121.1 Detailed Description

Descripcion del modulo.

Date

5 feb. 2020

Version

1.0

Author

Ing. Marcelo Trujillo

8.121.2 Function Documentation

8.121.2.1 IniciarPLL()

```
void IniciarPLL (
```

```
                  void )
```

Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

Author

Ing. Gustavo Fresno

Date

5 feb. 2020

8.122 dr_pll.h

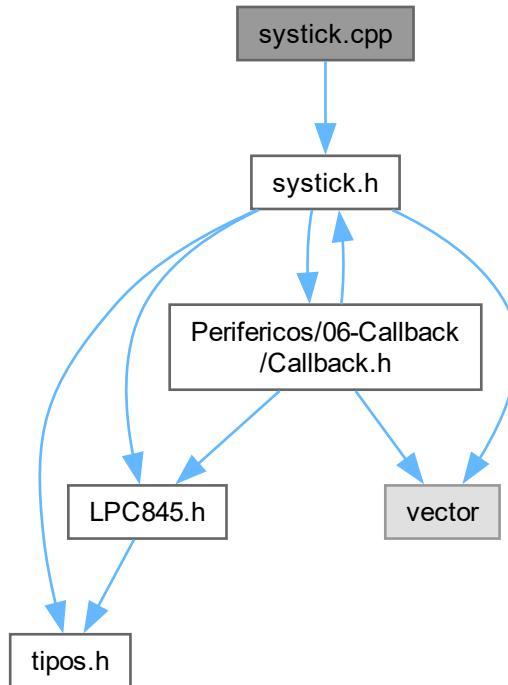
[Go to the documentation of this file.](#)

```
00001  /*****
00012  **** MODULO
00013  ***
00014  ****
00015
00016 #ifndef DRIVERS_DR_PLL_H_
00017 #define DRIVERS_DR_PLL_H_
00018
00019
00020  *** INCLUDES GLOBALES
00021
00022 #include <LPC845.h>
00023
00024  *** DEFINES GLOBALES
00025
00026
00027
00028  *** MACROS GLOBALES
00029
00030
00031
00032  *** TIPO DE DATOS GLOBALES
00033
00034
00035
00036  *** VARIABLES GLOBALES
00037
00038
00039
00040  *** PROTOTIPOS DE FUNCIONES GLOBALES
00041
00042 void Inicializar_PLL( void ) ;
00043
00044 #endif /* DRIVERS_DR_PLL_H_ */
```

8.123 systick.cpp File Reference

Firmware del systick.

```
#include "systick.h"
Include dependency graph for systick.cpp:
```



Macros

- `#define MAX_TICKS 0xffffffff`

Functions

- `void SysTick_Handler (void)`
Funcion handler de todos los systick.
- `uint32_t Inicializar_SysTick (uint32_t freq)`
Inicializa el systick en la frecuencia asignada.

Variables

- `uint32_t g_systick_freq = 0`

8.123.1 Detailed Description

Firmware del systick.

Date

20 abr. 2022

Version

2.0

Author

Marcelo y Técnico. Martinez Agustin (masteragus365@gmail.com)

8.123.2 Macro Definition Documentation

8.123.2.1 MAX_TICKS

```
#define MAX_TICKS 0xffffffff  
MAX ticks of SYSTICK clock.
```

8.123.3 Function Documentation

8.123.3.1 Inicializar_SysTick()

```
uint32_t Inicializar_SysTick (  
    uint32_t freq)
```

Inicializa el systick en la frecuencia asignada.

Parameters

in	<i>freq</i>	ticks por segundo.
----	-------------	--------------------

Returns

mensaje de error

8.123.4 Variable Documentation

8.123.4.1 g_systick_freq

```
uint32_t g_systick_freq = 0
```

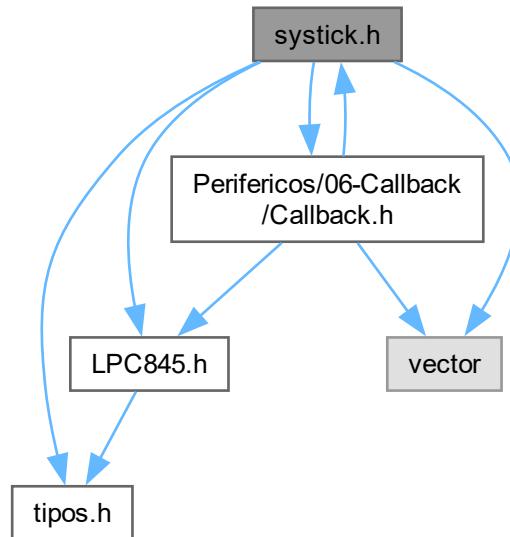
Variable global. Posee la frecuencia del systick configurada.

8.124 systick.h File Reference

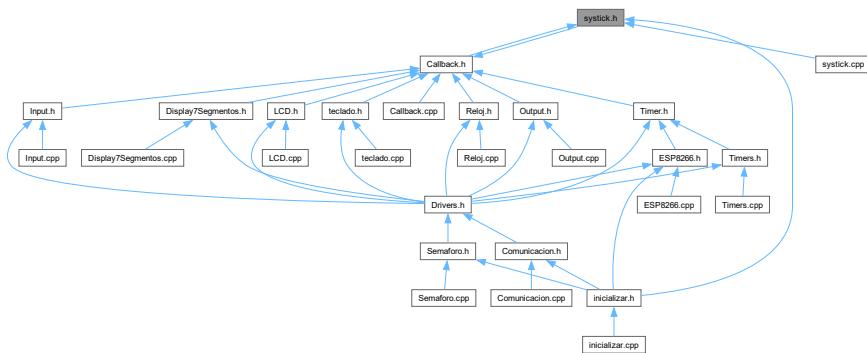
Firmware del systick.

```
#include <LPC845.h>  
#include <tipos.h>  
#include <Perifericos/06-Callback/Callback.h>  
#include <vector>
```

Include dependency graph for systick.h:



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



Functions

- `uint32_t Inicializar_SysTick (uint32_t freq)`
Inicializa el systick en la frecuencia asignada.

Variables

- `uint32_t g_systick_freq`

8.124.1 Detailed Description

Firmware del systick.

Date

20 abr. 2022

Version

2.0

Author

Marcelo y Técnico. Martinez Agustin (masteragus365@gmail.com)

8.124.2 Function Documentation

8.124.2.1 Inicializar_SysTick()

```
uint32_t Inicializar_SysTick (
    uint32_t freq)
```

Inicializa el systick en la frecuencia asignada.

Parameters

in	<i>freq</i>	ticks por segundo.
----	-------------	--------------------

Returns

mensaje de error

8.124.3 Variable Documentation

8.124.3.1 g_systick_freq

```
uint32_t g_systick_freq [extern]
```

Variable global. Posee la frecuencia del systick configurada.

8.125 systick.h

[Go to the documentation of this file.](#)

```
00001 /*****
00012 ****/
00013 *** MODULO
00014
00015 #ifndef SYSTICK_H_
00016 #define SYSTICK_H_
00017
00018 /**
00019 *** INCLUDES GLOBALES
00020 */
00021 #include <LPC845.h>
00022 #include <tipos.h>
00023 #include <Perifericos/06-Callback/Callback.h>
00024 #include <vector>
00025 using namespace std;
00026
00027 *** DEFINES GLOBALES
00028
00029
00030 /**
00031 *** MACROS GLOBALES
00032 */
00033
```

```

00034
00035     *** TIPO DE DATOS GLOBALES
00036
00037 extern uint32_t g_systick_freq;
00038
00039     *** IMPLANTACION DE LA CLASE
00040
00041 uint32_t Inicializar_Systick( uint32_t freq );
00042
00043 #if defined (__cplusplus)
00044     extern "C" {
00045         void SysTick_Handler(void);
00046     }
00047 #endif
00048
00049 #endif /* SYSTICK_H_ */

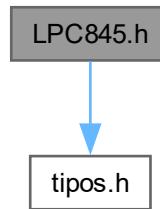
```

8.126 LPC845.h File Reference

Registros del LPC845.

```
#include "tipos.h"
```

Include dependency graph for LPC845.h:



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



Classes

- struct [SYSCON_Type](#)
Structure type to access the SYSCON.
- struct [GPIO_Type](#)
Structure type to access the General Purpose [Input Output](#) (GPIO).
- struct [IOCON_Type](#)
Structure type to access the IOCON.
- struct [SysTick_t](#)
Structure type to access the System [Timer](#) (SysTick).
- struct [MRT_t](#)
Structure type to access the MRT timer.

- struct **NVIC_Type**
Structure type to access the Nested Vectored Interrupt Controller (NVIC).
- struct **PIN_INTERRUPT_t**
Structure type to access the Pin Interrupt.
- struct **SCT_t**
Structure type to access the SCT.
- struct **USART_Type**
Structure type to access the Universal Serial Asincronical Reciver Transmitter (USART).
- struct **SWM_t**
Structure type to access the Switch Matriz (SWM).
- struct **ADC_Type**
Structure type to access the Analog Digital Convertos (ADC).
- struct **DAC_t**
Structure type to access the Digital Analog Convertor (DAC).
- struct **I2C_Type**
- struct **SPI_Type**

Macros

- #define **FCLKIN** (12000000UL)
- #define **FREQ_PRINCIPAL** 48000000UL
- #define **SYSCON_BASE** (0x40048000u)
- #define **SYSCON** ((**SYSCON_Type** *)**SYSCON_BASE**)
- #define **SYSCON_BASE_ADDRS** { **SYSCON_BASE** }
- #define **SYSCON_BASE_PTRS** { **SYSCON** }
- #define **SYSCON_IRQS** { **BOD_IRQn** }
- #define **GPIO** ((**GPIO_Type***) 0xA0000000UL)
- #define **IOCON** ((**IOCON_Type***) 0x40044000UL)
- #define **SysTick** ((**SysTick_t** *) 0xE000E010UL)
- #define **FREQ_SYSTICK** (1000)
- #define **MRT** ((**MRT_t** *) 0x40004000UL)
- #define **MRT0** ((**MRT_t** *) **MRT**)
- #define **MRT1** ((**MRT_t** *) **MRT0** + 1)
- #define **MRT2** ((**MRT_t** *) **MRT1** + 1)
- #define **MRT3** ((**MRT_t** *) **MRT2** + 1)
- #define **MRT_IDLE_CH** ((**_R uint32_t** *) 0x400040F4UL)
- #define **MRT_IRQ_FLAQ** ((**_RW uint32_t** *) 0x400040F8UL)
- #define **SCS_BASE** (0xE000E000UL)
- #define **SysTick_BASE** (**SCS_BASE** + 0x0010UL)
- #define **NVIC_BASE** (**SCS_BASE** + 0x0100UL)
- #define **SCB_BASE** (**SCS_BASE** + 0x0D00UL)
- #define **NVIC** ((**NVIC_Type** *) **NVIC_BASE**)
- #define **PIN_INTERRUPT** ((**PIN_INTERRUPT_t** *) 0xA0004000UL)
- #define **SCT** ((**SCT_t** *) 0x50004000UL)
- #define **USART0_BASE** (0x40064000u)
- #define **USART0** ((**USART_Type** *)**USART0_BASE**)
- #define **USART1_BASE** (0x40068000u)
- #define **USART1** ((**USART_Type** *)**USART1_BASE**)
- #define **USART2_BASE** (0x4006C000u)
- #define **USART2** ((**USART_Type** *)**USART2_BASE**)
- #define **USART3_BASE** (0x40070000u)
- #define **USART3** ((**USART_Type** *)**USART3_BASE**)
- #define **USART4_BASE** (0x40074000u)
- #define **USART4** ((**USART_Type** *)**USART4_BASE**)

- #define USART_BASE_ADDRS { USART0_BASE, USART1_BASE, USART2_BASE, USART3_BASE, USART4_BASE }
- #define USART_BASE_PTRS { USART0, USART1, USART2, USART3, USART4 }
- #define USART IRQS { USART0_IRQn, USART1_IRQn, USART2_IRQn, PIN_INT6_USART3_IRQn, PININT7_USART4_IRQn }
- #define SWM ((SWM_t *) 0x4000C000UL)
- #define ADC_SEQ_CTRL_COUNT (2U)
- #define ADC_SEQ_GDAT_COUNT (2U)
- #define ADC_DAT_COUNT (12U)
- #define ADC0_BASE (0x4001C000u)
- #define ADC0 ((ADC_Type *)ADC0_BASE)
- #define ADC_BASE_ADDRS { ADC0_BASE }
- #define ADC_BASE_PTRS { ADC0 }
- #define ADC_SEQ IRQS { ADC0_SEQA_IRQn, ADC0_SEQB_IRQn }
- #define DAC0 ((DAC_t*) 0x40014000UL)
- #define DAC1 ((DAC_t*) 0x40018000UL)
- #define I2C0_BASE (0x40050000u)
- #define I2C0 ((I2C_Type *)I2C0_BASE)
- #define I2C1_BASE (0x40054000u)
- #define I2C1 ((I2C_Type *)I2C1_BASE)
- #define I2C2_BASE (0x40030000u)
- #define I2C2 ((I2C_Type *)I2C2_BASE)
- #define I2C3_BASE (0x40034000u)
- #define I2C3 ((I2C_Type *)I2C3_BASE)
- #define I2C_BASE_ADDRS { I2C0_BASE, I2C1_BASE, I2C2_BASE, I2C3_BASE }
- #define I2C_BASE_PTRS { I2C0, I2C1, I2C2, I2C3 }
- #define SPI0_BASE (0x40058000u)
- #define SPI0 ((SPI_Type *)SPI0_BASE)
- #define SPI1_BASE (0x4005C000u)
- #define SPI1 ((SPI_Type *)SPI1_BASE)
- #define SPI_BASE_ADDRS { SPI0_BASE, SPI1_BASE }
- #define SPI_BASE_PTRS { SPI0, SPI1 }

CTRL - ADC Control register. Contains the clock divide value, resolution selection, sampling time selection, and mode controls.

- #define ADC_CTRL_CLKDIV(x)
- #define ADC_CTRL_ASYNMODE(x)
- #define ADC_CTRL_LPWRMODE(x)
- #define ADC_CTRL_CALMODE(x)

SEQ_CTRL - ADC Conversion Sequence-n control register: Controls triggering and channel selection for conversion sequence-n. Also specifies interrupt mode for sequence-n.

- #define ADC_SEQ_CTRL_CHANNELS(x)
- #define ADC_SEQ_CTRL_TRIGGER(x)
- #define ADC_SEQ_CTRL_TRIGPOL(x)
- #define ADC_SEQ_CTRL_SYNCBYPASS(x)
- #define ADC_SEQ_CTRL_START(x)
- #define ADC_SEQ_CTRL_BURST(x)
- #define ADC_SEQ_CTRL_SINGLESTEP(x)
- #define ADC_SEQ_CTRL_LOWPRIO(x)
- #define ADC_SEQ_CTRL_MODE(x)
- #define ADC_SEQ_CTRL_SEQ_ENA(x)

SEQ_GDAT - ADC Sequence-n Global Data register. This register contains the result of the most recent ADC conversion performed under sequence-n.

- #define ADC_SEQ_GDAT_RESULT(x)

- #define ADC_SEQ_GDAT_THCMPRANGE(x)
- #define ADC_SEQ_GDAT_THCMPCROSS(x)
- #define ADC_SEQ_GDAT_CHN(x)
- #define ADC_SEQ_GDAT_OVERRUN(x)
- #define ADC_SEQ_GDAT_DATAVALID(x)

DAT - ADC Channel N Data register. This register contains the result of the most recent conversion completed on channel N.

- #define ADC_DAT_RESULT(x)
- #define ADC_DAT_THCMPRANGE(x)
- #define ADC_DAT_THCMPCROSS(x)
- #define ADC_DAT_CHANNEL(x)
- #define ADC_DAT_OVERRUN(x)
- #define ADC_DAT_DATAVALID(x)

THR0_LOW - ADC Low Compare Threshold register 0: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0.

- #define ADC_THR0_LOW_THRLOW(x)

THR1_LOW - ADC Low Compare Threshold register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1.

- #define ADC_THR1_LOW_THRLOW(x)

THR0_HIGH - ADC High Compare Threshold register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0.

- #define ADC_THR0_HIGH_THRHIGH(x)

THR1_HIGH - ADC High Compare Threshold register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1.

- #define ADC_THR1_HIGH_THRHIGH(x)

CHAN_THRSEL - ADC Channel-Threshold Select register. Specifies which set of threshold compare registers are to be used for each channel

- #define ADC_CHAN_THRSEL_CH0_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH1_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH2_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH3_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH4_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH5_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH6_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH7_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH8_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH9_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH10_THRSEL(x)
- #define ADC_CHAN_THRSEL_CH11_THRSEL(x)

INTEN - ADC Interrupt Enable register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated.

- #define ADC_INTEN_SEQA_INTEN(x)
- #define ADC_INTEN_SEQB_INTEN(x)
- #define ADC_INTEN_OVR_INTEN(x)
- #define ADC_INTEN_ADCMPINTEN0(x)
- #define ADC_INTEN_ADCMPINTEN1(x)
- #define ADC_INTEN_ADCMPINTEN2(x)
- #define ADC_INTEN_ADCMPINTEN3(x)
- #define ADC_INTEN_ADCMPINTEN4(x)
- #define ADC_INTEN_ADCMPINTEN5(x)

- #define ADC_INTEN_ADCMPINTEN6(x)
- #define ADC_INTEN_ADCMPINTEN7(x)
- #define ADC_INTEN_ADCMPINTEN8(x)
- #define ADC_INTEN_ADCMPINTEN9(x)
- #define ADC_INTEN_ADCMPINTEN10(x)
- #define ADC_INTEN_ADCMPINTEN11(x)

FLAGS - ADC Flags register. Contains the four interrupt/DMA trigger flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers).

- #define ADC_FLAGS_THCMP0(x)
- #define ADC_FLAGS_THCMP1(x)
- #define ADC_FLAGS_THCMP2(x)
- #define ADC_FLAGS_THCMP3(x)
- #define ADC_FLAGS_THCMP4(x)
- #define ADC_FLAGS_THCMP5(x)
- #define ADC_FLAGS_THCMP6(x)
- #define ADC_FLAGS_THCMP7(x)
- #define ADC_FLAGS_THCMP8(x)
- #define ADC_FLAGS_THCMP9(x)
- #define ADC_FLAGS_THCMP10(x)
- #define ADC_FLAGS_THCMP11(x)
- #define ADC_FLAGS_OVERRUN0(x)
- #define ADC_FLAGS_OVERRUN1(x)
- #define ADC_FLAGS_OVERRUN2(x)
- #define ADC_FLAGS_OVERRUN3(x)
- #define ADC_FLAGS_OVERRUN4(x)
- #define ADC_FLAGS_OVERRUN5(x)
- #define ADC_FLAGS_OVERRUN6(x)
- #define ADC_FLAGS_OVERRUN7(x)
- #define ADC_FLAGS_OVERRUN8(x)
- #define ADC_FLAGS_OVERRUN9(x)
- #define ADC_FLAGS_OVERRUN10(x)
- #define ADC_FLAGS_OVERRUN11(x)
- #define ADC_FLAGS_SEQA_OVR(x)
- #define ADC_FLAGS_SEQB_OVR(x)
- #define ADC_FLAGS_SEQA_INT(x)
- #define ADC_FLAGS_SEQB_INT(x)
- #define ADC_FLAGS_THCMP_INT(x)
- #define ADC_FLAGS_OVR_INT(x)

TRM - ADC Startup register.

- #define ADC_TRM_VRANGE(x)

CFG - Configuration for shared functions.

- #define I2C_CFG_MSTEN(x)
- #define I2C_CFG_SLVEN(x)
- #define I2C_CFG_MONEN(x)
- #define I2C_CFG_TIMEOUTEN(x)
- #define I2C_CFG_MONCLKSTR(x)

STAT - Status register for Master, Slave, and Monitor functions.

- #define I2C_STAT_MSTPENDING(x)
- #define I2C_STAT_MSTSTATE(x)
- #define I2C_STAT_MSTARBLOSS(x)
- #define I2C_STAT_MSTSTSTPERR(x)
- #define I2C_STAT_SLVPENDING(x)
- #define I2C_STAT_SLVSTATE(x)
- #define I2C_STAT_SLVNOTSTR(x)
- #define I2C_STAT_SLVIDX(x)
- #define I2C_STAT_SLVSEL(x)

- #define I2C_STAT_SLVDESEL(x)
- #define I2C_STAT_MONRDY(x)
- #define I2C_STAT_MONOV(x)
- #define I2C_STAT_MONACTIVE(x)
- #define I2C_STAT_MONIDLE(x)
- #define I2C_STAT_EVENTTIMEOUT(x)
- #define I2C_STAT_SCLTIMEOUT(x)

INTENSET - Interrupt Enable Set and read register.

- #define I2C_INTENSET_MSTPENDINGEN(x)
- #define I2C_INTENSET_MSTARBLOSSEN(x)
- #define I2C_INTENSET_MSTSTSPERREN(x)
- #define I2C_INTENSET_SLVPENDINGEN(x)
- #define I2C_INTENSET_SLVNOTSTREN(x)
- #define I2C_INTENSET_SLVDESELEN(x)
- #define I2C_INTENSET_MONRDYEN(x)
- #define I2C_INTENSET_MONOVEN(x)
- #define I2C_INTENSET_MONIDLEEN(x)
- #define I2C_INTENSET_EVENTTIMEOUTEN(x)
- #define I2C_INTENSET_SCLTIMEOUTEN(x)

INTENCLR - Interrupt Enable Clear register.

TIMEOUT - Time-out value register.

CLKDIV - Clock pre-divider for the entire I2C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function.

INTSTAT - Interrupt Status register for Master, Slave, and Monitor functions.

MSTCTL - Master control register.

- #define I2C_MSTCTL_MSTCONTINUE(x)
- #define I2C_MSTCTL_MSTSTART(x)
- #define I2C_MSTCTL_MSTSTOP(x)
- #define I2C_MSTCTL_MSTDMA(x)

MSTTIME - Master timing configuration.

- #define I2C_MSTTIME_MSTSCLLOW(x)
- #define I2C_MSTTIME_MSTSCLHIGH(x)

MSTDAT - Combined Master receiver and transmitter data register.

SLVCTL - Slave control register.

- #define I2C_SLVCTL_SLCONTINUE(x)
- #define I2C_SLVCTL_SLVNACK(x)
- #define I2C_SLVCTL_SLVDMA(x)

SLVDAT - Combined Slave receiver and transmitter data register.

SLVADR - Slave address register.

- #define I2C_SLVADR_SADISABLE(x)

SLVQUAL0 - Slave Qualification for address 0.

- #define I2C_SLVQUAL0_QUALMODE0(x)

MONRXDAT - Monitor receiver data register.

- #define I2C_MONRXDAT_MONSTART(x)
- #define I2C_MONRXDAT_MONRESTART(x)

- #define I2C_MONRXDAT_MONNACK(x)

CFG - SPI Configuration register

- #define SPI_CFG_ENABLE(x)
- #define SPI_CFG_MASTER(x)
- #define SPI_CFG_LSBF(x)
- #define SPI_CFG_CPHA(x)
- #define SPI_CFG_CPOL(x)
- #define SPI_CFG_LOOP(x)
- #define SPI_CFG_SPOL0(x)
- #define SPI_CFG_SPOL1(x)
- #define SPI_CFG_SPOL2(x)
- #define SPI_CFG_SPOL3(x)

DLY - SPI Delay register

STAT - SPI Status. Some status flags can be cleared by writing a 1 to that bit position

INTENSET - SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.

- #define SPI_INTENSET_RXRDYEN(x)
- #define SPI_INTENSET_TXRDYEN(x)
- #define SPI_INTENSET_RXOVEN(x)
- #define SPI_INTENSET_TXUREN(x)
- #define SPI_INTENSET_SSAEN(x)
- #define SPI_INTENSET_SSSEN(x)

INTENCLR - SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.

RXDAT - SPI Receive Data

TXDATCTL - SPI Transmit Data with Control

- #define SPI_TXDATCTL_TXSSEL0_N(x)
- #define SPI_TXDATCTL_TXSSEL1_N(x)
- #define SPI_TXDATCTL_TXSSEL2_N(x)
- #define SPI_TXDATCTL_TXSSEL3_N(x)
- #define SPI_TXDATCTL_EOT(x)
- #define SPI_TXDATCTL_EOF(x)
- #define SPI_TXDATCTL_RXIGNORE(x)
- #define SPI_TXDATCTL_LEN(x)

TXDAT - SPI Transmit Data.

TXCTL - SPI Transmit Control

DIV - SPI clock Divider

INTSTAT - SPI Interrupt Status

8.126.1 Detailed Description

Registros del LPC845.

Date

10 ene. 2022

Version

1.0

Author

Técnico. Martínez Agustín (masteragus365@gmail.com)

8.127 LPC845.h

[Go to the documentation of this file.](#)

```

00001  /*****
00012  **** MODULO
00013  ***
00014  ****
00015 #ifndef LPC845_H_
00016 #define LPC845_H_
00017
00018  ****
00019  *** INCLUDES GLOBALES
00020
00021 #include "tipos.h"
00022
00023  ****
00024  *** DEFINES GLOBALES
00025
00026
00027
00028  ****
00029  *** MACROS GLOBALES
00030
00031
00032  ****
00033  *** TIPO DE DATOS GLOBALES
00034
00035
00036 #define FCLKIN      (12000000UL)
00037
00038 #define FREQ_PRINCIPAL 48000000UL //En Hz. Puede ser 24 MHz o 48 MHz
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052 typedef struct {
00053     __RW uint32_t SYSMEMREMAP;
00054     uint8_t RESERVED_0[4];
00055     __RW uint32_t SYSPLLCTRL;
00056     __R  uint32_t SYSPLLSTAT;
00057     uint8_t RESERVED_1[16];
00058     __RW uint32_t SYSSCCTRL;
00059     __RW uint32_t WDTOSCCTRL;
00060     __RW uint32_t FROOSCCTRL;
00061     uint8_t RESERVED_2[4];
00062     __RW uint32_t FRDIRECTCLKUEN;
00063     uint8_t RESERVED_3[4];
00064     __RW uint32_t SYSRSTSTAT;
00065     uint8_t RESERVED_4[4];
00066     __RW uint32_t SYSPLLCLKSEL;
00067     __RW uint32_t SYSPLLCLKUEN;
00068     __RW uint32_t MAINCLKPLLSEL;
00069     __RW uint32_t MAINCLKPLLUEN;
00070     __RW uint32_t MAINCLKSEL;
00071     __RW uint32_t MAINCLKUEN;
00072     __RW uint32_t SYSAHBCLKDIV;
00073     uint8_t RESERVED_5[4];
00074     __RW uint32_t CAPTCLKSEL;
00075     __RW uint32_t ADCCLKSEL;
00076     __RW uint32_t ADCCLKDIV;
00077     __RW uint32_t SCTCLKSEL;
00078     __RW uint32_t SCTCLKDIV;
00079     __RW uint32_t EXTCLKSEL;
00080     uint8_t RESERVED_6[8];
00081     __RW uint32_t SYSAHBCLKCTRL0;
00082     __RW uint32_t SYSAHBCLKCTRL1;
00083     __RW uint32_t PRESETCTRL0;
00084     __RW uint32_t PRESETCTRL1;
00085     __RW uint32_t FCLKSEL[11];
00086     uint8_t RESERVED_7[20];
00087     struct {
00088         __RW uint32_t FRGDIV;
00089         __RW uint32_t FRGMULT;
00090         __RW uint32_t FRGCLKSEL;
00091         uint8_t RESERVED_0[4];
00092     } FRG[2];
00093     __RW uint32_t CLKOUTSEL;
00094     __RW uint32_t CLKOUTDIV;
00095     uint8_t RESERVED_8[4];

```

```

00096     __RW uint32_t EXTTRACECMD;
00097     __R  uint32_t PIOPORCAP[2];
00098     uint8_t RESERVED_9[44];
00099     __RW uint32_t IOCONCLKDIV6;
00100     __RW uint32_t IOCONCLKDIV5;
00101     __RW uint32_t IOCONCLKDIV4;
00102     __RW uint32_t IOCONCLKDIV3;
00103     __RW uint32_t IOCONCLKDIV2;
00104     __RW uint32_t IOCONCLKDIV1;
00105     __RW uint32_t IOCONCLKDIV0;
00106     __RW uint32_t BODCTRL;
00107     __RW uint32_t SYSTCKCAL;
00108     uint8_t RESERVED_10[24];
00109     __RW uint32_t IRQLATENCY;
00110     __RW uint32_t NMISRC;
00111     __RW uint32_t PINTSEL[8];
00112     uint8_t RESERVED_11[108];
00113     __RW uint32_t STARTERPO;
00114     uint8_t RESERVED_12[12];
00115     __RW uint32_t STARTERPI;
00116     uint8_t RESERVED_13[24];
00117     __RW uint32_t PDSLEEPFCFG;
00118     __RW uint32_t PDWAKEFCFG;
00119     __RW uint32_t PDRUNCFG;
00120     uint8_t RESERVED_14[444];
00121     __R  uint32_t DEVICE_ID;
00122 } SYSCON_Type;
00123
00126 #define SYSCON_BASE           (0x40048000u)
00128 #define SYSCON    ((SYSCON_Type *) SYSCON_BASE)
00130 #define SYSCON_BASE_ADDRS
00132 #define SYSCON_BASE_PTRS
00134 #define SYSCON_IRQS
00138
00147 typedef struct {
00148     __RW uint8_t B[2][32];
00149     uint8_t RESERVED_0[4032];
00150     __RW uint32_t W[2][32];
00151     uint8_t RESERVED_1[3840];
00152     __RW uint32_t DIR[2];
00153     uint8_t RESERVED_2[120];
00154     __RW uint32_t MASK[2];
00155     uint8_t RESERVED_3[120];
00156     __RW uint32_t PIN[2];
00157     uint8_t RESERVED_4[120];
00158     __RW uint32_t MPIN[2];
00159     uint8_t RESERVED_5[120];
00160     __RW uint32_t SET[2];
00161     uint8_t RESERVED_6[120];
00162     __W  uint32_t CLR[2];
00163     uint8_t RESERVED_7[120];
00164     __W  uint32_t NOT[2];
00165     uint8_t RESERVED_8[120];
00166     __W  uint32_t DIRSET[2];
00167     uint8_t RESERVED_9[120];
00168     __W  uint32_t DIRCLR[2];
00169     uint8_t RESERVED_10[120];
00170     __W  uint32_t DIRNOT[2];
00171 } GPIO_Type;
00172
00173 #define GPIO      ( (GPIO_Type*) 0xA0000000UL )
00177
00186 typedef struct {
00187     __RW uint32_t PIO[56];
00188 } IOCON_Type;
00189
00190 #define IOCON  ( (IOCON_Type*) 0x40044000UL )
00194
00203 typedef struct
00204 {
00205     __RW uint32_t CTRL;
00206     __RW uint32_t RELOAD;
00207     __RW uint32_t CURR;
00208     __R  uint32_t CALIB;
00209 } SysTick_t;
00210
00212 #define SysTick   ( (SysTick_t *) 0xE000E010UL)
00213
00215 #define FREQ_SYSTICK   (1000)          //En Hz
00216
00218 #define STCTRL      SysTick->w_stctrl
00219     #define STCTRL_ENABLE      SysTick->ENABLE
00220     #define STCTRL_TICKINT     SysTick->TICKINT
00221     #define STCTRL_CLKSOURCE    SysTick->CLKSOURCE
00222     #define STCTRL_COUNTFLAG    SysTick->COUNTFLAG
00223 #define STRELOAD     SysTick->_STRELOAD
00224 #define STCURR       SysTick->_STCURR

```

```

00225 #define      STCALIB      SysTick->_STCALIB
00227
00231
00240 typedef struct
00241 {
00242     union{
00243         struct{
00244             __RW      uint32_t      IVALUE:31;
00245             __RW      uint32_t      LOAD:1;
00246         };
00247         __RW      uint32_t      INTVAL;
00248     };
00249     union{
00250         struct{
00251             __R       uint32_t      VALUE:31;
00252             __R       uint32_t      RESERVED0:1;
00253         };
00254         __R       uint32_t      TIMER;
00255     };
00256     union{
00257         struct{
00258             __RW      uint32_t      INTEN:1;
00259             __RW      uint32_t      MODE:2;
00260             __RW      uint32_t      RESERVED1:29;
00261         };
00262         __RW      uint32_t      CTRL;
00263     };
00264     union{
00265         struct{
00266             __RW      uint32_t      INTFLAG:1;
00267             __RW      uint32_t      RUN:1;
00268             __RW      uint32_t      RESERVED2:30;
00269         };
00270         __RW      uint32_t      STAT;
00271     };
00272 }MRT_t;
00273 #define MRT      ( (MRT_t *) 0x40004000UL)
00275 #define MRT0     ( (MRT_t *) MRT )
00277 #define MRT1     ( (MRT_t *) MRT0 + 1 )
00279 #define MRT2     ( (MRT_t *) MRT1 + 1 )
00281 #define MRT3     ( (MRT_t *) MRT2 + 1 )
00283 #define MRT_IDLE_CH ( (__R uint32_t *) 0x400040F4UL )
00285 #define MRT IRQ_FLAQ ( (__RW uint32_t *) 0x400040F8UL )
00289
00298 typedef struct
00299 {
00300     __RW uint32_t  ISER[1U];
00301     uint32_t    RESERVED0[31U];
00302     __RW uint32_t  ICER[1U];
00303     uint32_t    RSERVED1[31U];
00304     __RW uint32_t  ISPR[1U];
00305     uint32_t    RESERVED2[31U];
00306     __RW uint32_t  RESERVED3[31U];
00307     uint32_t    RESERVED4[64U];
00308     __RW uint32_t  IP[8U];
00309 }NVIC_Type;
00310
00311 /* Memory mapping of Core Hardware */
00313 #define SCS_BASE          (0xE000E000UL)
00315 #define SysTick_BASE      (SCS_BASE + 0x0010UL)
00317 #define NVIC_BASE         (SCS_BASE + 0x0100UL)
00319 #define SCB_BASE          (SCS_BASE + 0x0D00UL)
00321 #define NVIC             ((NVIC_Type      *)  NVIC_BASE      )
00325
00334 typedef struct
00335 {
00336     __RW      uint32_t      ISEL;
00337
00338     __RW      uint32_t      IENR;
00339     __W       uint32_t      SIENR;
00340     __W       uint32_t      CIENR;
00341
00342     __RW      uint32_t      IENF;
00343     __W       uint32_t      SIENF;
00344     __W       uint32_t      CIENF;
00345
00346     __RW      uint32_t      RISE;
00347     __RW      uint32_t      FALL;
00348     __RW      uint32_t      IST;
00349
00350     __RW      uint32_t      PMCTRL;
00351     __RW      uint32_t      PMSRC;
00352     __RW      uint32_t      PMCFG;
00353 }PIN_INTERRUPT_t;
00354
00355 #define PIN_INTERRUPT ( (PIN_INTERRUPT_t *) 0xA0004000UL )
00359

```

```

00368 typedef struct {
00369     __RW uint32_t CONFIG;
00370     __RW uint32_t CTRL;
00371     __RW uint32_t LIMIT;
00372     __RW uint32_t HALT;
00373     __RW uint32_t STOP;
00374     __RW uint32_t START;
00375     uint8_t RESERVED_0[40];
00376     __RW uint32_t COUNT;
00377     __RW uint32_t STATE;
00378     __R uint32_t INPUT;
00379     __RW uint32_t REGMODE;
00380     __RW uint32_t OUTPUT;
00381     __RW uint32_t OUTPUTDIRCTRL;
00382     __RW uint32_t RES;
00383     __RW uint32_t DMAREQ0;
00384     __RW uint32_t DMAREQ1;
00385     uint8_t RESERVED_1[140];
00386     __RW uint32_t EVEN;
00387     __RW uint32_t EVFLAG;
00388     __RW uint32_t CONEN;
00389     __RW uint32_t CONFLAG;
00390     union {                                     /* offset: 0x100 */
00391         __RW uint32_t CAP[8];
00392         __RW uint32_t MATCH[8];
00393     };
00394     uint8_t RESERVED_2[224];
00395     union {                                     /* offset: 0x200 */
00396         __RW uint32_t CAPCTRL[8];
00397         __RW uint32_t MATCHREL[8];
00398     };
00399     uint8_t RESERVED_3[224];
00400     struct {                                    /* offset: 0x300, array step: 0x8 */
00401         __RW uint32_t STATE;
00402         __RW uint32_t CTRL;
00403     } EV[8];
00404     uint8_t RESERVED_4[448];
00405     struct {                                    /* offset: 0x500, array step: 0x8 */
00406         __RW uint32_t SET;
00407         __RW uint32_t CLR;
00408     } OUT[7];
00409 } SCT_t;
00410
00412 #define SCT ( (SCT_t *) 0x50004000UL )
00416
00425 typedef struct {
00426     __RW uint32_t CFG;
00427     __RW uint32_t CTL;
00428     __RW uint32_t STAT;
00429     __RW uint32_t INTENSET;
00430     __W uint32_t INTENCLR;
00431     __R uint32_t RXDAT;
00432     __R uint32_t RXDATSTAT;
00433     __RW uint32_t TXDAT;
00434     __RW uint32_t BRG;
00435     __R uint32_t INTSTAT;
00436     __RW uint32_t OSR;
00437     __RW uint32_t ADDR;
00438 } USART_Type;
00439
00440 /* USART - Peripheral instance base addresses */
00442 #define USART0_BASE                         (0x40064000u)
00444 #define USART0                                ((USART_Type *)USART0_BASE)
00446 #define USART1_BASE                          (0x40068000u)
00448 #define USART1                                ((USART_Type *)USART1_BASE)
00450 #define USART2_BASE                          (0x4006C000u)
00452 #define USART2                                ((USART_Type *)USART2_BASE)
00454 #define USART3_BASE                          (0x40070000u)
00456 #define USART3                                ((USART_Type *)USART3_BASE)
00458 #define USART4_BASE                          (0x40074000u)
00460 #define USART4                                ((USART_Type *)USART4_BASE)
00462 #define USART_BASE_ADDRS
00463     USART4_BASE }
00464 #define USART_BASE_PTRS                      { USART0, USART1, USART2, USART3, USART4 }
00466 #define USART IRQS
00467     PIN_INT6_USART3_IRQn, PIN_INT7_USART4_IRQn }
00470
00471 /* -----
00472 -- SWM Peripheral Access Layer
00473 ----- */ */
00478 typedef struct {
00479     union {
00480         struct {
00481             __RW uint32_t PINASSIGN0;           /* offset: 0x0 */
00482             __RW uint32_t PINASSIGN1;           /* offset: 0x0 */
00483             __RW uint32_t PINASSIGN2;
00484             __RW uint32_t PINASSIGN3;

```

```

00489     __RW uint32_t PINASSIGN4;
00490     __RW uint32_t PINASSIGN5;
00491     __RW uint32_t PINASSIGN6;
00492     __RW uint32_t PINASSIGN7;
00493     __RW uint32_t PINASSIGN8;
00494     __RW uint32_t PINASSIGN9;
00495     __RW uint32_t PINASSIGN10;
00496     __RW uint32_t PINASSIGN11;
00497     __RW uint32_t PINASSIGN12;
00498     __RW uint32_t PINASSIGN13;
00499     __RW uint32_t PINASSIGN14;
00500 } PINASSIGN;
00501     __RW uint32_t PINASSIGN_DATA[15];
00502 };
00503     uint8_t RESERVED_0[388];
00504     __RW uint32_t PINENABLE0;
00505     __RW uint32_t PINENABLE1;
00506 } SWM_t;
00507
00508 #define SWM ( (SWM_t *) 0x4000C000UL )
00513
00514 /* -----
00515   -- ADC Peripheral Access Layer
00516 ----- */
00517
00518 #define ADC_CTRL_CLKDIV_MASK           (0xFFU)
00519 #define ADC_CTRL_CLKDIV_SHIFT          (0U)
00520
00521 #define ADC_CTRL_ASYNMODE_MASK         (0x100U)
00522 #define ADC_CTRL_ASYNMODE_SHIFT        (8U)
00523
00524 #define ADC_CTRL_LPWRMODE_MASK         (0x400U)
00525 #define ADC_CTRL_LPWRMODE_SHIFT        (10U)
00526
00527 #define ADC_CTRL_CALMODE_MASK          (0x40000000U)
00528 #define ADC_CTRL_CALMODE_SHIFT         (30U)
00529
00530 #define ADC_SEQ_CTRL_CHANNELS_MASK     (0xFFFFU)
00531 #define ADC_SEQ_CTRL_CHANNELS_SHIFT    (0U)
00532
00533 #define ADC_SEQ_CTRL_TRIGGER_MASK      (0x7000U)
00534 #define ADC_SEQ_CTRL_TRIGGER_SHIFT     (12U)
00535
00536 #define ADC_SEQ_CTRL_TRIGPOL_MASK     (0x40000U)
00537 #define ADC_SEQ_CTRL_TRIGPOL_SHIFT    (18U)
00538
00539 #define ADC_SEQ_CTRL_SYNCBYPASS_MASK   (0x80000U)
00540 #define ADC_SEQ_CTRL_SYNCBYPASS_SHIFT  (19U)
00541
00542 #define ADC_SEQ_CTRL_START_MASK        (0x40000000U)
00543 #define ADC_SEQ_CTRL_START_SHIFT       (26U)
00544
00545 #define ADC_SEQ_CTRL_BURST_MASK       (0x80000000U)
00546 #define ADC_SEQ_CTRL_BURST_SHIFT      (27U)
00547
00548 #define ADC_SEQ_CTRL_SINGLESTEP_MASK  (0x10000000U)
00549 #define ADC_SEQ_CTRL_SINGLESTEP_SHIFT (28U)
00550
00551 #define ADC_SEQ_CTRL_LOWPRIOR_MASK    (0x20000000U)
00552 #define ADC_SEQ_CTRL_LOWPRIOR_SHIFT   (29U)
00553
00554 #define ADC_SEQ_CTRL_MODE_MASK         (0x40000000U)
00555 #define ADC_SEQ_CTRL_MODE_SHIFT       (30U)
00556
00557 #define ADC_SEQ_CTRL_SEQ_ENA_MASK     (0x80000000U)
00558 #define ADC_SEQ_CTRL_SEQ_ENA_SHIFT    (31U)
00559
00560 #define ADC_SEQ_GDAT_RESULT_MASK      (0xFFFFU)
00561 #define ADC_SEQ_GDAT_RESULT_SHIFT     (4U)
00562
00563 #define ADC_SEQ_GDAT_THCMPRANGE_MASK  (0x3000U)
00564 #define ADC_SEQ_GDAT_THCMPRANGE_SHIFT (16U)
00565
00566 #define ADC_SEQ_GDAT_THCMP CROSS_MASK (0xC000U)
00567 #define ADC_SEQ_GDAT_THCMP CROSS_SHIFT (18U)
00568
00569 #define ADC_SEQ_GDAT_CHN_MASK         (0x3C000000U)
00570 #define ADC_SEQ_GDAT_CHN_SHIFT       (26U)
00571
00572 #define ADC_SEQ_GDAT_OVERRUN_MASK    (0x40000000U)
00573 #define ADC_SEQ_GDAT_OVERRUN_SHIFT   (30U)
00574
00575 #define ADC_SEQ_GDAT_DATAVALID_MASK  (0x80000000U)
00576 #define ADC_SEQ_GDAT_DATAVALID_SHIFT (31U)
00577
00578 #define ADC_DAT_RESULT_MASK          (0xFFFFU)
00579 #define ADC_DAT_RESULT_SHIFT        (4U)
00580

```

```

00581 #define ADC_DAT_THCMRANGE_MASK           (0x30000U)
00582 #define ADC_DAT_THCMRANGE_SHIFT         (16U)
00583
00584 #define ADC_DAT_THCMPCROSS_MASK          (0xC0000U)
00585 #define ADC_DAT_THCMPCROSS_SHIFT         (18U)
00586
00587 #define ADC_DAT_CHANNEL_MASK             (0x3C000000U)
00588 #define ADC_DAT_CHANNEL_SHIFT           (26U)
00589
00590 #define ADC_DAT_OVERRUN_MASK            (0x40000000U)
00591 #define ADC_DAT_OVERRUN_SHIFT           (30U)
00592
00593 #define ADC_DAT_DATAVALID_MASK          (0x80000000U)
00594 #define ADC_DAT_DATAVALID_SHIFT         (31U)
00595
00596 #define ADC_THR0_LOW_THRLLOW_MASK        (0xFFFF0U)
00597 #define ADC_THR0_LOW_THRLLOW_SHIFT      (4U)
00598 #define ADC_THR1_LOW_THRLLOW_MASK        (0xFFFF0U)
00599 #define ADC_THR1_LOW_THRLLOW_SHIFT      (4U)
00600
00601 #define ADC_THR0_HIGH_THRHIGH_MASK       (0xFFFF0U)
00602 #define ADC_THR0_HIGH_THRHIGH_SHIFT     (4U)
00603 #define ADC_THR1_HIGH_THRHIGH_MASK       (0xFFFF0U)
00604 #define ADC_THR1_HIGH_THRHIGH_SHIFT     (4U)
00605
00606 #define ADC_CHAN_THRSEL_CH0_THRSEL_MASK   (0x1U)
00607 #define ADC_CHAN_THRSEL_CH0_THRSEL_SHIFT  (0U)
00608 #define ADC_CHAN_THRSEL_CH1_THRSEL_MASK   (0x2U)
00609 #define ADC_CHAN_THRSEL_CH1_THRSEL_SHIFT  (1U)
00610 #define ADC_CHAN_THRSEL_CH2_THRSEL_MASK   (0x4U)
00611 #define ADC_CHAN_THRSEL_CH2_THRSEL_SHIFT  (2U)
00612 #define ADC_CHAN_THRSEL_CH3_THRSEL_MASK   (0x8U)
00613 #define ADC_CHAN_THRSEL_CH3_THRSEL_SHIFT  (3U)
00614 #define ADC_CHAN_THRSEL_CH4_THRSEL_MASK   (0x10U)
00615 #define ADC_CHAN_THRSEL_CH4_THRSEL_SHIFT  (4U)
00616 #define ADC_CHAN_THRSEL_CH5_THRSEL_MASK   (0x20U)
00617 #define ADC_CHAN_THRSEL_CH5_THRSEL_SHIFT  (5U)
00618 #define ADC_CHAN_THRSEL_CH6_THRSEL_MASK   (0x40U)
00619 #define ADC_CHAN_THRSEL_CH6_THRSEL_SHIFT  (6U)
00620 #define ADC_CHAN_THRSEL_CH7_THRSEL_MASK   (0x80U)
00621 #define ADC_CHAN_THRSEL_CH7_THRSEL_SHIFT  (7U)
00622 #define ADC_CHAN_THRSEL_CH8_THRSEL_MASK   (0x100U)
00623 #define ADC_CHAN_THRSEL_CH8_THRSEL_SHIFT  (8U)
00624 #define ADC_CHAN_THRSEL_CH9_THRSEL_MASK   (0x200U)
00625 #define ADC_CHAN_THRSEL_CH9_THRSEL_SHIFT  (9U)
00626 #define ADC_CHAN_THRSEL_CH10_THRSEL_MASK  (0x400U)
00627 #define ADC_CHAN_THRSEL_CH10_THRSEL_SHIFT (10U)
00628 #define ADC_CHAN_THRSEL_CH11_THRSEL_MASK  (0x800U)
00629 #define ADC_CHAN_THRSEL_CH11_THRSEL_SHIFT (11U)
00630
00631 #define ADC_INTEN_SEQA_INTEN_MASK         (0x1U)
00632 #define ADC_INTEN_SEQA_INTEN_SHIFT       (0U)
00633 #define ADC_INTEN_SEQB_INTEN_MASK         (0x2U)
00634 #define ADC_INTEN_SEQB_INTEN_SHIFT       (1U)
00635
00636 #define ADC_INTEN_OVR_INTEN_MASK         (0x4U)
00637 #define ADC_INTEN_OVR_INTEN_SHIFT       (2U)
00638
00639 #define ADC_INTEN_ADCMPINTENO_MASK       (0x18U)
00640 #define ADC_INTEN_ADCMPINTENO_SHIFT     (3U)
00641 #define ADC_INTEN_ADCMPINTEN1_MASK       (0x60U)
00642 #define ADC_INTEN_ADCMPINTEN1_SHIFT     (5U)
00643 #define ADC_INTEN_ADCMPINTEN2_MASK       (0x180U)
00644 #define ADC_INTEN_ADCMPINTEN2_SHIFT     (7U)
00645 #define ADC_INTEN_ADCMPINTEN3_MASK       (0x600U)
00646 #define ADC_INTEN_ADCMPINTEN3_SHIFT     (9U)
00647 #define ADC_INTEN_ADCMPINTEN4_MASK       (0x1800U)
00648 #define ADC_INTEN_ADCMPINTEN4_SHIFT     (11U)
00649 #define ADC_INTEN_ADCMPINTEN5_MASK       (0x6000U)
00650 #define ADC_INTEN_ADCMPINTEN5_SHIFT     (13U)
00651 #define ADC_INTEN_ADCMPINTEN6_MASK       (0x18000U)
00652 #define ADC_INTEN_ADCMPINTEN6_SHIFT     (15U)
00653 #define ADC_INTEN_ADCMPINTEN7_MASK       (0x60000U)
00654 #define ADC_INTEN_ADCMPINTEN7_SHIFT     (17U)
00655 #define ADC_INTEN_ADCMPINTEN8_MASK       (0x180000U)
00656 #define ADC_INTEN_ADCMPINTEN8_SHIFT     (19U)
00657 #define ADC_INTEN_ADCMPINTEN9_MASK       (0x600000U)
00658 #define ADC_INTEN_ADCMPINTEN9_SHIFT     (21U)
00659 #define ADC_INTEN_ADCMPINTEN10_MASK      (0x1800000U)
00660 #define ADC_INTEN_ADCMPINTEN10_SHIFT    (23U)
00661 #define ADC_INTEN_ADCMPINTEN11_MASK      (0x6000000U)
00662 #define ADC_INTEN_ADCMPINTEN11_SHIFT    (25U)
00663
00664 #define ADC_FLAGS_THCMPO_MASK            (0x1U)
00665 #define ADC_FLAGS_THCMPO_SHIFT          (0U)
00666 #define ADC_FLAGS_THCMPI1_MASK          (0x2U)
00667 #define ADC_FLAGS_THCMPI1_SHIFT        (1U)

```

```

00668 #define ADC_FLAGS_THCMP2_MASK           (0x4U)
00669 #define ADC_FLAGS_THCMP2_SHIFT         (2U)
00670 #define ADC_FLAGS_THCMP3_MASK           (0x8U)
00671 #define ADC_FLAGS_THCMP3_SHIFT         (3U)
00672 #define ADC_FLAGS_THCMP4_MASK           (0x10U)
00673 #define ADC_FLAGS_THCMP4_SHIFT         (4U)
00674 #define ADC_FLAGS_THCMP5_MASK           (0x20U)
00675 #define ADC_FLAGS_THCMP5_SHIFT         (5U)
00676 #define ADC_FLAGS_THCMP6_MASK           (0x40U)
00677 #define ADC_FLAGS_THCMP6_SHIFT         (6U)
00678 #define ADC_FLAGS_THCMP7_MASK           (0x80U)
00679 #define ADC_FLAGS_THCMP7_SHIFT         (7U)
00680 #define ADC_FLAGS_THCMP8_MASK           (0x100U)
00681 #define ADC_FLAGS_THCMP8_SHIFT         (8U)
00682 #define ADC_FLAGS_THCMP9_MASK           (0x200U)
00683 #define ADC_FLAGS_THCMP9_SHIFT         (9U)
00684 #define ADC_FLAGS_THCMP10_MASK          (0x400U)
00685 #define ADC_FLAGS_THCMP10_SHIFT        (10U)
00686 #define ADC_FLAGS_THCMP11_MASK          (0x800U)
00687 #define ADC_FLAGS_THCMP11_SHIFT        (11U)
00688
00689 #define ADC_FLAGS_OVERRUN0_MASK          (0x1000U)
00690 #define ADC_FLAGS_OVERRUN0_SHIFT        (12U)
00691 #define ADC_FLAGS_OVERRUN1_MASK          (0x2000U)
00692 #define ADC_FLAGS_OVERRUN1_SHIFT        (13U)
00693 #define ADC_FLAGS_OVERRUN2_MASK          (0x4000U)
00694 #define ADC_FLAGS_OVERRUN2_SHIFT        (14U)
00695 #define ADC_FLAGS_OVERRUN3_MASK          (0x8000U)
00696 #define ADC_FLAGS_OVERRUN3_SHIFT        (15U)
00697 #define ADC_FLAGS_OVERRUN4_MASK          (0x10000U)
00698 #define ADC_FLAGS_OVERRUN4_SHIFT        (16U)
00699 #define ADC_FLAGS_OVERRUN5_MASK          (0x20000U)
00700 #define ADC_FLAGS_OVERRUN5_SHIFT        (17U)
00701 #define ADC_FLAGS_OVERRUN6_MASK          (0x40000U)
00702 #define ADC_FLAGS_OVERRUN6_SHIFT        (18U)
00703 #define ADC_FLAGS_OVERRUN7_MASK          (0x80000U)
00704 #define ADC_FLAGS_OVERRUN7_SHIFT        (19U)
00705 #define ADC_FLAGS_OVERRUN8_MASK          (0x100000U)
00706 #define ADC_FLAGS_OVERRUN8_SHIFT        (20U)
00707 #define ADC_FLAGS_OVERRUN9_MASK          (0x200000U)
00708 #define ADC_FLAGS_OVERRUN9_SHIFT        (21U)
00709 #define ADC_FLAGS_OVERRUN10_MASK         (0x400000U)
00710 #define ADC_FLAGS_OVERRUN10_SHIFT       (22U)
00711 #define ADC_FLAGS_OVERRUN11_MASK         (0x800000U)
00712 #define ADC_FLAGS_OVERRUN11_SHIFT       (23U)
00713
00714 #define ADC_FLAGS_SEQA_OVR_MASK          (0x1000000U)
00715 #define ADC_FLAGS_SEQA_OVR_SHIFT        (24U)
00716
00717 #define ADC_FLAGS_SEQB_OVR_MASK          (0x2000000U)
00718 #define ADC_FLAGS_SEQB_OVR_SHIFT        (25U)
00719
00720 #define ADC_FLAGS_SEQA_INT_MASK         (0x10000000U)
00721 #define ADC_FLAGS_SEQA_INT_SHIFT       (28U)
00722 #define ADC_FLAGS_SEQB_INT_MASK         (0x20000000U)
00723 #define ADC_FLAGS_SEQB_INT_SHIFT       (29U)
00724 #define ADC_FLAGS_THCMP_INT_MASK        (0x40000000U)
00725 #define ADC_FLAGS_THCMP_INT_SHIFT      (30U)
00726 #define ADC_FLAGS_OVR_INT_MASK         (0x80000000U)
00727 #define ADC_FLAGS_OVR_INT_SHIFT       (31U)
00728 #define ADC_TRM_VRANGE_MASK           (0x20U)
00729 #define ADC_TRM_VRANGE_SHIFT         (5U)
00731
00740 typedef struct {
00741     __RW uint32_t CTRL;
00742     uint8_t RESERVED_0[4];
00743     __RW uint32_t SEQ_CTRL[2];
00744     __R  uint32_t SEQ_GDAT[2];
00745     uint8_t RESERVED_1[8];
00746     __R  uint32_t DAT[12];
00747     __RW uint32_t THR0_LOW;
00748     __RW uint32_t THR1_LOW;
00749     __RW uint32_t THR0_HIGH;
00750     __RW uint32_t THR1_HIGH;
00751     __RW uint32_t CHAN_THRSEL;
00752     __RW uint32_t INTEN;
00753     __RW uint32_t FLAGS;
00754     __RW uint32_t TRM;
00755 } ADC_Type;
00756
00757 /* -----
00758 -- ADC Register Masks
00759 ----- */
00760
00765
00768
00775 #define ADC_CTRL_CLKDIV(x)             (((uint32_t)((uint32_t)(x)) <<

```

```

ADC_CTRL_CLKDIV_SHIFT)) & ADC_CTRL_CLKDIV_MASK)
00776 #define ADC_CTRL_ASYNMODE(x) (((uint32_t)((uint32_t)(x)) <<
ADC_CTRL_ASYNMODE_SHIFT)) & ADC_CTRL_ASYNMODE_MASK)
00787
00800 #define ADC_CTRL_LPWRMODE(x) (((uint32_t)((uint32_t)(x)) <<
ADC_CTRL_LPWRMODE_SHIFT)) & ADC_CTRL_LPWRMODE_MASK)
00801
00808 #define ADC_CTRL_CALMODE(x) (((uint32_t)((uint32_t)(x)) <<
ADC_CTRL_CALMODE_SHIFT)) & ADC_CTRL_CALMODE_MASK)
00810
00813
00822 #define ADC_SEQ_CTRL_CHANNELS(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_CHANNELS_SHIFT)) & ADC_SEQ_CTRL_CHANNELS_MASK)
00823
00829 #define ADC_SEQ_CTRL_TRIGGER(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_TRIGGER_SHIFT)) & ADC_SEQ_CTRL_TRIGGER_MASK)
00830
00837 #define ADC_SEQ_CTRL_TRIGPOL(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_TRIGPOL_SHIFT)) & ADC_SEQ_CTRL_TRIGPOL_MASK)
00838
00854 #define ADC_SEQ_CTRL_SYNCBYPASS(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_SYNCBYPASS_SHIFT)) & ADC_SEQ_CTRL_SYNCBYPASS_MASK)
00855
00861 #define ADC_SEQ_CTRL_START(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_START_SHIFT)) & ADC_SEQ_CTRL_START_MASK)
00862
00868 #define ADC_SEQ_CTRL_BURST(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_BURST_SHIFT)) & ADC_SEQ_CTRL_BURST_MASK)
00869
00877 #define ADC_SEQ_CTRL_SINGLESTEP(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_SINGLESTEP_SHIFT)) & ADC_SEQ_CTRL_SINGLESTEP_MASK)
00878
00887 #define ADC_SEQ_CTRL_LOWPRIO(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_LOWPRIO_SHIFT)) & ADC_SEQ_CTRL_LOWPRIO_MASK)
00888
00903 #define ADC_SEQ_CTRL_MODE(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_MODE_SHIFT)) & ADC_SEQ_CTRL_MODE_MASK)
00904
00917 #define ADC_SEQ_CTRL_SEQ_ENA(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_CTRL_SEQ_ENA_SHIFT)) & ADC_SEQ_CTRL_SEQ_ENA_MASK)
00919
00921 #define ADC_SEQ_CTRL_COUNT (2U)
00922
00925
00934 #define ADC_SEQ_GDAT_RESULT(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_GDAT_RESULT_SHIFT)) & ADC_SEQ_GDAT_RESULT_MASK)
00935
00939 #define ADC_SEQ_GDAT_THCMPRANGE(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_GDAT_THCMPRANGE_SHIFT)) & ADC_SEQ_GDAT_THCMPRANGE_MASK)
00940
00944 #define ADC_SEQ_GDAT_THCMPCROSS(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_GDAT_THCMPCROSS_SHIFT)) & ADC_SEQ_GDAT_THCMPCROSS_MASK)
00945
00948 #define ADC_SEQ_GDAT_CHN(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_GDAT_CHN_SHIFT)) & ADC_SEQ_GDAT_CHN_MASK)
00949
00955 #define ADC_SEQ_GDAT_OVERRUN(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_GDAT_OVERRUN_SHIFT)) & ADC_SEQ_GDAT_OVERRUN_MASK)
00956
00961 #define ADC_SEQ_GDAT_DATAVALID(x) (((uint32_t)((uint32_t)(x)) <<
ADC_SEQ_GDAT_DATAVALID_SHIFT)) & ADC_SEQ_GDAT_DATAVALID_MASK)
00963
00965 #define ADC_SEQ_GDAT_COUNT (2U)
00966
00969
00975 #define ADC_DAT_RESULT(x) (((uint32_t)((uint32_t)(x)) << ADC_DAT_RESULT_SHIFT))
& ADC_DAT_RESULT_MASK)
00976
00984 #define ADC_DAT_THCMPRANGE(x) (((uint32_t)((uint32_t)(x)) <<
ADC_DAT_THCMPRANGE_SHIFT)) & ADC_DAT_THCMPRANGE_MASK)
00985
00996 #define ADC_DAT_THCMPCROSS(x) (((uint32_t)((uint32_t)(x)) <<
ADC_DAT_THCMPCROSS_SHIFT)) & ADC_DAT_THCMPCROSS_MASK)
00997
01001 #define ADC_DAT_CHANNEL(x) (((uint32_t)((uint32_t)(x)) <<
ADC_DAT_CHANNEL_SHIFT)) & ADC_DAT_CHANNEL_MASK)
01002
01012 #define ADC_DAT_OVERRUN(x) (((uint32_t)((uint32_t)(x)) <<
ADC_DAT_OVERRUN_SHIFT)) & ADC_DAT_OVERRUN_MASK)
01013
01020 #define ADC_DAT_DATAVALID(x) (((uint32_t)((uint32_t)(x)) <<
ADC_DAT_DATAVALID_SHIFT)) & ADC_DAT_DATAVALID_MASK)
01022
01024 #define ADC_DAT_COUNT (12U)
01025
01028

```

```

01030 #define ADC_THR0_LOW_THRLow(x) ((uint32_t)((uint32_t)(x) <<
01031     ADC_THR0_LOW_THRLow_SHIFT)) & ADC_THR0_LOW_THRLow_MASK)
01032
01035
01037 #define ADC_THR1_LOW_THRLow(x) ((uint32_t)((uint32_t)(x) <<
01038     ADC_THR1_LOW_THRLow_SHIFT)) & ADC_THR1_LOW_THRLow_MASK)
01039
01042
01044 #define ADC_THR0_HIGH_THRHIGH(x) ((uint32_t)((uint32_t)(x) <<
01045     ADC_THR0_HIGH_THRHIGH_SHIFT)) & ADC_THR0_HIGH_THRHIGH_MASK)
01046
01049
01051 #define ADC_THR1_HIGH_THRHIGH(x) ((uint32_t)((uint32_t)(x) <<
01052     ADC_THR1_HIGH_THRHIGH_SHIFT)) & ADC_THR1_HIGH_THRHIGH_MASK)
01053
01056
01060 #define ADC_CHAN_THRSEL_CH0_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01061     ADC_CHAN_THRSEL_CH0_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH0_THRSEL_MASK)
01062 #define ADC_CHAN_THRSEL_CH1_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01063     ADC_CHAN_THRSEL_CH1_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH1_THRSEL_MASK)
01064 #define ADC_CHAN_THRSEL_CH2_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01065     ADC_CHAN_THRSEL_CH2_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH2_THRSEL_MASK)
01066 #define ADC_CHAN_THRSEL_CH3_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01067     ADC_CHAN_THRSEL_CH3_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH3_THRSEL_MASK)
01068 #define ADC_CHAN_THRSEL_CH4_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01069     ADC_CHAN_THRSEL_CH4_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH4_THRSEL_MASK)
01070 #define ADC_CHAN_THRSEL_CH5_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01071     ADC_CHAN_THRSEL_CH5_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH5_THRSEL_MASK)
01072 #define ADC_CHAN_THRSEL_CH6_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01073     ADC_CHAN_THRSEL_CH6_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH6_THRSEL_MASK)
01074 #define ADC_CHAN_THRSEL_CH7_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01075     ADC_CHAN_THRSEL_CH7_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH7_THRSEL_MASK)
01076 #define ADC_CHAN_THRSEL_CH8_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01077     ADC_CHAN_THRSEL_CH8_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH8_THRSEL_MASK)
01078 #define ADC_CHAN_THRSEL_CH9_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01079     ADC_CHAN_THRSEL_CH9_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH9_THRSEL_MASK)
01080 #define ADC_CHAN_THRSEL_CH10_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01081     ADC_CHAN_THRSEL_CH10_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH10_THRSEL_MASK)
01082 #define ADC_CHAN_THRSEL_CH11_THRSEL(x) ((uint32_t)((uint32_t)(x) <<
01083     ADC_CHAN_THRSEL_CH11_THRSEL_SHIFT)) & ADC_CHAN_THRSEL_CH11_THRSEL_MASK)
01084
01087
01093 #define ADC_INTEN_SEQA_INTEN(x) ((uint32_t)((uint32_t)(x) <<
01094     ADC_INTEN_SEQA_INTEN_SHIFT)) & ADC_INTEN_SEQA_INTEN_MASK)
01094
01100 #define ADC_INTEN_SEQB_INTEN(x) ((uint32_t)((uint32_t)(x) <<
01101     ADC_INTEN_SEQB_INTEN_SHIFT)) & ADC_INTEN_SEQB_INTEN_MASK)
01108 #define ADC_INTEN_OVR_INTEN(x) ((uint32_t)((uint32_t)(x) <<
01109     ADC_INTEN_OVR_INTEN_SHIFT)) & ADC_INTEN_OVR_INTEN_MASK)
01115 #define ADC_INTEN_ADCMPINTENO(x) ((uint32_t)((uint32_t)(x) <<
01116     ADC_INTEN_ADCMPINTENO_SHIFT)) & ADC_INTEN_ADCMPINTENO_MASK)
01117 #define ADC_INTEN_ADCMPINTEN1(x) ((uint32_t)((uint32_t)(x) <<
01118     ADC_INTEN_ADCMPINTEN1_SHIFT)) & ADC_INTEN_ADCMPINTEN1_MASK)
01119 #define ADC_INTEN_ADCMPINTEN2(x) ((uint32_t)((uint32_t)(x) <<
01120     ADC_INTEN_ADCMPINTEN2_SHIFT)) & ADC_INTEN_ADCMPINTEN2_MASK)
01121 #define ADC_INTEN_ADCMPINTEN3(x) ((uint32_t)((uint32_t)(x) <<
01122     ADC_INTEN_ADCMPINTEN3_SHIFT)) & ADC_INTEN_ADCMPINTEN3_MASK)
01123 #define ADC_INTEN_ADCMPINTEN4(x) ((uint32_t)((uint32_t)(x) <<
01124     ADC_INTEN_ADCMPINTEN4_SHIFT)) & ADC_INTEN_ADCMPINTEN4_MASK)
01125 #define ADC_INTEN_ADCMPINTEN5(x) ((uint32_t)((uint32_t)(x) <<
01126     ADC_INTEN_ADCMPINTEN5_SHIFT)) & ADC_INTEN_ADCMPINTEN5_MASK)
01127 #define ADC_INTEN_ADCMPINTEN6(x) ((uint32_t)((uint32_t)(x) <<
01128     ADC_INTEN_ADCMPINTEN6_SHIFT)) & ADC_INTEN_ADCMPINTEN6_MASK)
01129 #define ADC_INTEN_ADCMPINTEN7(x) ((uint32_t)((uint32_t)(x) <<
01130     ADC_INTEN_ADCMPINTEN7_SHIFT)) & ADC_INTEN_ADCMPINTEN7_MASK)
01131 #define ADC_INTEN_ADCMPINTEN8(x) ((uint32_t)((uint32_t)(x) <<
01132     ADC_INTEN_ADCMPINTEN8_SHIFT)) & ADC_INTEN_ADCMPINTEN8_MASK)
01133 #define ADC_INTEN_ADCMPINTEN9(x) ((uint32_t)((uint32_t)(x) <<
01134     ADC_INTEN_ADCMPINTEN9_SHIFT)) & ADC_INTEN_ADCMPINTEN9_MASK)
01135 #define ADC_INTEN_ADCMPINTEN10(x) ((uint32_t)((uint32_t)(x) <<
01136     ADC_INTEN_ADCMPINTEN10_SHIFT)) & ADC_INTEN_ADCMPINTEN10_MASK)
01137 #define ADC_INTEN_ADCMPINTEN11(x) ((uint32_t)((uint32_t)(x) <<
01138     ADC_INTEN_ADCMPINTEN11_SHIFT)) & ADC_INTEN_ADCMPINTEN11_MASK)
01139
01142
01146 #define ADC_FLAGS_THCMP0(x) ((uint32_t)((uint32_t)(x) <<
01147     ADC_FLAGS_THCMP0_SHIFT)) & ADC_FLAGS_THCMP0_MASK)
01148 #define ADC_FLAGS_THCMP1(x) ((uint32_t)((uint32_t)(x) <<
01149     ADC_FLAGS_THCMP1_SHIFT)) & ADC_FLAGS_THCMP1_MASK)
01150 #define ADC_FLAGS_THCMP2(x) ((uint32_t)((uint32_t)(x) <<
01151     ADC_FLAGS_THCMP2_SHIFT)) & ADC_FLAGS_THCMP2_MASK)
01152 #define ADC_FLAGS_THCMP3(x) ((uint32_t)((uint32_t)(x) <<
01153     ADC_FLAGS_THCMP3_SHIFT)) & ADC_FLAGS_THCMP3_MASK)
01154 #define ADC_FLAGS_THCMP4(x) ((uint32_t)((uint32_t)(x) <<
01155     ADC_FLAGS_THCMP4_SHIFT)) & ADC_FLAGS_THCMP4_MASK)

```

```

01156 #define ADC_FLAGS_THCMP5(x) (((uint32_t)((uint32_t)(x)) <<
01157   ADC_FLAGS_THCMP5_SHIFT) & ADC_FLAGS_THCMP5_MASK)
01158 #define ADC_FLAGS_THCMP6(x) (((uint32_t)((uint32_t)(x)) <<
01159   ADC_FLAGS_THCMP6_SHIFT) & ADC_FLAGS_THCMP6_MASK)
01160 #define ADC_FLAGS_THCMP7(x) (((uint32_t)((uint32_t)(x)) <<
01161   ADC_FLAGS_THCMP7_SHIFT) & ADC_FLAGS_THCMP7_MASK)
01162 #define ADC_FLAGS_THCMP8(x) (((uint32_t)((uint32_t)(x)) <<
01163   ADC_FLAGS_THCMP8_SHIFT) & ADC_FLAGS_THCMP8_MASK)
01164 #define ADC_FLAGS_THCMP9(x) (((uint32_t)((uint32_t)(x)) <<
01165   ADC_FLAGS_THCMP9_SHIFT) & ADC_FLAGS_THCMP9_MASK)
01166 #define ADC_FLAGS_THCMP10(x) (((uint32_t)((uint32_t)(x)) <<
01167   ADC_FLAGS_THCMP10_SHIFT) & ADC_FLAGS_THCMP10_MASK)
01168 #define ADC_FLAGS_THCMP11(x) (((uint32_t)((uint32_t)(x)) <<
01169   ADC_FLAGS_THCMP11_SHIFT) & ADC_FLAGS_THCMP11_MASK)
01170 #define ADC_FLAGS_OVERRUN0(x) (((uint32_t)((uint32_t)(x)) <<
01171   ADC_FLAGS_OVERRUN0_SHIFT) & ADC_FLAGS_OVERRUN0_MASK)
01172 #define ADC_FLAGS_OVERRUN1(x) (((uint32_t)((uint32_t)(x)) <<
01173   ADC_FLAGS_OVERRUN1_SHIFT) & ADC_FLAGS_OVERRUN1_MASK)
01174 #define ADC_FLAGS_OVERRUN2(x) (((uint32_t)((uint32_t)(x)) <<
01175   ADC_FLAGS_OVERRUN2_SHIFT) & ADC_FLAGS_OVERRUN2_MASK)
01176 #define ADC_FLAGS_OVERRUN3(x) (((uint32_t)((uint32_t)(x)) <<
01177   ADC_FLAGS_OVERRUN3_SHIFT) & ADC_FLAGS_OVERRUN3_MASK)
01178 #define ADC_FLAGS_OVERRUN4(x) (((uint32_t)((uint32_t)(x)) <<
01179   ADC_FLAGS_OVERRUN4_SHIFT) & ADC_FLAGS_OVERRUN4_MASK)
01180 #define ADC_FLAGS_OVERRUN5(x) (((uint32_t)((uint32_t)(x)) <<
01181   ADC_FLAGS_OVERRUN5_SHIFT) & ADC_FLAGS_OVERRUN5_MASK)
01182 #define ADC_FLAGS_OVERRUN6(x) (((uint32_t)((uint32_t)(x)) <<
01183   ADC_FLAGS_OVERRUN6_SHIFT) & ADC_FLAGS_OVERRUN6_MASK)
01184 #define ADC_FLAGS_OVERRUN7(x) (((uint32_t)((uint32_t)(x)) <<
01185   ADC_FLAGS_OVERRUN7_SHIFT) & ADC_FLAGS_OVERRUN7_MASK)
01186 #define ADC_FLAGS_OVERRUN8(x) (((uint32_t)((uint32_t)(x)) <<
01187   ADC_FLAGS_OVERRUN8_SHIFT) & ADC_FLAGS_OVERRUN8_MASK)
01188 #define ADC_FLAGS_OVERRUN9(x) (((uint32_t)((uint32_t)(x)) <<
01189   ADC_FLAGS_OVERRUN9_SHIFT) & ADC_FLAGS_OVERRUN9_MASK)
01190 #define ADC_FLAGS_OVERRUN10(x) (((uint32_t)((uint32_t)(x)) <<
01191   ADC_FLAGS_OVERRUN10_SHIFT) & ADC_FLAGS_OVERRUN10_MASK)
01192 #define ADC_FLAGS_OVERRUN11(x) (((uint32_t)((uint32_t)(x)) <<
01193   ADC_FLAGS_OVERRUN11_SHIFT) & ADC_FLAGS_OVERRUN11_MASK)
01194 #define ADC_FLAGS_SEQA_OVR(x) (((uint32_t)((uint32_t)(x)) <<
01195   ADC_FLAGS_SEQA_OVR_SHIFT) & ADC_FLAGS_SEQA_OVR_MASK)
01196 #define ADC_FLAGS_SEQB_OVR(x) (((uint32_t)((uint32_t)(x)) <<
01197   ADC_FLAGS_SEQB_OVR_SHIFT) & ADC_FLAGS_SEQB_OVR_MASK)
01204 #define ADC_FLAGS_SEQA_INT(x) (((uint32_t)((uint32_t)(x)) <<
01205   ADC_FLAGS_SEQA_INT_SHIFT) & ADC_FLAGS_SEQA_INT_MASK)
01212 #define ADC_FLAGS_SEQB_INT(x) (((uint32_t)((uint32_t)(x)) <<
01213   ADC_FLAGS_SEQB_INT_SHIFT) & ADC_FLAGS_SEQB_INT_MASK)
01213
01219 #define ADC_FLAGS_THCMP_INT(x) (((uint32_t)((uint32_t)(x)) <<
01220   ADC_FLAGS_THCMP_INT_SHIFT) & ADC_FLAGS_THCMP_INT_MASK)
01226 #define ADC_FLAGS_OVR_INT(x) (((uint32_t)((uint32_t)(x)) <<
01227   ADC_FLAGS_OVR_INT_SHIFT) & ADC_FLAGS_OVR_INT_MASK)
01228
01231
01235 #define ADC_TRM_VRANGE(x) (((uint32_t)((uint32_t)(x)) << ADC_TRM_VRANGE_SHIFT))
01236   & ADC_TRM_VRANGE_MASK)
01237 /* end of group ADC_Register_Masks */
01241
01242
01243 /* ADC - Peripheral instance base addresses */
01245 #define ADC0_BASE (0x4001C000u)
01247 #define ADC0 ((ADC_Type *)ADC0_BASE)
01249 #define ADC_BASE_ADDRS { ADC0_BASE }
01251 #define ADC_BASE_PTRS { ADC0 }
01253 #define ADC_SEQ_IIRQS { ADC0_SEQA IRQn, ADC0_SEQB IRQn }
01255 #define ADC_THCMP_IIRQS { ADC0_THCMP IRQn }
01260 /* -----
01261   -- DAC Peripheral Access Layer
01262   ----- */
01271 typedef struct {
01272
01273   union {
01274     struct {
01275       __RW uint32_t RESERVED0 :6;
01276       __RW uint32_t VALUE :10;
01277       __RW uint32_t BIAS :1;
01278       __RW uint32_t RESERVED1 :15;
01279     };
01280     __RW uint32_t CR;
01281   };
01282   union{
01283     struct{
01284       __RW uint32_t INT_DMA_REQ :1;
01285       __RW uint32_t DBLBUF_ENA :1;
01286       __RW uint32_t CNT_ENA :1;
01287       __RW uint32_t DMA_FNA :1;
01288       __RW uint32_t RESERVED2 :28;

```

```

01289     );
01290     __RW uint32_t CTRL;
01291 };
01292
01293     __RW uint32_t CNTVAL;
01294
01295 } DAC_t;
01297 #define DAC0      ( (DAC_t*) 0x40014000UL )
01299 #define DAC1      ( (DAC_t*) 0x40018000UL )
01300 /* end of group DAC_Peripheral_Access_Layer */
01304
01305 /* -----
01306   -- I2C Peripheral Access Layer
01307   ----- */
01308
01313
01315 typedef struct {
01316     __RW uint32_t CFG;
01317     __RW uint32_t STAT;
01318     __RW uint32_t INTENSET;
01319     __W uint32_t INTENCLR;
01320     __RW uint32_t TIMEOUT;
01321     __RW uint32_t CLKDIV;
01322     __R uint32_t INTSTAT;
01323     uint8_t RESERVED_0[4];
01324     __RW uint32_t MSTCTL;
01325     __RW uint32_t MSTTIME;
01326     __RW uint32_t MSTDAT;
01327     uint8_t RESERVED_1[20];
01328     __RW uint32_t SLVCTL;
01329     __RW uint32_t SLVDAT;
01330     __RW uint32_t SLVADR[4];
01331     __RW uint32_t SLVQUAL0;
01332     uint8_t RESERVED_2[36];
01333     __R uint32_t MONRXDAT;
01334 } I2C_Type;
01335
01336 /* -----
01337   -- I2C Register Masks
01338   ----- */
01339
01344
01347 #define I2C_CFG_MSTEN_MASK          (0x1U)
01348 #define I2C_CFG_MSTEN_SHIFT         (0U)
01354 #define I2C_CFG_MSTEN(x)            (((uint32_t)((uint32_t)(x)) << I2C_CFG_MSTEN_SHIFT))
01355 #define I2C_CFG_SLVEN_MASK          (0x2U)
01356 #define I2C_CFG_SLVEN_SHIFT         (1U)
01362 #define I2C_CFG_SLVEN(x)            (((uint32_t)((uint32_t)(x)) << I2C_CFG_SLVEN_SHIFT))
01363 #define I2C_CFG_MONEN_MASK          (0x4U)
01364 #define I2C_CFG_MONEN_SHIFT         (2U)
01370 #define I2C_CFG_MONEN(x)            (((uint32_t)((uint32_t)(x)) << I2C_CFG_MONEN_SHIFT))
01371 #define I2C_CFG_TIMEOUTUTEN_MASK    (0x8U)
01372 #define I2C_CFG_TIMEOUTUTEN_SHIFT   (3U)
01378 #define I2C_CFG_TIMEOUTUTEN(x)      (((uint32_t)((uint32_t)(x)) <<
01379 #define I2C_CFG_MONCLKSTR_MASK     (0x10U)
01380 #define I2C_CFG_MONCLKSTR_SHIFT    (4U)
01388 #define I2C_CFG_MONCLKSTR(x)       (((uint32_t)((uint32_t)(x)) <<
01390
01393 #define I2C_STAT_MSTPENDING_MASK    (0x1U)
01394 #define I2C_STAT_MSTPENDING_SHIFT   (0U)
01405 #define I2C_STAT_MSTPENDING(x)      (((uint32_t)((uint32_t)(x)) <<
01406 #define I2C_STAT_MSTSTATE_MASK     (0xEU)
01407 #define I2C_STAT_MSTSTATE_SHIFT    (1U)
01418 #define I2C_STAT_MSTSTATE(x)       (((uint32_t)((uint32_t)(x)) <<
01419 #define I2C_STAT_MSTARLOSS_MASK    (0x10U)
01420 #define I2C_STAT_MSTARLOSS_SHIFT   (4U)
01428 #define I2C_STAT_MSTARLOSS(x)      (((uint32_t)((uint32_t)(x)) <<
01429 #define I2C_STAT_MSTSTSTPERR_MASK  (0x40U)
01430 #define I2C_STAT_MSTSTSTPERR_SHIFT (6U)
01439 #define I2C_STAT_MSTSTSTPERR(x)    (((uint32_t)((uint32_t)(x)) <<
01440 #define I2C_STAT_SLVPENDING_MASK   (0x100U)
01441 #define I2C_STAT_SLVPENDING_SHIFT (8U)
01455 #define I2C_STAT_SLVPENDING(x)     (((uint32_t)((uint32_t)(x)) <<
01456 #define I2C_STAT_SLVSTATE_MASK    (0x600U)
01457 #define I2C_STAT_SLVSTATE_SHIFT   (9U)
01466 #define I2C_STAT_SLVSTATE(x)      (((uint32_t)((uint32_t)(x)) <<
01467 #define I2C_STAT_SLVSTATE_SHIFT   (10U)
```

```

01467 #define I2C_STAT_SLVNOTSTR_MASK           (0x800U)
01468 #define I2C_STAT_SLVNOTSTR_SHIFT         (11U)
01476 #define I2C_STAT_SLVNOTSTR(x)             (((uint32_t)((uint32_t)(x)) <<
01477     I2C_STAT_SLVNOTSTR_SHIFT) & I2C_STAT_SLVNOTSTR_MASK)
01478 #define I2C_STAT_SLVIDX_MASK              (0x3000U)
01479 #define I2C_STAT_SLVIDX_SHIFT             (12U)
01488 #define I2C_STAT_SLVIDX(x)                (((uint32_t)((uint32_t)(x)) <<
01489     I2C_STAT_SLVIDX_SHIFT) & I2C_STAT_SLVIDX_MASK)
01490 #define I2C_STAT_SLVSEL_MASK              (0x4000U)
01491 #define I2C_STAT_SLVSEL_SHIFT             (14U)
01500 #define I2C_STAT_SLVSEL(x)                (((uint32_t)((uint32_t)(x)) <<
01501     I2C_STAT_SLVSEL_SHIFT) & I2C_STAT_SLVSEL_MASK)
01502 #define I2C_STAT_SLVDESEL_MASK            (0x8000U)
01503 #define I2C_STAT_SLVDESEL_SHIFT           (15U)
01510 #define I2C_STAT_SLVDESEL(x)               (((uint32_t)((uint32_t)(x)) <<
01511     I2C_STAT_SLVDESEL_SHIFT) & I2C_STAT_SLVDESEL_MASK)
01512 #define I2C_STAT_MONRDY_MASK              (0x10000U)
01513 #define I2C_STAT_MONRDY_SHIFT             (16U)
01517 #define I2C_STAT_MONRDY(x)                (((uint32_t)((uint32_t)(x)) <<
01518     I2C_STAT_MONRDY_SHIFT) & I2C_STAT_MONRDY_MASK)
01519 #define I2C_STAT_MONOV_MASK              (0x20000U)
01520 #define I2C_STAT_MONOV_SHIFT             (17U)
01525 #define I2C_STAT_MONOV(x)                (((uint32_t)((uint32_t)(x)) <<
01526     I2C_STAT_MONOV_SHIFT) & I2C_STAT_MONOV_MASK)
01527 #define I2C_STAT_MONACTIVE_MASK            (0x40000U)
01528 #define I2C_STAT_MONACTIVE_SHIFT          (18U)
01534 #define I2C_STAT_MONACTIVE(x)              (((uint32_t)((uint32_t)(x)) <<
01535     I2C_STAT_MONACTIVE_SHIFT) & I2C_STAT_MONACTIVE_MASK)
01536 #define I2C_STAT_MONIDLE_MASK              (0x80000U)
01537 #define I2C_STAT_MONIDLE_SHIFT             (19U)
01544 #define I2C_STAT_MONIDLE(x)                (((uint32_t)((uint32_t)(x)) <<
01545     I2C_STAT_MONIDLE_SHIFT) & I2C_STAT_MONIDLE_MASK)
01546 #define I2C_STAT_EVENTTIMEOUT_MASK        (0x1000000U)
01547 #define I2C_STAT_EVENTTIMEOUT_SHIFT       (24U)
01554 #define I2C_STAT_EVENTTIMEOUT(x)          (((uint32_t)((uint32_t)(x)) <<
01555     I2C_STAT_EVENTTIMEOUT_SHIFT) & I2C_STAT_EVENTTIMEOUT_MASK)
01556 #define I2C_STAT_SCLTIMEOUT_MASK            (0x2000000U)
01557 #define I2C_STAT_SCLTIMEOUT_SHIFT          (25U)
01562 #define I2C_STAT_SCLTIMEOUT(x)              (((uint32_t)((uint32_t)(x)) <<
01563     I2C_STAT_SCLTIMEOUT_SHIFT) & I2C_STAT_SCLTIMEOUT_MASK)
01564
01567 #define I2C_INTENSET_MSTPENDINGEN_MASK    (0x1U)
01568 #define I2C_INTENSET_MSTPENDINGEN_SHIFT   (0U)
01573 #define I2C_INTENSET_MSTPENDINGEN(x)       (((uint32_t)((uint32_t)(x)) <<
01574     I2C_INTENSET_MSTPENDINGEN_SHIFT) & I2C_INTENSET_MSTPENDINGEN_MASK)
01575 #define I2C_INTENSET_MSTARBLOSSEN_MASK    (0x10U)
01576 #define I2C_INTENSET_MSTARBLOSSEN_SHIFT   (4U)
01580 #define I2C_INTENSET_MSTARBLOSSEN(x)       (((uint32_t)((uint32_t)(x)) <<
01581     I2C_INTENSET_MSTARBLOSSEN_SHIFT) & I2C_INTENSET_MSTARBLOSSEN_MASK)
01582 #define I2C_INTENSET_MSTSTSTPERREN_MASK   (0x40U)
01583 #define I2C_INTENSET_MSTSTSTPERREN_SHIFT  (6U)
01587 #define I2C_INTENSET_MSTSTSTPERREN(x)       (((uint32_t)((uint32_t)(x)) <<
01588 #define I2C_INTENSET_SLVPENDINGEN_MASK    (0x100U)
01589 #define I2C_INTENSET_SLVPENDINGEN_SHIFT   (8U)
01594 #define I2C_INTENSET_SLVPENDINGEN(x)       (((uint32_t)((uint32_t)(x)) <<
01595 #define I2C_INTENSET_SLVNOTSTREN_MASK     (0x800U)
01596 #define I2C_INTENSET_SLVNOTSTREN_SHIFT   (11U)
01601 #define I2C_INTENSET_SLVNOTSTREN(x)       (((uint32_t)((uint32_t)(x)) <<
01602 #define I2C_INTENSET_SLVDESELEN_MASK     (0x8000U)
01603 #define I2C_INTENSET_SLVDESELEN_SHIFT   (15U)
01608 #define I2C_INTENSET_SLVDESELEN(x)       (((uint32_t)((uint32_t)(x)) <<
01609 #define I2C_INTENSET_MONRDYEN_MASK      (0x10000U)
01610 #define I2C_INTENSET_MONRDYEN_SHIFT     (16U)
01615 #define I2C_INTENSET_MONRDYEN(x)         (((uint32_t)((uint32_t)(x)) <<
01616 #define I2C_INTENSET_MONOVEN_MASK       (0x20000U)
01617 #define I2C_INTENSET_MONOVEN_SHIFT     (17U)
01622 #define I2C_INTENSET_MONOVEN(x)         (((uint32_t)((uint32_t)(x)) <<
01623 #define I2C_INTENSET_MONIDLEEN_MASK    (0x80000U)
01624 #define I2C_INTENSET_MONIDLEEN_SHIFT   (19U)
01629 #define I2C_INTENSET_MONIDLEEN(x)       (((uint32_t)((uint32_t)(x)) <<
01630 #define I2C_INTENSET_EVENTTIMEOUTEN_MASK (0x1000000U)
01631 #define I2C_INTENSET_EVENTTIMEOUTEN_SHIFT (24U)
01636 #define I2C_INTENSET_EVENTTIMEOUTEN(x)   (((uint32_t)((uint32_t)(x)) <<
01637 #define I2C_INTENSET_SCLTIMEOUTEN_MASK   (0x2000000U)
01638 #define I2C_INTENSET_SCLTIMEOUTEN_SHIFT (25U)
01643 #define I2C_INTENSET_SCLTIMEOUTEN(x)     (((uint32_t)((uint32_t)(x)) <<
01645 #define I2C_INTENCLR_MSTPENDINGCLR_MASK (0x1U)

```

```

01649 #define I2C_INTENCLR_MSTPENDINGCLR_SHIFT          (0U)
01650 #define I2C_INTENCLR_MSTPENDINGCLR(x)             (((uint32_t) (((uint32_t) (x)) <
01651   I2C_INTENCLR_MSTPENDINGCLR_SHIFT)) & I2C_INTENCLR_MSTPENDINGCLR_MASK)
01652 #define I2C_INTENCLR_MSTARBLOSSCLR_MASK           (0x10U)
01653 #define I2C_INTENCLR_MSTARBLOSSCLR_SHIFT          (4U)
01654 #define I2C_INTENCLR_MSTARBLOSSCLR(x)              (((uint32_t) (((uint32_t) (x)) <
01655   I2C_INTENCLR_MSTARBLOSSCLR_SHIFT)) & I2C_INTENCLR_MSTARBLOSSCLR_MASK)
01656 #define I2C_INTENCLR_MSTSTSTPERRCLR_MASK           (0x40U)
01657 #define I2C_INTENCLR_MSTSTSTPERRCLR_SHIFT          (6U)
01658 #define I2C_INTENCLR_MSTSTSTPERRCLR(x)              (((uint32_t) (((uint32_t) (x)) <
01659   I2C_INTENCLR_MSTSTSTPERRCLR_SHIFT)) & I2C_INTENCLR_MSTSTSTPERRCLR_MASK)
01660 #define I2C_INTENCLR_SLVPENDINGCLR_MASK            (0x100U)
01661 #define I2C_INTENCLR_SLVPENDINGCLR_SHIFT           (8U)
01662 #define I2C_INTENCLR_SLVPENDINGCLR(x)              (((uint32_t) (((uint32_t) (x)) <
01663   I2C_INTENCLR_SLVPENDINGCLR_SHIFT)) & I2C_INTENCLR_SLVPENDINGCLR_MASK)
01664 #define I2C_INTENCLR_SLVNOTSTRCLR_MASK             (0x800U)
01665 #define I2C_INTENCLR_SLVNOTSTRCLR_SHIFT            (11U)
01666 #define I2C_INTENCLR_SLVNOTSTRCLR(x)              (((uint32_t) (((uint32_t) (x)) <
01667   I2C_INTENCLR_SLVNOTSTRCLR_SHIFT)) & I2C_INTENCLR_SLVNOTSTRCLR_MASK)
01668 #define I2C_INTENCLR_SLVDESELCLR_MASK              (0x8000U)
01669 #define I2C_INTENCLR_SLVDESELCLR_SHIFT             (15U)
01670 #define I2C_INTENCLR_SLVDESELCLR(x)                (((uint32_t) (((uint32_t) (x)) <
01671   I2C_INTENCLR_SLVDESELCLR_SHIFT)) & I2C_INTENCLR_SLVDESELCLR_MASK)
01672 #define I2C_INTENCLR_MONRDYCLR_MASK                (0x10000U)
01673 #define I2C_INTENCLR_MONRDYCLR_SHIFT               (16U)
01674 #define I2C_INTENCLR_MONRDYCLR(x)                  (((uint32_t) (((uint32_t) (x)) <
01675   I2C_INTENCLR_MONRDYCLR_SHIFT)) & I2C_INTENCLR_MONRDYCLR_MASK)
01676 #define I2C_INTENCLR_MONOVCLR_MASK                 (0x20000U)
01677 #define I2C_INTENCLR_MONOVCLR_SHIFT                (17U)
01678 #define I2C_INTENCLR_MONOVCLR(x)                   (((uint32_t) (((uint32_t) (x)) <
01679   I2C_INTENCLR_MONOVCLR_SHIFT)) & I2C_INTENCLR_MONOVCLR_MASK)
01680 #define I2C_INTENCLR_MONVIDLECLR_MASK              (0x80000U)
01681 #define I2C_INTENCLR_MONVIDLECLR_SHIFT             (19U)
01682 #define I2C_INTENCLR_MONVIDLECLR(x)                (((uint32_t) (((uint32_t) (x)) <
01683   I2C_INTENCLR_MONVIDLECLR_SHIFT)) & I2C_INTENCLR_MONVIDLECLR_MASK)
01684 #define I2C_INTENCLR_EVENTTIMEOUTCLR_MASK          (0x1000000U)
01685 #define I2C_INTENCLR_EVENTTIMEOUTCLR_SHIFT         (24U)
01686 #define I2C_INTENCLR_EVENTTIMEOUTCLR(x)             (((uint32_t) (((uint32_t) (x)) <
01687   I2C_INTENCLR_EVENTTIMEOUTCLR_SHIFT)) & I2C_INTENCLR_EVENTTIMEOUTCLR_MASK)
01688 #define I2C_TIMEOUT_TO_MASK                         (0xFFFFU)
01689 #define I2C_TIMEOUT_TO_SHIFT                       (0U)
01690 #define I2C_TIMEOUT_TO(x)                          (((uint32_t) (((uint32_t) (x)) <
01691   I2C_TIMEOUT_TO_SHIFT))
01692 #define I2C_CLKDIV_DIVVAL_MASK                      (0xFFFFU)
01693 #define I2C_CLKDIV_DIVVAL_SHIFT                     (0U)
01694 #define I2C_CLKDIV_DIVVAL(x)                        (((uint32_t) (((uint32_t) (x)) <
01695   I2C_CLKDIV_DIVVAL_SHIFT)) & I2C_CLKDIV_DIVVAL_MASK)
01696 #define I2C_INTSTAT_MSTPENDING_MASK                (0x1U)
01697 #define I2C_INTSTAT_MSTPENDING_SHIFT               (0U)
01698 #define I2C_INTSTAT_MSTPENDING(x)                  (((uint32_t) (((uint32_t) (x)) <
01699   I2C_INTSTAT_MSTPENDING_SHIFT)) & I2C_INTSTAT_MSTPENDING_MASK)
01700 #define I2C_INTSTAT_MSTARBLOSS_MASK               (0x10U)
01701 #define I2C_INTSTAT_MSTARBLOSS_SHIFT              (4U)
01702 #define I2C_INTSTAT_MSTARBLOSS(x)                 (((uint32_t) (((uint32_t) (x)) <
01703   I2C_INTSTAT_MSTARBLOSS_SHIFT)) & I2C_INTSTAT_MSTARBLOSS_MASK)
01704 #define I2C_INTSTAT_MSTSTSTPERR_MASK              (0x40U)
01705 #define I2C_INTSTAT_MSTSTSTPERR_SHIFT             (6U)
01706 #define I2C_INTSTAT_MSTSTSTPERR(x)                (((uint32_t) (((uint32_t) (x)) <
01707   I2C_INTSTAT_MSTSTSTPERR_SHIFT)) & I2C_INTSTAT_MSTSTSTPERR_MASK)
01708 #define I2C_INTSTAT_MSTSTSTPERR_SHIFT             (6U)
01709 #define I2C_INTSTAT_MSTSTSTPERR(x)                (((uint32_t) (((uint32_t) (x)) <
01710   I2C_INTSTAT_MSTSTSTPERR_SHIFT)) & I2C_INTSTAT_MSTSTSTPERR_MASK)
01711 #define I2C_INTSTAT_SLVPENDING_MASK               (0x100U)
01712 #define I2C_INTSTAT_SLVPENDING_SHIFT              (8U)
01713 #define I2C_INTSTAT_SLVPENDING(x)                 (((uint32_t) (((uint32_t) (x)) <
01714   I2C_INTSTAT_SLVPENDING_SHIFT)) & I2C_INTSTAT_SLVPENDING_MASK)
01715 #define I2C_INTSTAT_SLVNNOTSTR_MASK              (0x800U)
01716 #define I2C_INTSTAT_SLVNNOTSTR_SHIFT             (11U)
01717 #define I2C_INTSTAT_SLVNNOTSTR(x)                (((uint32_t) (((uint32_t) (x)) <
01718   I2C_INTSTAT_SLVNNOTSTR_SHIFT)) & I2C_INTSTAT_SLVNNOTSTR_MASK)
01719 #define I2C_INTSTAT_SLVDESEL_MASK                (0x8000U)
01720 #define I2C_INTSTAT_SLVDESEL_SHIFT               (15U)
01721 #define I2C_INTSTAT_SLVDESEL(x)                  (((uint32_t) (((uint32_t) (x)) <
01722 #define I2C_INTSTAT_MONRDY_MASK                  (0x10000U)
01723 #define I2C_INTSTAT_MONRDY_SHIFT                 (16U)
01724 #define I2C_INTSTAT_MONRDY(x)                   (((uint32_t) (((uint32_t) (x)) <
01725   I2C_INTSTAT_MONRDY_SHIFT)) & I2C_INTSTAT_MONRDY_MASK)
01726 #define I2C_INTSTAT_MONOV_MASK                  (0x20000U)

```

```

01724 #define I2C_INTSTAT_MONOV_SHIFT          (17U)
01725 #define I2C_INTSTAT_MONOV(x)             (((uint32_t)((uint32_t)(x)) <<
01726     I2C_INTSTAT_MONOV_SHIFT)) & I2C_INTSTAT_MONOV_MASK)
01727 #define I2C_INTSTAT_MONIDLE_MASK         (0x800000U)
01728 #define I2C_INTSTAT_MONIDLE_SHIFT        (19U)
01729 #define I2C_INTSTAT_MONIDLE(x)           (((uint32_t)((uint32_t)(x)) <<
01730     I2C_INTSTAT_MONIDLE_SHIFT)) & I2C_INTSTAT_MONIDLE_MASK)
01731 #define I2C_INTSTAT_EVENTTIMEOUT_MASK    (0x1000000U)
01732 #define I2C_INTSTAT_EVENTTIMEOUT_SHIFT   (24U)
01733 #define I2C_INTSTAT_EVENTTIMEOUT(x)      (((uint32_t)((uint32_t)(x)) <<
01734     I2C_INTSTAT_EVENTTIMEOUT_SHIFT)) & I2C_INTSTAT_EVENTTIMEOUT_MASK)
01735 #define I2C_INTSTAT_SCLTIMEOUT_MASK      (0x2000000U)
01736 #define I2C_INTSTAT_SCLTIMEOUT_SHIFT     (25U)
01737 #define I2C_INTSTAT_SCLTIMEOUT(x)        (((uint32_t)((uint32_t)(x)) <<
01738     I2C_INTSTAT_SCLTIMEOUT_SHIFT)) & I2C_INTSTAT_SCLTIMEOUT_MASK)
01739 #define I2C_MSTCTL_MSTCONTINUE_MASK      (0x1U)
01740 #define I2C_MSTCTL_MSTCONTINUE_SHIFT     (0U)
01741 #define I2C_MSTCTL_MSTCONTINUE(x)        (((uint32_t)((uint32_t)(x)) <<
01742     I2C_MSTCTL_MSTCONTINUE_SHIFT)) & I2C_MSTCTL_MSTCONTINUE_MASK)
01743 #define I2C_MSTCTL_MSTSTART_MASK         (0x2U)
01744 #define I2C_MSTCTL_MSTSTART_SHIFT        (1U)
01745 #define I2C_MSTCTL_MSTSTART(x)           (((uint32_t)((uint32_t)(x)) <<
01746     I2C_MSTCTL_MSTSTART_SHIFT)) & I2C_MSTCTL_MSTSTART_MASK)
01747 #define I2C_MSTCTL_MSTSTOP_MASK          (0x4U)
01748 #define I2C_MSTCTL_MSTSTOP_SHIFT         (2U)
01749 #define I2C_MSTCTL_MSTSTOP(x)           (((uint32_t)((uint32_t)(x)) <<
01750     I2C_MSTCTL_MSTSTOP_SHIFT)) & I2C_MSTCTL_MSTSTOP_MASK)
01751 #define I2C_MSTCTL_MSTDMA_MASK          (0x8U)
01752 #define I2C_MSTCTL_MSTDMA_SHIFT         (3U)
01753 #define I2C_MSTCTL_MSTDMA(x)            (((uint32_t)((uint32_t)(x)) <<
01754     I2C_MSTCTL_MSTDMA_SHIFT)) & I2C_MSTCTL_MSTDMA_MASK)
01755 #define I2C_MSTTIME_MSTSCLLOW_MASK      (0x7U)
01756 #define I2C_MSTTIME_MSTSCLLOW_SHIFT     (0U)
01757 #define I2C_MSTTIME_MSTSCLLOW(x)        (((uint32_t)((uint32_t)(x)) <<
01758     I2C_MSTTIME_MSTSCLLOW_SHIFT)) & I2C_MSTTIME_MSTSCLLOW_MASK)
01759 #define I2C_MSTTIME_MSTSCLHIGH_MASK     (0x70U)
01760 #define I2C_MSTTIME_MSTSCLHIGH_SHIFT    (4U)
01761 #define I2C_MSTTIME_MSTSCLHIGH(x)       (((uint32_t)((uint32_t)(x)) <<
01762     I2C_MSTTIME_MSTSCLHIGH_SHIFT)) & I2C_MSTTIME_MSTSCLHIGH_MASK)
01763 #define I2C_MSTDAT_DATA_MASK            (0xFFU)
01764 #define I2C_MSTDAT_DATA_SHIFT          (0U)
01765 #define I2C_MSTDAT_DATA(x)             (((uint32_t)((uint32_t)(x)) <<
01766     I2C_MSTDAT_DATA_SHIFT)) & I2C_MSTDAT_DATA_MASK)
01767 #define I2C_SLVCTL_SLVCONTINUE_MASK    (0x1U)
01768 #define I2C_SLVCTL_SLVCONTINUE_SHIFT   (0U)
01769 #define I2C_SLVCTL_SLVCONTINUE(x)      (((uint32_t)((uint32_t)(x)) <<
01770     I2C_SLVCTL_SLVCONTINUE_SHIFT)) & I2C_SLVCTL_SLVCONTINUE_MASK)
01771 #define I2C_SLVCTL_SLVNACK_MASK        (0x2U)
01772 #define I2C_SLVCTL_SLVNACK_SHIFT       (1U)
01773 #define I2C_SLVCTL_SLVNACK(x)          (((uint32_t)((uint32_t)(x)) <<
01774     I2C_SLVCTL_SLVNACK_SHIFT)) & I2C_SLVCTL_SLVNACK_MASK)
01775 #define I2C_SLVCTL_SLVDMA_MASK         (0x8U)
01776 #define I2C_SLVCTL_SLVDMA_SHIFT        (3U)
01777 #define I2C_SLVCTL_SLVDMA(x)           (((uint32_t)((uint32_t)(x)) <<
01778     I2C_SLVCTL_SLVDMA_SHIFT)) & I2C_SLVCTL_SLVDMA_MASK)
01779 #define I2C_SLVADR_SADISABLE_MASK      (0x1U)
01780 #define I2C_SLVADR_SADISABLE_SHIFT     (0U)
01781 #define I2C_SLVADR_SADISABLE(x)        (((uint32_t)((uint32_t)(x)) <<
01782     I2C_SLVADR_SADISABLE_SHIFT)) & I2C_SLVADR_SADISABLE_MASK)
01783 #define I2C_SLVQUAL0_QUALMODE0_MASK    (0x1U)
01784 #define I2C_SLVQUAL0_QUALMODE0_SHIFT   (0U)
01785 #define I2C_SLVQUAL0_QUALMODE0(x)      (((uint32_t)((uint32_t)(x)) <<
01786     I2C_SLVQUAL0_QUALMODE0_SHIFT)) & I2C_SLVQUAL0_QUALMODE0_MASK)
01787 #define I2C_SLVQUAL0_SLVQUAL0_MASK     (0xFEU)
01788 #define I2C_SLVQUAL0_SLVQUAL0_SHIFT    (1U)
01789 #define I2C_SLVQUAL0_SLVQUAL0(x)       (((uint32_t)((uint32_t)(x)) <<
01790     I2C_SLVQUAL0_SLVQUAL0_SHIFT)) & I2C_SLVQUAL0_SLVQUAL0_MASK)
01791 #define I2C_MONRXDAT_MONRXDAT_MASK    (0xFFU)

```

```

01886 #define I2C_MONRXDAT_MONRXDAT__SHIFT          (0U)
01887 #define I2C_MONRXDAT_MONRXDAT(x)              (((uint32_t)((uint32_t)(x)) <<
01888     I2C_MONRXDAT_MONRXDAT__SHIFT) & I2C_MONRXDAT_MONRXDAT_MASK)
01889 #define I2C_MONRXDAT_MONSTART__MASK           (0x100U)
01890 #define I2C_MONRXDAT_MONSTART__SHIFT          (8U)
01894 #define I2C_MONRXDAT_MONSTART(x)              (((uint32_t)((uint32_t)(x)) <<
01895     I2C_MONRXDAT_MONSTART__SHIFT) & I2C_MONRXDAT_MONSTART_MASK)
01896 #define I2C_MONRXDAT_MONRESTART__MASK         (0x200U)
01897 #define I2C_MONRXDAT_MONRESTART__SHIFT        (9U)
01901 #define I2C_MONRXDAT_MONRESTART(x)            (((uint32_t)((uint32_t)(x)) <<
01902     I2C_MONRXDAT_MONRESTART__SHIFT) & I2C_MONRXDAT_MONRESTART_MASK)
01903 #define I2C_MONRXDAT_MONNACK__MASK            (0x400U)
01904 #define I2C_MONRXDAT_MONNACK__SHIFT           (10U)
01908 #define I2C_MONRXDAT_MONNACK(x)              (((uint32_t)((uint32_t)(x)) <<
01909     I2C_MONRXDAT_MONNACK__SHIFT) & I2C_MONRXDAT_MONNACK_MASK)
01910
01911 /* end of group I2C_Register_Masks */
01915
01916
01917 /* I2C - Peripheral instance base addresses */
01919 #define I2C0__BASE                         (0x40050000u)
01921 #define I2C0                             ((I2C_Type *)I2C0__BASE)
01923 #define I2C1__BASE                         (0x40054000u)
01925 #define I2C1                             ((I2C_Type *)I2C1__BASE)
01927 #define I2C2__BASE                         (0x40030000u)
01929 #define I2C2                             ((I2C_Type *)I2C2__BASE)
01931 #define I2C3__BASE                         (0x40034000u)
01933 #define I2C3                             ((I2C_Type *)I2C3__BASE)
01935 #define I2C_BASE_ADDRS                     { I2C0__BASE, I2C1__BASE, I2C2__BASE, I2C3__BASE }
01937 #define I2C_BASE_PTRS                      { I2C0, I2C1, I2C2, I2C3 }
01939 /* #define I2C_IRQS
NOT IMPLEMENTED */
01940 /* end of group I2C_Peripheral_Access_Layer */
01944
01945 /*
01946 -- SPI Peripheral Access Layer
01947 -----
01948
01953
01955 typedef struct {
01956     __RW    uint32_t CFG;
01957     __RW    uint32_t DLY;
01958     __RW    uint32_t STAT;
01959     __RW    uint32_t INTENSET;
01960     __W     uint32_t INTENCLR;
01961     __R     uint32_t RXDAT;
01962     __RW    uint32_t TXDATCTL;
01963     __RW    uint32_t TXDAT;
01964     __RW    uint32_t TXCTL;
01965     __RW    uint32_t DIV;
01966     __R     uint32_t INTSTAT;
01967 } SPI_Type;
01968
01969 /*
01970 -- SPI Register Masks
01971 -----
01972
01977
01980 #define SPI_CFG_ENABLE__MASK                (0x1U)
01981 #define SPI_CFG_ENABLE__SHIFT               (0U)
01986 #define SPI_CFG_ENABLE(x)                  (((uint32_t)((uint32_t)(x)) <<
01987     SPI_CFG_ENABLE__SHIFT) & SPI_CFG_ENABLE_MASK)
01988 #define SPI_CFG_MASTER__MASK               (0x4U)
01989 #define SPI_CFG_MASTER__SHIFT              (2U)
01993 #define SPI_CFG_MASTER(x)                  (((uint32_t)((uint32_t)(x)) <<
01994     SPI_CFG_MASTER__SHIFT) & SPI_CFG_MASTER_MASK)
01995 #define SPI_CFG_LSBF__MASK                (0x8U)
01996 #define SPI_CFG_LSBF__SHIFT               (3U)
02000 #define SPI_CFG_LSBF(x)                  (((uint32_t)((uint32_t)(x)) <<
02001     SPI_CFG_LSBF__SHIFT) & SPI_CFG_LSBF_MASK)
02002 #define SPI_CFG_CPHA__MASK               (0x10U)
02003 #define SPI_CFG_CPHA__SHIFT              (4U)
02009 #define SPI_CFG_CPHA(x)                  (((uint32_t)((uint32_t)(x)) <<
02010     SPI_CFG_CPHA__SHIFT) & SPI_CFG_CPHA_MASK)
02011 #define SPI_CFG_CPOL__MASK               (0x20U)
02012 #define SPI_CFG_CPOL__SHIFT              (5U)
02016 #define SPI_CFG_CPOL(x)                  (((uint32_t)((uint32_t)(x)) <<
02017     SPI_CFG_CPOL__SHIFT) & SPI_CFG_CPOL_MASK)
02018 #define SPI_CFG_LOOP__MASK               (0x80U)
02019 #define SPI_CFG_LOOP__SHIFT              (7U)
02024 #define SPI_CFG_LOOP(x)                  (((uint32_t)((uint32_t)(x)) <<
02025     SPI_CFG_LOOP__SHIFT) & SPI_CFG_LOOP_MASK)
02026 #define SPI_CFG_SPOLO__MASK              (0x100U)
02027 #define SPI_CFG_SPOLO__SHIFT             (8U)
02031 #define SPI_CFG_SPOLO(x)                (((uint32_t)((uint32_t)(x)) <<
02032 #define SPI_CFG_SPOL1__MASK             (0x200U)

```

```

02033 #define SPI_CFG_SPOL1_SHIFT          (9U)
02038 #define SPI_CFG_SPOL1(x)           (((uint32_t)((uint32_t)(x)) << SPI_CFG_SPOL1_SHIFT))
02039 #define SPI_CFG_SPOL1_MASK         (0x400U)
02040 #define SPI_CFG_SPOL2_SHIFT          (10U)
02045 #define SPI_CFG_SPOL2(x)           (((uint32_t)((uint32_t)(x)) << SPI_CFG_SPOL2_SHIFT))
02046 #define SPI_CFG_SPOL2_MASK         (0x800U)
02047 #define SPI_CFG_SPOL3_SHIFT          (11U)
02052 #define SPI_CFG_SPOL3(x)           (((uint32_t)((uint32_t)(x)) << SPI_CFG_SPOL3_SHIFT))
02053 #define SPI_CFG_SPOL3_MASK         (0x100U)
02057 #define SPI_DLY_PRE_DELAY_MASK      (0xFU)
02058 #define SPI_DLY_PRE_DELAY_SHIFT     (0U)
02059 #define SPI_DLY_PRE_DELAY(x)        (((uint32_t)((uint32_t)(x)) <<
02060   SPI_DLY_PRE_DELAY_SHIFT) & SPI_DLY_PRE_DELAY_MASK)
02061 #define SPI_DLY_POST_DELAY_MASK     (0xFOU)
02062 #define SPI_DLY_POST_DELAY_SHIFT    (4U)
02063 #define SPI_DLY_POST_DELAY(x)        (((uint32_t)((uint32_t)(x)) <<
02064   SPI_DLY_POST_DELAY_SHIFT) & SPI_DLY_POST_DELAY_MASK)
02065 #define SPI_DLY_FRAME_DELAY_MASK    (0xFO00U)
02066 #define SPI_DLY_FRAME_DELAY_SHIFT   (8U)
02067 #define SPI_DLY_FRAME_DELAY(x)      (((uint32_t)((uint32_t)(x)) <<
02068   SPI_DLY_FRAME_DELAY_SHIFT) & SPI_DLY_FRAME_DELAY_MASK)
02069 #define SPI_DLY_TRANSFER_DELAY_MASK (0xF000U)
02070 #define SPI_DLY_TRANSFER_DELAY_SHIFT (12U)
02071 #define SPI_DLY_TRANSFER_DELAY(x)   (((uint32_t)((uint32_t)(x)) <<
02072   SPI_DLY_TRANSFER_DELAY_SHIFT) & SPI_DLY_TRANSFER_DELAY_MASK)
02073 #define SPI_STAT_RXRDY_MASK         (0x1U)
02074 #define SPI_STAT_RXRDY_SHIFT       (0U)
02075 #define SPI_STAT_RXRDY(x)          (((uint32_t)((uint32_t)(x)) << SPI_STAT_RXRDY_SHIFT))
02076 #define SPI_STAT_TXRDY_MASK         (0x2U)
02077 #define SPI_STAT_TXRDY_SHIFT       (1U)
02078 #define SPI_STAT_TXRDY(x)          (((uint32_t)((uint32_t)(x)) << SPI_STAT_TXRDY_SHIFT))
02079 #define SPI_STAT_RXOV_MASK         (0x4U)
02080 #define SPI_STAT_RXOV_SHIFT       (2U)
02081 #define SPI_STAT_RXOV(x)          (((uint32_t)((uint32_t)(x)) << SPI_STAT_RXOV_SHIFT))
02082 #define SPI_STAT_TXUR_MASK         (0x8U)
02083 #define SPI_STAT_TXUR_SHIFT       (3U)
02084 #define SPI_STAT_TXUR(x)          (((uint32_t)((uint32_t)(x)) << SPI_STAT_TXUR_SHIFT))
02085 #define SPI_STAT_SSA_MASK          (0x10U)
02086 #define SPI_STAT_SSA_SHIFT        (4U)
02087 #define SPI_STAT_SSA(x)           (((uint32_t)((uint32_t)(x)) << SPI_STAT_SSA_SHIFT) &
02088 #define SPI_STAT_SSD_MASK          (0x20U)
02089 #define SPI_STAT_SSD_SHIFT        (5U)
02090 #define SPI_STAT_SSD(x)           (((uint32_t)((uint32_t)(x)) << SPI_STAT_SSD_SHIFT) &
02091 #define SPI_STAT_STALLED_MASK      (0x40U)
02092 #define SPI_STAT_STALLED_SHIFT    (6U)
02093 #define SPI_STAT_STALLED(x)        (((uint32_t)((uint32_t)(x)) <<
02094   SPI_STAT_STALLED_SHIFT) & SPI_STAT_STALLED_MASK)
02095 #define SPI_STAT_ENDTRANSFER_MASK  (0x80U)
02096 #define SPI_STAT_ENDTRANSFER_SHIFT (7U)
02097 #define SPI_STAT_MSTIDLE_MASK      (0x100U)
02098 #define SPI_STAT_MSTIDLE_SHIFT    (8U)
02099 #define SPI_STAT_MSTIDLE(x)        (((uint32_t)((uint32_t)(x)) <<
02100   SPI_STAT_MSTIDLE_SHIFT) & SPI_STAT_MSTIDLE_MASK)
02101 #define SPI_INTENSET_RXRDYEN_MASK  (0x1U)
02102 #define SPI_INTENSET_RXRDYEN_SHIFT (0U)
02103 #define SPI_INTENSET_RXRDYEN(x)    (((uint32_t)((uint32_t)(x)) <<
02104   SPI_INTENSET_RXRDYEN_SHIFT) & SPI_INTENSET_RXRDYEN_MASK)
02105 #define SPI_INTENSET_TXRDYEN_MASK  (0x2U)
02106 #define SPI_INTENSET_TXRDYEN_SHIFT (1U)
02107 #define SPI_INTENSET_TXRDYEN(x)    (((uint32_t)((uint32_t)(x)) <<
02108   SPI_INTENSET_TXRDYEN_SHIFT) & SPI_INTENSET_TXRDYEN_MASK)
02109 #define SPI_INTENSET_RXOVEN_MASK    (0x4U)
02110 #define SPI_INTENSET_RXOVEN_SHIFT   (2U)
02111 #define SPI_INTENSET_RXOVEN(x)     (((uint32_t)((uint32_t)(x)) <<
02112   SPI_INTENSET_RXOVEN_SHIFT) & SPI_INTENSET_RXOVEN_MASK)
02113 #define SPI_INTENSET_TXUREN(x)     (((uint32_t)((uint32_t)(x)) <<
02114   SPI_INTENSET_TXUREN_SHIFT) & SPI_INTENSET_TXUREN_MASK)
02115 #define SPI_INTENSET_SSAEN_MASK    (0x8U)
02116 #define SPI_INTENSET_SSAEN_SHIFT   (3U)
02117 #define SPI_INTENSET_SSAEN(x)      (((uint32_t)((uint32_t)(x)) <<
02118   SPI_INTENSET_SSAEN_SHIFT) & SPI_INTENSET_SSAEN_MASK)
02119 #define SPI_INTENSET_SSDEN_MASK    (0x20U)
02120 #define SPI_INTENSET_SSDEN_SHIFT   (4U)
02121 #define SPI_INTENSET_SSDEN(x)      (((uint32_t)((uint32_t)(x)) <<
02122   SPI_INTENSET_SSDEN_SHIFT) & SPI_INTENSET_SSDEN_MASK)

```

```

02144 #define SPI_INENSET_SSDEN_SHIFT          (5U)
02149 #define SPI_INENSET_SSDEN(x)           (((uint32_t)((uint32_t)(x)) <<
02150     SPI_INENSET_SSDEN_SHIFT) & SPI_INENSET_SSDEN_MASK)
02151
02154 #define SPI_INENCLR_RXRDYEN_MASK        (0x1U)
02155 #define SPI_INENCLR_RXRDYEN_SHIFT       (0U)
02156 #define SPI_INENCLR_RXRDYEN_SHIFT(x)    (((uint32_t)((uint32_t)(x)) <<
02157     SPI_INENCLR_RXRDYEN_SHIFT) & SPI_INENCLR_RXRDYEN_MASK)
02158 #define SPI_INENCLR_TXRDYEN_MASK        (0x2U)
02159 #define SPI_INENCLR_TXRDYEN_SHIFT       (1U)
02160 #define SPI_INENCLR_TXRDYEN(x)          (((uint32_t)((uint32_t)(x)) <<
02161     SPI_INENCLR_TXRDYEN_SHIFT) & SPI_INENCLR_TXRDYEN_MASK)
02162 #define SPI_INENCLR_RXOVEN_MASK         (0x4U)
02163 #define SPI_INENCLR_RXOVEN_SHIFT       (2U)
02164 #define SPI_INENCLR_RXOVEN(x)          (((uint32_t)((uint32_t)(x)) <<
02165     SPI_INENCLR_RXOVEN_SHIFT) & SPI_INENCLR_RXOVEN_MASK)
02166 #define SPI_INENCLR_TXUREN_MASK         (0x8U)
02167 #define SPI_INENCLR_TXUREN_SHIFT       (3U)
02168 #define SPI_INENCLR_TXUREN(x)          (((uint32_t)((uint32_t)(x)) <<
02169     SPI_INENCLR_TXUREN_SHIFT) & SPI_INENCLR_TXUREN_MASK)
02170 #define SPI_INENCLR_SSAEN_MASK         (0x10U)
02171 #define SPI_INENCLR_SSAEN_SHIFT       (4U)
02172 #define SPI_INENCLR_SSAEN(x)          (((uint32_t)((uint32_t)(x)) <<
02173     SPI_INENCLR_SSAEN_SHIFT) & SPI_INENCLR_SSAEN_MASK)
02174
02176 #define SPI_RXDAT_RXDAT_MASK          (0xFFFFU)
02177 #define SPI_RXDAT_RXDAT_SHIFT        (0U)
02178 #define SPI_RXDAT_RXDAT(x)          (((uint32_t)((uint32_t)(x)) <<
02179     SPI_RXDAT_RXDAT_SHIFT) & SPI_RXDAT_RXDAT_MASK)
02180 #define SPI_RXDAT_RXSEL0_N_MASK       (0x10000U)
02181 #define SPI_RXDAT_RXSEL0_N_SHIFT     (16U)
02182 #define SPI_RXDAT_RXSEL0_N(x)        (((uint32_t)((uint32_t)(x)) <<
02183     SPI_RXDAT_RXSEL0_N_SHIFT) & SPI_RXDAT_RXSEL0_N_MASK)
02184 #define SPI_RXDAT_RXSEL1_N_MASK       (0x20000U)
02185 #define SPI_RXDAT_RXSEL1_N_SHIFT     (17U)
02186 #define SPI_RXDAT_RXSEL2_N_MASK       (0x40000U)
02187 #define SPI_RXDAT_RXSEL2_N_SHIFT     (18U)
02188 #define SPI_RXDAT_RXSEL3_N_MASK       (0x80000U)
02189 #define SPI_RXDAT_RXSEL3_N_SHIFT     (19U)
02190 #define SPI_RXDAT_RXSEL3_N(x)        (((uint32_t)((uint32_t)(x)) <<
02191     SPI_RXDAT_RXSEL3_N_SHIFT) & SPI_RXDAT_RXSEL3_N_MASK)
02192 #define SPI_RXDAT_SOT_MASK          (0x1000000U)
02193 #define SPI_RXDAT_SOT_SHIFT        (20U)
02194 #define SPI_RXDAT_SOT(x)            (((uint32_t)((uint32_t)(x)) <<
02195     SPI_RXDAT_SOT_SHIFT) & SPI_RXDAT_SOT_MASK)
02196
02198 #define SPI_TXDATCTL_TXDAT_MASK       (0xFFFFU)
02199 #define SPI_TXDATCTL_TXDAT_SHIFT     (0U)
02200 #define SPI_TXDATCTL_TXDAT(x)        (((uint32_t)((uint32_t)(x)) <<
02201     SPI_TXDATCTL_TXDAT_SHIFT) & SPI_TXDATCTL_TXDAT_MASK)
02202 #define SPI_TXDATCTL_RXSEL0_N_MASK   (0x10000U)
02203 #define SPI_TXDATCTL_RXSEL0_N_SHIFT (16U)
02204 #define SPI_TXDATCTL_RXSEL0_N(x)    (((uint32_t)((uint32_t)(x)) <<
02205     SPI_TXDATCTL_RXSEL0_N_SHIFT) & SPI_TXDATCTL_RXSEL0_N_MASK)
02206 #define SPI_TXDATCTL_RXSEL1_N_MASK   (0x20000U)
02207 #define SPI_TXDATCTL_RXSEL1_N_SHIFT (17U)
02208 #define SPI_TXDATCTL_RXSEL1_N(x)    (((uint32_t)((uint32_t)(x)) <<
02209     SPI_TXDATCTL_RXSEL1_N_SHIFT) & SPI_TXDATCTL_RXSEL1_N_MASK)
02210 #define SPI_TXDATCTL_RXSEL2_N_MASK   (0x40000U)
02211 #define SPI_TXDATCTL_RXSEL2_N_SHIFT (18U)
02212 #define SPI_TXDATCTL_RXSEL2_N(x)    (((uint32_t)((uint32_t)(x)) <<
02213     SPI_TXDATCTL_RXSEL2_N_SHIFT) & SPI_TXDATCTL_RXSEL2_N_MASK)
02214 #define SPI_TXDATCTL_RXSEL3_N_MASK   (0x80000U)
02215 #define SPI_TXDATCTL_RXSEL3_N_SHIFT (19U)
02216 #define SPI_TXDATCTL_RXSEL3_N(x)    (((uint32_t)((uint32_t)(x)) <<
02217     SPI_TXDATCTL_RXSEL3_N_SHIFT) & SPI_TXDATCTL_RXSEL3_N_MASK)
02218 #define SPI_TXDATCTL_EOT_MASK        (0x1000000U)
02219 #define SPI_TXDATCTL_EOT_SHIFT      (20U)
02220 #define SPI_TXDATCTL_EOT(x)         (((uint32_t)((uint32_t)(x)) <<
02221     SPI_TXDATCTL_EOT_SHIFT) & SPI_TXDATCTL_EOT_MASK)
02222 #define SPI_TXDATCTL_EOT_SHIFT      (21U)
02223 #define SPI_TXDATCTL_EOT(x)         (((uint32_t)((uint32_t)(x)) <<
02224     SPI_TXDATCTL_EOT_SHIFT) & SPI_TXDATCTL_EOT_MASK)
02225 #define SPI_TXDATCTL_EOF_MASK        (0x2000000U)
02226 #define SPI_TXDATCTL_EOF_SHIFT      (21U)
02227 #define SPI_TXDATCTL_EOF(x)         (((uint32_t)((uint32_t)(x)) <<
02228     SPI_TXDATCTL_EOF_SHIFT) & SPI_TXDATCTL_EOF_MASK)
02229 #define SPI_TXDATCTL_RXIGNORE_MASK  (0x4000000U)
02230 #define SPI_TXDATCTL_RXIGNORE_SHIFT (22U)
02231 #define SPI_TXDATCTL_RXIGNORE(x)   (((uint32_t)((uint32_t)(x)) <<
02232     SPI_TXDATCTL_RXIGNORE_SHIFT) & SPI_TXDATCTL_RXIGNORE_MASK)
02233 #define SPI_TXDATCTL_LEN_MASK       (0xF000000U)

```

```

02266 #define SPI_TXDATCTL_LEN_SHIFT          (24U)
02267 #define SPI_TXDATCTL_LEN(x)             (((uint32_t)((uint32_t)(x)) <<
02268     SPI_TXDATCTL_LEN_SHIFT)) & SPI_TXDATCTL_LEN_MASK
02269
02270 #define SPI_TXDAT_DATA_MASK            (0xFFFFU)
02271 #define SPI_TXDAT_DATA_SHIFT           (0U)
02272 #define SPI_TXDAT_DATA(x)              (((uint32_t)((uint32_t)(x)) << SPI_TXDAT_DATA_SHIFT))
02273     & SPI_TXDAT_DATA_MASK)
02274
02275 #define SPI_TXCTL_TXSSEL0_N_MASK        (0x10000U)
02276 #define SPI_TXCTL_TXSSEL0_N_SHIFT       (16U)
02277 #define SPI_TXCTL_TXSSEL0_N(x)          (((uint32_t)((uint32_t)(x)) <<
02278     SPI_TXCTL_TXSSEL0_N_SHIFT)) & SPI_TXCTL_TXSSEL0_N_MASK
02279 #define SPI_TXCTL_TXSSEL1_N_MASK        (0x20000U)
02280 #define SPI_TXCTL_TXSSEL1_N_SHIFT       (17U)
02281 #define SPI_TXCTL_TXSSEL1_N(x)          (((uint32_t)((uint32_t)(x)) <<
02282     SPI_TXCTL_TXSSEL1_N_SHIFT)) & SPI_TXCTL_TXSSEL1_N_MASK
02283 #define SPI_TXCTL_TXSSEL2_N_MASK        (0x40000U)
02284 #define SPI_TXCTL_TXSSEL2_N_SHIFT       (18U)
02285 #define SPI_TXCTL_TXSSEL2_N(x)          (((uint32_t)((uint32_t)(x)) <<
02286     SPI_TXCTL_TXSSEL2_N_SHIFT)) & SPI_TXCTL_TXSSEL2_N_MASK
02287 #define SPI_TXCTL_TXSSEL3_N_MASK        (0x80000U)
02288 #define SPI_TXCTL_TXSSEL3_N_SHIFT       (19U)
02289 #define SPI_TXCTL_TXSSEL3_N(x)          (((uint32_t)((uint32_t)(x)) <<
02290     SPI_TXCTL_TXSSEL3_N_SHIFT)) & SPI_TXCTL_TXSSEL3_N_MASK
02291 #define SPI_TXCTL_EOT_MASK             (0x100000U)
02292 #define SPI_TXCTL_EOT_SHIFT            (20U)
02293 #define SPI_TXCTL_EOT(x)              (((uint32_t)((uint32_t)(x)) << SPI_TXCTL_EOT_SHIFT))
02294     & SPI_TXCTL_EOT_MASK)
02295 #define SPI_TXCTL_EOF_MASK             (0x200000U)
02296 #define SPI_TXCTL_EOF_SHIFT            (21U)
02297 #define SPI_TXCTL_EOF(x)              (((uint32_t)((uint32_t)(x)) << SPI_TXCTL_EOF_SHIFT))
02298     & SPI_TXCTL_EOF_MASK)
02299 #define SPI_TXCTL_RXIGNORE_MASK        (0x400000U)
02300 #define SPI_TXCTL_RXIGNORE_SHIFT       (22U)
02301 #define SPI_TXCTL_RXIGNORE(x)          (((uint32_t)((uint32_t)(x)) <<
02302     SPI_TXCTL_RXIGNORE_SHIFT)) & SPI_TXCTL_RXIGNORE_MASK
02303 #define SPI_TXCTL_LEN_MASK             (0xF000000U)
02304 #define SPI_TXCTL_LEN_SHIFT            (24U)
02305 #define SPI_TXCTL_LEN(x)              (((uint32_t)((uint32_t)(x)) << SPI_TXCTL_LEN_SHIFT))
02306     & SPI_TXCTL_LEN_MASK)
02307
02308 #define SPI_DIV_DIVVAL_MASK           (0xFFFFU)
02309 #define SPI_DIV_DIVVAL_SHIFT          (0U)
02310 #define SPI_DIV_DIVVAL(x)             (((uint32_t)((uint32_t)(x)) << SPI_DIV_DIVVAL_SHIFT))
02311     & SPI_DIV_DIVVAL_MASK)
02312 #define SPI_INTSTAT_RXRDY_MASK         (0x1U)
02313 #define SPI_INTSTAT_RXRDY_SHIFT        (0U)
02314 #define SPI_INTSTAT_RXRDY(x)          (((uint32_t)((uint32_t)(x)) <<
02315     SPI_INTSTAT_RXRDY_SHIFT)) & SPI_INTSTAT_RXRDY_MASK
02316 #define SPI_INTSTAT_TXRDY_MASK         (0x2U)
02317 #define SPI_INTSTAT_TXRDY_SHIFT        (1U)
02318 #define SPI_INTSTAT_TXRDY(x)          (((uint32_t)((uint32_t)(x)) <<
02319     SPI_INTSTAT_TXRDY_SHIFT)) & SPI_INTSTAT_TXRDY_MASK
02320 #define SPI_INTSTAT_RXOV_MASK          (0x4U)
02321 #define SPI_INTSTAT_RXOV_SHIFT         (2U)
02322 #define SPI_INTSTAT_RXOV(x)           (((uint32_t)((uint32_t)(x)) << SPI_INTSTAT_RXOV_SHIFT))
02323     & SPI_INTSTAT_RXOV_MASK)
02324
02325 #define SPI_INTSTAT_TXUR_MASK          (0x8U)
02326 #define SPI_INTSTAT_TXUR_SHIFT         (3U)
02327 #define SPI_INTSTAT_TXUR(x)           (((uint32_t)((uint32_t)(x)) <<
02328     SPI_INTSTAT_TXUR_SHIFT)) & SPI_INTSTAT_TXUR_MASK
02329 #define SPI_INTSTAT_SSA_MASK           (0x10U)
02330 #define SPI_INTSTAT_SSA_SHIFT          (4U)
02331 #define SPI_INTSTAT_SSA(x)            (((uint32_t)((uint32_t)(x)) <<
02332     SPI_INTSTAT_SSA_SHIFT)) & SPI_INTSTAT_SSA_MASK
02333 #define SPI_INTSTAT_SSD_MASK           (0x20U)
02334 #define SPI_INTSTAT_SSD_SHIFT          (5U)
02335 #define SPI_INTSTAT_SSD(x)            (((uint32_t)((uint32_t)(x)) <<
02336     SPI_INTSTAT_SSD_SHIFT)) & SPI_INTSTAT_SSD_MASK
02337 #define SPI_INTSTAT_MSTIDLE_MASK        (0x100U)
02338 #define SPI_INTSTAT_MSTIDLE_SHIFT       (8U)
02339 #define SPI_INTSTAT_MSTIDLE(x)          (((uint32_t)((uint32_t)(x)) <<
02340     SPI_INTSTAT_MSTIDLE_SHIFT)) & SPI_INTSTAT_MSTIDLE_MASK
02341
02342 /* end of group SPI_Register_Masks */
02343
02344 /* SPI - Peripheral instance base addresses */
02345 #define SPI0_BASE                    (0x40058000u)
02346 #define SPI0                         ((SPI_Type *)SPI0_BASE)
02347 #define SPI1_BASE                    (0x4005C000u)
02348 #define SPI1                         ((SPI_Type *)SPI1_BASE)
02349 #define SPI_BASE_ADDRS                { SPI0_BASE, SPI1_BASE }
02350 #define SPI_BASE_PTRS                 { SPI0, SPI1 }
02351
02352 /* end of group SPI_Peripheral_Access_Layer */

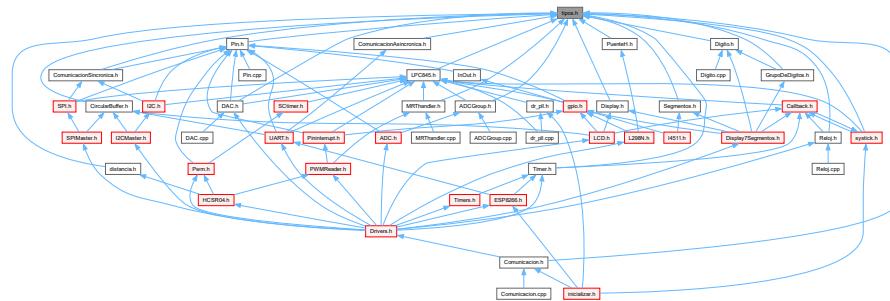
```

```
02381  
02383 #endif /* LPC845_H_ */
```

8.128 tipos.h File Reference

Definiciones de tipos de variables.

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



Macros

- #define __R volatile const
 - #define __W volatile
 - #define __RW volatile
 - #define INT32_MAX (2147483647)
 - #define INT32_MIN (-2147483648)
 - #define UINT32_MAX (4294967295)
 - #define INT8_MAX (127)
 - #define INT8_MIN (-128)
 - #define UINT8_MAX (255)

Typedefs

- `typedef unsigned char uint8_t`
 - `typedef char int8_t`
 - `typedef short unsigned int uint16_t`
 - `typedef short signed int int16_t`
 - `typedef unsigned int uint32_t`
 - `typedef signed int int32_t`

8.128.1 Detailed Description

Definiciones de tipos de variables.

Date

22 jun. 2022

Version

10

Author

Técnico. Martinez Agustín (masteragu365@gmail.com)

8.128.2 Macro Definition Documentation

8.128.2.1 __R

```
#define __R volatile const  
Defines 'read only' permission or 'Input'
```

8.128.2.2 __RW

```
#define __RW volatile  
Defines 'read / write' permission or 'Input / Output'
```

8.128.2.3 __W

```
#define __W volatile  
Defines 'write only' permission or 'Output'
```

8.128.2.4 INT32_MAX

```
#define INT32_MAX (2147483647)  
Valor máximo del INT32
```

8.128.2.5 INT32_MIN

```
#define INT32_MIN (-2147483648)  
Valor mínimo del INT32
```

8.128.2.6 INT8_MAX

```
#define INT8_MAX (127)  
Valor máximo del INT8
```

8.128.2.7 INT8_MIN

```
#define INT8_MIN (-128)  
Valor mínimo del INT8
```

8.128.2.8 UINT32_MAX

```
#define UINT32_MAX (4294967295)  
Valor máximo del UINT32
```

8.128.2.9 UINT8_MAX

```
#define UINT8_MAX (255)  
Valor máximo del UINT8
```

8.128.3 Typedef Documentation

8.128.3.1 int16_t

```
typedef short signed int int16_t  
Definicion del int16\_t
```

8.128.3.2 int32_t

```
typedef signed int int32_t  
Definicion del int32\_t
```

8.128.3.3 int8_t

```
typedef char int8_t
Definicion del int8_t
```

8.128.3.4 uint16_t

```
typedef short unsigned int uint16_t
Definicion del uint16_t
```

8.128.3.5 uint32_t

```
typedef unsigned int uint32_t
Definicion del uint32_t
```

8.128.3.6 uint8_t

```
typedef unsigned char uint8_t
Definicion del uint8_t
```

8.129 tipos.h

[Go to the documentation of this file.](#)

```
00001 /*****
00012 ****
00013 *** MODULO
00014 ****
00015
00016 #ifndef TIPOS_H_
00017 #define TIPOS_H_
00018
00019
00020 **** INCLUDES GLOBALES
00021 ****
00022
00023
00024 **** DEFINES GLOBALES
00025 ****
00026
00027
00028 **** MACROS GLOBALES
00029 ****
00030
00031
00032 **** TIPO DE DATOS GLOBALES
00033
00034
00035 #define __R volatile const
00036 #define __W volatile
00037 #define __RW volatile
00038
00039
00040
00041 typedef unsigned char uint8_t;
00042 typedef char int8_t;
00043 typedef short unsigned int uint16_t;
00044 typedef short signed int int16_t;
00045 typedef unsigned int uint32_t;
00046 typedef signed int int32_t;
00047
00048
00049 #define INT32_MAX (2147483647)
00050 #define INT32_MIN (-2147483648)
00051 #define UINT32_MAX (4294967295)
00052
00053
00054
00055 #define INT8_MAX (127)
00056 #define INT8_MIN (-128)
00057 #define UINT8_MAX (255)
00058
00059
00060
00061
```

```
00062 /*****  
00063 *** VARIABLES GLOBALES  
00064 *****  
00065  
00066 /*****  
00067 *** PROTOTIPOS DE FUNCIONES GLOBALES  
00068 *****  
00069  
00070 #endif /* TIPOS_H */
```


Index

- __R
 - tipos.h, 433
- __RW
 - tipos.h, 433
- __W
 - tipos.h, 433
- ~ADC_Group
 - ADC_Group, 86
- ~ComunicacionAsincronica
 - ComunicacionAsincronica, 95
- ~ComunicacionSincronica
 - ComunicacionSincronica, 98
- ~DAC
 - DAC, 102
- ~Display
 - Display, 110
- ~InOut
 - InOut, 156
- ~Input
 - Input, 161
- ~L298N
 - L298N, 166
- ~MRThandler
 - MRThandler, 175
- ~Output
 - Output, 182
- ~PinInterrupt
 - PinInterrupt, 192
- ~Puente_H
 - Puente_H, 195
- ~Pwm
 - Pwm, 200
- ~SCtimer
 - SCtimer, 216
- ~SPI
 - SPI, 221
- ~display7Segmentos
 - display7Segmentos, 113
- ~distancia
 - distancia, 115
- ~gpio
 - gpio, 123
- ~teclado
 - teclado, 242
- ~timers
 - timers, 250
- Abstracta, 18
- ACK
 - I2C, 139
- ACKAddr
 - I2C, 139
- activity_t
 - Pwm, 199
- ADC, 81
 - ADC, 82
 - Get, 83
 - Initialize, 83
 - IsResultReady, 83
 - Trigger, 83
- ADC Peripheral Access Layer, 31
 - ADC0, 32
 - ADC0_BASE, 32
 - ADC_BASE_ADDRS, 32
 - ADC_BASE_PTRS, 32
 - ADC_SEQ IRQS, 32
- ADC Register Masks, 33
 - ADC_CHAN_THRSEL_CH0_THRSEL, 36
 - ADC_CHAN_THRSEL_CH10_THRSEL, 36
 - ADC_CHAN_THRSEL_CH11_THRSEL, 36
 - ADC_CHAN_THRSEL_CH1_THRSEL, 36
 - ADC_CHAN_THRSEL_CH2_THRSEL, 36
 - ADC_CHAN_THRSEL_CH3_THRSEL, 37
 - ADC_CHAN_THRSEL_CH4_THRSEL, 37
 - ADC_CHAN_THRSEL_CH5_THRSEL, 37
 - ADC_CHAN_THRSEL_CH6_THRSEL, 37
 - ADC_CHAN_THRSEL_CH7_THRSEL, 37
 - ADC_CHAN_THRSEL_CH8_THRSEL, 38
 - ADC_CHAN_THRSEL_CH9_THRSEL, 38
 - ADC_CTRL_ASYNMODE, 38
 - ADC_CTRL_CALMODE, 38
 - ADC_CTRL_CLKDIV, 39
 - ADC_CTRL_LPWRMODE, 39
 - ADC_DAT_CHANNEL, 39
 - ADC_DAT_COUNT, 39
 - ADC_DAT_DATAVALID, 40
 - ADC_DAT_OVERRUN, 40
 - ADC_DAT_RESULT, 40
 - ADC_DAT_THCMPCROSS, 40
 - ADC_DAT_THCMPRANGE, 41
 - ADC_FLAGS_OVERRUN0, 41
 - ADC_FLAGS_OVERRUN1, 41
 - ADC_FLAGS_OVERRUN10, 41
 - ADC_FLAGS_OVERRUN11, 42
 - ADC_FLAGS_OVERRUN2, 42
 - ADC_FLAGS_OVERRUN3, 42
 - ADC_FLAGS_OVERRUN4, 42
 - ADC_FLAGS_OVERRUN5, 42
 - ADC_FLAGS_OVERRUN6, 43

ADC_FLAGS_OVERRUN7, 43
ADC_FLAGS_OVERRUN8, 43
ADC_FLAGS_OVERRUN9, 43
ADC_FLAGS_OVR_INT, 43
ADC_FLAGS_SEQA_INT, 44
ADC_FLAGS_SEQA_OVR, 44
ADC_FLAGS_SEQB_INT, 44
ADC_FLAGS_SEQB_OVR, 44
ADC_FLAGS_THCMP0, 45
ADC_FLAGS_THCMP1, 45
ADC_FLAGS_THCMP10, 45
ADC_FLAGS_THCMP11, 45
ADC_FLAGS_THCMP2, 45
ADC_FLAGS_THCMP3, 46
ADC_FLAGS_THCMP4, 46
ADC_FLAGS_THCMP5, 46
ADC_FLAGS_THCMP6, 46
ADC_FLAGS_THCMP7, 46
ADC_FLAGS_THCMP8, 47
ADC_FLAGS_THCMP9, 47
ADC_FLAGS_THCMP_INT, 47
ADC_INTEN_ADCMPINTEN0, 47
ADC_INTEN_ADCMPINTEN1, 47
ADC_INTEN_ADCMPINTEN10, 48
ADC_INTEN_ADCMPINTEN11, 48
ADC_INTEN_ADCMPINTEN2, 48
ADC_INTEN_ADCMPINTEN3, 48
ADC_INTEN_ADCMPINTEN4, 48
ADC_INTEN_ADCMPINTEN5, 49
ADC_INTEN_ADCMPINTEN6, 49
ADC_INTEN_ADCMPINTEN7, 49
ADC_INTEN_ADCMPINTEN8, 49
ADC_INTEN_ADCMPINTEN9, 49
ADC_INTEN_OVR_INTEN, 50
ADC_INTEN_SEQA_INTEN, 50
ADC_INTEN_SEQB_INTEN, 50
ADC_SEQ_CTRL_BURST, 50
ADC_SEQ_CTRL_CHANNELS, 51
ADC_SEQ_CTRL_COUNT, 51
ADC_SEQ_CTRL_LOWPRIOR, 51
ADC_SEQ_CTRL_MODE, 51
ADC_SEQ_CTRL_SEQ_ENA, 52
ADC_SEQ_CTRL_SINGLESTEP, 52
ADC_SEQ_CTRL_START, 52
ADC_SEQ_CTRL_SYNCBYPASS, 53
ADC_SEQ_CTRL_TRIGGER, 53
ADC_SEQ_CTRL_TRIGPOL, 53
ADC_SEQ_GDAT_CHN, 53
ADC_SEQ_GDAT_COUNT, 54
ADC_SEQ_GDAT_DATAVALID, 54
ADC_SEQ_GDAT_OVERRUN, 54
ADC_SEQ_GDAT_RESULT, 54
ADC_SEQ_GDAT_THCMPCROSS, 54
ADC_SEQ_GDAT_THCMPRANGE, 55
ADC_THR0_HIGH_THRHIGH, 55
ADC_THR0_LOW_THRLOW, 55
ADC_THR1_HIGH_THRHIGH, 55
ADC_THR1_LOW_THRLOW, 55
ADC_TRM_VRANGE, 56
ADC.cpp, 365
pin_index, 366
ADC.h, 367, 368
ADC0
 ADC Peripheral Access Layer, 32
ADC0_BASE
 ADC Peripheral Access Layer, 32
ADC_BASE_ADDRS
 ADC Peripheral Access Layer, 32
ADC_BASE_PTRS
 ADC Peripheral Access Layer, 32
ADC_CHAN_THRSEL_CH0_THRSEL
 ADC Register Masks, 36
ADC_CHAN_THRSEL_CH10_THRSEL
 ADC Register Masks, 36
ADC_CHAN_THRSEL_CH11_THRSEL
 ADC Register Masks, 36
ADC_CHAN_THRSEL_CH1_THRSEL
 ADC Register Masks, 36
ADC_CHAN_THRSEL_CH2_THRSEL
 ADC Register Masks, 36
ADC_CHAN_THRSEL_CH3_THRSEL
 ADC Register Masks, 37
ADC_CHAN_THRSEL_CH4_THRSEL
 ADC Register Masks, 37
ADC_CHAN_THRSEL_CH5_THRSEL
 ADC Register Masks, 37
ADC_CHAN_THRSEL_CH6_THRSEL
 ADC Register Masks, 37
ADC_CHAN_THRSEL_CH7_THRSEL
 ADC Register Masks, 37
ADC_CHAN_THRSEL_CH8_THRSEL
 ADC Register Masks, 38
ADC_CHAN_THRSEL_CH9_THRSEL
 ADC Register Masks, 38
ADC_CTRL_ASYNCMODE
 ADC Register Masks, 38
ADC_CTRL_CALMODE
 ADC Register Masks, 38
ADC_CTRL_CLKDIV
 ADC Register Masks, 39
ADC_CTRL_LPWRMODE
 ADC Register Masks, 39
ADC_DAT_CHANNEL
 ADC Register Masks, 39
ADC_DAT_COUNT
 ADC Register Masks, 39
ADC_DAT_DATAVALID
 ADC Register Masks, 40
ADC_DAT_OVERRUN
 ADC Register Masks, 40
ADC_DAT_RESULT
 ADC Register Masks, 40
ADC_DAT_THCMPCROSS
 ADC Register Masks, 40
ADC_DAT_THCMPRANGE
 ADC Register Masks, 41

ADC_FLAGS_OVERRUN0
 ADC Register Masks, 41
ADC_FLAGS_OVERRUN1
 ADC Register Masks, 41
ADC_FLAGS_OVERRUN10
 ADC Register Masks, 41
ADC_FLAGS_OVERRUN11
 ADC Register Masks, 42
ADC_FLAGS_OVERRUN2
 ADC Register Masks, 42
ADC_FLAGS_OVERRUN3
 ADC Register Masks, 42
ADC_FLAGS_OVERRUN4
 ADC Register Masks, 42
ADC_FLAGS_OVERRUN5
 ADC Register Masks, 42
ADC_FLAGS_OVERRUN6
 ADC Register Masks, 43
ADC_FLAGS_OVERRUN7
 ADC Register Masks, 43
ADC_FLAGS_OVERRUN8
 ADC Register Masks, 43
ADC_FLAGS_OVERRUN9
 ADC Register Masks, 43
ADC_FLAGS_OVR_INT
 ADC Register Masks, 43
ADC_FLAGS_SEQA_INT
 ADC Register Masks, 44
ADC_FLAGS_SEQA_OVR
 ADC Register Masks, 44
ADC_FLAGS_SEQB_INT
 ADC Register Masks, 44
ADC_FLAGS_SEQB_OVR
 ADC Register Masks, 44
ADC_FLAGS_THCMP0
 ADC Register Masks, 45
ADC_FLAGS_THCMP1
 ADC Register Masks, 45
ADC_FLAGS_THCMP10
 ADC Register Masks, 45
ADC_FLAGS_THCMP11
 ADC Register Masks, 45
ADC_FLAGS_THCMP2
 ADC Register Masks, 45
ADC_FLAGS_THCMP3
 ADC Register Masks, 46
ADC_FLAGS_THCMP4
 ADC Register Masks, 46
ADC_FLAGS_THCMP5
 ADC Register Masks, 46
ADC_FLAGS_THCMP6
 ADC Register Masks, 46
ADC_FLAGS_THCMP7
 ADC Register Masks, 46
ADC_FLAGS_THCMP8
 ADC Register Masks, 47
ADC_FLAGS_THCMP9
 ADC Register Masks, 47

ADC_FLAGS_THCMP_INT
 ADC Register Masks, 47
ADC_Group, 84
 ~ADC_Group, 86
 ADC_Group, 85
 adc_isr, 85
 DisableIRQ, 86
 EnableIRQ, 86
 error_t, 85
 GetValue, 86
 Handler, 86
 InitADCChanel, 87
 irq_source_inten, 85
 IsResultReady, 87
 RemoveADCChanel, 87
 SetLowPowerMode, 88

ADC_INTEN_ADCMPINTEN0
 ADC Register Masks, 47
ADC_INTEN_ADCMPINTEN1
 ADC Register Masks, 47
ADC_INTEN_ADCMPINTEN10
 ADC Register Masks, 48
ADC_INTEN_ADCMPINTEN11
 ADC Register Masks, 48
ADC_INTEN_ADCMPINTEN2
 ADC Register Masks, 48
ADC_INTEN_ADCMPINTEN3
 ADC Register Masks, 48
ADC_INTEN_ADCMPINTEN4
 ADC Register Masks, 48
ADC_INTEN_ADCMPINTEN5
 ADC Register Masks, 49
ADC_INTEN_ADCMPINTEN6
 ADC Register Masks, 49
ADC_INTEN_ADCMPINTEN7
 ADC Register Masks, 49
ADC_INTEN_ADCMPINTEN8
 ADC Register Masks, 49
ADC_INTEN_ADCMPINTEN9
 ADC Register Masks, 49
ADC_INTEN_OVR_INTEN
 ADC Register Masks, 50
ADC_INTEN_SEQA_INTEN
 ADC Register Masks, 50
ADC_INTEN_SEQB_INTEN
 ADC Register Masks, 50

adc_isr
 ADC_Group, 85
ADC_SEQ_CTRL_BURST
 ADC Register Masks, 50
ADC_SEQ_CTRL_CHANNELS
 ADC Register Masks, 51
ADC_SEQ_CTRL_COUNT
 ADC Register Masks, 51
ADC_SEQ_CTRL_LOWPRIO
 ADC Register Masks, 51
ADC_SEQ_CTRL_MODE
 ADC Register Masks, 51

ADC_SEQ_CTRL_SEQ_ENA
 ADC Register Masks, 52
ADC_SEQ_CTRL_SINGLESTEP
 ADC Register Masks, 52
ADC_SEQ_CTRL_START
 ADC Register Masks, 52
ADC_SEQ_CTRL_SYNCBYPASS
 ADC Register Masks, 53
ADC_SEQ_CTRL_TRIGGER
 ADC Register Masks, 53
ADC_SEQ_CTRL_TRIGPOL
 ADC Register Masks, 53
ADC_SEQ_GDAT_CHN
 ADC Register Masks, 53
ADC_SEQ_GDAT_COUNT
 ADC Register Masks, 54
ADC_SEQ_GDAT_DATAVALID
 ADC Register Masks, 54
ADC_SEQ_GDAT_OVERRUN
 ADC Register Masks, 54
ADC_SEQ_GDAT_THCMPCROSS
 ADC Register Masks, 54
ADC_SEQ_GDAT_THCMPRANGE
 ADC Register Masks, 55
ADC_SEQ IRQS
 ADC Peripheral Access Layer, 32
ADC_THR0_HIGH_THRHIGH
 ADC Register Masks, 55
ADC_THR0_LOW_THRLow
 ADC Register Masks, 55
ADC_THR1_HIGH_THRHIGH
 ADC Register Masks, 55
ADC_THR1_LOW_THRLow
 ADC Register Masks, 55
ADC_TRM_VRANGE
 ADC Register Masks, 56
ADC_Type, 88
 CHAN_THRSEL, 89
 CTRL, 89
 DAT, 89
 FLAGS, 89
 INTEN, 89
 RESERVED_0, 89
 RESERVED_1, 89
 SEQ_CTRL, 90
 SEQ_GDAT, 90
 THR0_HIGH, 90
 THR0_LOW, 90
 THR1_HIGH, 90
 THR1_LOW, 90
 TRM, 90
ADCCLKDIV
 SYSCON_Type, 234
ADCCLKSEL
 SYSCON_Type, 234
ADCGroup.cpp, 369
ADCGroup.h, 370
ADDR
 USART_Type, 251
AddSSEL
 SPI, 221
 SPIMaster, 228
Avanzar
 L298N, 166
 Puente_H, 195
B
 GPIO_Type, 128
barrido, 91
 Initialize, 91
 SetDigito, 91
Barrido.h, 268, 269
bases_t
 Timer, 245
BIAS
 DAC_t, 106
BODCTRL
 SYSCON_Type, 234
BRG
 USART_Type, 251
CALC_DISTANCIA
 HCSR04.h, 317
CALIB
 SysTick_t, 240
Callback, 92
 SetInterrupt, 93
 SWhandler, 93
 UnSetInterrupt, 93
Callback.cpp, 342
Callback.h, 343, 345
 TICK_MICROSECONDS, 345
 TICK_MILISECONDS, 345
 TICK_SECONDS, 345
CAP
 SCT_t, 212
CAPCTRL
 SCT_t, 212
CAPTCLKSEL
 SYSCON_Type, 234
CFG
 I2C_Type, 143
 SPI_Type, 224
 USART_Type, 251
CHAN_THRSEL
 ADC_Type, 89
CIENF
 PIN_INTERRUPT_t, 187
CIENR
 PIN_INTERRUPT_t, 187
CircularBuffer
 CircularBuffer< T >, 94
CircularBuffer< T >, 93
 CircularBuffer, 94
 isFull, 94

pop, 94
push, 95
CircularBuffer.h, 260, 262
Clear
 digito, 109
 Display, 111
 display7Segmentos, 113
 LCD, 170
ClearEOF
 SPI, 221
ClearEOT
 SPI, 221
ClearSSEL
 SPI, 221
CLKDIV
 I2C_Type, 143
CLKOUTDIV
 SYSCON_Type, 234
CLKOUTSEL
 SYSCON_Type, 234
CLR
 GPIO_Type, 128
 SCT_t, 212
ClrPin
 gpio, 124
 InOut, 156
CNT_ENA
 DAC_t, 106
CNTVAL
 DAC_t, 106
codigo_t
 digito, 108
Comunicacion.cpp, 253
Comunicacion.h, 253, 254
ComunicacionAsincronica, 95
 ~ComunicacionAsincronica, 95
 ComunicacionAsincronica, 95
 Read, 96
 UART_IRQHandler, 96
 Write, 96
ComunicacionAsincronica.h, 372, 373
ComunicacionSincronica, 97
 ~ComunicacionSincronica, 98
 ComunicacionSincronica, 98
 m_scl, 99
 Read, 98
 Write, 98
ComunicacionSincronica.h, 378, 380
conection_type
 ESP8266, 117
CONEN
 SCT_t, 212
CONFIG
 SCT_t, 212
CONFLAG
 SCT_t, 212
ConnectToServer
 ESP8266, 117
ConnectToWifi
 ESP8266, 117
Continue
 I2C, 139
COUNT
 SCT_t, 212
CR
 DAC_t, 106
CTL
 USART_Type, 251
CTRL
 ADC_Type, 89
 DAC_t, 106
 SCT_t, 213
 SysTick_t, 240
CURR
 SysTick_t, 240
DAC, 99
 ~DAC, 102
 DAC, 101
 dac_channel, 101
 dac_error, 101
 Get, 102
 GetMaxRange, 102
 Initialize, 102
 operator!=, 102
 operator<, 103
 operator<=, 103
 operator>, 104
 operator>=, 104
 operator=, 103
 operator==, 104
 Set, 105
 SetMaxRange, 105
DAC Peripheral Access Layer, 56
 DAC0, 57
 DAC1, 57
DAC.cpp, 360
DAC.h, 362, 364
 MAX_DAC_CHANNEL, 364
 MAX_DAC_VALUE, 364
DAC0
 DAC Peripheral Access Layer, 57
DAC1
 DAC Peripheral Access Layer, 57
dac_channel
 DAC, 101
dac_error
 DAC, 101
DAC_t, 105
 BIAS, 106
 CNT_ENA, 106
 CNTVAL, 106
 CR, 106
 CTRL, 106
 DBLBUF_ENA, 106
 DMA_ENA, 107
 INT_DMA_REQ, 107

RESERVED0, 107
 RESERVED1, 107
 RESERVED2, 107
 VALUE, 107
DAT
 ADC_Type, 89
DBLBUF_ENA
 DAC_t, 106
DEFAULT_ESP01_BAUDRATE
 ESP8266.h, 320
DEVICE_ID
 SYSCON_Type, 235
digito, 108
 Clear, 109
 codigo_t, 108
 digito, 109
 Get, 109
 modo_t, 108
 Set, 109
 SIMBOLOS, 108
Digito.cpp, 278
 Tabla_Digitos_BCD_7seg, 279
Digito.h, 279, 281
DIR
 GPIO_Type, 128
DIRCLR
 GPIO_Type, 128
DIRNOT
 GPIO_Type, 128
DIRSET
 GPIO_Type, 128
DisableInterrupt
 PinInterrupt, 193
DisableIRQ
 ADC_Group, 86
DisconnectToServer
 ESP8266, 118
DisconnectToWifi
 ESP8266, 118
Display, 110
 ~Display, 110
 Clear, 111
 Display, 110
 Write, 111
Display.h, 287, 288
display7Segmentos, 111
 ~display7Segmentos, 113
 Clear, 113
 display7Segmentos, 113
 Set, 113
 SWhandler, 114
 Write, 114
Display7Segmentos.cpp, 282
Display7Segmentos.h, 282, 284
 UPDATE_TICKS, 284
distancia, 114
 ~distancia, 115
 distancia, 115
 GetDistancia, 115
 operator==, 115
distancia.h, 311, 313
DISTANCIA_MAX
 HCSR04.h, 317
DIV
 SPI_Type, 224
DLY
 SPI_Type, 224
DMA_ENA
 DAC_t, 107
DMAREQ0
 SCT_t, 213
DMAREQ1
 SCT_t, 213
dr_pll.cpp, 399
 Inicializar_PLL, 400
dr_pll.h, 400, 402
 Inicializar_PLL, 401
Drivers, 15
 g_gpiohandler, 17
 g_Handler, 17
 g_MRThandler, 17
 IOCON_INDEX_PIO0, 18
 IOCON_INDEX_PIO1, 18
 Timer_Handler, 17
Drivers.h, 263, 264
EnableInterrupt
 PinInterrupt, 193
EnableIRQ
 ADC_Group, 86
error_t
 ADC_Group, 85
 Pin, 185
ESP8266, 116
 connection_type, 117
 ConnectToServer, 117
 ConnectToWifi, 117
 DisconnectToServer, 118
 DisconnectToWifi, 118
 GetIP, 118
 GetStatus, 118
 Initialize, 118
 IsConnectedToServer, 119
 IsConnectedToWifi, 119
 Read, 119
 SetIP, 119
 status_type, 117
 Write, 120
ESP8266.cpp, 318
ESP8266.h, 318, 320
 DEFAULT_ESP01_BAUDRATE, 320
 SEG_ESP01_TIMEOUT, 320
EV
 SCT_t, 213
EVEN
 SCT_t, 213
EVFLAG

SCT_t, 213
EXTCLKSEL
 SYSCON_Type, 235
EXTTRACECMD
 SYSCON_Type, 235

FALL
 PIN_INTERRUPT_t, 187
FCLKIN
 REGISTERS Peripheral Access Layer, 20
FCLKSEL
 SYSCON_Type, 235
FLAGS
 ADC_Type, 89
forceFinish
 SPI, 221
Frenar
 L298N, 166
 Puente_H, 195
FREQ_PRINCIPAL
 REGISTERS Peripheral Access Layer, 20
FREQ_SYSTICK
 SYSTICK Peripheral Access Layer, 24
FRG
 SYSCON_Type, 235
FRGCLKSEL
 SYSCON_Type, 235
FRGDIV
 SYSCON_Type, 235
FRGMULT
 SYSCON_Type, 235
FRODIRECTCLKUEN
 SYSCON_Type, 235
FROOSCCTRL
 SYSCON_Type, 235

g_gpiohandler
 Drivers, 17
g_Handler
 Drivers, 17
g_i2c
 I2C.cpp, 382
g_MRThandler
 Drivers, 17
g_spi
 SPI.cpp, 391
g_systick_freq
 systick.cpp, 404
 systick.h, 406
g_usart
 UART.cpp, 375
Get
 ADC, 83
 DAC, 102
 digito, 109
 teclado, 242
get
 Input, 162
GetDistancia
 distancia, 115
 HC_SR04, 134
GetHour
 Reloj, 210
GetIP
 ESP8266, 118
GetMaxRange
 DAC, 102
GetMin
 Reloj, 210
GetmrStandBy
 Timer, 245
GetPin
 gpio, 124
 InOut, 156
GetPulseOn
 PWM_Reader, 208
GetSeg
 Reloj, 210
GetState
 I2C, 139
GetStatus
 ESP8266, 118
GetTimer
 Timer, 245
GetTmrEvent
 Timer, 246
GetTmrRun
 Timer, 246
GetValue
 ADC_Group, 86
Girar
 L298N, 167
 Puente_H, 196
GirarDer
 L298N, 167
 Puente_H, 196
Girarlzq
 L298N, 167
 Puente_H, 196
GPIO
 GPIO Peripheral Access Layer, 22
gpio, 121
 ~gpio, 123
 ClrPin, 124
 GetPin, 124
 gpio, 123
 m_activity, 126
 m_direction, 126
 m_mode, 127
 operator=, 124
 SetDir, 125
 SetPin, 125
 SetPinMode, 125
 SetPinResistor, 125
 SetToggleDir, 126
 SetTogglePin, 126
 GPIO Peripheral Access Layer, 22

GPIO, 22
 gpio.cpp, 324
 gpio.h, 325, 327
 GPIO_Type, 127
 B, 128
 CLR, 128
 DIR, 128
 DIRCLR, 128
 DIRNOT, 128
 DIRSET, 128
 MASK, 128
 MPIN, 129
 NOT, 129
 PIN, 129
 RESERVED_0, 129
 RESERVED_1, 129
 RESERVED_10, 129
 RESERVED_2, 129
 RESERVED_3, 129
 RESERVED_4, 130
 RESERVED_5, 130
 RESERVED_6, 130
 RESERVED_7, 130
 RESERVED_8, 130
 RESERVED_9, 130
 SET, 130
 W, 130
 GpioHandler
 PinInterrupt, 193
 PWM_Reader, 208
 GrupoDeDigitos.h, 285, 286
 gruposdedigitos, 131
 gruposdedigitos, 131
 m_cantidad, 131
 m_comienzo, 131

 HALT
 SCT_t, 213
 Handler
 ADC_Group, 86
 hasData
 SPIMaster, 228
 HC_SR04, 132
 GetDistancia, 134
 HC_SR04, 133
 Initialize, 134
 Off, 134
 On, 134
 operator<, 134
 operator<=, 135
 operator>, 135
 operator>=, 136
 operator==, 135
 HCSR04.cpp, 313
 HCSR04.h, 315, 317
 CALC_DISTANCIA, 317
 DISTANCIA_MAX, 317
 PERIODO, 317

 I2C, 136
 ACK, 139
 ACKAddr, 139
 Continue, 139
 GetState, 139
 I2C, 138
 I2C_IRQHandler, 139
 I2C_states_t, 138
 Initialize, 139
 operator=, 140
 Read, 140
 SetTimeOut, 141
 Start, 141
 Stop, 141
 Write, 141
 I2C Peripheral Access Layer, 57
 I2C0, 58
 I2C0_BASE, 58
 I2C1, 58
 I2C1_BASE, 58
 I2C2, 58
 I2C2_BASE, 58
 I2C3, 58
 I2C3_BASE, 58
 I2C_BASE_ADDRS, 59
 I2C_BASE_PTRS, 59
 I2C Register Masks, 59
 I2C_CFG_MONCLKSTR, 61
 I2C_CFG_MONEN, 61
 I2C_CFG_MSTEN, 61
 I2C_CFG_SLVEN, 61
 I2C_CFG_TIMEOUTEN, 62
 I2C_INTENSET_EVENTTIMEOUTEN, 62
 I2C_INTENSET_MONIDLEEN, 62
 I2C_INTENSET_MONOVEN, 62
 I2C_INTENSET_MONRDYEN, 62
 I2C_INTENSET_MSTARBLOSSEN, 63
 I2C_INTENSET_MSTPENDINGEN, 63
 I2C_INTENSET_MSTSTSTPERREN, 63
 I2C_INTENSET_SCLTIMEOUTEN, 63
 I2C_INTENSET_SLVDESELEN, 63
 I2C_INTENSET_SLVNOTSTREN, 64
 I2C_INTENSET_SLVPENDINGEN, 64
 I2C_MONRXDAT_MONNACK, 64
 I2C_MONRXDAT_MONRESTART, 64
 I2C_MONRXDAT_MONSTART, 64
 I2C_MSTCTL_MSTCONTINUE, 65
 I2C_MSTCTL_MSTDMA, 65
 I2C_MSTCTL_MSTSTART, 65
 I2C_MSTCTL_MSTSTOP, 65
 I2C_MSTTIME_MSTSCLHIGH, 66
 I2C_MSTTIME_MSTSCLLOW, 66
 I2C_SLVADR_SADISABLE, 66
 I2C_SLVCTL_SLVCONTINUE, 66
 I2C_SLVCTL_SLVDMA, 67
 I2C_SLVCTL_SLVNACK, 67
 I2C_SLVQUAL0_QUALMODE0, 67
 I2C_STAT_EVENTTIMEOUT, 67

I2C_STAT_MONACTIVE, 68
I2C_STAT_MONIDLE, 68
I2C_STAT_MONOV, 68
I2C_STAT_MONRDY, 68
I2C_STAT_MSTARBLOSS, 69
I2C_STAT_MSTPENDING, 69
I2C_STAT_MSTSTATE, 69
I2C_STAT_MSTSTPERR, 69
I2C_STAT_SCLTIMEOUT, 70
I2C_STAT_SLVDESEL, 70
I2C_STAT_SLVIDX, 70
I2C_STAT_SLVNOTSTR, 70
I2C_STAT_SLVPENDING, 71
I2C_STAT_SLVSEL, 71
I2C_STAT_SLVSTATE, 71
I2C.cpp, 380
g_i2c, 382
MAX_IC2, 382
I2C.h, 382, 384
I2C_MAX_FREQ, 384
I2C0
 I2C Peripheral Access Layer, 58
I2C0_BASE
 I2C Peripheral Access Layer, 58
I2C1
 I2C Peripheral Access Layer, 58
I2C1_BASE
 I2C Peripheral Access Layer, 58
I2C2
 I2C Peripheral Access Layer, 58
I2C2_BASE
 I2C Peripheral Access Layer, 58
I2C3
 I2C Peripheral Access Layer, 58
I2C3_BASE
 I2C Peripheral Access Layer, 58
I2C_BASE_ADDRS
 I2C Peripheral Access Layer, 59
I2C_BASE_PTRS
 I2C Peripheral Access Layer, 59
I2C_CFG_MONCLKSTR
 I2C Register Masks, 61
I2C_CFG_MONEN
 I2C Register Masks, 61
I2C_CFG_MSTEN
 I2C Register Masks, 61
I2C_CFG_SLVEN
 I2C Register Masks, 61
I2C_CFG_TIMEOUTEN
 I2C Register Masks, 62
I2C_INTENSET_EVENTTIMEOUTEN
 I2C Register Masks, 62
I2C_INTENSET_MONIDLEEN
 I2C Register Masks, 62
I2C_INTENSET_MONOVEN
 I2C Register Masks, 62
I2C_INTENSET_MONRDYEN
 I2C Register Masks, 62
I2C_INTENSET_MSTARBLOSSEN
 I2C Register Masks, 63
I2C_INTENSET_MSTPENDINGEN
 I2C Register Masks, 63
I2C_INTENSET_MSTSTPERREN
 I2C Register Masks, 63
I2C_INTENSET_SCLTIMEOUTEN
 I2C Register Masks, 63
I2C_INTENSET_SLVDESELEN
 I2C Register Masks, 63
I2C_INTENSET_SLVNOTSTREN
 I2C Register Masks, 64
I2C_INTENSET_SLVPENDINGEN
 I2C Register Masks, 64
I2C_IRQHandler
 I2C, 139
I2C_MAX_FREQ
 I2C.h, 384
I2C_MONRXDAT_MONNACK
 I2C Register Masks, 64
I2C_MONRXDAT_MONRESTART
 I2C Register Masks, 64
I2C_MONRXDAT_MONSTART
 I2C Register Masks, 64
I2C_MSTCTL_MSTCONTINUE
 I2C Register Masks, 65
I2C_MSTCTL_MSTDMA
 I2C Register Masks, 65
I2C_MSTCTL_MSTSTART
 I2C Register Masks, 65
I2C_MSTCTL_MSTSTOP
 I2C Register Masks, 65
I2C_MSTTIME_MSTSCLHIGH
 I2C Register Masks, 66
I2C_MSTTIME_MSTSCLLOW
 I2C Register Masks, 66
I2C_SLVADR_SADISABLE
 I2C Register Masks, 66
I2C_SLVCTL_SLVCONTINUE
 I2C Register Masks, 66
I2C_SLVCTL_SLVDMA
 I2C Register Masks, 67
I2C_SLVCTL_SLVNACK
 I2C Register Masks, 67
I2C_SLVQUAL0_QUALMODE0
 I2C Register Masks, 67
I2C_STAT_EVENTTIMEOUT
 I2C Register Masks, 67
I2C_STAT_MONACTIVE
 I2C Register Masks, 68
I2C_STAT_MONIDLE
 I2C Register Masks, 68
I2C_STAT_MONOV
 I2C Register Masks, 68
I2C_STAT_MONRDY
 I2C Register Masks, 68
I2C_STAT_MSTARBLOSS
 I2C Register Masks, 69

I2C_STAT_MSTPENDING
 I2C Register Masks, 69

I2C_STAT_MSTSTATE
 I2C Register Masks, 69

I2C_STAT_MSTSTSTPERR
 I2C Register Masks, 69

I2C_STAT_SCLTIMEOUT
 I2C Register Masks, 70

I2C_STAT_SLVDESEL
 I2C Register Masks, 70

I2C_STAT_SLVIDX
 I2C Register Masks, 70

I2C_STAT_SLVNOTSTR
 I2C Register Masks, 70

I2C_STAT_SLPENDING
 I2C Register Masks, 71

I2C_STAT_SLVSEL
 I2C Register Masks, 71

I2C_STAT_SLVSTATE
 I2C Register Masks, 71

I2C_states_t
 I2C, 138

I2C_Type, 143
 CFG, 143
 CLKDIV, 143
 INTENCLR, 144
 INTENSET, 144
 INTSTAT, 144
 MONRXDAT, 144
 MSTCTL, 144
 MSTDAT, 144
 MSTTIME, 144
 RESERVED_0, 145
 RESERVED_1, 145
 RESERVED_2, 145
 SLVADR, 145
 SLVCTL, 145
 SLVDAT, 145
 SLVQUAL0, 145
 STAT, 145
 TIMEOUT, 146

I2CMaster, 146
 I2CMaster, 148
 Initialize, 149
 isIdle, 149
 Read, 149
 RequestRead, 150
 Write, 150

I2CMaster.cpp, 385

I2CMaster.h, 386, 389

I4017, 151
 I4017, 152
 SetClock, 152
 SetDigito, 152
 SetReset, 152

I4017.cpp, 270

I4017.h, 271, 273

I4511, 153

I4511, 154
 SetSegmentos, 154

I4511.cpp, 274

I4511.h, 274, 276

ICER
 NVIC_Type, 177

IENF
 PIN_INTERRUPT_t, 187

IENR
 PIN_INTERRUPT_t, 188

inicializar.cpp, 255

inicializar.h, 256, 257

Inicializar_PLL
 dr_pll.cpp, 400
 dr_pll.h, 401

Inicializar_SysTick
 systick.cpp, 404
 systick.h, 406

InitADCChanel
 ADC_Group, 87

Initialize
 ADC, 83
 barrido, 91
 DAC, 102
 ESP8266, 118
 HC_SR04, 134
 I2C, 139
 I2CMaster, 149
 Input, 162
 L298N, 167
 LCD, 170
 Output, 182
 Puente_H, 196
 Pwm, 200
 PWM_Reader, 208
 segmentos, 217
 SPI, 222
 SPIMaster, 229
 teclado, 242

InOut, 155
 ~InOut, 156
 ClrPin, 156
 GetPin, 156
 InOut, 156
 SetDir, 156
 SetPin, 156
 SetPinMode, 156
 SetPinResistor, 157
 SetToggleDir, 157
 SetTogglePin, 157

InOut.h, 328, 329

INPUT
 SCT_t, 213

Input, 158
 ~Input, 161
 get, 162
 Initialize, 162
 Input, 161

operator!=, 162
operator==, 162, 163
SWhandler, 163
Input.cpp, 334
 operator==, 335
Input.h, 335, 337
 MAX_BOUCNE, 337
INSTRUCTION_8BITS
 LCD.h, 292
INSTRUCTION_CLEAR_DISP
 LCD.h, 292
INSTRUCTION_DISP_CTRL
 LCD.h, 292
INSTRUCTION_ENTRY_MODE_SET
 LCD.h, 292
INSTRUCTION_FUNC_SET
 LCD.h, 292
INSTRUCTION_SET_POS
 LCD.h, 292
int16_t
 tipos.h, 433
INT32_MAX
 tipos.h, 433
INT32_MIN
 tipos.h, 433
int32_t
 tipos.h, 433
INT8_MAX
 tipos.h, 433
INT8_MIN
 tipos.h, 433
int8_t
 tipos.h, 433
INT_DMA_REQ
 DAC_t, 107
INTEN
 ADC_Type, 89
INTENCLR
 I2C_Type, 144
 SPI_Type, 224
 USART_Type, 251
INTENSET
 I2C_Type, 144
 SPI_Type, 224
 USART_Type, 251
INTSTAT
 I2C_Type, 144
 SPI_Type, 225
 USART_Type, 251
IOCON
 IOCON Peripheral Access Layer, 23
IOCON Peripheral Access Layer, 22
 IOCON, 23
IOCON_INDEX_PIO0
 Drivers, 18
IOCON_INDEX_PIO1
 Drivers, 18
IOCON_Type, 163
 PIO, 164
 IOCONCLKDIV0
 SYSCON_Type, 235
 IOCONCLKDIV1
 SYSCON_Type, 235
 IOCONCLKDIV2
 SYSCON_Type, 236
 IOCONCLKDIV3
 SYSCON_Type, 236
 IOCONCLKDIV4
 SYSCON_Type, 236
 IOCONCLKDIV5
 SYSCON_Type, 236
 IOCONCLKDIV6
 SYSCON_Type, 236
IP
 NVIC_Type, 177
irq_source_inten
 ADC_Group, 85
IRQLATENCY
 SYSCON_Type, 236
IsConnectedToServer
 ESP8266, 119
IsConnectedToWifi
 ESP8266, 119
ISEL
 PIN_INTERRUPT_t, 188
ISER
 NVIC_Type, 177
isFull
 CircularBuffer< T >, 94
IsIdle
 SPIMaster, 229
isIdle
 I2CMaster, 149
ISPR
 NVIC_Type, 177
IsResultReady
 ADC, 83
 ADC_Group, 87
isRxReady
 SPI, 222
IST
 PIN_INTERRUPT_t, 188
isTxReady
 SPI, 222
KITLPC845, 1
L298N, 164
 ~L298N, 166
 Avanzar, 166
 Frenar, 166
 Girar, 167
 GirarDer, 167
 Girarlzq, 167
 Initialize, 167
 L298N, 166
 Retroceder, 168

L298N.cpp, 293
L298N.h, 294, 295
LCD, 168
 Clear, 170
 Initialize, 170
 LCD, 170
 operator=, 171
 SWhandler, 172
 Write, 172
 WriteAt, 172, 173
LCD.cpp, 289
LCD.h, 290, 292
 INSTRUCTION_8BITS, 292
 INSTRUCTION_CLEAR_DISP, 292
 INSTRUCTION_DISP_CTRL, 292
 INSTRUCTION_ENTRY_MODE_SET, 292
 INSTRUCTION_FUNC_SET, 292
 INSTRUCTION_SET_POS, 292
 MAX_PIN_COUNT, 292
 ROW_OFFSET, 292
LIMIT
 SCT_t, 213
LPC845.h, 407, 414

m_activity
 gpio, 126
 Pwm, 202
m_bit
 Pin, 186
m_cant
 PinInterrupt, 193
m_cantidad
 gruposdedigitos, 131
m_comienzo
 gruposdedigitos, 131
m_direction
 gpio, 126
m_error
 Pin, 186
m_interrupt_mode
 PinInterrupt, 193
m_interrupt_number
 PinInterrupt, 194
m_mode
 gpio, 127
m_port
 Pin, 186
m_pwm_channel
 Pwm, 202
m_scl
 ComunicacionSincronica, 99
m_SPI_register
 SPI, 224
m_timer_channel
 MRThandler, 176
m_TmrBase
 Timer, 249
m_TmrEvent
 Timer, 249
 m_TmrHandler
 Timer, 249
 m_TmrRun
 Timer, 249
 m_TmrStandBy
 Timer, 249
 m_toff
 Pwm, 202
 m_ton
 Pwm, 202
MAINCLKPLLSEL
 SYSCON_Type, 236
MAINCLKPLLUN
 SYSCON_Type, 236
MAINCLKSEL
 SYSCON_Type, 236
MAINCLKUEN
 SYSCON_Type, 236
MASK
 GPIO_Type, 128
MATCH
 SCT_t, 213
MATCHREL
 SCT_t, 213
MAX_BOUNCE
 Input.h, 337
MAX_DAC_CHANNEL
 DAC.h, 364
MAX_DAC_VALUE
 DAC.h, 364
MAX_IC2
 I2C.cpp, 382
MAX_MRT_CHANNEL
 MRThandler.h, 355
MAX_PIN_COUNT
 LCD.h, 292
MAX_PININTERRUPT
 Pininterrupt.h, 333
MAX_SPI
 SPI.cpp, 391
MAX_SSEL_SPI0
 SPI.h, 393
MAX_SSEL_SPI1
 SPI.h, 393
MAX_TICKS
 systick.cpp, 404
Misc, 15
MODE
 MRT_t, 173
modo_t
 digito, 108
MONRXDAT
 I2C_Type, 144
MPIN
 GPIO_Type, 129
MRT
 MRT Peripheral Access Layer, 25
 MRT Peripheral Access Layer, 24

MRT, 25
MRT0, 25
MRT1, 25
MRT2, 25
MRT3, 25
MRT_IDLE_CH, 25
MRT_IRQ_FLAQ, 25
MRT0
 MRT Peripheral Access Layer, 25
MRT1
 MRT Peripheral Access Layer, 25
MRT2
 MRT Peripheral Access Layer, 25
MRT3
 MRT Peripheral Access Layer, 25
MRT_get_time
 MRThandler, 176
MRT_IDLE_CH
 MRT Peripheral Access Layer, 25
MRT_IRQ_FLAQ
 MRT Peripheral Access Layer, 25
MRT_MODES
 MRThandler.h, 355
MRT_reset_time
 MRThandler, 176
MRT_t, 173
 MODE, 173
 RESERVED0, 173
 RESERVED1, 174
 RESERVED2, 174
 RUN, 174
MRT_timer_channels
 MRThandler.h, 355
MRThandler
 MRThandler, 176
MRThandler, 174
 ~MRThandler, 175
 m_timer_channel, 176
 MRT_get_time, 176
 MRT_reset_time, 176
 MRThandler, 176
 MRThandler, 175
MRThandler.cpp, 352
MRThandler.h, 353, 355
 MAX_MRT_CHANNEL, 355
 MRT_MODES, 355
 MRT_timer_channels, 355
MSTCTL
 I2C_Type, 144
MSTDAT
 I2C_Type, 144
MSTTIME
 I2C_Type, 144
NMISRC
 SYSCON_Type, 236
NO_DATA
 SPIMaster.h, 398
NO_KEY

 teclado.h, 267
NOT
 GPIO_Type, 129
NVIC
 NVIC Peripheral Access Layer, 26
NVIC Peripheral Access Layer, 26
 NVIC, 26
 NVIC_BASE, 26
 SCB_BASE, 26
 SCS_BASE, 26
 SysTick_BASE, 27
NVIC_BASE
 NVIC Peripheral Access Layer, 26
NVIC_Type, 176
 ICER, 177
 IP, 177
 ISER, 177
 ISPR, 177
 RESERVED0, 177
 RESERVED2, 178
 RESERVED3, 178
 RESERVED4, 178
 RSERVED1, 178
Off
 HC_SR04, 134
 Output, 182
 Pwm, 200
 PWM_Reader, 208
On
 HC_SR04, 134
 Output, 183
 Pwm, 201
 PWM_Reader, 208
operator bool
 Timer, 246
operator!
 Timer, 246
operator!=
 DAC, 102
 Input, 162
operator<
 DAC, 103
 HC_SR04, 134
operator<<
 timers, 250
operator<=>
 DAC, 103
 HC_SR04, 135
operator>
 DAC, 104
 HC_SR04, 135
operator>=
 DAC, 104
 HC_SR04, 136
operator=>
 DAC, 103
 gpio, 124
 I2C, 140

LCD, 171
 Output, 183
 Timer, 246
operator==
 DAC, 104
 distancia, 115
 HC_SR04, 135
 Input, 162, 163
 Input.cpp, 335
 Output, 183
 Timer, 246, 248
OSR
 USART_Type, 251
OUT
 SCT_t, 213
OUTPUT
 SCT_t, 214
Output, 179
 ~Output, 182
 Initialize, 182
 Off, 182
 On, 183
 operator=, 183
 operator==, 183
 Output, 182
 SWhandler, 184
Output.cpp, 338
Output.h, 340, 341
OUTPUTDIRCTRL
 SCT_t, 214
PDAWAKECFG
 SYSCON_Type, 236
PDRUNCFG
 SYSCON_Type, 237
PDSLEEPcfg
 SYSCON_Type, 237
PERIODO
 HCSR04.h, 317
PIN
 GPIO_Type, 129
Pin, 184
 error_t, 185
 m_bit, 186
 m_error, 186
 m_port, 186
 Pin, 186
 port_t, 185
Pin.cpp, 321
Pin.h, 322, 323
pin_index
 ADC.cpp, 366
PIN_INT Peripheral Access Layer, 27
 PIN_INTERRUPT, 27
PIN_INTERRUPT
 PIN_INT Peripheral Access Layer, 27
PIN_INTERRUPT_t, 187
 CIENF, 187
 CIENR, 187
 FALL, 187
 IENF, 187
 IENR, 188
 ISEL, 188
 IST, 188
 PMCFG, 188
 PMCTRL, 188
 PMSRC, 188
 RISE, 188
 SIENF, 188
 SIENR, 189
PINASSIGN0
 SWM_t, 231
PINASSIGN1
 SWM_t, 231
PINASSIGN10
 SWM_t, 231
PINASSIGN11
 SWM_t, 231
PINASSIGN12
 SWM_t, 231
PINASSIGN13
 SWM_t, 231
PINASSIGN14
 SWM_t, 231
PINASSIGN2
 SWM_t, 231
PINASSIGN3
 SWM_t, 231
PINASSIGN4
 SWM_t, 232
PINASSIGN5
 SWM_t, 232
PINASSIGN6
 SWM_t, 232
PINASSIGN7
 SWM_t, 232
PINASSIGN8
 SWM_t, 232
PINASSIGN9
 SWM_t, 232
PINASSIGN_DATA
 SWM_t, 232
PINENABLE0
 SWM_t, 232
PINENABLE1
 SWM_t, 232
PinInterrupt, 189
 ~PinInterrupt, 192
 DisableInterrupt, 193
 EnableInterrupt, 193
 GpioHandler, 193
 m_cant, 193
 m_interrupt_mode, 193
 m_interrupt_number, 194
 PinInterrupt, 192
 PinInterrupt_Inicializar, 193
Pininterrupt.cpp, 329

Pininterrupt.h, 331, 333
 MAX_PININTERRUPT, 333
PinInterrupt_Inicializar
 PinInterrupt, 193
PINTSEL
 SYSCON_Type, 237
PIO
 IOCON_Type, 164
PIOPORCAP
 SYSCON_Type, 237
PMCFG
 PIN_INTERRUPT_t, 188
PMCTRL
 PIN_INTERRUPT_t, 188
PMSRC
 PIN_INTERRUPT_t, 188
pop
 CircularBuffer< T >, 94
port_t
 Pin, 185
PRESETCTRL0
 SYSCON_Type, 237
PRESETCTRL1
 SYSCON_Type, 237
Puente_H, 194
 ~Puente_H, 195
 Avanzar, 195
 Frenar, 195
 Girar, 196
 GirarDer, 196
 Girarlzq, 196
 Initialize, 196
 Puente_H, 195
 Retroceder, 196
PuenteH.h, 297, 299
push
 CircularBuffer< T >, 95
Pwm, 197
 ~Pwm, 200
 activity_t, 199
 Initialize, 200
 m_activity, 202
 m_pwm_channel, 202
 m_toff, 202
 m_ton, 202
 Off, 200
 On, 201
 Pwm, 200
 pwm_channel_t, 199
 pwm_time_unit_t, 199
 SetPeriod, 201
 SetTon, 201
Pwm.cpp, 303
Pwm.h, 304, 306
pwm_channel_t
 Pwm, 199
PWM_Reader, 202
 GetPulseOn, 208
 GpioHandler, 208
 Initialize, 208
 Off, 208
 On, 208
 PWM_Reader, 207
pwm_time_unit_t
 Pwm, 199
PWMReader.cpp, 307
PWMReader.h, 308, 310
Read
 ComunicacionAsincronica, 96
 ComunicacionSincronica, 98
 ESP8266, 119
 I2C, 140
 I2CMaster, 149
 SPI, 222
 SPIMaster, 229
REGISTERS Peripheral Access Layer, 19
 FCLKIN, 20
 FREQ_PRINCIPAL, 20
REGMODE
 SCT_t, 214
RELOAD
 SysTick_t, 240
Reloj, 209
 GetHour, 210
 GetMin, 210
 GetSeg, 210
 Reloj, 210
 Reset, 210
 SetTime, 211
 SWhandler, 211
Reloj.cpp, 299
Reloj.h, 300, 302
RemoveADCChanel
 ADC_Group, 87
RemoveSSEL
 SPI, 223
 SPIMaster, 229
RequestRead
 I2CMaster, 150
 SPIMaster, 230
RES
 SCT_t, 214
RESERVED0
 DAC_t, 107
 MRT_t, 173
 NVIC_Type, 177
RESERVED1
 DAC_t, 107
 MRT_t, 174
RESERVED2
 DAC_t, 107
 MRT_t, 174
 NVIC_Type, 178
RESERVED3
 NVIC_Type, 178
RESERVED4

NVIC_Type, 178
 RESERVED_0
 ADC_Type, 89
 GPIO_Type, 129
 I2C_Type, 145
 SCT_t, 214
 SWM_t, 232
 SYSCON_Type, 237
 RESERVED_1
 ADC_Type, 89
 GPIO_Type, 129
 I2C_Type, 145
 SCT_t, 214
 SYSCON_Type, 237
 RESERVED_10
 GPIO_Type, 129
 SYSCON_Type, 237
 RESERVED_11
 SYSCON_Type, 237
 RESERVED_12
 SYSCON_Type, 237
 RESERVED_13
 SYSCON_Type, 237
 RESERVED_14
 SYSCON_Type, 238
 RESERVED_2
 GPIO_Type, 129
 I2C_Type, 145
 SCT_t, 214
 SYSCON_Type, 238
 RESERVED_3
 GPIO_Type, 129
 SCT_t, 214
 SYSCON_Type, 238
 RESERVED_4
 GPIO_Type, 130
 SCT_t, 214
 SYSCON_Type, 238
 RESERVED_5
 GPIO_Type, 130
 SYSCON_Type, 238
 RESERVED_6
 GPIO_Type, 130
 SYSCON_Type, 238
 RESERVED_7
 GPIO_Type, 130
 SYSCON_Type, 238
 RESERVED_8
 GPIO_Type, 130
 SYSCON_Type, 238
 RESERVED_9
 GPIO_Type, 130
 SYSCON_Type, 238
 Reset
 Reloj, 210
 Retroceder
 L298N, 168
 Puente_H, 196
 RISE
 PIN_INTERRUPT_t, 188
 ROW_OFFSET
 LCD.h, 292
 RSERVED1
 NVIC_Type, 178
 RUN
 MRT_t, 174
 RXDAT
 SPI_Type, 225
 USART_Type, 252
 RXDATSTAT
 USART_Type, 252
 SCB_BASE
 NVIC Peripheral Access Layer, 26
 SCS_BASE
 NVIC Peripheral Access Layer, 26
 SCT
 SCT Peripheral Access Layer, 28
 SCT Peripheral Access Layer, 28
 SCT, 28
 SCT_t, 211
 CAP, 212
 CAPCTRL, 212
 CLR, 212
 CONEN, 212
 CONFIG, 212
 CONFLAG, 212
 COUNT, 212
 CTRL, 213
 DMAREQ0, 213
 DMAREQ1, 213
 EV, 213
 EVEN, 213
 EVFLAG, 213
 HALT, 213
 INPUT, 213
 LIMIT, 213
 MATCH, 213
 MATCHREL, 213
 OUT, 213
 OUTPUT, 214
 OUTPUTDIRCTRL, 214
 REGMODE, 214
 RES, 214
 RESERVED_0, 214
 RESERVED_1, 214
 RESERVED_2, 214
 RESERVED_3, 214
 RESERVED_4, 214
 SET, 214
 START, 214
 STATE, 214
 STOP, 215
 SCTCLKDIV
 SYSCON_Type, 238
 SCTCLKSEL
 SYSCON_Type, 238

SCTimer, 215
~SCTimer, 216
SetAutoLimit, 216
SetSwitchMatrizSCTOUT, 216
SetTime, 216
SetUnify, 216
StartTimer, 217
StopTimer, 217
SCTimer.cpp, 356
SCTimer.h, 358, 360
SEG_ESP01_TIMEOUT
 ESP8266.h, 320
segmentos, 217
 Initialize, 217
 SetSegmentos, 218
Segmentos.h, 276, 278
Semaforo.cpp, 258
Semaforo.h, 258, 259
SEQ_CTRL
 ADC_Type, 90
SEQ_GDAT
 ADC_Type, 90
SET
 GPIO_Type, 130
 SCT_t, 214
Set
 DAC, 105
 digito, 109
 display7Segmentos, 113
SetAutoLimit
 SCTimer, 216
SetClock
 I4017, 152
SetDigito
 barrido, 91
 I4017, 152
SetDir
 gpio, 125
 InOut, 156
SetEOT
 SPI, 223
SetInterrupt
 Callback, 93
SetIP
 ESP8266, 119
SetLowPowerMode
 ADC_Group, 88
SetMaxRange
 DAC, 105
SetmrStandBy
 Timer, 247
SetPeriod
 Pwm, 201
SetPin
 gpio, 125
 InOut, 156
SetPinMode
 gpio, 125
 InOut, 156
 SetPinResistor
 gpio, 125
 InOut, 157
 SetReset
 I4017, 152
 SetSegmentos
 I4511, 154
 segmentos, 218
 SetSSEL
 SPI, 223
 SetSwitchMatrizSCTOUT
 SCTimer, 216
 SetTime
 Reloj, 211
 SCTimer, 216
 SetTimeOut
 I2C, 141
 SetTimer
 Timer, 247
 SetTimerBase
 Timer, 247
 SetTmrHandler
 Timer, 247
 SetToggleDir
 gpio, 126
 InOut, 157
 SetTogglePin
 gpio, 126
 InOut, 157
 SetTon
 Pwm, 201
 SetUnify
 SCTimer, 216
SIENF
 PIN_INTERRUPT_t, 188
SIENR
 PIN_INTERRUPT_t, 189
SIMBOLOS
 digito, 108
SLVADR
 I2C_Type, 145
SLVCTL
 I2C_Type, 145
SLVDAT
 I2C_Type, 145
SLVQUAL0
 I2C_Type, 145
SPI, 218
 ~SPI, 221
 AddSSEL, 221
 ClearEOF, 221
 ClearEOT, 221
 ClearSSEL, 221
 forceFinish, 221
 Initialize, 222
 isRxReady, 222
 isTxReady, 222

m_SPI_register, 224
 Read, 222
 Removessel, 223
 SetEOT, 223
 SetSSEL, 223
 SPI, 220
 SPI_MODE_0, 220
 SPI_MODE_1, 220
 SPI_MODE_2, 220
 SPI_MODE_3, 220
 SPI_mode_t, 220
 SPI_role_t, 220
 Write, 223
 SPI Peripheral Access Layer, 72
 SPI0, 72
 SPI0_BASE, 72
 SPI1, 73
 SPI1_BASE, 73
 SPI_BASE_ADDRS, 73
 SPI_BASE_PTRS, 73
 SPI Register Masks, 73
 SPI_CFG_CPHA, 74
 SPI_CFG_CPOL, 74
 SPI_CFG_ENABLE, 75
 SPI_CFG_LOOP, 75
 SPI_CFG_LSBF, 75
 SPI_CFG_MASTER, 75
 SPI_CFG_SPOL0, 75
 SPI_CFG_SPOL1, 76
 SPI_CFG_SPOL2, 76
 SPI_CFG_SPOL3, 76
 SPI_INTENSET_RXOVEN, 76
 SPI_INTENSET_RXRDYEN, 76
 SPI_INTENSET_SSAEN, 77
 SPI_INTENSET_SSDEN, 77
 SPI_INTENSET_TXRDYEN, 77
 SPI_INTENSET_TXUREN, 77
 SPI_TXDATCTL_EOF, 78
 SPI_TXDATCTL_EOT, 78
 SPI_TXDATCTL_LEN, 78
 SPI_TXDATCTL_RXIGNORE, 78
 SPI_TXDATCTL_TXSSEL0_N, 79
 SPI_TXDATCTL_TXSSEL1_N, 79
 SPI_TXDATCTL_TXSSEL2_N, 79
 SPI_TXDATCTL_TXSSEL3_N, 79
 SPI.cpp, 389
 g_spi, 391
 MAX_SPI, 391
 SPI.h, 391, 393
 MAX_SSEL_SPI0, 393
 MAX_SSEL_SPI1, 393
 SPI0
 SPI Peripheral Access Layer, 72
 SPI0_BASE
 SPI Peripheral Access Layer, 72
 SPI1
 SPI Peripheral Access Layer, 73
 SPI1_BASE

SPI Peripheral Access Layer, 73
 SPI_BASE_ADDRS
 SPI Peripheral Access Layer, 73
 SPI_BASE_PTRS
 SPI Peripheral Access Layer, 73
 SPI_CFG_CPHA
 SPI Register Masks, 74
 SPI_CFG_CPOL
 SPI Register Masks, 74
 SPI_CFG_ENABLE
 SPI Register Masks, 75
 SPI_CFG_LOOP
 SPI Register Masks, 75
 SPI_CFG_LSBF
 SPI Register Masks, 75
 SPI_CFG_MASTER
 SPI Register Masks, 75
 SPI_CFG_SPOL0
 SPI Register Masks, 75
 SPI_CFG_SPOL1
 SPI Register Masks, 76
 SPI_CFG_SPOL2
 SPI Register Masks, 76
 SPI_CFG_SPOL3
 SPI Register Masks, 76
 SPI_INTENSET_RXOVEN
 SPI Register Masks, 76
 SPI_INTENSET_RXRDYEN
 SPI Register Masks, 76
 SPI_INTENSET_SSAEN
 SPI Register Masks, 77
 SPI_INTENSET_SSDEN
 SPI Register Masks, 77
 SPI_INTENSET_TXRDYEN
 SPI Register Masks, 77
 SPI_INTENSET_TXUREN
 SPI Register Masks, 77
 SPI_MODE_0
 SPI, 220
 SPI_MODE_1
 SPI, 220
 SPI_MODE_2
 SPI, 220
 SPI_MODE_3
 SPI, 220
 SPI_mode_t
 SPI, 220
 SPI_role_t
 SPI, 220
 SPI_TXDATCTL_EOF
 SPI Register Masks, 78
 SPI_TXDATCTL_EOT
 SPI Register Masks, 78
 SPI_TXDATCTL_LEN
 SPI Register Masks, 78
 SPI_TXDATCTL_RXIGNORE
 SPI Register Masks, 78
 SPI_TXDATCTL_TXSSEL0_N

SPI Register Masks, 79
SPI_TXDATCTL_TXSSEL1_N
 SPI Register Masks, 79
SPI_TXDATCTL_TXSSEL2_N
 SPI Register Masks, 79
SPI_TXDATCTL_TXSSEL3_N
 SPI Register Masks, 79
SPI_Type, 224
 CFG, 224
 DIV, 224
 DLY, 224
 INTENCLR, 224
 INTENSET, 224
 INTSTAT, 225
 RXDAT, 225
 STAT, 225
 TXCTL, 225
 TXDAT, 225
 TXDATCTL, 225
SPIMaster, 225
 AddSSEL, 228
 hasData, 228
 Initialize, 229
 IsIdle, 229
 Read, 229
 RemoveSSEL, 229
 RequestRead, 230
 SPIMaster, 228
 Write, 230
SPIMaster.cpp, 394
SPIMaster.h, 395, 398
 NO_DATA, 398
StandByTimer
 Timer, 247
START
 SCT_t, 214
Start
 I2C, 141
STARTERP0
 SYSCON_Type, 238
STARTERP1
 SYSCON_Type, 239
StartTimer
 SCtimer, 217
STAT
 I2C_Type, 145
 SPI_Type, 225
 USART_Type, 252
STATE
 SCT_t, 214
status_type
 ESP8266, 117
STOP
 SCT_t, 215
Stop
 I2C, 141
StopTimer
 SCtimer, 217

SWhandler
 Callback, 93
 display7Segmentos, 114
 Input, 163
 LCD, 172
 Output, 184
 Reloj, 211
 teclado, 242

SWM
 SWM Peripheral Access Layer, 31
SWM Peripheral Access Layer, 31
 SWM, 31
SWM_t, 231
 PINASSIGN0, 231
 PINASSIGN1, 231
 PINASSIGN10, 231
 PINASSIGN11, 231
 PINASSIGN12, 231
 PINASSIGN13, 231
 PINASSIGN14, 231
 PINASSIGN2, 231
 PINASSIGN3, 231
 PINASSIGN4, 232
 PINASSIGN5, 232
 PINASSIGN6, 232
 PINASSIGN7, 232
 PINASSIGN8, 232
 PINASSIGN9, 232
 PINASSIGN_DATA, 232
 PINENABLE0, 232
 PINENABLE1, 232
 RESERVED_0, 232

SYSAHBCLKCTRL0
 SYSCON_Type, 239
SYSAHBCLKCTRL1
 SYSCON_Type, 239
SYSAHBCLKDIV
 SYSCON_Type, 239
SYSCON
 SYSCON Peripheral Access Layer, 21
SYSCON Peripheral Access Layer, 20
 SYSCON, 21
 SYSCON_BASE, 21
 SYSCON_BASE_ADDRS, 21
 SYSCON_BASE_PTRS, 21
 SYSCON_IRQS, 21
 SYSCON_BASE
 SYSCON Peripheral Access Layer, 21
 SYSCON_BASE_ADDRS
 SYSCON Peripheral Access Layer, 21
 SYSCON_BASE_PTRS
 SYSCON Peripheral Access Layer, 21
 SYSCON_IRQS
 SYSCON Peripheral Access Layer, 21
 SYSCON_Type, 233
 ADCCLKDIV, 234
 ADCCLKSEL, 234
 BODCTRL, 234

CAPTCLKSEL, 234
 CLKOUTDIV, 234
 CLKOUTSEL, 234
 DEVICE_ID, 235
 EXTCLKSEL, 235
 EXTRACECMD, 235
 FCLKSEL, 235
 FRG, 235
 FRGCLKSEL, 235
 FRGDIV, 235
 FRGMULT, 235
 FRODIRECTCLKUEN, 235
 FROOSCCTRL, 235
 IOCONCLKDIV0, 235
 IOCONCLKDIV1, 235
 IOCONCLKDIV2, 236
 IOCONCLKDIV3, 236
 IOCONCLKDIV4, 236
 IOCONCLKDIV5, 236
 IOCONCLKDIV6, 236
 IRQLATENCY, 236
 MAINCLKPLLSEL, 236
 MAINCLKPLLUEN, 236
 MAINCLKSEL, 236
 MAINCLKUEN, 236
 NMISRC, 236
 PDAWAKECFG, 236
 PDRUNCFG, 237
 PDSLEEPCFG, 237
 PINTSEL, 237
 PIOPORCAP, 237
 PRESETCTRL0, 237
 PRESETCTRL1, 237
 RESERVED_0, 237
 RESERVED_1, 237
 RESERVED_10, 237
 RESERVED_11, 237
 RESERVED_12, 237
 RESERVED_13, 237
 RESERVED_14, 238
 RESERVED_2, 238
 RESERVED_3, 238
 RESERVED_4, 238
 RESERVED_5, 238
 RESERVED_6, 238
 RESERVED_7, 238
 RESERVED_8, 238
 RESERVED_9, 238
 SCTCLKDIV, 238
 SCTCLKSEL, 238
 STARTERP0, 238
 STARTERP1, 239
 SYSAHBCLKCTRL0, 239
 SYSAHBCLKCTRL1, 239
 SYSAHBCLKDIV, 239
 SYSMEMREMAP, 239
 SYSOSCCTRL, 239
 SYSPLLCLKSEL, 239
 SYSPLLCLKUEN, 239
 SYSPLLCTRL, 239
 SYSPLLSTAT, 239
 SYSRSTSTAT, 239
 SYSTCKCAL, 239
 WDTOSCCTRL, 240
 SYSMEMREMAP
 SYSCON_Type, 239
 SYSOSCCTRL
 SYSCON_Type, 239
 SYSPLLCLKSEL
 SYSCON_Type, 239
 SYSPLLCLKUEN
 SYSCON_Type, 239
 SYSPLLCTRL
 SYSCON_Type, 239
 SYSPLLSTAT
 SYSCON_Type, 239
 SYSRSTSTAT
 SYSCON_Type, 239
 SYSTCKCAL
 SYSCON_Type, 239
 SysTick
 SYSTICK Peripheral Access Layer, 24
 SYSTICK Peripheral Access Layer, 23
 FREQ_SYSTICK, 24
 SysTick, 24
 systick.cpp, 402
 g_systick_freq, 404
 Iniciar_SysTick, 404
 MAX_TICKS, 404
 systick.h, 404, 406
 g_systick_freq, 406
 Iniciar_SysTick, 406
 SysTick_BASE
 NVIC Peripheral Access Layer, 27
 SysTick_t, 240
 CALIB, 240
 CTRL, 240
 CURR, 240
 RELOAD, 240
 Tabla_Digitos_BCD_7seg
 Digito.cpp, 279
 teclado, 241
 ~teclado, 242
 Get, 242
 Initialize, 242
 SWhandler, 242
 teclado, 242
 teclado.cpp, 265
 teclado.h, 265, 268
 NO_KEY, 267
 THR0_HIGH
 ADC_Type, 90
 THR0_LOW
 ADC_Type, 90
 THR1_HIGH
 ADC_Type, 90

THR1_LOW
 ADC_Type, 90
TICK_MICROSECONDS
 Callback.h, 345
TICK_MILISECONDS
 Callback.h, 345
TICK_SECONDS
 Callback.h, 345
TIMEOUT
 I2C_Type, 146
Timer, 243
 bases_t, 245
 GetmrStandBy, 245
 GetTimer, 245
 GetTmrEvent, 246
 GetTmrRun, 246
 m_TmrBase, 249
 m_TmrEvent, 249
 m_TmrHandler, 249
 m_TmrRun, 249
 m_TmrStandBy, 249
 operator bool, 246
 operator!, 246
 operator=, 246
 operator==, 246, 248
 SetmrStandBy, 247
 SetTimer, 247
 SetTimerBase, 247
 SetTmrHandler, 247
 StandByTimer, 247
 Timer, 245
 TimerStart, 248
 TmrEvent, 248
Timer.h, 346, 347
Timer_Handler
 Drivers, 17
timers, 249
 ~timers, 250
 operator<<, 250
 timers, 250
 TmrEvent, 250
Timers.cpp, 349
Timers.h, 350, 351
TimerStart
 Timer, 248
tipos.h, 432, 434
 __R, 433
 __RW, 433
 __W, 433
 int16_t, 433
 INT32_MAX, 433
 INT32_MIN, 433
 int32_t, 433
 INT8_MAX, 433
 INT8_MIN, 433
 int8_t, 433
 uint16_t, 434
 UINT32_MAX, 433
uint32_t, 434
 UINT8_MAX, 433
 uint8_t, 434
UINT32_MAX, 433
 uint32_t, 434
 UINT8_MAX, 433
 uint8_t, 434
TmrEvent
 Timer, 248
 timers, 250
Trigger
 ADC, 83
TRM
 ADC_Type, 90
TXCTL
 SPI_Type, 225
TXDAT
 SPI_Type, 225
 USART_Type, 252
TXDATCTL
 SPI_Type, 225
Uart, 250
UART.cpp, 374
 g_usart, 375
UART.h, 375, 377
UART_IRQHandler
 ComunicacionAsincronica, 96
uint16_t
 tipos.h, 434
UINT32_MAX
 tipos.h, 433
uint32_t
 tipos.h, 434
UINT8_MAX
 tipos.h, 433
uint8_t
 tipos.h, 434
UnSetInterrupt
 Callback, 93
UPDATE_TICKS
 Display7Segmentos.h, 284
USART Peripheral Access Layer, 28
 USART0, 29
 USART0_BASE, 29
 USART1, 29
 USART1_BASE, 29
 USART2, 29
 USART2_BASE, 30
 USART3, 30
 USART3_BASE, 30
 USART4, 30
 USART4_BASE, 30
 USART_BASE_ADDRS, 30
 USART_BASE_PTRS, 30
 USART IRQS, 30
USART0
 USART Peripheral Access Layer, 29
USART0_BASE
 USART Peripheral Access Layer, 29
USART1
 USART Peripheral Access Layer, 29
USART1_BASE

USART Peripheral Access Layer, [29](#)
USART2
 USART Peripheral Access Layer, [29](#)
USART2_BASE
 USART Peripheral Access Layer, [30](#)
USART3
 USART Peripheral Access Layer, [30](#)
USART3_BASE
 USART Peripheral Access Layer, [30](#)
USART4
 USART Peripheral Access Layer, [30](#)
USART4_BASE
 USART Peripheral Access Layer, [30](#)
USART_BASE_ADDRS
 USART Peripheral Access Layer, [30](#)
USART_BASE_PTRS
 USART Peripheral Access Layer, [30](#)
USART IRQS
 USART Peripheral Access Layer, [30](#)
USART_Type, [250](#)
 ADDR, [251](#)
 BRG, [251](#)
 CFG, [251](#)
 CTL, [251](#)
 INTENCLR, [251](#)
 INTENSET, [251](#)
 INTSTAT, [251](#)
 OSR, [251](#)
 RXDAT, [252](#)
 RXDATSTAT, [252](#)
 STAT, [252](#)
 TXDAT, [252](#)

VALUE
 DAC_t, [107](#)

W
 GPIO_Type, [130](#)
WDTOSCCTRL
 SYSCON_Type, [240](#)

Write
 ComunicacionAsincronica, [96](#)
 ComunicacionSincronica, [98](#)
 Display, [111](#)
 display7Segmentos, [114](#)
 ESP8266, [120](#)
 I2C, [141](#)
 I2CMaster, [150](#)
 LCD, [172](#)
 SPI, [223](#)
 SPIMaster, [230](#)

WriteAt
 LCD, [172](#), [173](#)