

Battery doc

Generated by Doxygen 1.9.3



---

<b>1 Control de Versiones</b>	<b>1</b>
1.1 Nomenclatura . . . . .	1
1.2 Tabla de Versiones . . . . .	1
1.2.1 B1 . . . . .	1
1.3 Descripcion de Versiones B1 . . . . .	2
1.3.1 v15 Interaccion con la introduccion de la bateria . . . . .	2
1.3.2 v14 Alerta de Error . . . . .	2
1.3.3 V13 Aviso Bateria Baja . . . . .	2
1.3.4 V12 Temporizador de No Uso . . . . .	3
1.3.5 V11 Estado de conexion USB . . . . .	3
1.3.6 V10 Modo Sleep y bajo consumo . . . . .	3
1.3.7 V9 Profiling . . . . .	3
1.3.8 V8 Power Bar y Brillo Tenue . . . . .	4
1.3.9 V7 Lectura Capacidad & Fix Error . . . . .	4
1.3.9.1 Fix Error . . . . .	4
1.3.10 V6 Diagnostico Basico . . . . .	5
1.3.11 V5 (Buzzer Sounds) . . . . .	5
1.3.12 V4 (Funcionalidades Basicas Botoneras y Display) . . . . .	5
1.3.12.1 V4.1 <em>Display</em> . . . . .	6
1.3.13 V3 (Proteccion UnderVoltage y Proteccion OverPower.) . . . . .	7
1.3.14 V2 (Proteccion de sobreconsumo y Tiempo de Arranque) . . . . .	7
1.3.15 V1 . . . . .	8
<b>2 Directional PAD</b>	<b>9</b>
2.1 Eventos posibles . . . . .	9
2.2 Funcionalidades implementadas . . . . .	9
2.3 Condiciones . . . . .	9
2.4 Conflictos . . . . .	9
2.5 Tests . . . . .	10
2.5.1 Test 0 . . . . .	10
<b>3 DISPLAY LED</b>	<b>11</b>
3.1 Funciones Privadas . . . . .	11
3.2 Funciones Publicas . . . . .	11
3.3 TESTs . . . . .	14
3.3.1 TEST 0 . . . . .	14
3.3.2 TEST 1 . . . . .	14
3.3.3 Test 2 . . . . .	14
3.3.4 Test 3 . . . . .	14
3.3.5 Test 4 . . . . .	14
3.3.6 Test 5 . . . . .	14
3.3.7 Test 6 . . . . .	14
3.3.8 Test 7 . . . . .	14

---

3.3.9 Test 8 . . . . .	14
<b>4 BUZZER</b>	<b>15</b>
4.1 Funciones . . . . .	15
4.2 TEST . . . . .	15
<b>5 DCDC</b>	<b>17</b>
5.1 Contantes . . . . .	17
5.2 Metodos . . . . .	17
5.3 TESTs . . . . .	17
5.3.1 Test 0 . . . . .	17
<b>6 Heath Monitor</b>	<b>19</b>
6.1 Metodos . . . . .	19
6.2 Test . . . . .	19
6.2.1 Test 0 . . . . .	19
<b>7 Power Bar</b>	<b>21</b>
7.1 Funciones . . . . .	21
7.2 TESTs . . . . .	21
<b>8 Arduino LowPower</b>	<b>23</b>
8.1 Funciones . . . . .	23
8.2 TESTs . . . . .	23
<b>9 Logging Tech Note</b>	<b>25</b>
9.1 Arquitectura de la memoria del SAMD21G18 . . . . .	25
9.1.1 Memoria Externa I2C . . . . .	25
9.2 Funciones Privadas . . . . .	25
9.3 * Guardado en EEPROM (X,B,C). X no se guarda, B se guarda en la Bateria y C se guarda en el Chasis. . . . .	26
9.4 * @param address Direccion en la memoria RAM de la variable. . . . .	26
9.5 * @return: valor del elemento de diagnostico. . . . .	26
9.6 * @param: Incremento, address. . . . .	26
9.7 * @param: type_print. . . . .	26
9.8 * @brief: Impresion por el puerto serie de los datos estaticos. . . . .	26
9.9 * False: La memoria esta vacia. . . . .	26
9.10 * @param: wats . . . . .	27
9.11 * @brief: Actualizacion de la memoria EEPROM de la Bateria. Antes de relajar el guardado se comprueba que el valor a guardar sea distinto del valor ya almacenado. . . . .	27
9.12 * @brief: Muestreo por el puerto serie de los valores acumulados de las estadisticas de potencia y voltage.Ademas de algunos datos estaticos guardados en eeprom como el numero de serio o el modelo. . . . .	27
9.13 TESTs . . . . .	27
9.13.1 TEST_0 . . . . .	27
9.13.2 TEST_1 . . . . .	27

---

9.13.3 TEST_2 . . . . .	27
9.13.4 TEST_3 . . . . .	27
<b>10 FlashStorage library for Arduino</b>	<b>29</b>
10.1 Supported hardware . . . . .	29
10.2 Limited number of writes . . . . .	29
10.3 Usage . . . . .	29
10.3.1 Using the alternative EEPROM-like API . . . . .	29
10.4 License . . . . .	30
10.5 FAQ . . . . .	30
10.5.1 Can I use a single FlashStorage object to store more stuff? . . . . .	30
10.5.2 The content of the FlashStorage is erased each time a new sketch is uploaded? . . . . .	30
10.5.3 Do you recommend to use FLASH instead of EEPROM? . . . . .	30
<b>11 Hierarchical Index</b>	<b>31</b>
11.1 Class Hierarchy . . . . .	31
<b>12 Class Index</b>	<b>33</b>
12.1 Class List . . . . .	33
<b>13 File Index</b>	<b>35</b>
13.1 File List . . . . .	35
<b>14 Class Documentation</b>	<b>39</b>
14.1 Adafruit_GFX Class Reference . . . . .	39
14.1.1 Detailed Description . . . . .	43
14.1.2 Constructor & Destructor Documentation . . . . .	43
14.1.2.1 Adafruit_GFX() . . . . .	43
14.1.3 Member Function Documentation . . . . .	43
14.1.3.1 charBounds() . . . . .	43
14.1.3.2 cp437() . . . . .	44
14.1.3.3 drawBitmap() [1/4] . . . . .	44
14.1.3.4 drawBitmap() [2/4] . . . . .	44
14.1.3.5 drawBitmap() [3/4] . . . . .	45
14.1.3.6 drawBitmap() [4/4] . . . . .	45
14.1.3.7 drawChar() [1/2] . . . . .	46
14.1.3.8 drawChar() [2/2] . . . . .	46
14.1.3.9 drawCircle() . . . . .	47
14.1.3.10 drawCircleHelper() . . . . .	47
14.1.3.11 drawFastHLine() . . . . .	47
14.1.3.12 drawFastVLine() . . . . .	48
14.1.3.13 drawGrayscaleBitmap() [1/4] . . . . .	48
14.1.3.14 drawGrayscaleBitmap() [2/4] . . . . .	48
14.1.3.15 drawGrayscaleBitmap() [3/4] . . . . .	49

14.1.3.16 drawGrayscaleBitmap()	[4/4]	49
14.1.3.17 drawLine()		50
14.1.3.18 drawPixel()		50
14.1.3.19 drawRect()		50
14.1.3.20 drawRGBBitmap()	[1/4]	51
14.1.3.21 drawRGBBitmap()	[2/4]	51
14.1.3.22 drawRGBBitmap()	[3/4]	52
14.1.3.23 drawRGBBitmap()	[4/4]	52
14.1.3.24 drawRoundRect()		52
14.1.3.25 drawTriangle()		53
14.1.3.26 drawXBitmap()		53
14.1.3.27 endWrite()		54
14.1.3.28 fillCircle()		54
14.1.3.29 fillCircleHelper()		54
14.1.3.30 fillRect()		55
14.1.3.31 fillRoundRect()		55
14.1.3.32 fillScreen()		55
14.1.3.33 fillTriangle()		56
14.1.3.34 getCursorX()		56
14.1.3.35 getCursorY()		56
14.1.3.36 getRotation()		56
14.1.3.37 getTextBounds()	[1/3]	57
14.1.3.38 getTextBounds()	[2/3]	57
14.1.3.39 getTextBounds()	[3/3]	58
14.1.3.40 height()		58
14.1.3.41 invertDisplay()		58
14.1.3.42 setCursor()		58
14.1.3.43 setFont()		59
14.1.3.44 setRotation()		59
14.1.3.45 setTextColor()	[1/2]	59
14.1.3.46 setTextColor()	[2/2]	59
14.1.3.47 setTextSize()	[1/2]	60
14.1.3.48 setTextSize()	[2/2]	60
14.1.3.49 setTextWrap()		60
14.1.3.50 startWrite()		61
14.1.3.51 width()		61
14.1.3.52 write()		61
14.1.3.53 writeFastHLine()		61
14.1.3.54 writeFastVLine()		61
14.1.3.55 writeFillRect()		62
14.1.3.56 writeLine()		62
14.1.3.57 writePixel()		63

---

14.1.4 Member Data Documentation . . . . .	63
14.1.4.1 _cp437 . . . . .	63
14.1.4.2 _height . . . . .	63
14.1.4.3 _width . . . . .	63
14.1.4.4 cursor_x . . . . .	63
14.1.4.5 cursor_y . . . . .	63
14.1.4.6 gfxFont . . . . .	63
14.1.4.7 HEIGHT . . . . .	64
14.1.4.8 rotation . . . . .	64
14.1.4.9 textbgcolor . . . . .	64
14.1.4.10 textcolor . . . . .	64
14.1.4.11 textSize_x . . . . .	64
14.1.4.12 textSize_y . . . . .	64
14.1.4.13 WIDTH . . . . .	64
14.1.4.14 wrap . . . . .	64
14.2 Adafruit_GFX_Button Class Reference . . . . .	65
14.2.1 Detailed Description . . . . .	65
14.2.2 Constructor & Destructor Documentation . . . . .	65
14.2.2.1 Adafruit_GFX_Button() . . . . .	65
14.2.3 Member Function Documentation . . . . .	66
14.2.3.1 contains() . . . . .	66
14.2.3.2 drawButton() . . . . .	66
14.2.3.3 initButton() [1/2] . . . . .	66
14.2.3.4 initButton() [2/2] . . . . .	67
14.2.3.5 initButtonUL() [1/2] . . . . .	67
14.2.3.6 initButtonUL() [2/2] . . . . .	68
14.2.3.7 isPressed() . . . . .	69
14.2.3.8 justPressed() . . . . .	69
14.2.3.9 justReleased() . . . . .	69
14.2.3.10 press() . . . . .	69
14.3 Adafruit_IS31FL3731 Class Reference . . . . .	69
14.3.1 Detailed Description . . . . .	70
14.3.2 Constructor & Destructor Documentation . . . . .	71
14.3.2.1 Adafruit_IS31FL3731() . . . . .	71
14.3.3 Member Function Documentation . . . . .	71
14.3.3.1 audioSync() . . . . .	71
14.3.3.2 begin() . . . . .	71
14.3.3.3 clear() . . . . .	71
14.3.3.4 displayFrame() . . . . .	72
14.3.3.5 drawPixel() . . . . .	72
14.3.3.6 readRegister8() . . . . .	72
14.3.3.7 selectBank() . . . . .	72

---

14.3.3.8 setFrame()	73
14.3.3.9 setLEDPWM()	73
14.3.3.10 writeRegister8()	73
14.3.4 Member Data Documentation	73
14.3.4.1 _frame	74
14.3.4.2 _i2caddr	74
14.4 Adafruit_IS31FL3731_Wing Class Reference	74
14.4.1 Detailed Description	74
14.4.2 Constructor & Destructor Documentation	74
14.4.2.1 Adafruit_IS31FL3731_Wing()	75
14.4.3 Member Function Documentation	75
14.4.3.1 drawPixel()	75
14.5 ArduinoLowPowerClass Class Reference	75
14.5.1 Detailed Description	75
14.5.2 Member Function Documentation	76
14.5.2.1 attachInterruptWakeup()	76
14.5.2.2 deatchRTCInterrupt()	76
14.5.2.3 deepSleep() [1/3]	76
14.5.2.4 deepSleep() [2/3]	76
14.5.2.5 deepSleep() [3/3]	76
14.5.2.6 idle() [1/3]	76
14.5.2.7 idle() [2/3]	76
14.5.2.8 idle() [3/3]	76
14.5.2.9 sleep() [1/3]	77
14.5.2.10 sleep() [2/3]	77
14.5.2.11 sleep() [3/3]	77
14.6 dc当地控件 Class Reference	77
14.6.1 Detailed Description	77
14.6.2 Constructor & Destructor Documentation	77
14.6.2.1 dc当地控件()	77
14.6.3 Member Function Documentation	78
14.6.3.1 EnableDCDC()	78
14.6.3.2 SetVoltage()	78
14.6.4 Member Data Documentation	78
14.6.4.1 encoderPos	78
14.6.4.2 pin_enable	78
14.6.4.3 TPIValue	78
14.7 EEPROM_EMULATION Struct Reference	79
14.7.1 Detailed Description	79
14.7.2 Member Data Documentation	79
14.7.2.1 data [1/2]	79
14.7.2.2 data [2/2]	79

---

14.7.2.3 valid . . . . .	79
14.8 EEPROMClass Class Reference . . . . .	79
14.8.1 Detailed Description . . . . .	80
14.8.2 Constructor & Destructor Documentation . . . . .	80
14.8.2.1 EEPROMClass() . . . . .	80
14.8.3 Member Function Documentation . . . . .	80
14.8.3.1 commit() . . . . .	80
14.8.3.2 isValid() . . . . .	80
14.8.3.3 length() . . . . .	80
14.8.3.4 read() . . . . .	80
14.8.3.5 update() . . . . .	81
14.8.3.6 write() . . . . .	81
14.9 Element Struct Reference . . . . .	81
14.9.1 Detailed Description . . . . .	81
14.9.2 Member Data Documentation . . . . .	81
14.9.2.1 category . . . . .	81
14.9.2.2 name . . . . .	82
14.9.2.3 value . . . . .	82
14.10 FlashClass Class Reference . . . . .	82
14.10.1 Detailed Description . . . . .	82
14.10.2 Constructor & Destructor Documentation . . . . .	82
14.10.2.1 FlashClass() . . . . .	82
14.10.3 Member Function Documentation . . . . .	82
14.10.3.1 erase() [1/2] . . . . .	82
14.10.3.2 erase() [2/2] . . . . .	83
14.10.3.3 read() [1/2] . . . . .	83
14.10.3.4 read() [2/2] . . . . .	83
14.10.3.5 write() [1/2] . . . . .	83
14.10.3.6 write() [2/2] . . . . .	83
14.11 FlashStorageClass< T > Class Template Reference . . . . .	83
14.11.1 Detailed Description . . . . .	84
14.11.2 Constructor & Destructor Documentation . . . . .	84
14.11.2.1 FlashStorageClass() . . . . .	84
14.11.3 Member Function Documentation . . . . .	84
14.11.3.1 read() [1/2] . . . . .	84
14.11.3.2 read() [2/2] . . . . .	84
14.11.3.3 write() . . . . .	84
14.12 GFXcanvas1 Class Reference . . . . .	84
14.12.1 Detailed Description . . . . .	85
14.12.2 Constructor & Destructor Documentation . . . . .	85
14.12.2.1 GFXcanvas1() . . . . .	85
14.12.2.2 ~GFXcanvas1() . . . . .	86

14.12.3 Member Function Documentation	86
14.12.3.1 drawFastHLine()	86
14.12.3.2 drawFastRawHLine()	86
14.12.3.3 drawFastRawVLine()	86
14.12.3.4 drawFastVLine()	87
14.12.3.5 drawPixel()	87
14.12.3.6 fillScreen()	87
14.12.3.7 getBuffer()	88
14.12.3.8 getPixel()	88
14.12.3.9 getRawPixel()	88
14.13 GFXcanvas16 Class Reference	89
14.13.1 Detailed Description	89
14.13.2 Constructor & Destructor Documentation	90
14.13.2.1 GFXcanvas16()	90
14.13.2.2 ~GFXcanvas16()	90
14.13.3 Member Function Documentation	90
14.13.3.1 byteSwap()	90
14.13.3.2 drawFastHLine()	90
14.13.3.3 drawFastRawHLine()	91
14.13.3.4 drawFastRawVLine()	91
14.13.3.5 drawFastVLine()	91
14.13.3.6 drawPixel()	92
14.13.3.7 fillScreen()	92
14.13.3.8 getBuffer()	92
14.13.3.9 getPixel()	92
14.13.3.10 getRawPixel()	93
14.14 GFXcanvas8 Class Reference	93
14.14.1 Detailed Description	94
14.14.2 Constructor & Destructor Documentation	94
14.14.2.1 GFXcanvas8()	94
14.14.2.2 ~GFXcanvas8()	94
14.14.3 Member Function Documentation	94
14.14.3.1 drawFastHLine()	95
14.14.3.2 drawFastRawHLine()	95
14.14.3.3 drawFastRawVLine()	95
14.14.3.4 drawFastVLine()	96
14.14.3.5 drawPixel()	96
14.14.3.6 fillScreen()	96
14.14.3.7 getBuffer()	96
14.14.3.8 getPixel()	97
14.14.3.9 getRawPixel()	97
14.15 GFXfont Struct Reference	97

---

14.15.1 Detailed Description . . . . .	98
14.15.2 Member Data Documentation . . . . .	98
14.15.2.1 bitmap . . . . .	98
14.15.2.2 first . . . . .	98
14.15.2.3 glyph . . . . .	98
14.15.2.4 last . . . . .	98
14.15.2.5 yAdvance . . . . .	98
14.16 GFXglyph Struct Reference . . . . .	99
14.16.1 Detailed Description . . . . .	99
14.16.2 Member Data Documentation . . . . .	99
14.16.2.1 bitmapOffset . . . . .	99
14.16.2.2 height . . . . .	99
14.16.2.3 width . . . . .	99
14.16.2.4 xAdvance . . . . .	99
14.16.2.5 xOffset . . . . .	100
14.16.2.6 yOffset . . . . .	100
14.17 HealthMonitor Class Reference . . . . .	100
14.17.1 Detailed Description . . . . .	100
14.17.2 Constructor & Destructor Documentation . . . . .	100
14.17.2.1 HealthMonitor() . . . . .	101
14.17.3 Member Function Documentation . . . . .	101
14.17.3.1 check() . . . . .	101
14.17.3.2 getCounter() . . . . .	101
14.17.3.3 getSample() . . . . .	101
14.17.3.4 setCounter() . . . . .	102
14.17.4 Member Data Documentation . . . . .	102
14.17.4.1 alarm . . . . .	102
14.17.4.2 limit . . . . .	102
14.17.4.3 rampDOWNdec . . . . .	102
14.17.4.4 rampUPinc . . . . .	102
14.17.4.5 sense_values . . . . .	102
14.17.4.6 threshold . . . . .	102
14.18 MilliTimer Class Reference . . . . .	103
14.18.1 Detailed Description . . . . .	103
14.18.2 Constructor & Destructor Documentation . . . . .	103
14.18.2.1 MilliTimer() . . . . .	103
14.18.3 Member Function Documentation . . . . .	103
14.18.3.1 idle() . . . . .	103
14.18.3.2 poll() . . . . .	103
14.18.3.3 remaining() . . . . .	103
14.18.3.4 set() . . . . .	104
14.19 Person Struct Reference . . . . .	104

---

14.19.1 Detailed Description . . . . .	104
14.19.2 Member Data Documentation . . . . .	104
14.19.2.1 name . . . . .	104
14.19.2.2 surname . . . . .	104
14.19.2.3 valid . . . . .	104
14.20 RTCZero Class Reference . . . . .	104
14.20.1 Detailed Description . . . . .	105
14.20.2 Member Enumeration Documentation . . . . .	106
14.20.2.1 Alarm_Match . . . . .	106
14.20.3 Constructor & Destructor Documentation . . . . .	106
14.20.3.1 RTCZero() . . . . .	106
14.20.4 Member Function Documentation . . . . .	106
14.20.4.1 attachInterrupt() . . . . .	106
14.20.4.2 begin() . . . . .	106
14.20.4.3 detachInterrupt() . . . . .	106
14.20.4.4 disableAlarm() . . . . .	106
14.20.4.5 enableAlarm() . . . . .	107
14.20.4.6 getAlarmDay() . . . . .	107
14.20.4.7 getAlarmHours() . . . . .	107
14.20.4.8 getAlarmMinutes() . . . . .	107
14.20.4.9 getAlarmMonth() . . . . .	107
14.20.4.10 getAlarmSeconds() . . . . .	107
14.20.4.11 getAlarmYear() . . . . .	107
14.20.4.12 getDay() . . . . .	107
14.20.4.13 getEpoch() . . . . .	107
14.20.4.14 getHours() . . . . .	107
14.20.4.15 getMinutes() . . . . .	108
14.20.4.16 getMonth() . . . . .	108
14.20.4.17 getSeconds() . . . . .	108
14.20.4.18 getY2kEpoch() . . . . .	108
14.20.4.19 getYear() . . . . .	108
14.20.4.20 isConfigured() . . . . .	108
14.20.4.21 setAlarmDate() . . . . .	108
14.20.4.22 setAlarmDay() . . . . .	108
14.20.4.23 setAlarmEpoch() . . . . .	108
14.20.4.24 setAlarmHours() . . . . .	109
14.20.4.25 setAlarmMinutes() . . . . .	109
14.20.4.26 setAlarmMonth() . . . . .	109
14.20.4.27 setAlarmSeconds() . . . . .	109
14.20.4.28 setAlarmTime() . . . . .	109
14.20.4.29 setAlarmYear() . . . . .	109
14.20.4.30 setDate() . . . . .	109

---

14.20.4.31 setDay() . . . . .	109
14.20.4.32 setEpoch() . . . . .	110
14.20.4.33 setHours() . . . . .	110
14.20.4.34 setMinutes() . . . . .	110
14.20.4.35 setMonth() . . . . .	110
14.20.4.36 setSeconds() . . . . .	110
14.20.4.37 setTime() . . . . .	110
14.20.4.38 setY2kEpoch() . . . . .	110
14.20.4.39 setYear() . . . . .	110
14.20.4.40 standbyMode() . . . . .	111
<b>14.21 SlowSoftI2CMaster Class Reference . . . . .</b>	<b>111</b>
<b>14.21.1 Detailed Description . . . . .</b>	<b>111</b>
<b>14.21.2 Constructor &amp; Destructor Documentation . . . . .</b>	<b>111</b>
14.21.2.1 SlowSoftI2CMaster() [1/2] . . . . .	111
14.21.2.2 SlowSoftI2CMaster() [2/2] . . . . .	111
<b>14.21.3 Member Function Documentation . . . . .</b>	<b>111</b>
14.21.3.1 i2c_init() . . . . .	112
14.21.3.2 i2c_read() . . . . .	112
14.21.3.3 i2c_rep_start() . . . . .	112
14.21.3.4 i2c_start() . . . . .	112
14.21.3.5 i2c_start_wait() . . . . .	112
14.21.3.6 i2c_stop() . . . . .	112
14.21.3.7 i2c_write() . . . . .	112
<b>14.21.4 Member Data Documentation . . . . .</b>	<b>112</b>
14.21.4.1 error . . . . .	112
<b>15 File Documentation . . . . .</b>	<b>115</b>
<b>15.1 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/CODE/B1/B1.ino File Reference . . . . .</b>	<b>115</b>
<b>15.1.1 Detailed Description . . . . .</b>	<b>119</b>
<b>15.1.2 Function Documentation . . . . .</b>	<b>121</b>
15.1.2.1 irq_DeathBattery_Handler() . . . . .	121
15.1.2.2 irq_USB_Handler() . . . . .	121
15.1.2.3 irqCenterButtonHandler() . . . . .	121
15.1.2.4 loop() . . . . .	121
15.1.2.5 setup() . . . . .	121
15.1.2.6 voltage_input_protection() . . . . .	121
<b>15.1.3 Variable Documentation . . . . .</b>	<b>122</b>
15.1.3.1 afk_counter . . . . .	122
15.1.3.2 afk_timer . . . . .	122
15.1.3.3 boost_check . . . . .	122
15.1.3.4 button_event . . . . .	122
15.1.3.5 buzzer_error_state . . . . .	122

---

15.1.3.6 <a href="#">buzzer_state</a>	122
15.1.3.7 <a href="#">C_CMD_DIAGNOSTIC_STOP</a>	122
15.1.3.8 <a href="#">C_CMD_DISPLAY_ENDING_OFF</a>	123
15.1.3.9 <a href="#">C_CMD_DISPLAY_ENDING_ON</a>	123
15.1.3.10 <a href="#">C_CMD_DISPLAY_STARTING_OFF</a>	123
15.1.3.11 <a href="#">C_CMD_DISPLAY_STARTING_ON</a>	123
15.1.3.12 <a href="#">C_CMD_LIVE_OFF</a>	123
15.1.3.13 <a href="#">C_CMD_LIVE_ON</a>	123
15.1.3.14 <a href="#">C_CMD_SOUND_ENDING_OFF</a>	123
15.1.3.15 <a href="#">C_CMD_SOUND_ENDING_ON</a>	123
15.1.3.16 <a href="#">C_CMD_SOUND_FULL_CHARGE_OFF</a>	123
15.1.3.17 <a href="#">C_CMD_SOUND_FULL_CHARGE_ON</a>	123
15.1.3.18 <a href="#">C_CMD_SOUND_STARTING_OFF</a>	124
15.1.3.19 <a href="#">C_CMD_SOUND_STARTING_ON</a>	124
15.1.3.20 <a href="#">C_CMS_SOUND_CHARGE_START_OFF</a>	124
15.1.3.21 <a href="#">C_CMS_SOUND_CHARGE_START_ON</a>	124
15.1.3.22 <a href="#">C_CMS_SOUND_DEATH_BATTERY_OFF</a>	124
15.1.3.23 <a href="#">C_CMS_SOUND_DEATH_BATTERY_ON</a>	124
15.1.3.24 <a href="#">C_DIAGNOSTIC_PASSWORD</a>	124
15.1.3.25 <a href="#">C_IDLE_TIMER_COUNT</a>	124
15.1.3.26 <a href="#">C_LIMIT_COMSUPTION_PROT</a>	124
15.1.3.27 <a href="#">C_LIMIT_OVERPOWER_PROT</a>	124
15.1.3.28 <a href="#">C_LIMIT_UNDERVOLTAGE_PROT</a>	125
15.1.3.29 <a href="#">C_LIMIT_VOLTAGE_INPUT</a>	125
15.1.3.30 <a href="#">C_LIVE_CMD_OFF_LIVE_VIEW</a>	125
15.1.3.31 <a href="#">C_LIVE_CMD_OUT_OFF</a>	125
15.1.3.32 <a href="#">C_LIVE_CMD_OUT_ON</a>	125
15.1.3.33 <a href="#">C_LIVE_CMD_VOLT_DOWN_1</a>	125
15.1.3.34 <a href="#">C_LIVE_CMD_VOLT_DOWN_10</a>	125
15.1.3.35 <a href="#">C_LIVE_CMD_VOLT_UP_1</a>	125
15.1.3.36 <a href="#">C_LIVE_CMD_VOLT_UP_10</a>	125
15.1.3.37 <a href="#">C_LOW_BATTERY_LEVEL</a>	125
15.1.3.38 <a href="#">C_MASK_ACTIVATE</a>	126
15.1.3.39 <a href="#">C_MASK_DEACTIVATE</a>	126
15.1.3.40 <a href="#">C_MAX_VOLT</a>	126
15.1.3.41 <a href="#">C_MIN_VOLT</a>	126
15.1.3.42 <a href="#">C_NUMB_PRINTS_STATIC_DATA</a>	126
15.1.3.43 <a href="#">C_OFF</a>	126
15.1.3.44 <a href="#">C_ON</a>	126
15.1.3.45 <a href="#">C_PIN_DEATH_BATTERY_CHECK</a>	126
15.1.3.46 <a href="#">C_PIN_EN_DCDC</a>	126
15.1.3.47 <a href="#">C_PIN_FLAG_BATTLOW</a>	126

---

15.1.3.48 C_PIN_FLAG_CHARG . . . . .	127
15.1.3.49 C_PIN_I_OUT . . . . .	127
15.1.3.50 C_PIN_SCL_2 . . . . .	127
15.1.3.51 C_PIN_SDA_2 . . . . .	127
15.1.3.52 C_PIN_SHIPPING_MOD . . . . .	127
15.1.3.53 C_PIN_TEST_MODE . . . . .	127
15.1.3.54 C_PIN_USB_CONNEXION . . . . .	127
15.1.3.55 C_PIN_USB_SEL . . . . .	127
15.1.3.56 C_PIN_V_IN . . . . .	127
15.1.3.57 C_PIN_V_OUT . . . . .	127
15.1.3.58 C_RETRY_750_COUNT . . . . .	128
15.1.3.59 C_SW_ST_CAPACITY . . . . .	128
15.1.3.60 C_SW_ST_DIAGNOSTIC . . . . .	128
15.1.3.61 C_SW_ST_ERROR . . . . .	128
15.1.3.62 C_SW_ST_RUN . . . . .	128
15.1.3.63 C_SW_ST_SLEEP . . . . .	128
15.1.3.64 C_SW_ST_START_UP . . . . .	128
15.1.3.65 C_SW_ST_STOP . . . . .	128
15.1.3.66 C_SW_ST_USB . . . . .	128
15.1.3.67 C_TIME_TO_LIGHT_DOWN . . . . .	128
15.1.3.68 C_TIMER_ARMED . . . . .	129
15.1.3.69 C_TIMER_DONE . . . . .	129
15.1.3.70 C_TIMER_IDLE . . . . .	129
15.1.3.71 C_USB_CONNECTED . . . . .	129
15.1.3.72 C_USB_DISCONNECTED . . . . .	129
15.1.3.73 C_WATCH_DOG_TIME_MS . . . . .	129
15.1.3.74 capacity . . . . .	129
15.1.3.75 capcaity_ask_off . . . . .	129
15.1.3.76 change_text_answered . . . . .	129
15.1.3.77 click_events . . . . .	129
15.1.3.78 cmd_go_sleep . . . . .	130
15.1.3.79 consmptn_event_protection_counter . . . . .	130
15.1.3.80 cont_idle_timer . . . . .	130
15.1.3.81 cont_sec . . . . .	130
15.1.3.82 counter_prints_static_data . . . . .	130
15.1.3.83 data . . . . .	130
15.1.3.84 DCDC . . . . .	130
15.1.3.85 delay_live_view . . . . .	130
15.1.3.86 diag_check . . . . .	130
15.1.3.87 diagnostic_msg . . . . .	130
15.1.3.88 display_error_status . . . . .	131
15.1.3.89 display_status . . . . .	131

15.1.3.90 enable_charge_sound . . . . .	131
15.1.3.91 enable_death_battery_sound . . . . .	131
15.1.3.92 enable_ending_sound . . . . .	131
15.1.3.93 enable_ending_text . . . . .	131
15.1.3.94 enable_full_charge_sound . . . . .	131
15.1.3.95 enable_live_view . . . . .	131
15.1.3.96 enable_starting_sound . . . . .	131
15.1.3.97 enable_starting_text . . . . .	131
15.1.3.98 error_msg . . . . .	132
15.1.3.99 flag_arranque . . . . .	132
15.1.3.100 flag_cap_one_shot . . . . .	132
15.1.3.101 flag_diag_header_printed . . . . .	132
15.1.3.102 flag_display_capacity . . . . .	132
15.1.3.103 flag_error . . . . .	132
15.1.3.104 flag_initialize . . . . .	132
15.1.3.105 flag_irq_center_button . . . . .	132
15.1.3.106 flag_irq_death_battery . . . . .	132
15.1.3.107 flag_irq_singleshot . . . . .	132
15.1.3.108 flag_low_battery . . . . .	133
15.1.3.109 flag_msg_init . . . . .	133
15.1.3.110 flag_msg_sleep . . . . .	133
15.1.3.111 flag_return . . . . .	133
15.1.3.112 flag_sleep . . . . .	133
15.1.3.113 flag_sound . . . . .	133
15.1.3.114 flag_test . . . . .	133
15.1.3.115 flag_timer_low_bright . . . . .	133
15.1.3.116 flag_update_eeprom . . . . .	133
15.1.3.117 flag_usb_change . . . . .	133
15.1.3.118 flag_waiting . . . . .	134
15.1.3.119 flag_work . . . . .	134
15.1.3.120 go_sleep . . . . .	134
15.1.3.121 hw_output . . . . .	134
15.1.3.122 ledmatrix . . . . .	134
15.1.3.123 low_batt_display . . . . .	134
15.1.3.124 low_batt_sound . . . . .	134
15.1.3.125 mask_protection_state . . . . .	134
15.1.3.126 MAX_NUM_PROTECTION_EVENTS . . . . .	134
15.1.3.127 mode_bright_display . . . . .	134
15.1.3.128 msg_sleep . . . . .	135
15.1.3.129 new_text_received . . . . .	135
15.1.3.130 OP_event_protection_counter . . . . .	135
15.1.3.131 output_mode . . . . .	135

---

15.1.3.132 over_consumption_protection . . . . .	135
15.1.3.133 over_power_protection . . . . .	135
15.1.3.134 prev_state . . . . .	135
15.1.3.135 prev_volt . . . . .	136
15.1.3.136 print_diagnostic_static_data . . . . .	136
15.1.3.137 program_tick . . . . .	136
15.1.3.138 protection_event_delay . . . . .	136
15.1.3.139 protection_event_delay_flag . . . . .	136
15.1.3.140 reset_capacity_off_text . . . . .	136
15.1.3.141 reset_diag_text . . . . .	136
15.1.3.142 reset_error_text . . . . .	136
15.1.3.143 reset_init_text . . . . .	136
15.1.3.144 reset_sleep_text . . . . .	136
15.1.3.145 sample_IOut . . . . .	137
15.1.3.146 sample_POut . . . . .	137
15.1.3.147 sample_Vin . . . . .	137
15.1.3.148 sample_VOut . . . . .	137
15.1.3.149 sound . . . . .	137
15.1.3.150 stage_capacity_1 . . . . .	137
15.1.3.151 stage_capacity_2 . . . . .	137
15.1.3.152 start_string . . . . .	137
15.1.3.153 state_to_return . . . . .	137
15.1.3.154 sw_output . . . . .	137
15.1.3.155 sw_status . . . . .	138
15.1.3.156 t0 . . . . .	138
15.1.3.157 t1 . . . . .	138
15.1.3.158 test_enable . . . . .	138
15.1.3.159 theory_Vout . . . . .	138
15.1.3.160 timer_arranque . . . . .	138
15.1.3.161 timer_diagnostic_querist . . . . .	138
15.1.3.162 timer_display_capacity . . . . .	138
15.1.3.163 timer_display_error . . . . .	138
15.1.3.164 timer_idle . . . . .	138
15.1.3.165 timer_irq_button_center . . . . .	139
15.1.3.166 timer_low_bright . . . . .	139
15.1.3.167 timer_print_diagnostic . . . . .	139
15.1.3.168 timer_recover_voltage . . . . .	139
15.1.3.169 timer_sec_count . . . . .	139
15.1.3.170 timer_wait_sleep . . . . .	139
15.1.3.171 trigger_Display_volt . . . . .	139
15.1.3.172 under_voltage_protection . . . . .	139
15.1.3.173 usb_status . . . . .	140

---

15.1.3.174 user_output . . . . .	140
15.1.3.175 UV_event_protection_counter . . . . .	140
15.1.3.176 voltage_input_event_protection_counter . . . . .	140
15.2 B1.ino . . . . .	140
15.3 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/CODE/README.md File Reference . . . . .	161
15.4 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0001-BUTTONS/README.md File Reference . . . . .	161
15.5 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/README.md File Reference . . . . .	161
15.6 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0003-BUZZER/README.md File Reference . . . . .	161
15.7 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0004-DCDC/README.md File Reference . . . . .	161
15.8 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0005-SENSING/README.md File Reference . . . . .	161
15.9 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0006-POWERBAR/README.md File Reference . . . . .	161
15.10 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/README.md File Reference . . . . .	161
15.11 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/README.md File Reference . . . . .	161
15.12 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/readme.md File Reference . . . . .	161
15.13 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0000-EXAMPLE/librarie.h File Reference . . . . .	161
15.14 librarie.h . . . . .	161
15.15 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0001-BUTTONS/Dpad.h File Reference . . . . .	162
15.15.1 Function Documentation . . . . .	162
15.15.1.1 ReadDirPad() . . . . .	162
15.15.2 Variable Documentation . . . . .	162
15.15.2.1 button_not_pressed . . . . .	162
15.15.2.2 button_pressed . . . . .	163
15.15.2.3 C_CLICK_CENTER . . . . .	163
15.15.2.4 C_CLICK_DOWN . . . . .	163
15.15.2.5 C_CLICK_UP . . . . .	163
15.15.2.6 C_LP_CENTER . . . . .	163
15.15.2.7 C_LP_DOWN . . . . .	163
15.15.2.8 C_LP_UP . . . . .	163
15.15.2.9 C_NONE_EVENT . . . . .	163
15.15.2.10 C_PIN_BUTT_CENTER . . . . .	163
15.15.2.11 C_PIN_BUTT_DOWN . . . . .	163
15.15.2.12 C_PIN_BUTT_UP . . . . .	164
15.15.2.13 C_TIMER_DEBOUNCE . . . . .	164

---

15.15.2.14 C_TIMER_LONGPRESS . . . . .	164
15.15.2.15 C_TIMER_LONGPRESS_LOOP . . . . .	164
15.16 Dpad.h . . . . .	164
15.17 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0001-BUTTONS/examples/TEST_0/TEST_0.ino File Reference . . . . .	167
15.17.1 Detailed Description . . . . .	167
15.17.2 Function Documentation . . . . .	167
15.17.2.1 loop() . . . . .	167
15.17.2.2 setup() . . . . .	168
15.18 TEST_0.ino . . . . .	168
15.19 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_0/TEST_0.ino File Reference . . . . .	169
15.19.1 Detailed Description . . . . .	169
15.19.2 Function Documentation . . . . .	170
15.19.2.1 loop() . . . . .	170
15.19.2.2 setup() . . . . .	170
15.19.3 Variable Documentation . . . . .	170
15.19.3.1 ledmatrix . . . . .	170
15.20 TEST_0.ino . . . . .	170
15.21 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0003-BUZZER/examples/TEST_0/TEST_0.ino File Reference . . . . .	171
15.21.1 Detailed Description . . . . .	172
15.21.2 Function Documentation . . . . .	172
15.21.2.1 loop() . . . . .	172
15.21.2.2 setup() . . . . .	172
15.21.3 Variable Documentation . . . . .	172
15.21.3.1 i . . . . .	172
15.21.3.2 t0 . . . . .	172
15.21.3.3 t1 . . . . .	173
15.22 TEST_0.ino . . . . .	173
15.23 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0004-DCDC/exmples/TEST_0/TEST_0.ino File Reference . . . . .	173
15.23.1 Detailed Description . . . . .	174
15.23.2 Function Documentation . . . . .	174
15.23.2.1 loop() . . . . .	174
15.23.2.2 setup() . . . . .	174
15.23.3 Variable Documentation . . . . .	174
15.23.3.1 C_PIN_EN_DCDC . . . . .	174
15.23.3.2 C_PIN_I_IN . . . . .	174
15.23.3.3 C_PIN_V_OUT . . . . .	175
15.23.3.4 DCDC . . . . .	175
15.23.3.5 i_sense . . . . .	175
15.23.3.6 v_diff . . . . .	175

15.23.3.7 v_sense . . . . .	175
15.24 TEST_0.ino . . . . .	175
15.25 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0005-SENSING/examples/TEST_0/TEST_0.ino File Reference . . . . .	176
15.25.1 Detailed Description . . . . .	176
15.25.2 Macro Definition Documentation . . . . .	176
15.25.2.1 DAC_PIN . . . . .	177
15.25.3 Function Documentation . . . . .	177
15.25.3.1 loop() . . . . .	177
15.25.3.2 setup() . . . . .	177
15.25.4 Variable Documentation . . . . .	177
15.25.4.1 count . . . . .	177
15.25.4.2 dac_output . . . . .	177
15.25.4.3 sample . . . . .	177
15.25.4.4 sensor_1 . . . . .	177
15.25.4.5 trigger_timer . . . . .	177
15.26 TEST_0.ino . . . . .	178
15.27 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0006-POWERBAR/examples/TEST_0/TEST_0.ino File Reference . . . . .	178
15.27.1 Detailed Description . . . . .	179
15.27.2 Function Documentation . . . . .	179
15.27.2.1 loop() . . . . .	179
15.27.2.2 setup() . . . . .	179
15.27.3 Variable Documentation . . . . .	179
15.27.3.1 ledmatrix . . . . .	179
15.28 TEST_0.ino . . . . .	179
15.29 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/examples/TEST_0/TEST_0.ino File Reference . . . . .	180
15.29.1 Detailed Description . . . . .	180
15.29.2 Function Documentation . . . . .	180
15.29.2.1 loop() . . . . .	180
15.29.2.2 setup() . . . . .	181
15.30 TEST_0.ino . . . . .	181
15.31 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/bitmaps.h File Reference . . . . .	181
15.31.1 Detailed Description . . . . .	183
15.31.2 Macro Definition Documentation . . . . .	183
15.31.2.1 C_DISPLAY_HEIGHT . . . . .	183
15.31.2.2 C_DISPLAY_WIDTH . . . . .	183
15.31.2.3 DISPLAY_SIZE . . . . .	184
15.31.2.4 POWERBAR_LOCATION . . . . .	184
15.31.3 Function Documentation . . . . .	184
15.31.3.1 Char2Bitmap() . . . . .	184

---

15.31.4 Variable Documentation . . . . .	184
15.31.4.1 ACCENTchar . . . . .	184
15.31.4.2 Achar . . . . .	184
15.31.4.3 ANPERSANchar . . . . .	184
15.31.4.4 APOSTROPHEchar . . . . .	185
15.31.4.5 ASTERISKchar . . . . .	185
15.31.4.6 ATchar . . . . .	185
15.31.4.7 BACKSLASHchar . . . . .	185
15.31.4.8 Bchar . . . . .	185
15.31.4.9 Cchar . . . . .	186
15.31.4.10 CIRCUMFLEXchar . . . . .	186
15.31.4.11 CLOSECURLYBRAChar . . . . .	186
15.31.4.12 CLOSEPARENTESISchar . . . . .	186
15.31.4.13 CLOSESQUAREBRAQUETchar . . . . .	186
15.31.4.14 COLONchar . . . . .	187
15.31.4.15 COMMAchar . . . . .	187
15.31.4.16 Dchar . . . . .	187
15.31.4.17 DOLLARchar . . . . .	187
15.31.4.18 DOTchar . . . . .	187
15.31.4.19 Echar . . . . .	188
15.31.4.20 EIGHTchar . . . . .	188
15.31.4.21 EQUALchar . . . . .	188
15.31.4.22 EXCLMARKchar . . . . .	188
15.31.4.23 EYES2_1 . . . . .	188
15.31.4.24 EYES2_2 . . . . .	189
15.31.4.25 EYES_1 . . . . .	189
15.31.4.26 EYES_2 . . . . .	189
15.31.4.27 FACE_1 . . . . .	189
15.31.4.28 FACE_2 . . . . .	189
15.31.4.29 Fchar . . . . .	190
15.31.4.30 FIVEchar . . . . .	190
15.31.4.31 FOURchar . . . . .	190
15.31.4.32 Frame_Batt_1 . . . . .	190
15.31.4.33 Frame_Batt_2 . . . . .	190
15.31.4.34 Frame_Batt_Low_1 . . . . .	191
15.31.4.35 Frame_Batt_Low_2 . . . . .	191
15.31.4.36 Frame_Error_1 . . . . .	191
15.31.4.37 Frame_Error_2 . . . . .	191
15.31.4.38 Frame_USB_1 . . . . .	191
15.31.4.39 Frame_USB_2 . . . . .	192
15.31.4.40 Gchar . . . . .	192
15.31.4.41 GREATERTHANchar . . . . .	192

---

15.31.4.42 HASTAGchar . . . . .	192
15.31.4.43 Hchar . . . . .	192
15.31.4.44 HYPHENchar . . . . .	193
15.31.4.45 Ichar . . . . .	193
15.31.4.46 Jchar . . . . .	193
15.31.4.47 Kchar . . . . .	193
15.31.4.48 Lchar . . . . .	193
15.31.4.49 LESSTHANchar . . . . .	194
15.31.4.50 Mchar . . . . .	194
15.31.4.51 Nchar . . . . .	194
15.31.4.52 NINEchar . . . . .	194
15.31.4.53 Ochar . . . . .	194
15.31.4.54 ONEchar . . . . .	195
15.31.4.55 OPENCURLYBRAchar . . . . .	195
15.31.4.56 OPENPARENTESISchar . . . . .	195
15.31.4.57 Pchar . . . . .	195
15.31.4.58 PERCENTchar . . . . .	195
15.31.4.59 PLUSchar . . . . .	196
15.31.4.60 Qchar . . . . .	196
15.31.4.61 QUESTIONchar . . . . .	196
15.31.4.62 QUOTMARKchar . . . . .	196
15.31.4.63 Rchar . . . . .	196
15.31.4.64 Schar . . . . .	197
15.31.4.65 SEMICOLONchar . . . . .	197
15.31.4.66 SEVENchar . . . . .	197
15.31.4.67 SIXchar . . . . .	197
15.31.4.68 SLASHchar . . . . .	197
15.31.4.69 SPACEchar . . . . .	198
15.31.4.70 SQUAREBRAQUETchar . . . . .	198
15.31.4.71 SWUNGchar . . . . .	198
15.31.4.72 Tchar . . . . .	198
15.31.4.73 THREEmchar . . . . .	198
15.31.4.74 TWOchar . . . . .	199
15.31.4.75 Uchar . . . . .	199
15.31.4.76 UNDERSCOREchar . . . . .	199
15.31.4.77 Vchar . . . . .	199
15.31.4.78 VERTICALBARchar . . . . .	199
15.31.4.79 Wchar . . . . .	200
15.31.4.80 width_bitmap . . . . .	200
15.31.4.81 Xchar . . . . .	200
15.31.4.82 Ychar . . . . .	200
15.31.4.83 Zchar . . . . .	200

---

15.31.4.84 ZEROchar . . . . .	200
15.32 bitmaps.h . . . . .	201
15.33 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/display.h File Reference . . . . .	221
15.33.1 Detailed Description . . . . .	222
15.33.2 Function Documentation . . . . .	222
15.33.2.1 DisplayBattCharging() . . . . .	222
15.33.2.2 DisplayCap() . . . . .	223
15.33.2.3 DisplayDiagnosticMode() . . . . .	223
15.33.2.4 DisplayError() . . . . .	223
15.33.2.5 DisplayLowBattery() . . . . .	223
15.33.2.6 DisplayText() . . . . .	223
15.33.2.7 DisplayUsbIn() . . . . .	224
15.33.2.8 DisplayUsbOut() . . . . .	224
15.33.2.9 DisplayVolt() . . . . .	224
15.33.2.10 LedWork() . . . . .	225
15.33.2.11 ScreenON() . . . . .	225
15.33.2.12 SwitchScreenOff() . . . . .	225
15.33.3 Variable Documentation . . . . .	225
15.33.3.1 blink_OFF . . . . .	225
15.33.3.2 blink_ON . . . . .	225
15.33.3.3 C_DELAY_BLINK_LOW_BATTERY . . . . .	225
15.33.3.4 C_DISPLAY_OFF . . . . .	226
15.33.3.5 C_DISPLAY_ON . . . . .	226
15.33.3.6 C_DISPLAY_ST_BUSSY . . . . .	226
15.33.3.7 C_DISPLAY_ST_NOT_BUSSY . . . . .	226
15.33.3.8 C_HIGH_BRIGHTNESS . . . . .	226
15.33.3.9 C_LOW_BRIGHTNESS . . . . .	226
15.33.3.10 C_MODE_HIGH_BRIGHT . . . . .	226
15.33.3.11 C_MODE_LOW_BRIGHT . . . . .	226
15.33.3.12 C_NUMBER_BLINK_LOW_BATTERY . . . . .	226
15.33.3.13 C_PIXEL_START_STRING . . . . .	226
15.33.3.14 C_PIXEL_STOP_STRING . . . . .	227
15.33.3.15 C_PWBAR_Y_AXE . . . . .	227
15.33.3.16 C_TEXT_BITMAP_WEIGHT . . . . .	227
15.33.3.17 C_TEXT_CHAR_HEIGHT . . . . .	227
15.33.3.18 C_TEXT_CHAR_OFFSET_Y . . . . .	227
15.33.3.19 C_TIMER_BTW_FRAMES . . . . .	227
15.34 display.h . . . . .	227
15.35 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_1/TEST_1.ino File Reference . . . . .	233
15.35.1 Detailed Description . . . . .	233
15.35.2 Function Documentation . . . . .	234

---

15.35.2.1 loop()	234
15.35.2.2 setup()	234
15.35.3 Variable Documentation	234
15.35.3.1 ledmatrix	234
15.36 TEST_1.ino	234
15.37 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0004-DCDC/exmples/TEST_1/TEST_1.ino File Reference	235
15.37.1 Detailed Description	235
15.37.2 Function Documentation	235
15.37.2.1 loop()	235
15.37.2.2 setup()	236
15.37.3 Variable Documentation	236
15.37.3.1 C_PIN_EN_DCDC	236
15.37.3.2 DCDC	236
15.38 TEST_1.ino	236
15.39 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/examples/TEST_1/TEST_1.ino File Reference	236
15.39.1 Detailed Description	236
15.39.2 Function Documentation	237
15.39.2.1 loop()	237
15.39.2.2 setup()	237
15.40 TEST_1.ino	237
15.41 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_2/TEST_2.ino File Reference	238
15.41.1 Detailed Description	238
15.41.2 Function Documentation	238
15.41.2.1 loop()	238
15.41.2.2 setup()	238
15.41.3 Variable Documentation	238
15.41.3.1 ledmatrix	239
15.42 TEST_2.ino	239
15.43 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/examples/TEST_2/TEST_2.ino File Reference	239
15.43.1 Detailed Description	239
15.43.2 Function Documentation	240
15.43.2.1 loop()	240
15.43.2.2 setup()	240
15.44 TEST_2.ino	240
15.45 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_3/TEST_3.ino File Reference	241
15.45.1 Detailed Description	241
15.45.2 Function Documentation	241
15.45.2.1 loop()	241

---

15.45.2.2 setup() . . . . .	241
15.45.3 Variable Documentation . . . . .	242
15.45.3.1 ledmatrix . . . . .	242
15.46 TEST_3.ino . . . . .	242
15.47 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/examples/TEST_3/TEST_3.ino File Reference . . . . .	243
15.47.1 Detailed Description . . . . .	243
15.47.2 Function Documentation . . . . .	243
15.47.2.1 loop() . . . . .	243
15.47.2.2 setup() . . . . .	243
15.48 TEST_3.ino . . . . .	243
15.49 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_4/TEST_4.ino File Reference . . . . .	244
15.49.1 Detailed Description . . . . .	245
15.49.2 Function Documentation . . . . .	245
15.49.2.1 loop() . . . . .	245
15.49.2.2 setup() . . . . .	245
15.49.3 Variable Documentation . . . . .	245
15.49.3.1 ledmatrix . . . . .	245
15.50 TEST_4.ino . . . . .	246
15.51 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_5/TEST_5.ino File Reference . . . . .	246
15.51.1 Detailed Description . . . . .	246
15.51.2 Function Documentation . . . . .	247
15.51.2.1 loop() . . . . .	247
15.51.2.2 setup() . . . . .	247
15.51.3 Variable Documentation . . . . .	247
15.51.3.1 ledmatrix . . . . .	247
15.52 TEST_5.ino . . . . .	247
15.53 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_6/TEST_6.ino File Reference . . . . .	248
15.53.1 Detailed Description . . . . .	248
15.53.2 Function Documentation . . . . .	248
15.53.2.1 loop() . . . . .	248
15.53.2.2 setup() . . . . .	249
15.53.3 Variable Documentation . . . . .	249
15.53.3.1 display_status . . . . .	249
15.53.3.2 ledmatrix . . . . .	249
15.53.3.3 mode_bright_display . . . . .	249
15.53.3.4 reset_init_text . . . . .	249
15.54 TEST_6.ino . . . . .	249
15.55 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_7/TEST_7.ino File Reference . . . . .	250

---

15.55.1 Detailed Description . . . . .	250
15.55.2 Function Documentation . . . . .	250
15.55.2.1 loop() . . . . .	250
15.55.2.2 setup() . . . . .	250
15.55.3 Variable Documentation . . . . .	251
15.55.3.1 display_status . . . . .	251
15.55.3.2 ledmatrix . . . . .	251
15.55.3.3 mode_bright_display . . . . .	251
15.55.3.4 reset_init_text . . . . .	251
15.56 TEST_7.ino . . . . .	251
15.57 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0002-DISPLAY/examples/TEST_8/TEST_8.ino File Reference . . . . .	251
15.57.1 Detailed Description . . . . .	252
15.57.2 Function Documentation . . . . .	252
15.57.2.1 loop() . . . . .	252
15.57.2.2 setup() . . . . .	252
15.57.3 Variable Documentation . . . . .	252
15.57.3.1 ledmatrix . . . . .	252
15.57.3.2 timer . . . . .	252
15.58 TEST_8.ino . . . . .	253
15.59 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0003-BUZZER/Buzzer.h File Reference . . . . .	253
15.59.1 Detailed Description . . . . .	255
15.59.2 Macro Definition Documentation . . . . .	255
15.59.2.1 NOTE_B9 . . . . .	255
15.59.3 Function Documentation . . . . .	255
15.59.3.1 getSound() . . . . .	255
15.59.3.2 InitBuzzer() . . . . .	255
15.59.3.3 playSound() . . . . .	256
15.59.4 Variable Documentation . . . . .	256
15.59.4.1 C_BUZZER_BUSSY . . . . .	256
15.59.4.2 C_BUZZER_NOT_BUSSY . . . . .	256
15.59.4.3 C_MODE_DEFAULT . . . . .	256
15.59.4.4 C_MODE_HACK_ALTERNATIVE . . . . .	256
15.59.4.5 C_NOTES_CHARGE_IN . . . . .	256
15.59.4.6 C_NOTES_CHARGE_OUT . . . . .	257
15.59.4.7 C_NOTES_DEATHBATTERY . . . . .	257
15.59.4.8 C_NOTES_DOWN_1 . . . . .	257
15.59.4.9 C_NOTES_END_1 . . . . .	257
15.59.4.10 C_NOTES_ERROR . . . . .	257
15.59.4.11 C_NOTES_FULL_CHARGE . . . . .	257
15.59.4.12 C_NOTES_LOW_BATTERY . . . . .	257
15.59.4.13 C_NOTES_OFF_1 . . . . .	257

---

15.59.4.14 C_NOTES_ON_1 . . . . .	257
15.59.4.15 C_NOTES_START_1 . . . . .	258
15.59.4.16 C_NOTES_UP_1 . . . . .	258
15.59.4.17 C_PIN_BUZZER . . . . .	258
15.59.4.18 C_SOUND_CHARGE_IN . . . . .	258
15.59.4.19 C_SOUND_CHARGE_OUT . . . . .	258
15.59.4.20 C_SOUND_DEATH_BATTERY . . . . .	258
15.59.4.21 C_SOUND_DOWN . . . . .	258
15.59.4.22 C_SOUND_END . . . . .	258
15.59.4.23 C_SOUND_ERROR . . . . .	258
15.59.4.24 C_SOUND_FULL_CHARGE . . . . .	258
15.59.4.25 C_SOUND_LOW_BATTERY . . . . .	259
15.59.4.26 C_SOUND_MUTE . . . . .	259
15.59.4.27 C_SOUND_OFF . . . . .	259
15.59.4.28 C_SOUND_ON . . . . .	259
15.59.4.29 C_SOUND_START . . . . .	259
15.59.4.30 C_SOUND_UP . . . . .	259
15.59.4.31 size_sound_charge_in . . . . .	259
15.59.4.32 size_sound_charge_out . . . . .	259
15.59.4.33 size_sound_death_battery . . . . .	259
15.59.4.34 size_sound_down . . . . .	259
15.59.4.35 size_sound_end . . . . .	260
15.59.4.36 size_sound_error . . . . .	260
15.59.4.37 size_sound_full_charge . . . . .	260
15.59.4.38 size_sound_low_battery . . . . .	260
15.59.4.39 size_sound_off . . . . .	260
15.59.4.40 size_sound_on . . . . .	260
15.59.4.41 size_sound_start . . . . .	260
15.59.4.42 size_sound_up . . . . .	260
15.59.4.43 SOUND_CHARGE_IN . . . . .	260
15.59.4.44 SOUND_CHARGE_OUT . . . . .	260
15.59.4.45 SOUND_DEATH_BATTERY . . . . .	261
15.59.4.46 SOUND_DOWN . . . . .	261
15.59.4.47 SOUND_END . . . . .	261
15.59.4.48 SOUND_ERROR . . . . .	261
15.59.4.49 SOUND_FULL_CHARGE . . . . .	261
15.59.4.50 SOUND_LOW_BATTERY . . . . .	261
15.59.4.51 SOUND_OFF . . . . .	261
15.59.4.52 SOUND_ON . . . . .	261
15.59.4.53 SOUND_START . . . . .	261
15.59.4.54 SOUND_UP . . . . .	261
15.59.4.55 timer_buzzer . . . . .	262

15.60 Buzzer.h . . . . .	262
15.61 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0003-BUZZER/notes.h File Reference . . . . .	264
15.61.1 Detailed Description . . . . .	266
15.61.2 Macro Definition Documentation . . . . .	267
15.61.2.1 NOTE_A1 . . . . .	267
15.61.2.2 NOTE_A2 . . . . .	267
15.61.2.3 NOTE_A3 . . . . .	267
15.61.2.4 NOTE_A4 . . . . .	267
15.61.2.5 NOTE_A5 . . . . .	267
15.61.2.6 NOTE_A6 . . . . .	267
15.61.2.7 NOTE_A7 . . . . .	267
15.61.2.8 NOTE_A8 . . . . .	267
15.61.2.9 NOTE_A9 . . . . .	268
15.61.2.10 NOTE_AS1 . . . . .	268
15.61.2.11 NOTE_AS2 . . . . .	268
15.61.2.12 NOTE_AS3 . . . . .	268
15.61.2.13 NOTE_AS4 . . . . .	268
15.61.2.14 NOTE_AS5 . . . . .	268
15.61.2.15 NOTE_AS6 . . . . .	268
15.61.2.16 NOTE_AS7 . . . . .	268
15.61.2.17 NOTE_AS8 . . . . .	268
15.61.2.18 NOTE_AS9 . . . . .	268
15.61.2.19 NOTE_B0 . . . . .	269
15.61.2.20 NOTE_B1 . . . . .	269
15.61.2.21 NOTE_B2 . . . . .	269
15.61.2.22 NOTE_B3 . . . . .	269
15.61.2.23 NOTE_B4 . . . . .	269
15.61.2.24 NOTE_B5 . . . . .	269
15.61.2.25 NOTE_B6 . . . . .	269
15.61.2.26 NOTE_B7 . . . . .	269
15.61.2.27 NOTE_B8 . . . . .	269
15.61.2.28 NOTE_C1 . . . . .	269
15.61.2.29 NOTE_C2 . . . . .	270
15.61.2.30 NOTE_C3 . . . . .	270
15.61.2.31 NOTE_C4 . . . . .	270
15.61.2.32 NOTE_C5 . . . . .	270
15.61.2.33 NOTE_C6 . . . . .	270
15.61.2.34 NOTE_C7 . . . . .	270
15.61.2.35 NOTE_C8 . . . . .	270
15.61.2.36 NOTE_C9 . . . . .	270
15.61.2.37 NOTE_CS1 . . . . .	270

---

15.61.2.38 NOTE_CS2 . . . . .	270
15.61.2.39 NOTE_CS3 . . . . .	271
15.61.2.40 NOTE_CS4 . . . . .	271
15.61.2.41 NOTE_CS5 . . . . .	271
15.61.2.42 NOTE_CS6 . . . . .	271
15.61.2.43 NOTE_CS7 . . . . .	271
15.61.2.44 NOTE_CS8 . . . . .	271
15.61.2.45 NOTE_CS9 . . . . .	271
15.61.2.46 NOTE_D1 . . . . .	271
15.61.2.47 NOTE_D2 . . . . .	271
15.61.2.48 NOTE_D3 . . . . .	271
15.61.2.49 NOTE_D4 . . . . .	272
15.61.2.50 NOTE_D5 . . . . .	272
15.61.2.51 NOTE_D6 . . . . .	272
15.61.2.52 NOTE_D7 . . . . .	272
15.61.2.53 NOTE_D8 . . . . .	272
15.61.2.54 NOTE_D9 . . . . .	272
15.61.2.55 NOTE_DS1 . . . . .	272
15.61.2.56 NOTE_DS2 . . . . .	272
15.61.2.57 NOTE_DS3 . . . . .	272
15.61.2.58 NOTE_DS4 . . . . .	272
15.61.2.59 NOTE_DS5 . . . . .	273
15.61.2.60 NOTE_DS6 . . . . .	273
15.61.2.61 NOTE_DS7 . . . . .	273
15.61.2.62 NOTE_DS8 . . . . .	273
15.61.2.63 NOTE_DS9 . . . . .	273
15.61.2.64 NOTE_E1 . . . . .	273
15.61.2.65 NOTE_E2 . . . . .	273
15.61.2.66 NOTE_E3 . . . . .	273
15.61.2.67 NOTE_E4 . . . . .	273
15.61.2.68 NOTE_E5 . . . . .	273
15.61.2.69 NOTE_E6 . . . . .	274
15.61.2.70 NOTE_E7 . . . . .	274
15.61.2.71 NOTE_E8 . . . . .	274
15.61.2.72 NOTE_E9 . . . . .	274
15.61.2.73 NOTE_F1 . . . . .	274
15.61.2.74 NOTE_F2 . . . . .	274
15.61.2.75 NOTE_F3 . . . . .	274
15.61.2.76 NOTE_F4 . . . . .	274
15.61.2.77 NOTE_F5 . . . . .	274
15.61.2.78 NOTE_F6 . . . . .	274
15.61.2.79 NOTE_F7 . . . . .	275

15.61.2.80 NOTE_F8 . . . . .	275
15.61.2.81 NOTE_F9 . . . . .	275
15.61.2.82 NOTE_FS1 . . . . .	275
15.61.2.83 NOTE_FS2 . . . . .	275
15.61.2.84 NOTE_FS3 . . . . .	275
15.61.2.85 NOTE_FS4 . . . . .	275
15.61.2.86 NOTE_FS5 . . . . .	275
15.61.2.87 NOTE_FS6 . . . . .	275
15.61.2.88 NOTE_FS7 . . . . .	275
15.61.2.89 NOTE_FS8 . . . . .	276
15.61.2.90 NOTE_FS9 . . . . .	276
15.61.2.91 NOTE_G1 . . . . .	276
15.61.2.92 NOTE_G2 . . . . .	276
15.61.2.93 NOTE_G3 . . . . .	276
15.61.2.94 NOTE_G4 . . . . .	276
15.61.2.95 NOTE_G5 . . . . .	276
15.61.2.96 NOTE_G6 . . . . .	276
15.61.2.97 NOTE_G7 . . . . .	276
15.61.2.98 NOTE_G8 . . . . .	276
15.61.2.99 NOTE_G9 . . . . .	277
15.61.2.100 NOTE_GS1 . . . . .	277
15.61.2.101 NOTE_GS2 . . . . .	277
15.61.2.102 NOTE_GS3 . . . . .	277
15.61.2.103 NOTE_GS4 . . . . .	277
15.61.2.104 NOTE_GS5 . . . . .	277
15.61.2.105 NOTE_GS6 . . . . .	277
15.61.2.106 NOTE_GS7 . . . . .	277
15.61.2.107 NOTE_GS8 . . . . .	277
15.61.2.108 NOTE_GS9 . . . . .	277
15.62 notes.h . . . . .	278
15.63 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0004-DCDC/DCDC.h File Reference . . . . .	279
15.63.1 Detailed Description . . . . .	279
15.63.2 Variable Documentation . . . . .	280
15.63.2.1 ADDR_I2C_DCDC . . . . .	280
15.63.2.2 BOOST_ARRAY . . . . .	280
15.63.2.3 C_BOOST_MODE . . . . .	280
15.63.2.4 C_NON_BOOST_MODE . . . . .	280
15.63.2.5 LenDCDCvalues . . . . .	280
15.63.2.6 NON_BOOST_ARRAY . . . . .	281
15.64 DCDC.h . . . . .	281
15.65 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0005-SENSING/HealthMonitor.h File Reference . . . . .	282

15.65.1 Detailed Description . . . . .	282
15.65.2 Variable Documentation . . . . .	282
15.65.2.1 C_COUNTER_LIMIT . . . . .	282
15.65.2.2 C_RISE_COUNT . . . . .	282
15.65.2.3 C_SAMPLING_TIME . . . . .	282
15.65.2.4 C_TIME_HEALTH_MONITORING . . . . .	283
15.66 HealthMonitor.h . . . . .	283
15.67 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0006-POWERBAR/power_bar.h File Reference . . . . .	284
15.67.1 Detailed Description . . . . .	284
15.67.2 Function Documentation . . . . .	285
15.67.2.1 PowerBar() . . . . .	285
15.67.2.2 UpdatePowerBar() . . . . .	285
15.67.3 Variable Documentation . . . . .	285
15.67.3.1 LEDS_IN_POWERBAR . . . . .	285
15.67.3.2 MAX_POWER DISPLAYED . . . . .	285
15.67.3.3 power_by_led . . . . .	285
15.67.3.4 refresh_timer . . . . .	286
15.68 power_bar.h . . . . .	286
15.69 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0007-SlowSoftI2CMaster/batt_SlowSoftI2CMaster.cpp File Reference . . . . .	286
15.70 batt_SlowSoftI2CMaster.cpp . . . . .	287
15.71 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0007-SlowSoftI2CMaster/batt_SlowSoftI2CMaster.h File Reference . . . . .	288
15.71.1 Macro Definition Documentation . . . . .	289
15.71.1.1 BUFFER_LENGTH . . . . .	289
15.71.1.2 DELAY . . . . .	289
15.71.1.3 I2C_MAXWAIT . . . . .	289
15.71.1.4 I2C_READ . . . . .	289
15.71.1.5 I2C_WRITE . . . . .	289
15.71.2 Function Documentation . . . . .	289
15.72 batt_SlowSoftI2CMaster.h . . . . .	289
15.73 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_Adafruit_GFX.cpp File Reference . . . . .	290
15.73.1 Macro Definition Documentation . . . . .	290
15.73.1.1 _swap_int16_t . . . . .	290
15.73.1.2 min . . . . .	291
15.73.1.3 pgm_read_byte . . . . .	291
15.73.1.4 pgm_read_dword . . . . .	291
15.73.1.5 pgm_read_pointer . . . . .	291
15.73.1.6 pgm_read_word . . . . .	291
15.73.2 Function Documentation . . . . .	291
15.73.2.1 pgm_read_bitmap_ptr() . . . . .	291
15.73.2.2 pgm_read_glyph_ptr() . . . . .	291

---

15.74 batt_Adafruit_GFX.cpp . . . . .	292
15.75 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_Adafruit_GFX.h File Reference . . . . .	315
15.76 batt_Adafruit_GFX.h . . . . .	315
15.77 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_gfxfont.h File Reference . . . . .	319
15.78 batt_gfxfont.h . . . . .	319
15.79 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_glcdfont.c File Reference . . . . .	319
15.79.1 Macro Definition Documentation . . . . .	319
15.79.1.1 FONT5X7_H . . . . .	319
15.79.1.2 PROGMEM . . . . .	320
15.80 batt_glcdfont.c . . . . .	320
15.81 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/batt_ArduinoLowPower.cpp File Reference . . . . .	321
15.82 batt_ArduinoLowPower.cpp . . . . .	321
15.83 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/batt_ArduinoLowPower.h File Reference . . . . .	325
15.83.1 Macro Definition Documentation . . . . .	325
15.83.1.1 RTC_ALARM_WAKEUP . . . . .	325
15.83.2 Typedef Documentation . . . . .	326
15.83.2.1 irq_mode . . . . .	326
15.83.2.2 onOffFuncPtr . . . . .	326
15.83.3 Enumeration Type Documentation . . . . .	326
15.83.3.1 wakeup_reason . . . . .	326
15.83.4 Variable Documentation . . . . .	326
15.83.4.1 LowPower . . . . .	326
15.84 batt_ArduinoLowPower.h . . . . .	326
15.85 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/examples/AdcWakeups/AdcWakeups.ino File Reference . . . . .	328
15.85.1 Function Documentation . . . . .	328
15.85.1.1 loop() . . . . .	328
15.85.1.2 repetitionsIncrease() . . . . .	328
15.85.1.3 setup() . . . . .	328
15.85.2 Variable Documentation . . . . .	328
15.85.2.1 margin . . . . .	328
15.85.2.2 pin . . . . .	329
15.85.2.3 repetitions . . . . .	329
15.86 AdcWakeups.ino . . . . .	329
15.87 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/examples/ExternalWakeups/ExternalWakeups.ino File Reference . . . . .	330
15.87.1 Function Documentation . . . . .	330
15.87.1.1 loop() . . . . .	330

15.87.1.2 repetitionsIncrease()	330
15.87.1.3 setup()	330
15.87.2 Variable Documentation	330
15.87.2.1 pin	330
15.87.2.2 repetitions	331
15.88 ExternalWakeups.ino	331
15.89 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/examples/PrimoDeepSleep/PrimoDeepSleep.ino File Reference	332
15.89.1 Function Documentation	332
15.89.1.1 doMyStuff()	332
15.89.1.2 doMyStuffWithNFC()	332
15.89.1.3 doOtherStuff()	332
15.89.1.4 loop()	332
15.89.1.5 setup()	332
15.89.1.6 StmEspPM()	333
15.89.2 Variable Documentation	333
15.89.2.1 analogPin	333
15.89.2.2 digitalPin	333
15.90 PrimoDeepSleep.ino	333
15.91 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/examples/TianStandby/TianStandby.ino File Reference	334
15.91.1 Macro Definition Documentation	334
15.91.1.1 MIPS_PIN	334
15.91.2 Function Documentation	335
15.91.2.1 loop()	335
15.91.2.2 MipsPM()	335
15.91.2.3 onWakeups()	335
15.91.2.4 setup()	335
15.92 TianStandby.ino	335
15.93 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0009-LOWPOWER/examples/TimedWakeups/TimedWakeups.ino File Reference	336
15.93.1 Function Documentation	336
15.93.1.1 dummy()	336
15.93.1.2 loop()	336
15.93.1.3 setup()	336
15.94 TimedWakeups.ino	336
15.95 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/diagnostic.h File Reference	337
15.95.1 Detailed Description	338
15.95.2 Macro Definition Documentation	338
15.95.2.1 EEPROM_EMULATION_SIZE	338

---

15.95.3 Function Documentation . . . . .	339
15.95.3.1 FlashStorage() . . . . .	339
15.95.3.2 IncrementDiagnosticData() . . . . .	339
15.95.3.3 Init_diagnostic_elements() . . . . .	339
15.95.3.4 isValid() . . . . .	339
15.95.3.5 LogDiagnosticData() . . . . .	339
15.95.3.6 PrintDiagnosticData() . . . . .	340
15.95.3.7 PrintStaticData() . . . . .	340
15.95.3.8 PrintStats() . . . . .	340
15.95.3.9 ReadDiagnosticData() . . . . .	340
15.95.3.10 ResetBateriaEeprom() . . . . .	340
15.95.3.11 SaveEepromChasis() . . . . .	340
15.95.3.12 Stats() . . . . .	340
15.95.3.13 UpdateEepromBatory() . . . . .	341
15.95.4 Variable Documentation . . . . .	341
15.95.4.1 _dirty . . . . .	341
15.95.4.2 _eeprom_RAM . . . . .	341
15.95.4.3 eeprom_index . . . . .	341
15.95.4.4 elements . . . . .	341
15.95.4.5 init_flag . . . . .	341
15.95.4.6 MAX_VOLTAGE . . . . .	341
15.95.4.7 MAX_WATS . . . . .	341
15.95.4.8 MIN_VOLTAGE . . . . .	342
15.95.4.9 MIN_WATS . . . . .	342
15.95.4.10 power_values . . . . .	342
15.95.4.11 voltage_values . . . . .	342
15.96 diagnostic.h . . . . .	342
15.97 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/Eeprom_LUT.h File Reference . . . . .	346
15.97.1 Detailed Description . . . . .	347
15.97.2 Variable Documentation . . . . .	347
15.97.2.1 C_CAPACITY . . . . .	347
15.97.2.2 C_CONSUMPTION_ERROR . . . . .	347
15.97.2.3 C_HACK_CHARGE_SOUND . . . . .	348
15.97.2.4 C_HACK_DEATH_BATTERY_SOUND . . . . .	348
15.97.2.5 C_HACK_END_DISPLAY . . . . .	348
15.97.2.6 C_HACK_END_SOUND . . . . .	348
15.97.2.7 C_HACK_FULL_CHARGE_SOUND . . . . .	348
15.97.2.8 C_HACK_START_DISPLAY . . . . .	348
15.97.2.9 C_HACK_START_SOUND . . . . .	348
15.97.2.10 C_INIT_STATE . . . . .	348
15.97.2.11 C_INSTANT_CURRENT . . . . .	348

15.97.2.12 C_INSTANT_POWER . . . . .	348
15.97.2.13 C_INSTANT_VOLTAGE . . . . .	349
15.97.2.14 C_MODEL . . . . .	349
15.97.2.15 C_NUM_ELEMENTS . . . . .	349
15.97.2.16 C_NUMBER_CYCLES . . . . .	349
15.97.2.17 C_POWER_ERROR . . . . .	349
15.97.2.18 C_SERIAL_NUMBER . . . . .	349
15.97.2.19 C THEORY_VOLTAGE . . . . .	349
15.97.2.20 C_VOLTAGE_ERROR . . . . .	349
15.97.2.21 C_VOLTAGE_INPUT_ERROR . . . . .	349
15.97.2.22 C_WORK_TIME . . . . .	349
15.97.2.23 POS_EEPROM_CONSUMPTION_ERROR . . . . .	350
15.97.2.24 POS_EEPROM_INIT . . . . .	350
15.97.2.25 POS_EEPROM_MODEL . . . . .	350
15.97.2.26 POS_EEPROM_NUMBER_CYCLES . . . . .	350
15.97.2.27 POS_EEPROM_POWER_ARRAY . . . . .	350
15.97.2.28 POS_EEPROM_POWER_ERROR . . . . .	350
15.97.2.29 POS_EEPROM_SERIAL_NUMBER . . . . .	350
15.97.2.30 POS_EEPROM_VOLTAGE_ERROR . . . . .	350
15.97.2.31 POS_EEPROM_VOLTAGE_INPUT_ERROR . . . . .	350
15.97.2.32 POS_EEPROM_VOLTS_ARRAY . . . . .	350
15.97.2.33 POS_EEPROM_WORK_TIME . . . . .	351
15.98 Eeprom_LUT.h . . . . .	351
15.99 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0010-LOGGING/EepromBitBang.h File Reference . . . . .	351
15.99.1 Macro Definition Documentation . . . . .	352
15.99.1.1 EEPROMADDR . . . . .	352
15.99.1.2 SCL_PIN . . . . .	352
15.99.1.3 SDA_PIN . . . . .	352
15.99.2 Function Documentation . . . . .	352
15.99.2.1 ReadArrayEEPROM() . . . . .	352
15.99.2.2 readEEPROM() . . . . .	352
15.99.2.3 ReadWordEEPROM() . . . . .	352
15.99.2.4 WriteArrayEEPROM() . . . . .	352
15.99.2.5 writeEEPROM() . . . . .	353
15.99.2.6 WriteWordEEPROM() . . . . .	353
15.99.3 Variable Documentation . . . . .	353
15.99.3.1 si . . . . .	353
15.100 EepromBitBang.h . . . . .	353
15.101 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/batt_Adafruit_IS31FL3731.cpp File Reference . . . . .	354
15.101.1 Macro Definition Documentation . . . . .	354

---

15.101.1.1 _swap_int16_t . . . . .	354
15.102 batt_Adafruit_IS31FL3731.cpp . . . . .	355
15.103 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/batt_Adafruit_IS31FL3731.h File Reference . . . . .	357
15.103.1 Macro Definition Documentation . . . . .	358
15.103.1.1 ISSI_ADDR_DEFAULT . . . . .	358
15.103.1.2 ISSI_BANK_FUNCTIONREG . . . . .	358
15.103.1.3 ISSI_COMMANDREGISTER . . . . .	358
15.103.1.4 ISSI_CONF_AUDIOMODE . . . . .	358
15.103.1.5 ISSI_CONF_AUTOFRAMEMODE . . . . .	358
15.103.1.6 ISSI_CONF_PICTUREMODE . . . . .	359
15.103.1.7 ISSI_REG_AUDIOSYNC . . . . .	359
15.103.1.8 ISSI_REG_CONFIG . . . . .	359
15.103.1.9 ISSI_REG_CONFIG_AUDIOPLAYMODE . . . . .	359
15.103.1.10 ISSI_REG_CONFIG_AUTOPLAYMODE . . . . .	359
15.103.1.11 ISSI_REG_CONFIG_PICTUREMODE . . . . .	359
15.103.1.12 ISSI_REG_PICTUREFRAME . . . . .	359
15.103.1.13 ISSI_REG_SHUTDOWN . . . . .	359
15.104 batt_Adafruit_IS31FL3731.h . . . . .	359
15.105 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/examples/gfxdemo/gfxdemo.ino File Reference . . . . .	360
15.105.1 Function Documentation . . . . .	360
15.105.1.1 loop() . . . . .	360
15.105.1.2 setup() . . . . .	361
15.105.2 Variable Documentation . . . . .	361
15.105.2.1 matrix . . . . .	361
15.106 gfxdemo.ino . . . . .	361
15.107 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/examples/manualanim/manualanim.ino File Reference . . . . .	362
15.107.1 Function Documentation . . . . .	363
15.107.1.1 loop() . . . . .	363
15.107.1.2 setup() . . . . .	363
15.107.2 Variable Documentation . . . . .	363
15.107.2.1 ledmatrix . . . . .	363
15.107.2.2 x . . . . .	363
15.108 manualanim.ino . . . . .	363
15.109 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/examples/swirldemo/swirldemo.ino File Reference . . . . .	364
15.109.1 Function Documentation . . . . .	364
15.109.1.1 loop() . . . . .	364

---

15.109.1.2 setup()	364
15.109.2 Variable Documentation	364
15.109.2.1 ledmatrix	364
15.109.2.2 sweep	364
15.110 swirldemo.ino	365
15.111 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/examples/EmulateEEPROM/EmulateEEPROM.ino File Reference	365
15.111.1 Function Documentation	365
15.111.1.1 loop()	365
15.111.1.2 setup()	365
15.112 EmulateEEPROM.ino	366
15.113 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/examples/FlashStoreAndRetrieve/FlashStoreAndRetrieve.ino File Reference	366
15.113.1 Function Documentation	367
15.113.1.1 FlashStorage()	367
15.113.1.2 loop()	367
15.113.1.3 setup()	367
15.114 FlashStoreAndRetrieve.ino	367
15.115 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/examples/StoreNameAndSurname/StoreNameAndSurname.ino File Reference	368
15.115.1 Function Documentation	368
15.115.1.1 FlashStorage()	368
15.115.1.2 loop()	368
15.115.1.3 setup()	368
15.116 StoreNameAndSurname.ino	368
15.117 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashAsEEPROM.cpp File Reference	369
15.117.1 Function Documentation	370
15.117.1.1 FlashStorage()	370
15.117.2 Variable Documentation	370
15.117.2.1 EEPROM	370
15.118 batt_FlashAsEEPROM.cpp	370
15.119 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashAsEEPROM.h File Reference	371
15.119.1 Macro Definition Documentation	371
15.119.1.1 EEPROM_EMULATION_SIZE	371
15.119.2 Variable Documentation	371
15.119.2.1 EEPROM	371
15.120 batt_FlashAsEEPROM.h	372
15.121 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashStorage.cpp File Reference	372
15.122 batt_FlashStorage.cpp	373

---

15.123 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashStorage.h File Reference . . . . .	374
15.123.1 Macro Definition Documentation . . . . .	375
15.123.1.1 Flash . . . . .	375
15.123.1.2 FlashStorage . . . . .	375
15.123.1.3 PPCAT . . . . .	375
15.123.1.4 PPCAT_NX . . . . .	375
15.124 batt_FlashStorage.h . . . . .	375
15.125 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0013-SAMD_AnalogCorrection/examples/CorrectADCResponse/← CorrectADCResponse.ino File Reference . . . . .	377
15.125.1 Macro Definition Documentation . . . . .	377
15.125.1.1 ADC_3V3_PIN . . . . .	377
15.125.1.2 ADC_GND_PIN . . . . .	377
15.125.1.3 ADC_MAX_GAIN . . . . .	377
15.125.1.4 ADC_MIN_GAIN . . . . .	377
15.125.1.5 ADC_RANGE . . . . .	378
15.125.1.6 ADC_READS_COUNT . . . . .	378
15.125.1.7 ADC_READS_SHIFT . . . . .	378
15.125.1.8 ADC_RESOLUTION_BITS . . . . .	378
15.125.1.9 ADC_TOP_VALUE . . . . .	378
15.125.1.10 ADC_UNITY_GAIN . . . . .	378
15.125.1.11 MAX_TOP_VALUE_READS . . . . .	378
15.125.2 Function Documentation . . . . .	378
15.125.2.1 loop() . . . . .	378
15.125.2.2 read3V3Level() . . . . .	378
15.125.2.3 readGndLevel() . . . . .	379
15.125.2.4 setup() . . . . .	379
15.126 CorrectADCResponse.ino . . . . .	379
15.127 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0013-SAMD_AnalogCorrection/src/batt_SAMD_AnalogCorrection.cpp File Reference . . . . .	381
15.127.1 Function Documentation . . . . .	381
15.127.1.1 analogReadCorrection() . . . . .	382
15.128 batt_SAMD_AnalogCorrection.cpp . . . . .	382
15.129 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0013-SAMD_AnalogCorrection/src/batt_SAMD_AnalogCorrection.h File Reference . . . . .	382
15.129.1 Function Documentation . . . . .	382
15.129.1.1 analogReadCorrection() . . . . .	382
15.130 batt_SAMD_AnalogCorrection.h . . . . .	383
15.131 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0014-MilliTimer/MilliTimer.cpp File Reference . . . . .	383
15.132 MilliTimmer.cpp . . . . .	383

---

15.133 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0014-MilliTimer/MilliTimer.h File Reference . . . . .	384
15.133.1 Macro Definition Documentation . . . . .	384
15.133.1.1 C_TIMER_NOT_EXPIRED . . . . .	384
15.134 MilliTimer.h . . . . .	384
15.135 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0015-RTCZero/examples/Epoch/Epoch.ino File Reference . . . . .	384
15.135.1 Function Documentation . . . . .	385
15.135.1.1 loop() . . . . .	385
15.135.1.2 print2digits() . . . . .	385
15.135.1.3 setup() . . . . .	385
15.135.2 Variable Documentation . . . . .	385
15.135.2.1 rtc . . . . .	385
15.136 Epoch.ino . . . . .	385
15.137 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0015-RTCZero/examples/SimpleRTC/SimpleRTC.ino File Reference . . . . .	386
15.137.1 Function Documentation . . . . .	386
15.137.1.1 loop() . . . . .	386
15.137.1.2 print2digits() . . . . .	387
15.137.1.3 setup() . . . . .	387
15.137.2 Variable Documentation . . . . .	387
15.137.2.1 day . . . . .	387
15.137.2.2 hours . . . . .	387
15.137.2.3 minutes . . . . .	387
15.137.2.4 month . . . . .	387
15.137.2.5 rtc . . . . .	387
15.137.2.6 seconds . . . . .	387
15.137.2.7 year . . . . .	387
15.138 SimpleRTC.ino . . . . .	388
15.139 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0015-RTCZero/examples/SimpleRTCAlarm/SimpleRTCAlarm.ino File Reference . . . . .	389
15.139.1 Function Documentation . . . . .	389
15.139.1.1 alarmMatch() . . . . .	389
15.139.1.2 loop() . . . . .	389
15.139.1.3 setup() . . . . .	389
15.139.2 Variable Documentation . . . . .	389
15.139.2.1 day . . . . .	389
15.139.2.2 hours . . . . .	390
15.139.2.3 minutes . . . . .	390
15.139.2.4 month . . . . .	390
15.139.2.5 rtc . . . . .	390
15.139.2.6 seconds . . . . .	390

---

15.139.2.7 year . . . . .	390
15.140 SimpleRTCAlarm.ino . . . . .	390
15.141 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0015-RTCZero/examples/SleepRTCAlarm/SleepRTCAlarm.ino File Reference . . . . .	391
15.141.1 Function Documentation . . . . .	391
15.141.1.1 alarmMatch() . . . . .	391
15.141.1.2 loop() . . . . .	391
15.141.1.3 setup() . . . . .	392
15.141.2 Variable Documentation . . . . .	392
15.141.2.1 day . . . . .	392
15.141.2.2 hours . . . . .	392
15.141.2.3 minutes . . . . .	392
15.141.2.4 month . . . . .	392
15.141.2.5 rtc . . . . .	392
15.141.2.6 seconds . . . . .	392
15.141.2.7 year . . . . .	392
15.142 SleepRTCAlarm.ino . . . . .	392
15.143 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0015-RTCZero/src/batt_RTCZero.cpp File Reference . . . . .	393
15.143.1 Macro Definition Documentation . . . . .	394
15.143.1.1 DEFAULT_DAY . . . . .	394
15.143.1.2 DEFAULT_HOUR . . . . .	394
15.143.1.3 DEFAULT_MINUTE . . . . .	394
15.143.1.4 DEFAULT_MONTH . . . . .	394
15.143.1.5 DEFAULT_SECOND . . . . .	394
15.143.1.6 DEFAULT_YEAR . . . . .	394
15.143.1.7 EPOCH_TIME_OFF . . . . .	394
15.143.1.8 EPOCH_TIME_YEAR_OFF . . . . .	394
15.143.2 Function Documentation . . . . .	395
15.143.2.1 RTC_Handler() . . . . .	395
15.143.3 Variable Documentation . . . . .	395
15.143.3.1 RTC_callBack . . . . .	395
15.144 batt_RTCZero.cpp . . . . .	395
15.145 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/LIBRARIES/0015-RTCZero/src/batt_RTCZero.h File Reference . . . . .	401
15.145.1 Typedef Documentation . . . . .	401
15.145.1.1 voidFuncPtr . . . . .	401
15.146 batt_RTCZero.h . . . . .	401
15.147 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/CPU/Validacion Diseño/ARDUINO/Test_Shield_Validator_CPU/Test_Shield_Validator_CPU.ino File Reference . . . . .	403
15.148 Test_Shield_Validator_CPU.ino . . . . .	403

---

15.149 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/CPU/Validacion CPU.ino File Reference . . . . .	403
15.150 PCB_Validator_CPU.ino . . . . .	403
15.151 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/CPU/Validacion Produccion/ARDUINO/Test_Production/Test_Production.ino File Reference . . . . .	403
15.151.1 Detailed Description . . . . .	404
15.151.2 Function Documentation . . . . .	404
15.151.2.1 loop() . . . . .	404
15.151.2.2 setup() . . . . .	404
15.151.3 Variable Documentation . . . . .	405
15.151.3.1 C_PIN_ENABLE_LDO . . . . .	405
15.151.3.2 C_PIN_IOUT_SENSE . . . . .	405
15.151.3.3 C_PIN_ISENSE_RAW . . . . .	405
15.151.3.4 C_PIN_LED_1 . . . . .	405
15.151.3.5 C_PIN_LED_2 . . . . .	405
15.151.3.6 C_PIN_LED_3 . . . . .	405
15.151.3.7 C_PIN_SCL_2 . . . . .	405
15.151.3.8 C_PIN_SDA_2 . . . . .	405
15.151.3.9 C_PIN_SW1 . . . . .	405
15.151.3.10 C_PIN_SW2 . . . . .	406
15.151.3.11 C_PIN_VCC . . . . .	406
15.151.3.12 C_PIN_VCC_2 . . . . .	406
15.151.3.13 data . . . . .	406
15.151.3.14 sample_iout_sense . . . . .	406
15.151.3.15 sample_vcc . . . . .	406
15.151.3.16 sample_vcc_2 . . . . .	406
15.151.3.17 scl_pin . . . . .	406
15.151.3.18 sda_pin . . . . .	406
15.151.3.19 test_start . . . . .	406
15.152 Test_Production.ino . . . . .	407
15.153 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/CPU/Validacion Produccion/LABVIEW/CPU TEST/sketch_jul14a/sketch_jul14a.ino File Reference . . . . .	409
15.153.1 Function Documentation . . . . .	409
15.153.1.1 loop() . . . . .	409
15.153.1.2 setup() . . . . .	409
15.153.2 Variable Documentation . . . . .	409
15.153.2.1 C_PIN_ENABLE_LDO . . . . .	409
15.153.2.2 C_PIN_IOUT_SENSE . . . . .	410
15.153.2.3 C_PIN_ISENSE_RAW . . . . .	410
15.153.2.4 C_PIN_LED_1 . . . . .	410
15.153.2.5 C_PIN_LED_2 . . . . .	410

---

15.153.2.6 C_PIN_LED_3 . . . . .	410
15.153.2.7 C_PIN_SCL_2 . . . . .	410
15.153.2.8 C_PIN_SDA_2 . . . . .	410
15.153.2.9 C_PIN_SW1 . . . . .	410
15.153.2.10 C_PIN_SW2 . . . . .	410
15.153.2.11 C_PIN_VCC . . . . .	410
15.153.2.12 C_PIN_VCC_2 . . . . .	411
15.153.2.13 data . . . . .	411
15.153.2.14 i . . . . .	411
15.153.2.15 iout_sense . . . . .	411
15.153.2.16 vcc . . . . .	411
15.153.2.17 vcc_2 . . . . .	411
15.154 sketch_jul14a.ino . . . . .	411
15.155 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/DCDC/Validacion Diseño/Arduino/Test_Shield_Validacion_DCDC/Test_Shield_Validacion_DCDC.ino File Reference . . . . .	413
15.156 Test_Shield_Validacion_DCDC.ino . . . . .	413
15.157 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/DISPLAY/Validacion Diseño/Arduino/Test_Shield_Validacion_Display/Test_Shield_Validacion_Display.ino File Reference . . . . .	414
15.158 Test_Shield_Validacion_Display.ino . . . . .	414
15.159 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat_sw/TEST/INPUT/Validacion Diseño/Test_Shield_Validacion_Input/Test_Shield_Validacion_Input.ino File Reference . . . . .	414
15.160 Test_Shield_Validacion_Input.ino . . . . .	414
<b>Index</b>	<b>415</b>

# Chapter 1

## Control de Versiones

### 1.1 Nomenclatura

Dentro de esta carpeta se encontraran archivos o ficheros que pueden corresponder a distintas ramas de trabajo por ello se decide mantener una coherencia en la nomenclatura que antiende a las siguientes reglas: **Bx\_Rx\_Vx**

- Bx : Rama principal.
- Rx : Rama secundaria.
- Vx : Version.

Ej: B2\_R3\_V5. Version 5 de la rama secundaria 3 de la rama principal 2.

### 1.2 Tabla de Versiones

#### 1.2.1 B1

Version	Features Added	Commit	Board	IDE
B1_V15	Interaccion con la bateria	cebfe41:Merge BAT-96	Feather M0	VSCODE
B1_V14	Alerta de Error	564718c:Merge BAT-95	Feather M0	VSCODE
B1_V13	Aviso Bateria Baja	5eac8e8:Merge BAT-94	Feather M0	VSCODE
B1_V12	Temporizador de Inactividad	aae4b20:Merge branch bat 94	Feather M0	VSCODE
B1_V11	Estado de Conexion USB	5d9c458:Version 11	Feather M0	VSCODE
B1_V10	Modo Sleep y bajo consumo	58g65ed:B1_V10_Sleep Mode & External Interrupt	Feather M0	VSCODE
B1_V9	Profiling	db26dc1:B1_V9 Profiling	Feather M0	VSCODE
B1_V8	Power Bar y Brillo Tenue	cf2be86:B1_V8 Power Bar & Low Brightness	Feather M0	VSCODE
B1_V7	Lectura Capacidad & Fix Error	68dcdad:B1_V7 Lectura Capacidad & Fix Error	Feather M0	VSCODE
B1_V6	Diagnostico Basico	2dbfe4a:B1_V6	Feather M0	VSCODE
B1_V5	Sonidos Basicos	07cce4f:B_V5:Buzzers Sounds	Feather M0	VSCODE

Version	Features Added	Commit	Board	IDE
B1_V4	Funcionalidades Basicas Botoneras y Display	001e90f:B1_V4	Feather M0	VSCODE
B1_V3	Proteccion UnderVoltage y Proteccion OverPower	fab392e:B1_V3: UnderVoltage & OverPower Protection	Feather M0	VSCODE
B1_V2	Proteccion SobreCorriente y Tiempo de Arranque	1028ac8: B1_V2 Consumption Protection	Feather M0	Arduino 1.8.12
B1_V1	Gestion Boost-No Boost	d34e26a: B1_V1	Metro M0 express	Arduino 1.8.12

## 1.3 Descripcion de Versiones B1

### 1.3.1 v15 Interaccion con la introduccion de la bateria

al insertarse la bateria en el body, es necesario que haya un feedback para el usuario. Se ha decidido de manera inicial, incluir un leve brillo en la pantalla seguido de mostrar el nivel de la bateria. Tras ello la bateria procederia a apagarse y entrar en el estado de Sleep. Esta interaccion tiene que tener lugar previamente a entra en el ciclo de control ya que solo se realizara cada vez que la CPU reciba alimentacion.

### 1.3.2 v14 Alerta de Error

Cuando una de las protecciones salta, el sistema se dirige el estado de error. Es necesario crear una alerta visual y sonora que alerte al ususario que el fncionamiento no esta siendo el adecuado. Tras reprodicir la alerta el sistema se redigira al estado de STOP.

### 1.3.3 V13 Aviso Bateria Baja

Siempre que la bateria se encuentre por debajo de cierto voltaje, no se puede asegurar un correcto funcionamiento, debido a que si la maquina que se conecta a la bateria demanda ucha corriente, el voltaje de l abateria puede llegar a caer por debajo de los 2,9V llegando a provocar un apagado de la CPU por fallo en la alimentacion. Por esta razon, se decide incluir un aviso visual y sonoro siempre que al mostrar el porcentaje de bateri, este se enceuntre por debajo del umbral.

La imagen elegida para representar esta falta de bateria es un simbolo de un abateria tachada y parpadeando.

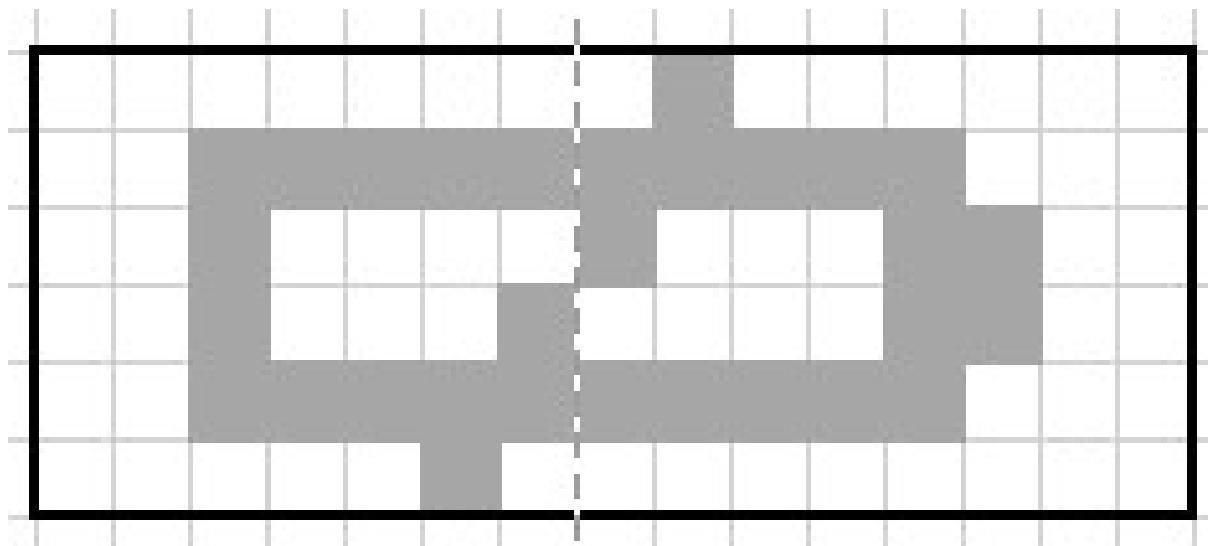


Figure 1.1 LowBatt

El sonido elegido para representar la falta de bateria son 3 pitidos consecutivos.

### 1.3.4 V12 Temporizador de No Uso

Siempre que la bateria se encuentre en el estado de STOP y no se haya pulsado ningun boton ni haya ocurrido un evento, la bateria se apagara automaticamente pasado X tiempo. La finalidad de esta medida es evitar que la bateria se pudiera desgastar al quedarse encendida demasiado tiempo.

### 1.3.5 V11 Estado de conexion USB

Cuando se conecte el USB al battery pack el chasis reaccionara en un estado que únicamente mostrara una animacion por el display indicando la conexion, y seguidamente se quedara permanentemente intentando conectarse al puerto serie. Si se consigue conectar al puerto serie, empezara a retransmitir la informacion del modo diagnostico (Aun por concretar esta informacion.)

Se volvera al modo sleep tras la desconexion del USB.

La detección de la conexión y desconexión se realizará mediante una interrupción hardware.

### 1.3.6 V10 Modo Sleep y bajo consumo

Con el fin de aumentar la vida de la batería, durante el estado de sleep, se introducirá al micro en un estado de lowpower o Standby. Para despertar el micro se utilizará una interrupción hardware provocada por el botón central. Tras el despertar se analizará la rutina de atención de la interrupción y se continuará el programa justo después de la intrucción que duerme al micro.

### 1.3.7 V9 Profiling

Para poder realizar un perfil de tiempos, se utilizan 6 pines que actualmente no tienen uso para, utilizando un analizador lógico, se pueda observar el consumo temporal de las distintas funciones como:

- Duración del ciclo de control.
- Lectura de la botonera.
- Funciones del Buzzer.
- Funciones del Display.
- Funciones de Diagnóstico.
- Función de la Power Bar.

Debido a esta incorporación se han realizado un par de correcciones para evitar ciclos de control más largos de lo necesario.

### 1.3.8 V8 Power Bar y Brillo Tenue

Manejo de la barra de potencia, formada por 14 leds puestos en linea recta. Colocada en la parte inferior del Display. La barra de potencia mostrara de forma grafica el valor de la potencia instantanea, de forma que con un fondo de escala ajustable (incialmente en 2500 mW) y una granularidad de 13 leds, se encienden los leds de manera fluida.

En los estados de inicio, apagado, error y diagnostico, la barra de potencia estara completamente apagada.

Por otro lado, se ha determinando la necesidad de incluir un modo de brillo tenue que se active a los 5 segundos de que no se produzca ninguna interaccion con la bateria, que disminuya considerablemente el brillo del Display. En el momento que se pulse cualquiera de los botones, la pantalla recupera el brillo.

La disminucion del brillo solo afecta al estado de Work y Capaciti, deminuyendo el brillo del display y la barra de potencia.

### 1.3.9 V7 Lectura Capacidad & Fix Error

Se incluye el calculo de la capacidad de la bateria partiendo de la lectura del voltaje. Esta lectura de realiza siempre cuando la salida se encuentra descoenctada para evitar que la carga afecte al voltaje, realizando la medida en vacio. Se situan los extremos en 4,2V y 3,5 como maximo y minimo respectivamente.

Propuesta de muestra de la informacion de la capacidad en 2 lugares:

- En la inicializacion.
- Si se mantiene apretado el boton central, se muestra la capacidad durante un breve tiempo antes de mostrar un mensaje de apagado. Si se mantiene apretado termina entrando en modo sleep. En el caso de soltarse, vuelve al estado previo (Stop/Run).

Se incluye la proteccion del voltage de entrada para prevenir que la bateria baje por debajo de un umbral que no asegure un correcto funcionamiento del sistema.

Caracteristicas:

- Umbral: 3300 mV.
- Tiempo: 500 ms.
- Sistema de doble pendiente.

#### 1.3.9.1 Fix Error

Se ha detectado que si no se levanta el boton central, el sistema se enciende y se apaga constantemente. Como solucion, al apagar se espera a que se levante el boton para permitir volver a pulsar y encender el sistema.

### 1.3.10 V6 Diagnostico Basico

Se decide realizar un seguimiento de ciertas variables que se consideran de interés. Para ello se utilizan 2 memorias EEPROM, una situada en la parte del chasis y otra en la parte de la batería.

En la batería se guardaran datos relacionados con la identidad y vida de la batería como son:

- Modelo.
- Número de serie.
- Número de ciclos/Tiempo de Uso.
- Estadísticas de uso en voltaje y potencia.
- Log de errores que ha podido sufrir.

En el chasis se almacena información relacionada con la configuración del chasis como son:

- Selección de los sonidos(Encendido,Run/Stop,Up,Down).
- Voltaje de funcionamiento.
- Número de serie del chasis.

La consulta de estos datos se realiza por el puerto serie, habiendo leído las memorias previamente.

Librería Usada: `diagnostic.h`

### 1.3.11 V5 (Buzzer Sounds)

Se decide añadir el módulo del control del Zumbador/Buzzer. Con este módulo se incorporan sonidos a determinados eventos. Los sonidos se seleccionarán entre alternativas en una función de inicialización, según el atributo de entrada.

Los eventos que irán acompañados de sonidos en esta integración son:

- Inicio.
- Apagado.
- Subida de Voltaje.
- Bajada de Voltaje.
- Run/Stop.

### 1.3.12 V4 (Funcionalidades Básicas Botoneras y Display)

Para clarificar la integración del display y de la botonera y se dividirá la integración en 2 pasos.

Primero se incluirá el display con las funcionalidades de: Mensaje de inicio y visualizar Voltaje.

Y luego, se incluirá la botonera con las funcionalidades de : Subir/bajar voltaje y Run/Stop.

### 1.3.12.1 V4.1 <em>Display</em>

La parte del display integrara 2 funcionalidades.

#### Funcionalidades.

- Mensaje de Inicio: Al encendido de la bateria se mostrara en la pantalla una cadena de caracteres predeterminada.
- Visualizar Voltaje: En el ciclo de control se mostrara en la pantalla el voltaje con el siguiente formato:

#### Librerias usadas:

- Display ("\*display.h\*","\*bitmaps.h\*").
- Dpad ("\*Dpad.h\*")

#### Test.

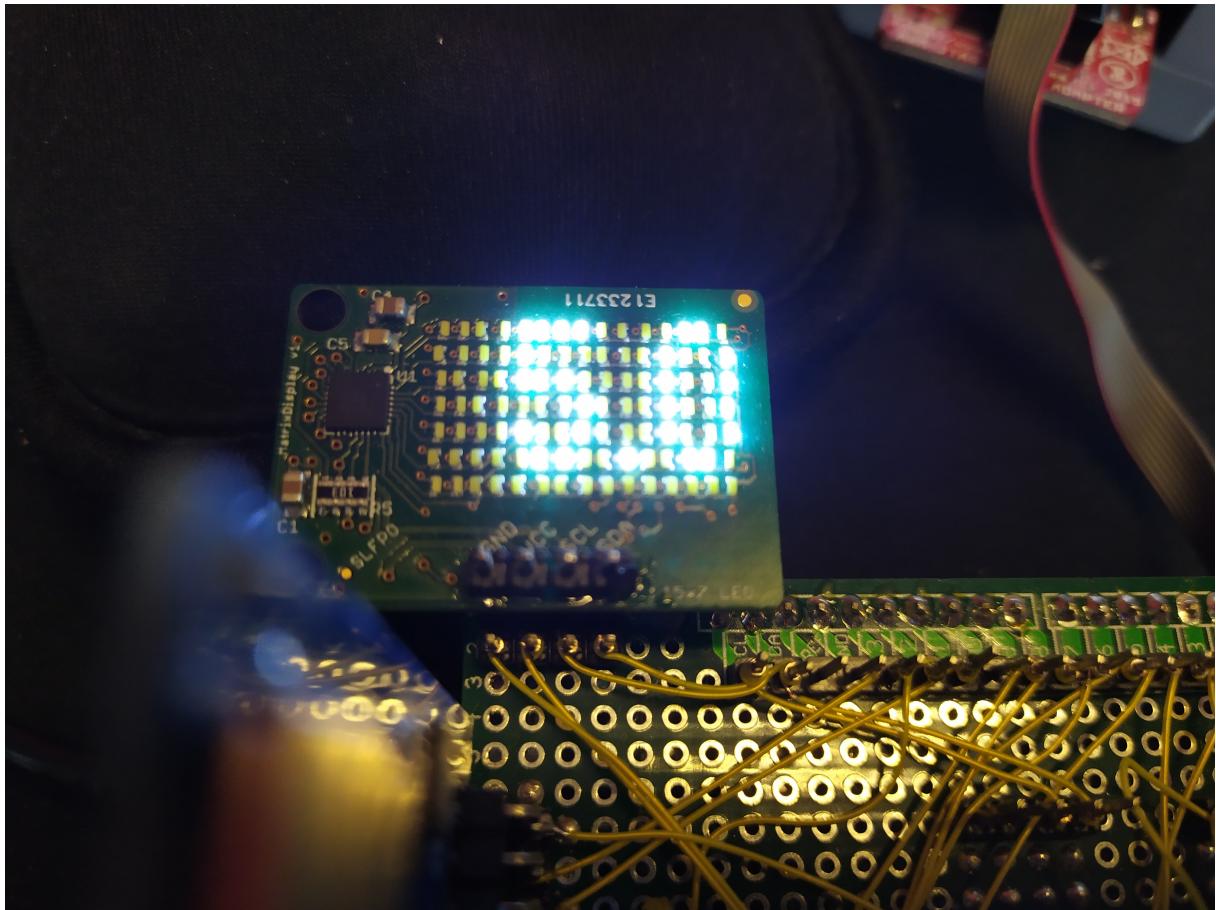


Figure 1.2 "IMAGEN NOT FOUND"

### 1.3.12.1.1 V4 <em>Display & Buttons</em> Funcionalidades.

- State Machine: Diferenciacion de distintos estados y las condiciones de cambio correspondientes:
  - Initialization: Mensaje de bienvenida y arranque.
  - Run: Voltaje por display y activacion de la salida.
  - Stop: Voltaje por Display y salida desactivada.
  - Sleep: Mensaje de despedida y bateria "apagada".
  - ERROR: Mensaje de error.
- Subida y bajada de voltaje con los botones.
- Activacion y desactavacion de la salida con los botones.

### 1.3.13 V3 (Proteccion UnderVoltage y Proteccion OverPower.)

Se crean 2 helathMonitor independientes para cada proteccion. El funcionamiento de estas protecciones son equivalentes a la proteccion de sobreconsumo.

Supuestos:

- **Voltage.** Sucederá si la máquina se queda bloqueada y el DCDC no puede mantener la salida.
- **Potencia** Considerando la Vout la teórica, no la Vo\_sense (más restrictivo).

#### Caracteristicas.

- Proteccion de UnderVoltage.
  - Umbral: 2 Voltios
  - Tiempo: 1 segundos
  - Sistema de doble pendiente.
- Proteccion de SobrePotencia.
  - Umbral: 4 Watos
  - Tiempo: 2 segundos
  - Sistema de doble pendiente.

#### Test.





### 1.3.14 V2 (Proteccion de sobreconsumo y Tiempo de Arranque)

Un Health Monitor sera el encargado de ir checkeando que el consumo no supera el limite durante un determinado tiempo. En caso de activarse, se abrirá un abanico de timepo en que se cortará la salida, tras el cual se volverá a checkear la salida. Si la proteccion salta varias veces, se considerará que la bateria se encuentra en un estado de mal funcionamiento/error. En versiones futuras se crearan las rutinas de atencion correspondientes.

Supuestos:

- Supuesto 1: la salida está a 8V, la máquina se queda bloqueada en el arranque (una máquina bloqueada suele consumir unos 500mA bloqueada).  $8V \times 500mA = 4W$ , lo que estaría en el límite de la primera protección, toda salida por debajo de 8V no haría saltar ninguna protección.
- Supuesto 2: el cargador corta y reintenta cada 60ms -> no saltaría esta protección pq la salida no llegaría a subir

- Supuesto 3: Cortocircuito al introducir el conector -> poco tiempo, no saltaría, pero tampoco queremos que salte -> ergo, es bien.

#### Caracteristicas:

- Proteccion:
  - Umbral: 450 mA.
  - Tiempo: 1500 ms.
  - Sistema de doble pendiente.
- Tiempo de Arranque: Durante el tiempo que dura un arranque (200ms aprox.) se desactivaran las comprobaciones de las protecciones.

#### Test:

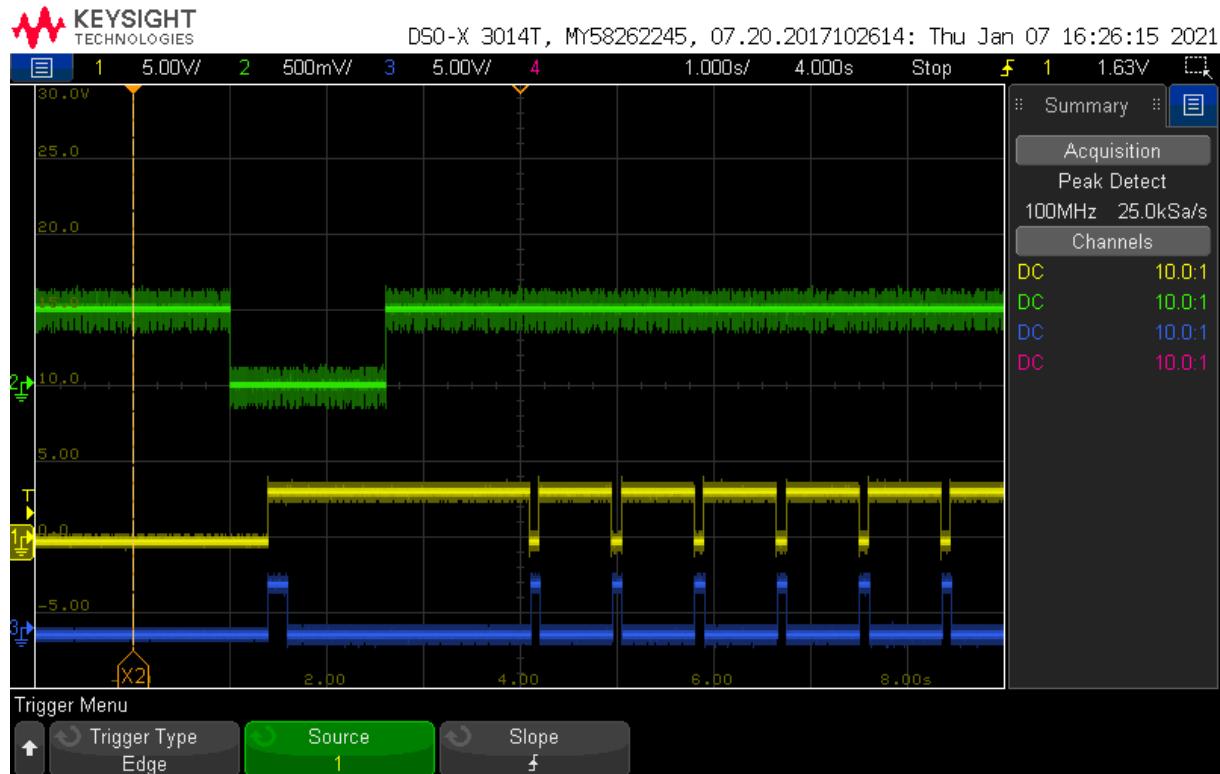


Figure 1.3 Protecion de consumo

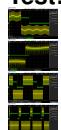
#### 1.3.15 V1

Gestion y detección del modo Boost/No boost. Cuando la corriente de salida sea superior a un umbral durante un determinado timep, se deberá activar la salida en modo BOOST, mientras que si se encuentra por debajo del umbral durante otro determinado tiempo, se deberá desactivar la salida modo BOOSt.

#### Caracteristicas:

- Umbral: 15 mA
- Tiempo Boost: 100 ms
- Tiempo No Boost: 1s.

#### Test:



## Chapter 2

# Directional PAD

Se dispone de 3 botones dispuestos 2 de ellos debajo de la pantalla y un tercero en la parte superior de la bateria.  
La funcion de estos botones sera:

- Control del voltage de salida.
- Activacion y desactivacion de la salida de voltaje.
- Apagado y encendido de la bateria.

### 2.1 Eventos posibles

- Click.
- Pulsacion larga.

### 2.2 Funcionalidades implementadas

- Click boton superior.
- Click boton Inferior.
- Click boton central.
- Pulsacion larga boton superior.
- Pulsacion larga boton inferior.
- Pulsacion larga boton central.

### 2.3 Condiciones

- Solo se permite la deteccion de un evento a la vez. Si se producen 2 o mas eventos al mismo tiempo, el resultado sera como si no se hubiera producido ningun evento. [Pendiente de Revision.]

### 2.4 Conflictos

- Durante una pulsacion larga se producen eventos en botones distintos al de la pulsacion larga. [Pendiente de Revision.]

## 2.5 Tests

### 2.5.1 Test 0

En el primer test se trata de comprobar el correcto funcionamiento de todos los botones y su funcionalidades de Click y Pulsacion larga.

Para ello, se inicializara el puerto serie, a atraves del cual podremos observar y verificar si al realizar un click o una pulsacion larga, este evento es reconocido y muestreado con un mensajoo con el correspondiente evento.

Adicionalmente, se mediraá el tiempo de computa de cada evento.

# Chapter 3

## DISPLAY LED

: Esta funcion recoge un caracter y devuelve su bitmap correspondiente.  
Display led de tamaño 15x7 leds blancos. Este display se "partira" en dos, de manerra que se quede una pantalla de 14x7 que se utilizara para la interfaz de la bateria con el uso de caracteres ASCII, y ademas quedara una "barra" de leds de 14x1 que se usara como medidor visual del consumo e potencia isntantanea de la bateria.

### 3.1 Funciones Privadas

- **Char2Bitmap**(char character, uint8\_t \*bitmap, int16\_t array\_len):

— \*

Parameters

character.

Parameters

\*bitmap.

—

Parameters

array\_len.

—

### 3.2 Funciones Publicas

- **SwitchScreenOff**

— : Apagado total de la pantalla.

- 
- **DisplayText(<String>,reset, brillo):**

— : Esta funcion muestra en el display una cadena de caracteres en un scroll lateral der-izq.No permite la introduccion de un cadena nueva hasta que no se haya terminado de imprimir la previa.

Parameters

cadena.

–

#### Parameters

<input type="checkbox"/>	reset. Permite reiniciar el muestreado por la pantalla.
--------------------------	---

–

#### Parameters

<input type="checkbox"/>	brillo. Ajusta el brillo de los leds.
--------------------------	---------------------------------------

–

#### Returns

– : bool.

- \* True: Si esta listo para imprimir una nueva cadena.
- \* False: Si se encuentra ocupado en la impresion de una cadena.

#### Tiempo de ejecucion:

- \* Not\_Busy: ~ 80us.
- \* Busy en espera: ~ 30us;
- \* Busy escribiendo: ~ 25ms.

---

- **DisplayVolt(<integer>,brillo):**

– : Esta funcion imprime la pantalla de voltaje, que muestra el voltaje al que se encuentra la salida de la bateria, con un decimal.

#### Parameters

<input type="checkbox"/>	Volts.
--------------------------	--------

–

#### Parameters

<input type="checkbox"/>	Brillo. Permite ajustar el brillo de la pantalla.
--------------------------	---

–

#### Tiempo de ejecucion:

~20ms.

---

- **PowerBar(<integer>, brillo):**

– : Esta funcion se encarga de encender y apagar los leds necesarios en la barra de potencia.

#### Parameters

<input type="checkbox"/>	Leds.
--------------------------	-------

–

#### Parameters

<input type="checkbox"/>	Brillo. Permite ajustar el brillo de los leds.
--------------------------	--

–

#### Tiempo de ejecucion:

~2,6ms.

---

- **DisplayCap(<integer>,brillo):**

- : Esta funcion muestra la pantalla de porcentaje de capacidad de bateria. si la capacidad es 100% muestra la pantalla FULL.

Parameters

<input type="checkbox"/>	capacity.
--------------------------	-----------

–

Parameters

<input type="checkbox"/>	Brillo. Permite ajustar el brillo de los leds.
--------------------------	--

- **Tiempo de ejecucion:** ~20ms.
- 

- **DisplayBattCharging(<integer>):**

- : Esta funcion muestra el marco de una bateria, se encarga de ir rellenando de manera solidia la capacidad recibida, y deja parpadeando la que seria el proximo 10% que se encuentra cargando.

Parameters

<input type="checkbox"/>	Capacity.
--------------------------	-----------

- **Tiempo de ejecucion:** 7-16 ms.

- **LedWork(state,brillo):**

- Funcion que permite encender y apagar el boton que representa el estado de la salida del voltage.

Parameters

<input type="checkbox"/>	state_led
--------------------------	-----------

–

Parameters

<input type="checkbox"/>	mode_bright
--------------------------	-------------

–

- **DisplayUsbIn:**

- Animacion de Entrada del USB. A los 2 segundos disminuye su brillo.

Parameters

<input type="checkbox"/>	ledmatrix
--------------------------	-----------

–

- **DisplayUsbOut:**

- Anuimacion de salida del USB.

#### Parameters

<code>ledmatrix</code>	<input type="text"/>
------------------------	----------------------

–

- **Display Diagnostic Mode ():**

- : Animacion del modo diagnostico.

### 3.3 TESTs

#### 3.3.1 TEST 0

Testeo de la funcion DisplayTest. PArar ponerla a prueba se lanzara funcion con distintas cadenas de texto. El resultado debe ser la impresion de cad cadena de manera consecutiva sin ningun tipo de solapamiento.

#### 3.3.2 TEST 1

Test de la funcion DisplayVolts. Este test imprime todos los posibles valores de voltaje de manera ascendente y despues de manera descendente.

#### 3.3.3 Test 2

Este test simula el funcionamiento de la barra de potencia creando un movimeinto senoidal de manera ciclica.

#### 3.3.4 Test 3

Este test pone a prueba la compatibilidad y convivencia de la barra de potencia y el display de informacion, en este caso mostrando cambios de voltaje.

#### 3.3.5 Test 4

Test de la funcion DisplayCap. Este test imprime toda la capacidad posible de la bateria ( 0-100 ). Ascendente y descendente, de manera ciclica.

#### 3.3.6 Test 5

Test de la funcion DisplayBattCharging. Este test muestra la evolucion de la carga de la bateria, desde el 0% al 100%.

#### 3.3.7 Test 6

Testeo de la funcion DisplayTest mostrando todos los caracteres imprimibles posibles.

#### 3.3.8 Test 7

Testeo del boton que marca el estado de la salida de voltage que forma parte de la pantalla.

#### 3.3.9 Test 8

Test que prueba las animaciones de entrada y salida de un USB. En el caso de la entrada pasados 2 segundos el brillo disminuye.

## Chapter 4

# BUZZER

Buzzer o Zumbador que permite realizar breves sonidos/effectos. el control de este componente se realizara con la libreria de arduino tone.

Esta libreria utiliza los timers internos de la CPU para generar una señal PWM a traves del Pin seleccionado. Esta señal se puede modular en frecuencia (generando las distintas notas) y duracion.

```
tone(PIN, FREQ, [DURATION])
```

Para generar una melodía o sonido, es necesario controlar el tiempo que dura la señal, ya que esta sentencia no es bloqueante y por lo tanto si se ejecuta otra sentencia de tono sobre el mismo pin se sobreescibiría y se cortaría la señal previa. Lo normal es la utilización de delay(), ya que se suelen reproducir de manera continuada las melodías. En este caso se utilizará la librería millis() y poder realizar otras tareas mientras se reproducen los sonidos.

### 4.1 Funciones

- **playSound(int):** Esta función reproduce el sonido indicado por el argumento.
- **getSound(array,int):** Función que selecciona qué array de notas se devuelve para tocar.(Brazo del tocacícos).
- **InitBuzzer(mode):** Inicialización del zumbador y selección de las pistas correspondientes para cada sonido.

### 4.2 TEST

- Test 0

Este test prueba todos los sonidos. por defecto.

El siguiente diagrama ejemplifica cómo se hace la reproducción de sonidos de manera no bloqueante:

```

```



# Chapter 5

## DCDC

LA librería del DCDC alberga la clase [\*dcdc\\_controller\*](#). Esta clase permite inicializar un objeto para controlar el TPIC que sirve de decodificador para fijar el voltaje de salida del DCDC controller.

### 5.1 Contantes

- Dirección I2C (ADDR\_I2C\_DCDC) = 0
- BOOST MODE = True
- NO BOOST MODE = False
- BOOST ARRAY: Conjunto de valores del decodificador (TPIC) para dar una salida de 0.4v superior al valor demandado. Ej. Teorico: 4v -> Salida: 4.4v
- NO BOOST ARRAY: Conjunto de Valores de decodificador (TPIC) para dar una salida de igual valor que el valor demandado. Ej. Teorico: 4v -> Salida: 4v

### 5.2 Metodos

- Constructor: [\*dcdc\\_controller\(pin enable\)\*](#).
- SetVoltage(voltaje, modo):
  - Modo Boost <true> El voltaje de salida es 0.4 V mayor que el que se muestra.
  - Modo No Boost <false> El voltaje de salida es exactamente el que se muestra.
- EnableDCDC (Bool): Activar o desactiva el enable.

### 5.3 TESTs

#### 5.3.1 Test 0

En este test se comprueba que el array de los comandos a decodificar por el TPIC se corresponden correctamente con los voltajes.

Para ello se utiliza la placa del makinator, donde aparte del DCDC y del TPIC, se encuentran 3 pines de sensado: Corriente a la entrada (I\_batt), Corriente a la salida (I\_OUT) y voltaje de salida (V\_out). Por ultimo se utiliza la galga de corriente como método de validación de esa corriente de entrada.

La salida de este test se trata de un conjunto de gráficas que muestran:



# Chapter 6

## Heath Monitor

Esta libreria trata la clase *HealthMonitor*. Esta clase permita la monitorizacion o segunto de una señal, permitiendo lanzar una alarma si dicha señal se mantiene por encima de un umbral durante un determinado tiempo, mediante la implementacion del algotimo de *pediente de subida pendiente de bajada*.

### 6.1 Metodos

- Constructor. ***HealthMonitor( Umbral, valor de incremento, valor de decremento, limite del contador)***

Para la programacion de estos obetos es necesario tener en cuenta la siguiente formula de la que se puede desgranar cada uno de las variables. Para ello hay que definir dos constantes previas:

- Tiempo de muestreo.
- Tiempo tras el que salta la alarma.

Con ello podemos aplicar la siguiente formula:

Limite del contador = valor de incremento \* (Tiempo de alarma/Tiempo de muestreo)

Jugando con estos valores se obtiene el limite del contador.

- ***setCounter(valor)***: Cambia el contador a un valor deseado. Tamaño 16 bits.
- ***getCounter()***: Devuevle el valor del contador.
- ***Check( sample )***: Funcion principal que compar ala muestra con el umbral y aumenta o decrementa el contador en funcion de si lo sobrepasa o no. Ademas comparará el contador con el limite y devovera una alarma si este es superado.
- ***getSample( ADCpin )***: Devuelve una muestra obtenida del ADC pin indicado.

---

### 6.2 Test

#### 6.2.1 Test 0

Este test inicializa un health monitor. El ciclo del programa se encargara de extraer una muestra del canal A1. Checkea la muestra y si salta la alarma se lanza un pulso de 10 ms por el pin A2. Paralelamente, por el puerto de salida analogica se esta muestreando el contador del health monitor.

- Para la creacion de la señal a muestrear, se utiliza la herramienta de LabView para la generacion de esta señal. El sketch mostrara un panel de control para modificar 2 señales senosoidales, 2 pulsos y el ruido. de menera que se imita la señal de la corriente, la frecuencia de la pulsacion deuna mano utilizando la maquina y los pulsos posibles errores de la señal que deberian lanzar la alarma del health monitor.

Esta señal, se almacena en un excel. El cual se utilizara para la generacion de la señal con la yuda del Analog Discovery.

- WaveForm. Dentro del programa en la parte de WaveGen se selecciona Custom e Import.

Seleccionando el excel previamente generado y configurando el separador y el simbolo decimal.

Una vez generada la señal y con las sondas del osciloscopio colocadas correctamente:

Corroboramos el funcionamiento:

- Canal 1: Pin A0 (Salida Analogica del ardunion con el "valor" del contador).
- Canal 2: Señal muestrada del wavegen.
- Canal digital 7: Pin A1 (Salida digital del arduino con el pulso de alarma.)

# Chapter 7

## Power Bar

Actualiza la cantidad de leds encendidos en la barra de potencia segun el parametro de entrada "power\_sample". Esta libreria contiene las funciones y constantes necesarias para controlar la ultima fila de leds de matriz de leds de 15x7, con el funcion de reproducir el funcionamiento de una barra de potencia. Esta barra iluminara mas leds o menos en funcion de la potencia que se este suministrando por parte de la bateria a la maquina.

### 7.1 Funciones

- [UpdatePowerBar\(\)](#)
- 

Parameters

<i>power_sample</i>	<input type="text"/>
---------------------	----------------------

•

Parameters

<i>ledmatrix</i>	<input type="text"/>
------------------	----------------------

•

Parameters

<i>mode_bright_display</i>	<input type="text"/>
----------------------------	----------------------

•

### 7.2 TESTs

- Test 0

Test para probar la actualizacion de la barra de potencia. Oscila los valores entre el maximo y el minimo de potencia que puede mostrar la barra.



# Chapter 8

## Arduino LowPower

Funcion principal de esta libreria. Comanda a la CPU a un estado de Stanby (Bajo consumo) hasta que una interrupcion configurada para ello, levante la levante/desperte.

La libreria Arduino LowPower permite configurar nuestra CPU para que entre en modos de bajo consumo, ideal para el proyecto cuando la bateria se encuentre "apagada".

### 8.1 Funciones

- **sleep():**
- 
- **attachInterruptWakeup():**
- Activamos la posibilidad de disparo de una interrupcion hardware en un determinado pin ante una determinada condicion (Rise Edge, Fall Edge, High state, Low State).

Parameters

<i>pin</i>	<input type="text"/>
------------	----------------------

•

Parameters

<i>callback</i>	<input type="text"/>
-----------------	----------------------

•

Parameters

<i>mode</i>	<input type="text"/>
-------------	----------------------

•

### 8.2 TESTs

- ExternalWakeUp.

This sketch demonstrates the usage of External Interrupts (on pins) to wakeup a chip in sleep mode. Sleep modes allow a significant drop in the power usage of a board while it does nothing waiting for an event to happen. Battery powered application can take advantage of these modes to enhance battery life significantly.

In this sketch, shorting pin 8 to a GND will wake up the board. Please note that, if the processor is sleeping, a new sketch can't be uploaded. To overcome this, manually reset the board (usually with a single or double tap to the RESET button)

This example code is in the public domain.

# Chapter 9

## Logging Tech Note

Inicializacion del valor de los elementos de diagnostico. Lectura y comprobacion de las memorias EEPROM, tanto la del chasis como la de la bateria. Cada elemento posee el valor, el nombre y la categoria:

### 9.1 Arquitectura de la memoria del SAMD21G18

La memoria de un MCU SAM D21 está mapeada en cuatro secciones:

- Memoria Flash
- Memoria Flash con capacidad de lectura y escritura (RWW)
- RAM interna
- Periféricos y puertos de E/S



En nuestro caso usaremos el MCU SAMD21G18. Por lo tanto tendremos una memoria de 256Kbytes.

La memoria de programa esta dividida en columnas (*rows*) y páginas (*pages*). Cada columna contiene 4 páginas. El tamaño de estas páginas están predefinidas a 64 bytes, a un tamaño de 32 bits (4 bytes) por palabra, tenemos que en cada página hay 16 variables.



Debido a este formato reducimos las variables útiles a 15.

Debido a la falta de una sección de memoria no volátil para el usuario, es posible configurar una zona de emulación de memoria EEPROM al final del stack de la memoria flash.

Este espacio puede ser de los siguientes tamaños:



En nuestro caso haremos uso de una memoria emulada de 256 bytes. Lo que se traduce en 1 columna de 4 páginas, un total de  $4 \times 64$  bytes = 256 bytes / 4 bytes per word - 1 word per header per word = 60 variables.

Estas variables se almacenarán en la parte no volátil de la memoria. Sin embargo esta memoria tiene una vida útil limitada, por lo que es necesario que se controle muy bien el momento de la escritura.

#### 9.1.1 Memoria Externa I2C

El acceso a la memoria externa situada en la batería, se realiza por I2C utilizando la técnica de BitBang debido a la compartición de los canales de I2C con el puerto USB y el pulsador. Por esta razón es aconsejable separar la comunicación de los puertos por defecto.

## 9.2 Funciones Privadas

- `Init_diagnostic_elements()`
- \* Serial Port (X o S). Con X no se muestra con S si.

### 9.3 \* Guardado en EEPROM (X,B,C). X no se guarda, B se guarda en la Bateria y C se guarda en el Chasis.

- **LogDiagnosticData** ( <integer> , <integer> )
  - Actualizar el valor en Ram de una variable de adiagnostico

Parameters

<i>data</i>	Valor del dato.
-------------	-----------------

–

### 9.4 \* @param address Direccion en la memoria RAM de la variable.

- **ReadDiagnosticData(Address)**
  - : Lectura del valor de un elemento de diagnostico.

Parameters

<i>Address</i> .	Direccion de almacenamiento en el array de elementos en RAM.
------------------	--

–

### 9.5 \* @return: valor del elemento de diagnostico.

- **IncrementDiagnosticdata** (<integer>,<integer>)
  - : Incrementa en una cantidad el valor de un dato de diagnostico.

### 9.6 \* @param: Incremento, address.

- **PrintDiagnosticData** (<integer>)
  - : Muestra por el puerto serie o guarda en EEPROM los datos de diagnostico.

### 9.7 \* @param: type\_print.

- **PrintStaticData**

### 9.8 \* @brief: Impresion por el puerto serie de los datos estaticos.

- **isValid()**
    - : Devuelve un booleano indicando si la memoria EEPROM contiene datos o esta vacia.
- Returns
- : bool
    - \* True: La memoria contiene valores.

### 9.9 \* False: La memoria esta vacia.

- **Stats(int,int)**

- : Incremento de las estadisticas del valor de voltaje y potencia usado. Se incremeneta en 1 el valor de la posicion de un array correspondiente. Ej: 12.3 v -> voltages\_values[ 123 ] ++

Parameters

<input type="text"/>	volts
----------------------	-------

–

## 9.10 \* @param: wats

- **UpdateEepromBatery():**

## 9.11 \* @brief: Actualizacion de la memoria EEPROM de la Bateria.

**Antes de relaizar el guardado se comprueba que el valor a guardar sea distinto del valor ya almacenado.**

- **PrintStats():**

## 9.12 \* @brief: Muestreo por el puerto serie de los valores acumulados de las estadisticas de potencia y voltage. Ademas de algunos datos estaticos guardados en eeprom como el numero de serio o el modelo.

- **ResetBateriaEeprom():**

- : Puesta a 0 de las primeras 1024 posiciones de la EEPROM de la Bateria.

## 9.13 TESTs

### 9.13.1 TEST\_0

Testeo de las funciones de LogDiagnosticData, PrintHeader and PrintDiagnosticData (Serial Port).

### 9.13.2 TEST\_1

Testeo de la funcionalidad de guardado en EEMPROM. Prueba de las funciones [isValid\(\)](#), [Incrementeent Diagnostic← Data](#).

### 9.13.3 TEST\_2

Test del guardado y lectura de la eeprom chasis. Para ello se utiliza y prueba la funcion de estadisticas.

### 9.13.4 TEST\_3

Test que permite medir la velocidad de escritura de 1, 10 y 20 bytes, a traves de la libreria de BitBang I2c.



# Chapter 10

## FlashStorage library for Arduino

The FlashStorage library aims to provide a convenient way to store and retrieve user's data using the non-volatile flash memory of microcontrollers.

The flash memory, due to it's properties, is generally used to store the firmware code, but it can also be used to store user data.

### 10.1 Supported hardware

Currently, ATSAMD21 and ATSAMD51 cpu are supported (and consequently every board based on this cpu like the Arduino Zero or Aduino MKR1000).

### 10.2 Limited number of writes

Flash memory has a limited amount of write cycles. Typical flash memory can perform about 10000 writes cycles to the same flash block before starting to "wear out" and begin to lose the ability to retain data.

**So BEWARE: IMPROPER USE OF THIS LIBRARY CAN QUICKLY AND PERMANENTLY DESTROY THE FLASH MEMORY OF YOUR MICRO,** in particular you should avoid to call the `write()` function too often and make sure that in the entire life of the micro the number of calls to `write` stay well below the above limit of 10000 (it's a good rule-of-thumb to keep that number in mind even if the manufacturer of the micro guarantees a bigger number of cycles).

The same caution must be taken if you're using the EEPROM API emulation (see below) with the `EEPROM.commit()` function.

### 10.3 Usage

First of all you must declare a global `FlashStorage` object for each piece of data you intend to store in the flash memory. For example if you want to store the age of a person you must declare an `age_storage` like this:

```
{c++}
FlashStorage(age_storage, int);
```

this instruction means "create a `FlashStorage` to store an `int` variable and call it `age\_storage`". Now you can use `age_storage` as a place to safely store an integer:

```
{c++}
void readAndStoreUserAge() {
    Serial.println("Please enter your age:");
    String age = Serial.readStringUntil('\n');
    age_storage.write(age.toInt()); // <-- save the age
}
```

after a reset of the microcontroller you can retrieve the stored age by using:

```
{c++}
int user_age = age_storage.read();
```

#### 10.3.1 Using the alternative EEPROM-like API

If you include `FlashAsEEPROM.h` you'll get an EEPROM emulation with the internal flash memory. See `EmulateEEPROM` sketch for an example.

The API is very similar to the well known `Arduino EEPROM.h` API but with two additional functions:

- EEPROM.isValid() returns `true` if data in the EEPROM is valid or, in other words, if the data has been written at least once, otherwise EEPROM data is "undefined" and the function returns `false`.
- EEPROM.commit() store the EEPROM data in flash. Use this with care: Every call writes the complete EEPROM data to flash. This will reduce the remaining flash-write-cycles. Don't call this method in a loop or you will kill your flash soon.

## 10.4 License

This library is released under LGPL-2.1.

## 10.5 FAQ

### 10.5.1 Can I use a single FlashStorage object to store more stuff?

Yes, you can declare a `struct` with more fields and create a `FlashStorage` object to store the entire structure. See the [StoreNameAndSurname](#) sketch for an example on how to do it.

### 10.5.2 The content of the FlashStorage is erased each time a new sketch is uploaded?

Yes, every time you upload a new sketch, the previous content of the FlashStorage is erased and filled with 0's. The FlashStorage library does not allow to set another default value.

### 10.5.3 Do you recommend to use FLASH instead of EEPROM?

No. If your micro provides an EEPROM it's almost always better to use that because it's a kind of memory designed with the specific purpose to store user data (it has a longer lifetime, number of write cycles, etc...).

In the absence of an EEPROM you can use this library to use a piece of the flash memory as an alternative to EEPROM. However, you must always keep in mind of its limits.

# Chapter 11

## Hierarchical Index

### 11.1 Class Hierarchy

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

Adafruit_GFX_Button . . . . .	65
ArduinoLowPowerClass . . . . .	75
dcdc_controller . . . . .	77
EEPROM_EMULATION . . . . .	79
EEPROMClass . . . . .	79
Element . . . . .	81
FlashClass . . . . .	82
FlashStorageClass< T > . . . . .	83
GFXfont . . . . .	97
GFXglyph . . . . .	99
HealthMonitor . . . . .	100
MilliTimer . . . . .	103
Person . . . . .	104
Print	
Adafruit_GFX . . . . .	39
Adafruit_IS31FL3731 . . . . .	69
Adafruit_IS31FL3731_Wing . . . . .	74
GFXcanvas1 . . . . .	84
GFXcanvas16 . . . . .	89
GFXcanvas8 . . . . .	93
RTCZero . . . . .	104
SlowSoftI2CMaster . . . . .	111



# Chapter 12

## Class Index

### 12.1 Class List

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

<a href="#">Adafruit_GFX</a> . . . . .	39
<a href="#">Adafruit_GFX_Button</a>	
A simple drawn button UI element . . . . .	65
<a href="#">Adafruit_IS31FL3731</a>	
Constructor for generic IS31FL3731 breakout version . . . . .	69
<a href="#">Adafruit_IS31FL3731_Wing</a>	
Constructor for FeatherWing IS31FL3731 version . . . . .	74
<a href="#">ArduinoLowPowerClass</a> . . . . .	75
<a href="#">dcdc_controller</a> . . . . .	77
<a href="#">EEPROM_EMULATION</a> . . . . .	79
<a href="#">EEPROMClass</a> . . . . .	79
<a href="#">Element</a> . . . . .	81
<a href="#">FlashClass</a> . . . . .	82
<a href="#">FlashStorageClass&lt; T &gt;</a> . . . . .	83
<a href="#">GFXcanvas1</a>	
A GFX 1-bit canvas context for graphics . . . . .	84
<a href="#">GFXcanvas16</a>	
A GFX 16-bit canvas context for graphics . . . . .	89
<a href="#">GFXcanvas8</a>	
A GFX 8-bit canvas context for graphics . . . . .	93
<a href="#">GFXfont</a>	
Data stored for FONT AS A WHOLE . . . . .	97
<a href="#">GFXglyph</a>	
Font data stored PER GLYPH . . . . .	99
<a href="#">HealthMonitor</a> . . . . .	100
<a href="#">MilliTimer</a> . . . . .	103
<a href="#">Person</a> . . . . .	104
<a href="#">RTCZero</a> . . . . .	104
<a href="#">SlowSoftI2CMaster</a> . . . . .	111



# Chapter 13

## File Index

### 13.1 File List

Here is a list of all files with brief descriptions:

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/CODE/B1/ <a href="#">B1.ino</a>	115
V1: - Gestión y detección del modo Boost/No boost. Cuando la corriente de salida sea superior a un umbral durante un determinado timep, se deberá activar la salida en modo BOOST, mientras que si se encuentra por debajo del umbral durante otro determinado tiempo, se deberá desactivar la salida modo BOOST . . . . .	115
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0000-EXAMPLE/ <a href="#">librarie.h</a>	161
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0001-BUTTONS/ <a href="#">Dpad.h</a>	162
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0001-BUTTONS/examples/TEST_0/ <a href="#">TEST_0.ino</a>	167
Este test comprobaba si se detectan los eventos de del DPAD . . . . .	167
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/ <a href="#">bitmaps.h</a>	181
Conjunto de Bitmaps, funciones y constantes relacionadas con el display Musotoku 15x7 . . .	181
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/ <a href="#">display.h</a>	221
Librería MUSOTOKU para display 15x7 . . . . .	221
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_0/ <a href="#">TEST_0.ino</a>	169
Testeo de la función DisplayTest . . . . .	169
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_1/ <a href="#">TEST_1.ino</a>	233
Test de la función DisplayVolts . . . . .	233
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_2/ <a href="#">TEST_2.ino</a>	238
Este test simula el funcionamiento de la barra de potencia creando un movimiento senoidal de manera cíclica . . . . .	238
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_3/ <a href="#">TEST_3.ino</a>	241
Este test pone a prueba la compatibilidad y convivencia de la barra de potencia y el display de información, en este caso mostrando cambios de voltaje . . . . .	241
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_4/ <a href="#">TEST_4.ino</a>	244
Este test imprime toda la capacidad posible de la batería ( 0-100 ). Ascendente y descendente, de manera cíclica . . . . .	244

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_5/TEST_5.ino	246
Test de la funcion DisplayBattCharging. Este test muestra la evolucion de la carga de la bateria, desde el 0% al 100%	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_6/TEST_6.ino	248
Testeo de la funcion DisplayTest mostrando todos los caracteres imprimibles posibles	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_7/TEST_7.ino	250
Testeo del boton que marca el estado de la salida de voltage que forma parte de la pantalla	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0002-DISPLAY/examples/TEST_8/TEST_8.ino	251
Test de la funcion DisplayUsbIn y DisplayUsbOut	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0003-BUZZER/Buzzer.h	253
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0003-BUZZER/notes.h	264
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0003-BUZZER/examples/TEST_0/TEST_0.ino	171
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0004-DCDC/DCDC.h	279
Libreria que describe la clase del DCDC	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0004-DCDC/exmples/TEST_0/TEST_0.ino	173
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0004-DCDC/exmples/TEST_1/TEST_1.ino	235
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0005-SENSING/HealthMonitor.h	282
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0005-SENSING/examples/TEST_0/TEST_0.ino	176
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0006-POWERBAR/power_bar.h	284
Esta libreria contiene las funciones y cosntantes encesarias para controlar la ultima fila de leds de matriz de leds de 15x7, con el funcion de reproducir el funcionamiento de una barra de potencia. Esta barra iluminara mas leds o menos en funcion de la potencia que se este suminsitrandio por parte de la bateria a la maquina	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0006-POWERBAR/examples/TEST_0/TEST_0.ino	178
Test para probar la actualizacion de la barra de potencia. Oscila los valores entre el maximo y el minimo de potencia que puede mostrar la barra	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0007-SlowSoftI2CMaster/batt_SlowSoftI2CMaster.cpp	286
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0007-SlowSoftI2CMaster/batt_SlowSoftI2CMaster.h	288
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_Adafruit_GFX.cpp	290
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_Adafruit_GFX.h	315
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_gfxfont.h	319
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0008-Adafruit_GFX_Library/batt_glcdfont.c	319
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0009-LOWPOWER/batt_ArduinoLowPower.cpp	321
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0009-LOWPOWER/batt_ArduinoLowPower.h	325
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←_sw/LIBRARIES/0009-LOWPOWER/examples/AdcWakeup/AdcWakeup.ino	328

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0009-LOWPOWER/examples/ExternalWakeUp/ <a href="#">ExternalWakeUp.ino</a>	330
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0009-LOWPOWER/examples/PrimoDeepSleep/ <a href="#">PrimoDeepSleep.ino</a>	332
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0009-LOWPOWER/examples/TianStandby/ <a href="#">TianStandby.ino</a>	334
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0009-LOWPOWER/examples/TimedWakeUp/ <a href="#">TimedWakeup.ino</a>	336
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/ <a href="#">diagnostic.h</a>	337
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/ <a href="#">Eeprom_LUT.h</a>	346
Look Up Table de direcciones de memoria de la eeprom de la bateria	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/ <a href="#">EepromBitBang.h</a>	351
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/examples/TEST_0/ <a href="#">TEST_0.ino</a>	180
Test de las funciones de PrintHeader, PrintDiagnosticData y LogDiagnosticData	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/examples/TEST_1/ <a href="#">TEST_1.ino</a>	236
Testeo de la funcionalidad de guardado en EEMPRON tanto Chasis como Batt Pack. Prueba de las funciones <code>isValid()</code> , Increment DiagnosticData	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/examples/TEST_2/ <a href="#">TEST_2.ino</a>	239
Test del guardado y lectura de la eeprom chasis. Para ello se utiliza y prueba la función de estadísticas	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0010-LOGGING/examples/TEST_3/ <a href="#">TEST_3.ino</a>	243
Test que mide los tiempos de lectura y escritura de distintas cantidades de bytes en la memoria EEPROM del Battery Pack	
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/batt_Adafuit_IS31FL3731.cpp	354
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/batt_Adafuit_IS31FL3731.h	357
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/examples/gfxdemo/ <a href="#">gfxdemo.ino</a>	360
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/examples/manualanim/ <a href="#">manualanim.ino</a>	362
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0011-Adafruit_IS31FL3731_Library/examples/swirldemo/ <a href="#">swirldemo.ino</a>	364
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/examples/EmulateEEPROM/ <a href="#">EmulateEEPROM.ino</a>	365
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/examples/FlashStoreAndRetrieve/ <a href="#">FlashStoreAndRetrieve.ino</a>	366
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/examples/StoreNameAndSurname/ <a href="#">StoreNameAndSurname.ino</a>	368
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashAsEEPROM.cpp	369
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashAsEEPROM.h	371
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashStorage.cpp	372
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat/_sw/LIBRARIES/0012-batt_FlashStorage/src/batt_FlashStorage.h	374

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/examples/CorrectADCResponse/[CorrectADCResponse.ino](#)  
    377

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/src/[batt\\_SAMD\\_AnalogCorrection.cpp](#) . . . . . 381

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/src/[batt\\_SAMD\\_AnalogCorrection.h](#) . . . . . 382

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0014-MilliTimer/[MilliTimer.cpp](#) . . . . . 383

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0014-MilliTimer/[MilliTimer.h](#) . . . . . 384

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0015-RTCZero/examples/Epoch/[Epoch.ino](#) . . . . . 384

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0015-RTCZero/examples/SimpleRTC/[SimpleRTC.ino](#) . . . . . 386

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0015-RTCZero/examples/SimpleRTCAlarm/[SimpleRTCAlarm.ino](#) . . . . . 389

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0015-RTCZero/examples/SleepRTCAlarm/[SleepRTCAlarm.ino](#) . . . . . 391

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0015-RTCZero/src/[batt\\_RTCZero.cpp](#) . . . . . 393

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/LIBRARIES/0015-RTCZero/src/[batt\\_RTCZero.h](#) . . . . . 401

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/CPU/Validacion Diseño/ARDUINO/Test\_Shield\_Validacion\_CPU/[Test\\_Shield\\_Validacion\\_CPU.ino](#)  
    403

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/CPU/Validacion Diseño/SAMD/PCB\_Validacion\_CPU/[PCB\\_Validacion\\_CPU.ino](#) . . . . . 403

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/CPU/Validacion Produccion/ARDUINO/Test\_Production/[Test\\_Production.ino](#) . . . . . 403

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/CPU/Validacion Produccion/LABVIEW/CPU TEST/sketch\_jul14a/[sketch\\_jul14a.ino](#) 409

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/DCDC/Validacion Diseño/Arduino/Test\_Shield\_Validacion\_DCDC/[Test\\_Shield\\_Validacion\\_DCDC.ino](#)  
    413

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/DISPLAY/Validacion Diseño/Arduino/Test\_Shield\_Validacion\_Display/[Test\\_Shield\\_Validacion\\_Display.ino](#)  
    414

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat←  
  \_sw/TEST/INPUT/Validacion Diseño/Test\_Shield\_Validacion\_Input/[Test\\_Shield\\_Validacion\\_Input.ino](#)  
    414

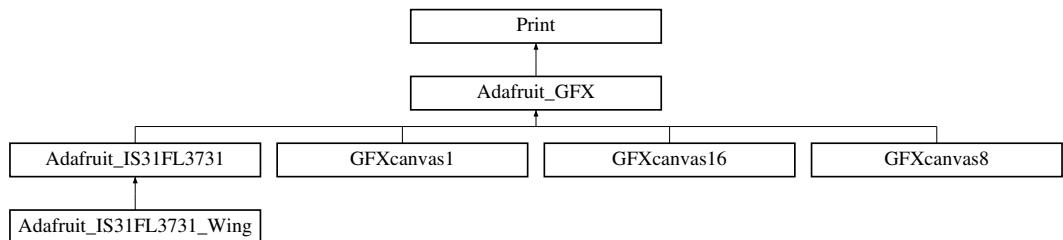
# Chapter 14

## Class Documentation

### 14.1 Adafruit\_GFX Class Reference

```
#include <batt_Adafuit_GFX.h>
```

Inheritance diagram for Adafruit\_GFX:



### Public Member Functions

- `Adafruit_GFX (int16_t w, int16_t h)`  
*Instantiate a GFX context for graphics! Can only be done by a superclass.*
- virtual void `drawPixel (int16_t x, int16_t y, uint16_t color)=0`  
*Draw to the screen/framebuffer/etc. Must be overridden in subclass.*
- virtual void `startWrite (void)`  
*Start a display-writing routine, overwrite in subclasses.*
- virtual void `writePixel (int16_t x, int16_t y, uint16_t color)`  
*Write a pixel, overwrite in subclasses if startWrite is defined!*
- virtual void `writeFillRect (int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)`  
*Write a rectangle completely with one color, overwrite in subclasses if startWrite is defined!*
- virtual void `writeFastVLine (int16_t x, int16_t y, int16_t h, uint16_t color)`  
*Write a perfectly vertical line, overwrite in subclasses if startWrite is defined!*
- virtual void `writeFastHLine (int16_t x, int16_t y, int16_t w, uint16_t color)`  
*Write a perfectly horizontal line, overwrite in subclasses if startWrite is defined!*
- virtual void `writeLine (int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color)`  
*Write a line. Bresenham's algorithm - thx wikipedia.*
- virtual void `endWrite (void)`  
*End a display-writing routine, overwrite in subclasses if startWrite is defined!*
- virtual void `setRotation (uint8_t r)`  
*Set rotation setting for display.*
- virtual void `invertDisplay (bool i)`  
*Invert the display (ideally using built-in hardware command)*
- virtual void `drawFastVLine (int16_t x, int16_t y, int16_t h, uint16_t color)`

- virtual void `drawFastHLine` (int16\_t x, int16\_t y, int16\_t w, uint16\_t color)
 

*Draw a perfectly vertical line (this is often optimized in a subclass!)*
- virtual void `fillRect` (int16\_t x, int16\_t y, int16\_t w, int16\_t h, uint16\_t color)
 

*Fill a rectangle completely with one color. Update in subclasses if desired!*
- virtual void `fillScreen` (uint16\_t color)
 

*Fill the screen completely with one color. Update in subclasses if desired!*
- virtual void `drawLine` (int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1, uint16\_t color)
 

*Draw a line.*
- virtual void `drawRect` (int16\_t x, int16\_t y, int16\_t w, int16\_t h, uint16\_t color)
 

*Draw a rectangle with no fill color.*
- void `drawCircle` (int16\_t x0, int16\_t y0, int16\_t r, uint16\_t color)
 

*Draw a circle outline.*
- void `drawCircleHelper` (int16\_t x0, int16\_t y0, int16\_t r, uint8\_t cornername, uint16\_t color)
 

*Quarter-circle drawer, used to do circles and roundrects.*
- void `fillCircle` (int16\_t x0, int16\_t y0, int16\_t r, uint16\_t color)
 

*Draw a circle with filled color.*
- void `fillCircleHelper` (int16\_t x0, int16\_t y0, int16\_t r, uint8\_t cornername, int16\_t delta, uint16\_t color)
 

*Quarter-circle drawer with fill, used for circles and roundrects.*
- void `drawTriangle` (int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1, int16\_t x2, int16\_t y2, uint16\_t color)
 

*Draw a triangle with no fill color.*
- void `fillTriangle` (int16\_t x0, int16\_t y0, int16\_t x1, int16\_t y1, int16\_t x2, int16\_t y2, uint16\_t color)
 

*Draw a triangle with color-fill.*
- void `drawRoundRect` (int16\_t x0, int16\_t y0, int16\_t w, int16\_t h, int16\_t radius, uint16\_t color)
 

*Draw a rounded rectangle with no fill color.*
- void `fillRoundRect` (int16\_t x0, int16\_t y0, int16\_t w, int16\_t h, int16\_t radius, uint16\_t color)
 

*Draw a rounded rectangle with fill color.*
- void `drawBitmap` (int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w, int16\_t h, uint16\_t color)
 

*Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent).*
- void `drawBitmap` (int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w, int16\_t h, uint16\_t color, uint16\_t bg)
 

*Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors.*
- void `drawBitmap` (int16\_t x, int16\_t y, uint8\_t \*bitmap, int16\_t w, int16\_t h, uint16\_t color)
 

*Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent).*
- void `drawBitmap` (int16\_t x, int16\_t y, uint8\_t \*bitmap, int16\_t w, int16\_t h, uint16\_t color, uint16\_t bg)
 

*Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors.*
- void `drawXBitmap` (int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w, int16\_t h, uint16\_t color)
 

*Draw PROGMEM-resident XBitMap Files (\*.xbm), exported from GIMP. Usage: Export from GIMP to \*.xbm, rename \*.xbm to \*.c and open in editor. C Array can be directly used with this function. There is no RAM-resident version of this function; if generating bitmaps in RAM, use the format defined by `drawBitmap()` and call that instead.*
- void `drawGrayscaleBitmap` (int16\_t x, int16\_t y, const uint8\_t bitmap[], int16\_t w, int16\_t h)
 

*Draw a PROGMEM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.*
- void `drawGrayscaleBitmap` (int16\_t x, int16\_t y, uint8\_t \*bitmap, int16\_t w, int16\_t h)
 

*Draw a RAM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.*
- void `drawGrayscaleBitmap` (int16\_t x, int16\_t y, const uint8\_t bitmap[], const uint8\_t mask[], int16\_t w, int16\_t h)
 

*Draw a RAM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.*

*Draw a PROGMEM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be PROGMEM-resident. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.*

- void **drawGrayscaleBitmap** (int16\_t x, int16\_t y, uint8\_t \*bitmap, uint8\_t \*mask, int16\_t w, int16\_t h)
 

*Draw a RAM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be RAM-resident, no mix-and-match. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.*
- void **drawRGBBitmap** (int16\_t x, int16\_t y, const uint16\_t bitmap[], int16\_t w, int16\_t h)
 

*Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed.*
- void **drawRGBBitmap** (int16\_t x, int16\_t y, uint16\_t \*bitmap, int16\_t w, int16\_t h)
 

*Draw a RAM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed.*
- void **drawRGBBitmap** (int16\_t x, int16\_t y, const uint16\_t bitmap[], const uint8\_t mask[], int16\_t w, int16\_t h)
 

*Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be PROGMEM-resident. For 16-bit display devices; no color reduction performed.*
- void **drawRGBBitmap** (int16\_t x, int16\_t y, uint16\_t \*bitmap, uint8\_t \*mask, int16\_t w, int16\_t h)
 

*Draw a RAM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be RAM-resident. For 16-bit display devices; no color reduction performed.*
- void **drawChar** (int16\_t x, int16\_t y, unsigned char c, uint16\_t color, uint16\_t bg, uint8\_t size)
 

*Draw a single character.*
- void **drawChar** (int16\_t x, int16\_t y, unsigned char c, uint16\_t color, uint16\_t bg, uint8\_t size\_x, uint8\_t size\_y)
 

*Draw a single character.*
- void **getTextBounds** (const char \*string, int16\_t x, int16\_t y, int16\_t \*x1, int16\_t \*y1, uint16\_t \*w, uint16\_t \*h)
 

*Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H.*
- void **getTextBounds** (const \_\_FlashStringHelper \*s, int16\_t x, int16\_t y, int16\_t \*x1, int16\_t \*y1, uint16\_t \*w, uint16\_t \*h)
 

*Helper to determine size of a PROGMEM string with current font/size. Pass string and a cursor position, returns UL corner and W,H.*
- void **getTextBounds** (const String &str, int16\_t x, int16\_t y, int16\_t \*x1, int16\_t \*y1, uint16\_t \*w, uint16\_t \*h)
 

*Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H.*
- void **setTextSize** (uint8\_t s)
 

*Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger.*
- void **setTextSize** (uint8\_t sx, uint8\_t sy)
 

*Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger.*
- void **setFont** (const GFXfont \*f=NULL)
 

*Set the font to display when print()ing, either custom or default.*
- void **setCursor** (int16\_t x, int16\_t y)
 

*Set text cursor location.*
- void **setTextColor** (uint16\_t c)
 

*Set text font color with transparent background.*
- void **setTextColor** (uint16\_t c, uint16\_t bg)
 

*Set text font color with custom background color.*
- void **setTextWrap** (bool w)
 

*Set whether text that is too long for the screen width should automatically wrap around to the next line (else clip right).*
- void **cp437** (bool x=true)
 

*Enable (or disable) Code Page 437-compatible charset. There was an error in glcdfont.c for the longest time – one character (#176, the 'light shade' block) was missing – this threw off the index of every character that followed it. But a TON of code has been written with the erroneous character indices. By default, the library uses the original 'wrong' behavior and old sketches will still work. Pass 'true' to this function to use correct CP437 character values in your code.*

- virtual size\_t `write` (uint8\_t)
 

*Print one byte/character of data, used to support print()*
- int16\_t `width` (void) const
 

*Get width of the display, accounting for current rotation.*
- int16\_t `height` (void) const
 

*Get height of the display, accounting for current rotation.*
- uint8\_t `getRotation` (void) const
 

*Get rotation setting for display.*
- int16\_t `getCursorX` (void) const
 

*Get text cursor X location.*
- int16\_t `getCursorY` (void) const
 

*Get text cursor Y location.*

## Protected Member Functions

- void `charBounds` (unsigned char c, int16\_t \*x, int16\_t \*y, int16\_t \*minx, int16\_t \*miny, int16\_t \*maxx, int16\_t \*maxy)
 

*Helper to determine size of a character with current font/size. Broke this out as it's used by both the PROGMEM- and RAM-resident `getTextBounds()` functions.*

## Protected Attributes

- int16\_t `WIDTH`

*This is the 'raw' display width - never changes.*
- int16\_t `HEIGHT`

*This is the 'raw' display height - never changes.*
- int16\_t `_width`

*Display width as modified by current rotation.*
- int16\_t `_height`

*Display height as modified by current rotation.*
- int16\_t `cursor_x`

*x location to start print()ing text*
- int16\_t `cursor_y`

*y location to start print()ing text*
- uint16\_t `textcolor`

*16-bit background color for print()*
- uint16\_t `textbgcolor`

*16-bit text color for print()*
- uint8\_t `textsize_x`

*Desired magnification in X-axis of text to print()*
- uint8\_t `textsize_y`

*Desired magnification in Y-axis of text to print()*
- uint8\_t `rotation`

*Display rotation (0 thru 3)*
- bool `wrap`

*If set, 'wrap' text at right edge of display.*
- bool `_cp437`

*If set, use correct CP437 charset (default is off)*
- GFXfont \* `gfxFont`

*Pointer to special font.*

### 14.1.1 Detailed Description

A generic graphics superclass that can handle all sorts of drawing. At a minimum you can subclass and provide [drawPixel\(\)](#). At a maximum you can do a ton of overriding to optimize. Used for any/all Adafruit displays!  
Definition at line 15 of file [batt\\_Adafruit\\_GFX.h](#).

### 14.1.2 Constructor & Destructor Documentation

#### 14.1.2.1 Adafruit\_GFX()

```
Adafruit_GFX::Adafruit_GFX (
    int16_t w,
    int16_t h )
```

Instantiate a GFX context for graphics! Can only be done by a superclass.

#### Parameters

w	Display width, in pixels
h	Display height, in pixels

Definition at line 110 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.1.3 Member Function Documentation

#### 14.1.3.1 charBounds()

```
void Adafruit_GFX::charBounds (
    unsigned char c,
    int16_t * x,
    int16_t * y,
    int16_t * minx,
    int16_t * miny,
    int16_t * maxx,
    int16_t * maxy ) [protected]
```

Helper to determine size of a character with current font/size. Broke this out as it's used by both the PROGMEM- and RAM-resident [getTextBounds\(\)](#) functions.

#### Parameters

c	The ASCII character in question
x	Pointer to x location of character. Value is modified by this function to advance to next character.
y	Pointer to y location of character. Value is modified by this function to advance to next character.
minx	Pointer to minimum X coordinate, passed in to AND returned by this function – this is used to incrementally build a bounding rectangle for a string.
miny	Pointer to minimum Y coord, passed in AND returned.
maxx	Pointer to maximum X coord, passed in AND returned.
maxy	Pointer to maximum Y coord, passed in AND returned.

Definition at line 1371 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.1.3.2 cp437()

```
void Adafruit_GFX::cp437 (
    bool x = true ) [inline]
```

Enable (or disable) Code Page 437-compatible charset. There was an error in glcdfont.c for the longest time – one character (#176, the 'light shade' block) was missing – this threw off the index of every character that followed it. But a TON of code has been written with the erroneous character indices. By default, the library uses the original 'wrong' behavior and old sketches will still work. Pass 'true' to this function to use correct CP437 character values in your code.

#### Parameters

x	true = enable (new behavior), false = disable (old behavior)
---	--

Definition at line 178 of file [batt\\_Adafruit\\_GFX.h](#).

### 14.1.3.3 drawBitmap() [1/4]

```
void Adafruit_GFX::drawBitmap (
    int16_t x,
    int16_t y,
    const uint8_t bitmap[],
    int16_t w,
    int16_t h,
    uint16_t color )
```

Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent).

#### Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate
bitmap	byte array with monochrome bitmap
w	Width of bitmap in pixels
h	Height of bitmap in pixels
color	16-bit 5-6-5 Color to draw with

Definition at line 717 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.1.3.4 drawBitmap() [2/4]

```
void Adafruit_GFX::drawBitmap (
    int16_t x,
    int16_t y,
    const uint8_t bitmap[],
    int16_t w,
    int16_t h,
    uint16_t color,
    uint16_t bg )
```

Draw a PROGMEM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors.

#### Parameters

x	Top left corner x coordinate
y	Top left corner y coordinate

**Parameters**

<i>bitmap</i>	byte array with monochrome bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels
<i>color</i>	16-bit 5-6-5 Color to draw pixels with
<i>bg</i>	16-bit 5-6-5 Color to draw background with

Definition at line 751 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.5 drawBitmap() [3/4]**

```
void Adafruit_GFX::drawBitmap (
    int16_t x,
    int16_t y,
    uint8_t * bitmap,
    int16_t w,
    int16_t h,
    uint16_t color )
```

Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground color (unset bits are transparent).

**Parameters**

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with monochrome bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 783 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.6 drawBitmap() [4/4]**

```
void Adafruit_GFX::drawBitmap (
    int16_t x,
    int16_t y,
    uint8_t * bitmap,
    int16_t w,
    int16_t h,
    uint16_t color,
    uint16_t bg )
```

Draw a RAM-resident 1-bit image at the specified (x,y) position, using the specified foreground (for set bits) and background (unset bits) colors.

**Parameters**

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with monochrome bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

**Parameters**

<i>color</i>	16-bit 5-6-5 Color to draw pixels with
<i>bg</i>	16-bit 5-6-5 Color to draw background with

Definition at line 817 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.7 drawChar() [1/2]**

```
void Adafruit_GFX::drawChar (
    int16_t x,
    int16_t y,
    unsigned char c,
    uint16_t color,
    uint16_t bg,
    uint8_t size )
```

Draw a single character.

**Parameters**

<i>x</i>	Bottom left corner x coordinate
<i>y</i>	Bottom left corner y coordinate
<i>c</i>	The 8-bit font-indexed character (likely ascii)
<i>color</i>	16-bit 5-6-5 Color to draw character with
<i>bg</i>	16-bit 5-6-5 Color to fill background with (if same as color, no background)
<i>size</i>	Font magnification level, 1 is 'original' size

Definition at line 1115 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.8 drawChar() [2/2]**

```
void Adafruit_GFX::drawChar (
    int16_t x,
    int16_t y,
    unsigned char c,
    uint16_t color,
    uint16_t bg,
    uint8_t size_x,
    uint8_t size_y )
```

Draw a single character.

**Parameters**

<i>x</i>	Bottom left corner x coordinate
<i>y</i>	Bottom left corner y coordinate
<i>c</i>	The 8-bit font-indexed character (likely ascii)
<i>color</i>	16-bit 5-6-5 Color to draw character with
<i>bg</i>	16-bit 5-6-5 Color to fill background with (if same as color, no background)
<i>size_x</i>	Font magnification level in X-axis, 1 is 'original' size
<i>size_y</i>	Font magnification level in Y-axis, 1 is 'original' size

Definition at line 1134 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.9 drawCircle()**

```
void Adafruit_GFX::drawCircle (
    int16_t x0,
    int16_t y0,
    int16_t r,
    uint16_t color )
```

Draw a circle outline.

**Parameters**

<i>x0</i>	Center-point x coordinate
<i>y0</i>	Center-point y coordinate
<i>r</i>	Radius of circle
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 357 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.10 drawCircleHelper()**

```
void Adafruit_GFX::drawCircleHelper (
    int16_t x0,
    int16_t y0,
    int16_t r,
    uint8_t cornername,
    uint16_t color )
```

Quarter-circle drawer, used to do circles and roundrects.

**Parameters**

<i>x0</i>	Center-point x coordinate
<i>y0</i>	Center-point y coordinate
<i>r</i>	Radius of circle
<i>cornername</i>	Mask bit #1 or bit #2 to indicate which quarters of the circle we're doing
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 407 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.11 drawFastHLine()**

```
void Adafruit_GFX::drawFastHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [virtual]
```

Draw a perfectly horizontal line (this is often optimized in a subclass!)

**Parameters**

<i>x</i>	Left-most x coordinate
<i>y</i>	Left-most y coordinate
<i>w</i>	Width in pixels
<i>color</i>	16-bit 5-6-5 Color to fill with

Reimplemented in [GFXcanvas1](#), [GFXcanvas8](#), and [GFXcanvas16](#).  
 Definition at line 282 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.12 drawFastVLine()

```
void Adafruit_GFX::drawFastVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [virtual]
```

Draw a perfectly vertical line (this is often optimized in a subclass!)

##### Parameters

<i>x</i>	Top-most x coordinate
<i>y</i>	Top-most y coordinate
<i>h</i>	Height in pixels
<i>color</i>	16-bit 5-6-5 Color to fill with

Reimplemented in [GFXcanvas1](#), [GFXcanvas8](#), and [GFXcanvas16](#).  
 Definition at line 265 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.13 drawGrayscaleBitmap() [1/4]

```
void Adafruit_GFX::drawGrayscaleBitmap (
    int16_t x,
    int16_t y,
    const uint8_t bitmap[],
    const uint8_t mask[],
    int16_t w,
    int16_t h )
```

Draw a PROGMEM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be PROGMEM-resident. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.

##### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with grayscale bitmap
<i>mask</i>	byte array with mask bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 935 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.14 drawGrayscaleBitmap() [2/4]

```
void Adafruit_GFX::drawGrayscaleBitmap (
    int16_t x,
    int16_t y,
    const uint8_t bitmap[],
    int16_t w,
    int16_t h )
```

Draw a PROGMEM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.

#### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with grayscale bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 885 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.15 drawGrayscaleBitmap() [3/4]

```
void Adafruit_GFX::drawGrayscaleBitmap (
    int16_t x,
    int16_t y,
    uint8_t * bitmap,
    int16_t w,
    int16_t h )
```

Draw a RAM-resident 8-bit image (grayscale) at the specified (x,y) pos. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.

#### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with grayscale bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 909 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.16 drawGrayscaleBitmap() [4/4]

```
void Adafruit_GFX::drawGrayscaleBitmap (
    int16_t x,
    int16_t y,
    uint8_t * bitmap,
    uint8_t * mask,
    int16_t w,
    int16_t h )
```

Draw a RAM-resident 8-bit image (grayscale) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (grayscale and mask) must be RAM-resident, no mix-and-match. Specifically for 8-bit display devices such as IS31FL3731; no color reduction/expansion is performed.

#### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with grayscale bitmap
<i>mask</i>	byte array with mask bitmap
<i>w</i>	Width of bitmap in pixels

**Parameters**

<i>h</i>	Height of bitmap in pixels
----------	----------------------------

Definition at line 971 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.17 drawLine()**

```
void Adafruit_GFX::drawLine (
    int16_t x0,
    int16_t y0,
    int16_t x1,
    int16_t y1,
    uint16_t color ) [virtual]
```

Draw a line.

**Parameters**

<i>x0</i>	Start point x coordinate
<i>y0</i>	Start point y coordinate
<i>x1</i>	End point x coordinate
<i>y1</i>	End point y coordinate
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 330 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.18 drawPixel()**

```
virtual void Adafruit_GFX::drawPixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [pure virtual]
```

Draw to the screen/framebuffer/etc. Must be overridden in subclass.

**Parameters**

<i>x</i>	X coordinate in pixels
<i>y</i>	Y coordinate in pixels
<i>color</i>	16-bit pixel color.

Implemented in [GFXcanvas1](#), [GFXcanvas8](#), [GFXcanvas16](#), [Adafruit\\_IS31FL3731](#), and [Adafruit\\_IS31FL3731\\_Wing](#).

**14.1.3.19 drawRect()**

```
void Adafruit_GFX::drawRect (
    int16_t x,
    int16_t y,
    int16_t w,
    int16_t h,
    uint16_t color ) [virtual]
```

Draw a rectangle with no fill color.

**Parameters**

<i>x</i>	Top left corner x coordinate
----------	------------------------------

**Parameters**

<i>y</i>	Top left corner y coordinate
<i>w</i>	Width in pixels
<i>h</i>	Height in pixels
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 523 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.20 drawRGBBitmap() [1/4]**

```
void Adafruit_GFX::drawRGBBitmap (
    int16_t x,
    int16_t y,
    const uint16_t bitmap[],
    const uint8_t mask[],
    int16_t w,
    int16_t h )
```

Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be PROGMEM-resident. For 16-bit display devices; no color reduction performed.

**Parameters**

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with 16-bit color bitmap
<i>mask</i>	byte array with monochrome mask bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 1048 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.21 drawRGBBitmap() [2/4]**

```
void Adafruit_GFX::drawRGBBitmap (
    int16_t x,
    int16_t y,
    const uint16_t bitmap[],
    int16_t w,
    int16_t h )
```

Draw a PROGMEM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed.

**Parameters**

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with 16-bit color bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 1001 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.22 drawRGBBitmap() [3/4]

```
void Adafruit_GFX::drawRGBBitmap (
    int16_t x,
    int16_t y,
    uint16_t * bitmap,
    int16_t w,
    int16_t h )
```

Draw a RAM-resident 16-bit image (RGB 5/6/5) at the specified (x,y) position. For 16-bit display devices; no color reduction performed.

##### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with 16-bit color bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 1023 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.23 drawRGBBitmap() [4/4]

```
void Adafruit_GFX::drawRGBBitmap (
    int16_t x,
    int16_t y,
    uint16_t * bitmap,
    uint8_t * mask,
    int16_t w,
    int16_t h )
```

Draw a RAM-resident 16-bit image (RGB 5/6/5) with a 1-bit mask (set bits = opaque, unset bits = clear) at the specified (x,y) position. BOTH buffers (color and mask) must be RAM-resident. For 16-bit display devices; no color reduction performed.

##### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with 16-bit color bitmap
<i>mask</i>	byte array with monochrome mask bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels

Definition at line 1081 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.24 drawRoundRect()

```
void Adafruit_GFX::drawRoundRect (
    int16_t x,
    int16_t y,
    int16_t w,
    int16_t h,
```

```
    int16_t r,
    uint16_t color )
```

Draw a rounded rectangle with no fill color.

#### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>w</i>	Width in pixels
<i>h</i>	Height in pixels
<i>r</i>	Radius of corner rounding
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 544 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.25 drawTriangle()

```
void Adafruit_GFX::drawTriangle (
    int16_t x0,
    int16_t y0,
    int16_t x1,
    int16_t y1,
    int16_t x2,
    int16_t y2,
    uint16_t color )
```

Draw a triangle with no fill color.

#### Parameters

<i>x0</i>	Vertex #0 x coordinate
<i>y0</i>	Vertex #0 y coordinate
<i>x1</i>	Vertex #1 x coordinate
<i>y1</i>	Vertex #1 y coordinate
<i>x2</i>	Vertex #2 x coordinate
<i>y2</i>	Vertex #2 y coordinate
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 600 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.26 drawXBitmap()

```
void Adafruit_GFX::drawXBitmap (
    int16_t x,
    int16_t y,
    const uint8_t bitmap[],
    int16_t w,
    int16_t h,
    uint16_t color )
```

Draw PROGMEM-resident XBitmap Files (\*.xbm), exported from GIMP. Usage: Export from GIMP to \*.xbm, rename \*.xbm to \*.c and open in editor. C Array can be directly used with this function. There is no RAM-resident version of this function; if generating bitmaps in RAM, use the format defined by [drawBitmap\(\)](#) and call that instead.

#### Parameters

<i>x</i>	Top left corner x coordinate
----------	------------------------------

**Parameters**

<i>y</i>	Top left corner y coordinate
<i>bitmap</i>	byte array with monochrome bitmap
<i>w</i>	Width of bitmap in pixels
<i>h</i>	Height of bitmap in pixels
<i>color</i>	16-bit 5-6-5 Color to draw pixels with

Definition at line 851 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.27 endWrite()**

```
void Adafruit_GFX::endWrite (
    void ) [virtual]
```

End a display-writing routine, overwrite in subclasses if startWrite is defined!

Definition at line 253 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.28 fillCircle()**

```
void Adafruit_GFX::fillCircle (
    int16_t x0,
    int16_t y0,
    int16_t r,
    uint16_t color )
```

Draw a circle with filled color.

**Parameters**

<i>x0</i>	Center-point x coordinate
<i>y0</i>	Center-point y coordinate
<i>r</i>	Radius of circle
<i>color</i>	16-bit 5-6-5 Color to fill with

Definition at line 452 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.29 fillCircleHelper()**

```
void Adafruit_GFX::fillCircleHelper (
    int16_t x0,
    int16_t y0,
    int16_t r,
    uint8_t corners,
    int16_t delta,
    uint16_t color )
```

Quarter-circle drawer with fill, used for circles and roundrects.

**Parameters**

<i>x0</i>	Center-point x coordinate
<i>y0</i>	Center-point y coordinate
<i>r</i>	Radius of circle
<i>corners</i>	Mask bits indicating which quarters we're doing
<i>delta</i>	Offset from center-point, used for round-rects
<i>color</i>	16-bit 5-6-5 Color to fill with

Definition at line 471 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.30 fillRect()

```
void Adafruit_GFX::fillRect (
    int16_t x,
    int16_t y,
    int16_t w,
    int16_t h,
    uint16_t color ) [virtual]
```

Fill a rectangle completely with one color. Update in subclasses if desired!

##### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>w</i>	Width in pixels
<i>h</i>	Height in pixels
<i>color</i>	16-bit 5-6-5 Color to fill with

Definition at line 300 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.31 fillRoundRect()

```
void Adafruit_GFX::fillRoundRect (
    int16_t x,
    int16_t y,
    int16_t w,
    int16_t h,
    int16_t r,
    uint16_t color )
```

Draw a rounded rectangle with fill color.

##### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>w</i>	Width in pixels
<i>h</i>	Height in pixels
<i>r</i>	Radius of corner rounding
<i>color</i>	16-bit 5-6-5 Color to draw/fill with

Definition at line 574 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.32 fillScreen()

```
void Adafruit_GFX::fillScreen (
    uint16_t color ) [virtual]
```

Fill the screen completely with one color. Update in subclasses if desired!

##### Parameters

<i>color</i>	16-bit 5-6-5 Color to fill with
--------------	---------------------------------

Reimplemented in [GFXcanvas1](#), [GFXcanvas8](#), and [GFXcanvas16](#).  
 Definition at line 316 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.33 `fillTriangle()`

```
void Adafruit_GFX::fillTriangle (
    int16_t x0,
    int16_t y0,
    int16_t x1,
    int16_t y1,
    int16_t x2,
    int16_t y2,
    uint16_t color )
```

Draw a triangle with color-fill.

##### Parameters

<code>x0</code>	Vertex #0 x coordinate
<code>y0</code>	Vertex #0 y coordinate
<code>x1</code>	Vertex #1 x coordinate
<code>y1</code>	Vertex #1 y coordinate
<code>x2</code>	Vertex #2 x coordinate
<code>y2</code>	Vertex #2 y coordinate
<code>color</code>	16-bit 5-6-5 Color to fill/draw with

Definition at line 619 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.34 `getCursorX()`

```
int16_t Adafruit_GFX::getCursorX (
    void ) const [inline]
```

Get text cursor X location.

##### Returns

X coordinate in pixels

Definition at line 219 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.3.35 `getCursorY()`

```
int16_t Adafruit_GFX::getCursorY (
    void ) const [inline]
```

Get text cursor Y location.

##### Returns

Y coordinate in pixels

Definition at line 227 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.3.36 `getRotation()`

```
uint8_t Adafruit_GFX::getRotation (
    void ) const [inline]
```

Get rotation setting for display.

**Returns**

0 thru 3 corresponding to 4 cardinal rotations

Definition at line 209 of file [batt\\_Adafruit\\_GFX.h](#).

**14.1.3.37 getTextBounds() [1/3]**

```
void Adafruit_GFX::getTextBounds (
    const __FlashStringHelper * str,
    int16_t x,
    int16_t y,
    int16_t * x1,
    int16_t * y1,
    uint16_t * w,
    uint16_t * h )
```

Helper to determine size of a PROGMEM string with current font/size. Pass string and a cursor position, returns UL corner and W,H.

**Parameters**

<i>str</i>	The flash-memory ascii string to measure
<i>x</i>	The current cursor X
<i>y</i>	The current cursor Y
<i>x1</i>	The boundary X coordinate, set by function
<i>y1</i>	The boundary Y coordinate, set by function
<i>w</i>	The boundary width, set by function
<i>h</i>	The boundary height, set by function

Definition at line 1510 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.38 getTextBounds() [2/3]**

```
void Adafruit_GFX::getTextBounds (
    const char * str,
    int16_t x,
    int16_t y,
    int16_t * x1,
    int16_t * y1,
    uint16_t * w,
    uint16_t * h )
```

Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H.

**Parameters**

<i>str</i>	The ASCII string to measure
<i>x</i>	The current cursor X
<i>y</i>	The current cursor Y
<i>x1</i>	The boundary X coordinate, returned by function
<i>y1</i>	The boundary Y coordinate, returned by function
<i>w</i>	The boundary width, returned by function
<i>h</i>	The boundary height, returned by function

Definition at line 1448 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.39 `getTextBounds()` [3/3]

```
void Adafruit_GFX::getTextBounds (
    const String & str,
    int16_t x,
    int16_t y,
    int16_t * x1,
    int16_t * y1,
    uint16_t * w,
    uint16_t * h )
```

Helper to determine size of a string with current font/size. Pass string and a cursor position, returns UL corner and W,H.

##### Parameters

<code>str</code>	The ascii string to measure (as an arduino String() class)
<code>x</code>	The current cursor X
<code>y</code>	The current cursor Y
<code>x1</code>	The boundary X coordinate, set by function
<code>y1</code>	The boundary Y coordinate, set by function
<code>w</code>	The boundary width, set by function
<code>h</code>	The boundary height, set by function

Definition at line 1489 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.40 `height()`

```
int16_t Adafruit_GFX::height (
    void ) const [inline]
```

Get height of the display, accounting for current rotation.

##### Returns

Height in pixels

Definition at line 201 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.3.41 `invertDisplay()`

```
void Adafruit_GFX::invertDisplay (
    bool i ) [virtual]
```

Invert the display (ideally using built-in hardware command)

##### Parameters

<code>i</code>	True if you want to invert, false to make 'normal'
----------------	--

Definition at line 1540 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.42 `setCursor()`

```
void Adafruit_GFX::setCursor (
    int16_t x,
```

```
    int16_t y ) [inline]
```

Set text cursor location.

#### Parameters

<i>x</i>	X coordinate in pixels
<i>y</i>	Y coordinate in pixels

Definition at line 128 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.3.43 setFont()

```
void Adafruit_GFX::setFont (
    const GFXfont * f = NULL )
```

Set the font to display when print()ing, either custom or default.

#### Parameters

<i>f</i>	The <a href="#">GFXfont</a> object, if NULL use built in 6x8 font
----------	---

Definition at line 1338 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.44 setRotation()

```
void Adafruit_GFX::setRotation (
    uint8_t x ) [virtual]
```

Set rotation setting for display.

#### Parameters

<i>x</i>	0 thru 3 corresponding to 4 cardinal rotations
----------	--

Definition at line 1316 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.45 setTextColor() [1/2]

```
void Adafruit_GFX::setTextColor (
    uint16_t c ) [inline]
```

Set text font color with transparent background.

#### Parameters

<i>c</i>	16-bit 5-6-5 Color to draw text with
----------	--------------------------------------

#### Note

For 'transparent' background, background and foreground are set to same color rather than using a separate flag.

Definition at line 141 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.3.46 setTextColor() [2/2]

```
void Adafruit_GFX::setTextColor (
```

```
    uint16_t c,
    uint16_t bg ) [inline]
```

Set text font color with custom background color.

#### Parameters

<i>c</i>	16-bit 5-6-5 Color to draw text with
<i>bg</i>	16-bit 5-6-5 Color to draw background/fill with

Definition at line 150 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.3.47 setTextSize() [1/2]

```
void Adafruit_GFX::setTextSize (
    uint8_t s )
```

Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger.

#### Parameters

<i>s</i>	Desired text size. 1 is default 6x8, 2 is 12x16, 3 is 18x24, etc
----------	--

Definition at line 1295 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.48 setTextSize() [2/2]

```
void Adafruit_GFX::setTextSize (
    uint8_t s_x,
    uint8_t s_y )
```

Set text 'magnification' size. Each increase in s makes 1 pixel that much bigger.

#### Parameters

<i>s_x</i>	Desired text width magnification level in X-axis. 1 is default
<i>s_y</i>	Desired text width magnification level in Y-axis. 1 is default

Definition at line 1305 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.49 setTextWrap()

```
void Adafruit_GFX::setTextWrap (
    bool w ) [inline]
```

Set whether text that is too long for the screen width should automatically wrap around to the next line (else clip right).

#### Parameters

<i>w</i>	true for wrapping, false for clipping
----------	---------------------------------------

Definition at line 162 of file [batt\\_Adafruit\\_GFX.h](#).

**14.1.3.50 startWrite()**

```
void Adafruit_GFX::startWrite (
    void ) [virtual]
```

Start a display-writing routine, overwrite in subclasses.  
Definition at line 180 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.51 width()**

```
int16_t Adafruit_GFX::width (
    void ) const [inline]
```

Get width of the display, accounting for current rotation.

**Returns**

Width in pixels

Definition at line 193 of file [batt\\_Adafruit\\_GFX.h](#).

**14.1.3.52 write()**

```
size_t Adafruit_GFX::write (
    uint8_t c ) [virtual]
```

Print one byte/character of data, used to support print()

**Parameters**

<i>c</i>	The 8-bit ascii character to write
----------	------------------------------------

Definition at line 1242 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.53 writeFastHLine()**

```
void Adafruit_GFX::writeFastHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [virtual]
```

Write a perfectly horizontal line, overwrite in subclasses if startWrite is defined!

**Parameters**

<i>x</i>	Left-most x coordinate
<i>y</i>	Left-most y coordinate
<i>w</i>	Width in pixels
<i>color</i>	16-bit 5-6-5 Color to fill with

Definition at line 222 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.1.3.54 writeFastVLine()**

```
void Adafruit_GFX::writeFastVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [virtual]
```

Write a perfectly vertical line, overwrite in subclasses if startWrite is defined!

#### Parameters

<i>x</i>	Top-most x coordinate
<i>y</i>	Top-most y coordinate
<i>h</i>	Height in pixels
<i>color</i>	16-bit 5-6-5 Color to fill with

Definition at line 204 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.55 writeFillRect()

```
void Adafruit_GFX::writeFillRect (
    int16_t x,
    int16_t y,
    int16_t w,
    int16_t h,
    uint16_t color ) [virtual]
```

Write a rectangle completely with one color, overwrite in subclasses if startWrite is defined!

#### Parameters

<i>x</i>	Top left corner x coordinate
<i>y</i>	Top left corner y coordinate
<i>w</i>	Width in pixels
<i>h</i>	Height in pixels
<i>color</i>	16-bit 5-6-5 Color to fill with

Definition at line 241 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.56 writeLine()

```
void Adafruit_GFX::writeLine (
    int16_t x0,
    int16_t y0,
    int16_t x1,
    int16_t y1,
    uint16_t color ) [virtual]
```

Write a line. Bresenham's algorithm - thx wikipedia.

#### Parameters

<i>x0</i>	Start point x coordinate
<i>y0</i>	Start point y coordinate
<i>x1</i>	End point x coordinate
<i>y1</i>	End point y coordinate
<i>color</i>	16-bit 5-6-5 Color to draw with

Definition at line 132 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.1.3.57 writePixel()

```
void Adafruit_GFX::writePixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [virtual]
```

Write a pixel, overwrite in subclasses if startWrite is defined!

##### Parameters

x	x coordinate
y	y coordinate
color	16-bit 5-6-5 Color to fill with

Definition at line 190 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.1.4 Member Data Documentation

#### 14.1.4.1 \_cp437

```
bool Adafruit_GFX::_cp437 [protected]  
If set, use correct CP437 charset (default is off)  
Definition at line 244 of file batt\_Adafruit\_GFX.h.
```

#### 14.1.4.2 \_height

```
int16_t Adafruit_GFX::_height [protected]  
Display height as modified by current rotation.  
Definition at line 235 of file batt\_Adafruit\_GFX.h.
```

#### 14.1.4.3 \_width

```
int16_t Adafruit_GFX::_width [protected]  
Display width as modified by current rotation.  
Definition at line 234 of file batt\_Adafruit\_GFX.h.
```

#### 14.1.4.4 cursor\_x

```
int16_t Adafruit_GFX::cursor_x [protected]  
x location to start print()ing text  
Definition at line 236 of file batt\_Adafruit\_GFX.h.
```

#### 14.1.4.5 cursor\_y

```
int16_t Adafruit_GFX::cursor_y [protected]  
y location to start print()ing text  
Definition at line 237 of file batt\_Adafruit\_GFX.h.
```

#### 14.1.4.6 gfxFont

```
GFXfont* Adafruit_GFX::gfxFont [protected]  
Pointer to special font.
```

Definition at line 245 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.7 HEIGHT

`int16_t Adafruit_GFX::HEIGHT [protected]`

This is the 'raw' display height - never changes.

Definition at line 233 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.8 rotation

`uint8_t Adafruit_GFX::rotation [protected]`

Display rotation (0 thru 3)

Definition at line 242 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.9 textbgcolor

`uint16_t Adafruit_GFX::textbgcolor [protected]`

16-bit text color for print()

Definition at line 239 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.10 textcolor

`uint16_t Adafruit_GFX::textcolor [protected]`

16-bit background color for print()

Definition at line 238 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.11 textsize\_x

`uint8_t Adafruit_GFX::textsize_x [protected]`

Desired magnification in X-axis of text to print()

Definition at line 240 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.12 textsize\_y

`uint8_t Adafruit_GFX::textsize_y [protected]`

Desired magnification in Y-axis of text to print()

Definition at line 241 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.13 WIDTH

`int16_t Adafruit_GFX::WIDTH [protected]`

This is the 'raw' display width - never changes.

Definition at line 232 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.1.4.14 wrap

`bool Adafruit_GFX::wrap [protected]`

If set, 'wrap' text at right edge of display.

Definition at line 243 of file [batt\\_Adafruit\\_GFX.h](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_↔ sw/LIBRARIES/0008-Adafruit\_GFX\_Library/batt\_Adafruit\_GFX.h
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_↔ sw/LIBRARIES/0008-Adafruit\_GFX\_Library/batt\_Adafruit\_GFX.cpp

## 14.2 Adafruit\_GFX\_Button Class Reference

A simple drawn button UI element.

```
#include <batt_Adafruit_GFX.h>
```

### Public Member Functions

- **Adafruit\_GFX\_Button (void)**  
*Create a simple drawn button UI element.*
- **void initButton (Adafruit\_GFX \*gfx, int16\_t x, int16\_t y, uint16\_t w, uint16\_t h, uint16\_t outline, uint16\_t fill, uint16\_t textcolor, char \*label, uint8\_t textsize)**  
*Initialize button with our desired color/size/settings.*
- **void initButton (Adafruit\_GFX \*gfx, int16\_t x, int16\_t y, uint16\_t w, uint16\_t h, uint16\_t outline, uint16\_t fill, uint16\_t textcolor, char \*label, uint8\_t textsize\_x, uint8\_t textsize\_y)**  
*Initialize button with our desired color/size/settings.*
- **void initButtonUL (Adafruit\_GFX \*gfx, int16\_t x1, int16\_t y1, uint16\_t w, uint16\_t h, uint16\_t outline, uint16\_t fill, uint16\_t textcolor, char \*label, uint8\_t textsize)**  
*Initialize button with our desired color/size/settings, with upper-left coordinates.*
- **void initButtonUL (Adafruit\_GFX \*gfx, int16\_t x1, int16\_t y1, uint16\_t w, uint16\_t h, uint16\_t outline, uint16\_t fill, uint16\_t textcolor, char \*label, uint8\_t textsize\_x, uint8\_t textsize\_y)**  
*Initialize button with our desired color/size/settings, with upper-left coordinates.*
- **void drawButton (bool inverted=false)**  
*Draw the button on the screen.*
- **bool contains (int16\_t x, int16\_t y)**  
*Helper to let us know if a coordinate is within the bounds of the button.*
- **void press (bool p)**  
*Sets button state, should be done by some touch function.*
- **bool justPressed ()**  
*Query whether the button was pressed since we last checked state.*
- **bool justReleased ()**  
*Query whether the button was released since we last checked state.*
- **bool isPressed (void)**  
*Query whether the button is currently pressed.*

### 14.2.1 Detailed Description

A simple drawn button UI element.

Definition at line 249 of file [batt\\_Adafruit\\_GFX.h](#).

### 14.2.2 Constructor & Destructor Documentation

#### 14.2.2.1 Adafruit\_GFX\_Button()

```
Adafruit_GFX_Button::Adafruit_GFX_Button (
    void )
```

Create a simple drawn button UI element.

Definition at line 1552 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.2.3 Member Function Documentation

#### 14.2.3.1 contains()

```
bool Adafruit_GFX_Button::contains (
    int16_t x,
    int16_t y )
```

Helper to let us know if a coordinate is within the bounds of the button.

##### Parameters

<i>x</i>	The X coordinate to check
<i>y</i>	The Y coordinate to check

##### Returns

True if within button graphics outline

Definition at line 1706 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.2.3.2 drawButton()

```
void Adafruit_GFX_Button::drawButton (
    bool inverted = false )
```

Draw the button on the screen.

##### Parameters

<i>inverted</i>	Whether to draw with fill/text swapped to indicate 'pressed'
-----------------	--

Definition at line 1673 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.2.3.3 initButton() [1/2]

```
void Adafruit_GFX_Button::initButton (
    Adafruit_GFX * gfx,
    int16_t x,
    int16_t y,
    uint16_t w,
    uint16_t h,
    uint16_t outline,
    uint16_t fill,
    uint16_t textcolor,
    char * label,
    uint8_t textsize )
```

Initialize button with our desired color/size/settings.

##### Parameters

<i>gfx</i>	Pointer to our display so we can draw to it!
<i>x</i>	The X coordinate of the center of the button
<i>y</i>	The Y coordinate of the center of the button
<i>w</i>	Width of the button
<i>h</i>	Height of the button

**Parameters**

<i>outline</i>	Color of the outline (16-bit 5-6-5 standard)
<i>fill</i>	Color of the button fill (16-bit 5-6-5 standard)
<i>textcolor</i>	Color of the button label (16-bit 5-6-5 standard)
<i>label</i>	Ascii string of the text inside the button
<i>textsize</i>	The font magnification of the label text

Definition at line 1570 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.2.3.4 initButton() [2/2]**

```
void Adafruit_GFX_Button::initButton (
    Adafruit_GFX * gfx,
    int16_t x,
    int16_t y,
    uint16_t w,
    uint16_t h,
    uint16_t outline,
    uint16_t fill,
    uint16_t textcolor,
    char * label,
    uint8_t textsize_x,
    uint8_t textsize_y )
```

Initialize button with our desired color/size/settings.

**Parameters**

<i>gfx</i>	Pointer to our display so we can draw to it!
<i>x</i>	The X coordinate of the center of the button
<i>y</i>	The Y coordinate of the center of the button
<i>w</i>	Width of the button
<i>h</i>	Height of the button
<i>outline</i>	Color of the outline (16-bit 5-6-5 standard)
<i>fill</i>	Color of the button fill (16-bit 5-6-5 standard)
<i>textcolor</i>	Color of the button label (16-bit 5-6-5 standard)
<i>label</i>	Ascii string of the text inside the button
<i>textsize_x</i>	The font magnification in X-axis of the label text
<i>textsize_y</i>	The font magnification in Y-axis of the label text

Definition at line 1596 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.2.3.5 initButtonUL() [1/2]**

```
void Adafruit_GFX_Button::initButtonUL (
    Adafruit_GFX * gfx,
    int16_t x1,
    int16_t y1,
    uint16_t w,
    uint16_t h,
    uint16_t outline,
```

```
    uint16_t fill,
    uint16_t textcolor,
    char * label,
    uint8_t textsize )
```

Initialize button with our desired color/size/settings, with upper-left coordinates.

#### Parameters

<i>gfx</i>	Pointer to our display so we can draw to it!
<i>x1</i>	The X coordinate of the Upper-Left corner of the button
<i>y1</i>	The Y coordinate of the Upper-Left corner of the button
<i>w</i>	Width of the button
<i>h</i>	Height of the button
<i>outline</i>	Color of the outline (16-bit 5-6-5 standard)
<i>fill</i>	Color of the button fill (16-bit 5-6-5 standard)
<i>textcolor</i>	Color of the button label (16-bit 5-6-5 standard)
<i>label</i>	Ascii string of the text inside the button
<i>textsize</i>	The font magnification of the label text

Definition at line 1622 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.2.3.6 `initButtonUL()` [2/2]

```
void Adafruit_GFX_Button::initButtonUL (
    Adafruit_GFX * gfx,
    int16_t x1,
    int16_t y1,
    uint16_t w,
    uint16_t h,
    uint16_t outline,
    uint16_t fill,
    uint16_t textcolor,
    char * label,
    uint8_t textsize_x,
    uint8_t textsize_y )
```

Initialize button with our desired color/size/settings, with upper-left coordinates.

#### Parameters

<i>gfx</i>	Pointer to our display so we can draw to it!
<i>x1</i>	The X coordinate of the Upper-Left corner of the button
<i>y1</i>	The Y coordinate of the Upper-Left corner of the button
<i>w</i>	Width of the button
<i>h</i>	Height of the button
<i>outline</i>	Color of the outline (16-bit 5-6-5 standard)
<i>fill</i>	Color of the button fill (16-bit 5-6-5 standard)
<i>textcolor</i>	Color of the button label (16-bit 5-6-5 standard)
<i>label</i>	Ascii string of the text inside the button
<i>textsize_x</i>	The font magnification in X-axis of the label text
<i>textsize_y</i>	The font magnification in Y-axis of the label text

Definition at line 1648 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.2.3.7 isPressed()

```
bool Adafruit_GFX_Button::isPressed  
    void ) [inline]
```

Query whether the button is currently pressed.

##### Returns

True if pressed

Definition at line 292 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.2.3.8 justPressed()

```
bool Adafruit_GFX_Button::justPressed ( )
```

Query whether the button was pressed since we last checked state.

##### Returns

True if was not-pressed before, now is.

Definition at line 1717 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.2.3.9 justReleased()

```
bool Adafruit_GFX_Button::justReleased ( )
```

Query whether the button was released since we last checked state.

##### Returns

True if was pressed before, now is not.

Definition at line 1725 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.2.3.10 press()

```
void Adafruit_GFX_Button::press  
    bool p ) [inline]
```

Sets button state, should be done by some touch function.

##### Parameters

<i>p</i>	True for pressed, false for not.
----------	----------------------------------

Definition at line 278 of file [batt\\_Adafruit\\_GFX.h](#).

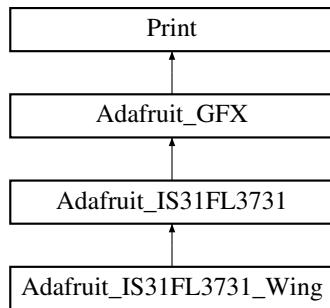
The documentation for this class was generated from the following files:

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.cpp](#)

## 14.3 Adafruit\_IS31FL3731 Class Reference

Constructor for generic IS31FL3731 breakout version.

```
#include <batt_Adafruit_IS31FL3731.h>
Inheritance diagram for Adafruit_IS31FL3731:
```



## Public Member Functions

- `Adafruit_IS31FL3731 (uint8_t x=16, uint8_t y=9)`  
*Constructor for breakout version.*
- `bool begin (uint8_t addr=ISSI_ADDR_DEFAULT)`  
*Initialize hardware and clear display.*
- `void drawPixel (int16_t x, int16_t y, uint16_t color)`  
*Adafruit GFX low level accesssor - sets a 8-bit PWM pixel value handles rotation and pixel arrangement, unlike setLEDPWM.*
- `void clear (void)`  
*Sets all LEDs on & 0 PWM for current frame.*
- `void setLEDPWM (uint8_t lednum, uint8_t pwm, uint8_t bank=0)`  
*Low level accesssor - sets a 8-bit PWM pixel value to a bank location does not handle rotation, x/y or any rearrangements!*
- `void audioSync (bool sync)`  
*Enable the audio 'sync' for brightness pulsing (not really tested)*
- `void setFrame (uint8_t b)`  
*Set's this object's frame tracker (does not talk to the chip)*
- `void displayFrame (uint8_t frame)`  
*Have the chip set the display to the contents of a frame.*

## Protected Member Functions

- `void selectBank (uint8_t bank)`  
*Switch to a given bank in the chip memory for future reads.*
- `void writeRegister8 (uint8_t bank, uint8_t reg, uint8_t data)`  
*Write one byte to a register located in a given bank.*
- `uint8_t readRegister8 (uint8_t bank, uint8_t reg)`  
*Read one byte from a register located in a given bank.*

## Protected Attributes

- `uint8_t _i2caddr`  
*The I2C address we expect to find the chip.*
- `uint8_t _frame`  
*The frame (of 8) we are currently addressing.*

### 14.3.1 Detailed Description

Constructor for generic IS31FL3731 breakout version.  
Definition at line 32 of file [batt\\_Adafruit\\_IS31FL3731.h](#).

### 14.3.2 Constructor & Destructor Documentation

#### 14.3.2.1 Adafruit\_IS31FL3731()

```
Adafruit_IS31FL3731::Adafruit_IS31FL3731 (
    uint8_t width = 16,
    uint8_t height = 9 )
```

Constructor for breakout version.

##### Parameters

<i>width</i>	Desired width of led display
<i>height</i>	Desired height of led display

Definition at line 23 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

### 14.3.3 Member Function Documentation

#### 14.3.3.1 audioSync()

```
void Adafruit_IS31FL3731::audioSync (
    bool sync )
```

Enable the audio 'sync' for brightness pulsing (not really tested)

##### Parameters

<i>sync</i>	True to enable, False to disable
-------------	----------------------------------

Definition at line 237 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

#### 14.3.3.2 begin()

```
bool Adafruit_IS31FL3731::begin (
    uint8_t addr = ISSI\_ADDR\_DEFAULT )
```

Initialize hardware and clear display.

##### Parameters

<i>addr</i>	The I2C address we expect to find the chip at
-------------	---

##### Returns

True on success, false if chip isn't found

Definition at line 41 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

#### 14.3.3.3 clear()

```
void Adafruit_IS31FL3731::clear (
    void )
```

Sets all LEDs on & 0 PWM for current frame.

Definition at line 86 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

#### 14.3.3.4 `displayFrame()`

```
void Adafruit_IS31FL3731::displayFrame (
    uint8_t frame )
```

Have the chip set the display to the contents of a frame.

##### Parameters

<i>frame</i>	Ranges from 0 - 7 for the 8 frames
--------------	------------------------------------

Definition at line 212 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

#### 14.3.3.5 `drawPixel()`

```
void Adafruit_IS31FL3731::drawPixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [virtual]
```

Adafruit GFX low level accesssor - sets a 8-bit PWM pixel value handles rotation and pixel arrangement, unlike `setLEDPWM`.

##### Parameters

<i>x</i>	The x position, starting with 0 for left-most side
<i>y</i>	The y position, starting with 0 for top-most side
<i>color</i>	Despite being a 16-bit value, takes 0 (off) to 255 (max on)

Implements [Adafruit\\_GFX](#).

Reimplemented in [Adafruit\\_IS31FL3731\\_Wing](#).

Definition at line 170 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

#### 14.3.3.6 `readRegister8()`

```
uint8_t Adafruit_IS31FL3731::readRegister8 (
    uint8_t bank,
    uint8_t reg ) [protected]
```

Read one byte from a register located in a given bank.

##### Parameters

<i>bank</i>	The IS31 bank to read the register location
<i>reg</i>	the offset into the bank to read

##### Returns

1 byte value

Definition at line 273 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

#### 14.3.3.7 `selectBank()`

```
void Adafruit_IS31FL3731::selectBank (
    uint8_t bank ) [protected]
```

Switch to a given bank in the chip memory for future reads.

**Parameters**

<i>bank</i>	The IS31 bank to switch to
-------------	----------------------------

Definition at line 224 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

**14.3.3.8 setFrame()**

```
void Adafruit_IS31FL3731::setFrame (
    uint8_t frame )
```

Set's this object's frame tracker (does not talk to the chip)

**Parameters**

<i>frame</i>	Ranges from 0 - 7 for the 8 frames
--------------	------------------------------------

Definition at line 204 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

**14.3.3.9 setLEDPWM()**

```
void Adafruit_IS31FL3731::setLEDPWM (
    uint8_t lednum,
    uint8_t pwm,
    uint8_t bank = 0 )
```

Low level accesssor - sets a 8-bit PWM pixel value to a bank location does not handle rotation, x/y or any rearrangements!

**Parameters**

<i>lednum</i>	The offset into the bank that corresponds to the LED
<i>bank</i>	The bank/frame we will set the data in
<i>pwm</i>	brightnes, from 0 (off) to 255 (max on)

Definition at line 109 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

**14.3.3.10 writeRegister8()**

```
void Adafruit_IS31FL3731::writeRegister8 (
    uint8_t bank,
    uint8_t reg,
    uint8_t data ) [protected]
```

Write one byte to a register located in a given bank.

**Parameters**

<i>bank</i>	The IS31 bank to write the register location
<i>reg</i>	the offset into the bank to write
<i>data</i>	The byte value

Definition at line 253 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

**14.3.4 Member Data Documentation**

#### 14.3.4.1 `_frame`

`uint8_t Adafruit_IS31FL3731::_frame [protected]`

The frame (of 8) we are currently addressing.

Definition at line 49 of file [batt\\_Adafruit\\_IS31FL3731.h](#).

#### 14.3.4.2 `_i2caddr`

`uint8_t Adafruit_IS31FL3731::_i2caddr [protected]`

The I2C address we expect to find the chip.

Definition at line 48 of file [batt\\_Adafruit\\_IS31FL3731.h](#).

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

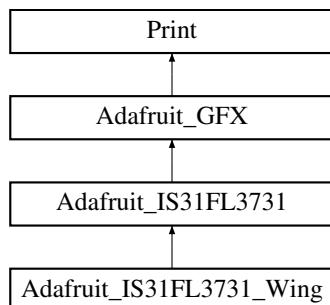
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_Library/[batt\\_Adafruit\\_IS31FL3731.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_Library/[batt\\_Adafruit\\_IS31FL3731.cpp](#)

## 14.4 Adafruit\_IS31FL3731\_Wing Class Reference

Constructor for FeatherWing IS31FL3731 version.

#include <batt\_Adafruit\_IS31FL3731.h>

Inheritance diagram for Adafruit\_IS31FL3731\_Wing:



### Public Member Functions

- [Adafruit\\_IS31FL3731\\_Wing \(void\)](#)  
*Constructor for FeatherWing version (15x7 LEDs)*
- void [drawPixel \(int16\\_t x, int16\\_t y, uint16\\_t color\)](#)  
*Adafruit GFX low level accesssor - sets a 8-bit PWM pixel value handles rotation and pixel arrangement, unlike setLEDPWM.*

### Additional Inherited Members

#### 14.4.1 Detailed Description

Constructor for FeatherWing IS31FL3731 version.

Definition at line 57 of file [batt\\_Adafruit\\_IS31FL3731.h](#).

#### 14.4.2 Constructor & Destructor Documentation

#### 14.4.2.1 Adafruit\_IS31FL3731\_Wing()

```
Adafruit_IS31FL3731_Wing::Adafruit_IS31FL3731_Wing (
    void )
```

Constructor for FeatherWing version (15x7 LEDs)

Definition at line 31 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

### 14.4.3 Member Function Documentation

#### 14.4.3.1 drawPixel()

```
void Adafruit_IS31FL3731_Wing::drawPixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [virtual]
```

Adafruit GFX low level accesssor - sets a 8-bit PWM pixel value handles rotation and pixel arrangement, unlike setLEDPWM.

#### Parameters

<i>x</i>	The x position, starting with 0 for left-most side
<i>y</i>	The y position, starting with 0 for top-most side
<i>color</i>	Despite being a 16-bit value, takes 0 (off) to 255 (max on)

Reimplemented from [Adafruit\\_IS31FL3731](#).

Definition at line 124 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_Library/[batt\\_Adafruit\\_IS31FL3731.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_Library/[batt\\_Adafruit\\_IS31FL3731.cpp](#)

## 14.5 ArduinoLowPowerClass Class Reference

```
#include <batt_ArduinoLowPower.h>
```

### Public Member Functions

- void [idle](#) (void)
- void [idle](#) (uint32\_t millis)
- void [idle](#) (int millis)
- void [sleep](#) (void)
- void [sleep](#) (uint32\_t millis)
- void [sleep](#) (int millis)
- void [deepSleep](#) (void)
- void [deepSleep](#) (uint32\_t millis)
- void [deepSleep](#) (int millis)
- void [attachInterruptWakeup](#) (uint32\_t pin, voidFuncPtr callback, [irq\\_mode](#) mode)
- void [deatchRTCInterrupt](#) (void)

#### 14.5.1 Detailed Description

Definition at line 48 of file [batt\\_ArduinoLowPower.h](#).

## 14.5.2 Member Function Documentation

### 14.5.2.1 attachInterruptWakeup()

```
void ArduinoLowPowerClass::attachInterruptWakeup (
    uint32_t pin,
    voidFuncPtr callback,
    irq_mode mode )
```

### 14.5.2.2 deatchRTCInterrupt()

```
void ArduinoLowPowerClass::deatchRTCInterrupt (
    void )
```

### 14.5.2.3 deepSleep() [1/3]

```
void ArduinoLowPowerClass::deepSleep (
    int millis ) [inline]
```

Definition at line 67 of file [batt\\_ArduinoLowPower.h](#).

### 14.5.2.4 deepSleep() [2/3]

```
void ArduinoLowPowerClass::deepSleep (
    uint32_t millis )
```

### 14.5.2.5 deepSleep() [3/3]

```
void ArduinoLowPowerClass::deepSleep (
    void )
```

### 14.5.2.6 idle() [1/3]

```
void ArduinoLowPowerClass::idle (
    int millis ) [inline]
```

Definition at line 53 of file [batt\\_ArduinoLowPower.h](#).

### 14.5.2.7 idle() [2/3]

```
void ArduinoLowPowerClass::idle (
    uint32_t millis )
```

### 14.5.2.8 idle() [3/3]

```
void ArduinoLowPowerClass::idle (
    void )
```

**14.5.2.9 sleep() [1/3]**

```
void ArduinoLowPowerClass::sleep (
    int millis ) [inline]
```

Definition at line 60 of file [batt\\_ArduinoLowPower.h](#).

**14.5.2.10 sleep() [2/3]**

```
void ArduinoLowPowerClass::sleep (
    uint32_t millis )
```

**14.5.2.11 sleep() [3/3]**

```
void ArduinoLowPowerClass::sleep (
    void )
```

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/[batt\\_ArduinoLowPower.h](#)

## 14.6 dcdc\_controller Class Reference

```
#include <DCDC.h>
```

### Public Member Functions

- [dcdc\\_controller \(int16\\_t pin\)](#)  
*Construct a new dcdc controller object.*
- void [SetVoltage \(int volt, bool mode\)](#)  
*Set the Voltage object.*
- void [EnableDCDC \(bool enable\)](#)  
*Habilita la activacion del DCDC.*

### Public Attributes

- uint16\_t [encoderPos](#)
- byte [TPICvalue](#)
- int16\_t [pin\\_enable](#)

#### 14.6.1 Detailed Description

Definition at line 21 of file [DCDC.h](#).

#### 14.6.2 Constructor & Destructor Documentation

##### 14.6.2.1 dcdc\_controller()

```
dcdc_controller::dcdc_controller (
    int16_t pin ) [inline]
```

Construct a new dcdc controller object.

#### Parameters

<i>pin</i>	Pin conectado al enable del DCDC.
------------	-----------------------------------

Definition at line 48 of file [DCDC.h](#).

### 14.6.3 Member Function Documentation

#### 14.6.3.1 EnableDCDC()

```
void dcdc_controller::EnableDCDC (
    bool enable ) [inline]
```

Habilita la activacion del DCDC.

##### Parameters

<i>enable</i>	<input type="checkbox"/>
---------------	--------------------------

Definition at line 82 of file [DCDC.h](#).

#### 14.6.3.2 SetVoltage()

```
void dcdc_controller::SetVoltage (
    int volt,
    bool mode ) [inline]
```

Set the Voltage object.

##### Parameters

<i>volt</i>	<input type="checkbox"/>
<i>mode</i>	<input type="checkbox"/>

Definition at line 62 of file [DCDC.h](#).

### 14.6.4 Member Data Documentation

#### 14.6.4.1 encoderPos

```
uint16_t dc当地控制器::encoderPos
```

Definition at line 39 of file [DCDC.h](#).

#### 14.6.4.2 pin\_enable

```
int16_t dc当地控制器::pin_enable
```

Definition at line 41 of file [DCDC.h](#).

#### 14.6.4.3 TPICvalue

```
byte dc当地控制器::TPICvalue
```

Definition at line 40 of file [DCDC.h](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0004-DCDC/DCDC.h

## 14.7 EEPROM\_EMULATION Struct Reference

```
#include <diagnostic.h>
```

### Public Attributes

- int16\_t data [EEPROM\_EMULATION\_SIZE]
- boolean valid
- byte data [EEPROM\_EMULATION\_SIZE]

#### 14.7.1 Detailed Description

##### VARIABLES

Definition at line 36 of file [diagnostic.h](#).

#### 14.7.2 Member Data Documentation

##### 14.7.2.1 data [1/2]

```
int16_t EEPROM_EMULATION::data[EEPROM_EMULATION_SIZE]
```

Definition at line 38 of file [diagnostic.h](#).

##### 14.7.2.2 data [2/2]

```
byte EEPROM_EMULATION::data[EEPROM_EMULATION_SIZE]
```

Definition at line 32 of file [batt\\_FlashAsEEPROM.h](#).

##### 14.7.2.3 valid

```
boolean EEPROM_EMULATION::valid
```

Definition at line 39 of file [diagnostic.h](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0010-LOGGING/[diagnostic.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/[batt\\_FlashAsEEPROM.h](#)

## 14.8 EEPROMClass Class Reference

```
#include <batt_FlashAsEEPROM.h>
```

### Public Member Functions

- [EEPROMClass](#) (void)
- uint8\_t [read](#) (int)
- void [write](#) (int, uint8\_t)
- void [update](#) (int, uint8\_t)
- bool [isValid](#) ()
- void [commit](#) ()
- uint16\_t [length](#) ()

### 14.8.1 Detailed Description

Definition at line 37 of file [batt\\_FlashAsEEPROM.h](#).

### 14.8.2 Constructor & Destructor Documentation

#### 14.8.2.1 EEPROMClass()

```
EEPROMClass::EEPROMClass ( void )
```

Definition at line 26 of file [batt\\_FlashAsEEPROM.cpp](#).

### 14.8.3 Member Function Documentation

#### 14.8.3.1 commit()

```
void EEPROMClass::commit ( )
```

Write previously made eeprom changes to the underlying flash storage Use this with care: Each and every commit will harm the flash and reduce it's lifetime (like with every flash memory)

Definition at line 65 of file [batt\\_FlashAsEEPROM.cpp](#).

#### 14.8.3.2 isValid()

```
bool EEPROMClass::isValid ( )
```

Check whether the eeprom data is valid

##### Returns

true, if eeprom data is valid (has been written at least once), false if not

Definition at line 59 of file [batt\\_FlashAsEEPROM.cpp](#).

#### 14.8.3.3 length()

```
uint16_t EEPROMClass::length ( ) [inline]
```

Definition at line 75 of file [batt\\_FlashAsEEPROM.h](#).

#### 14.8.3.4 read()

```
uint8_t EEPROMClass::read ( int address )
```

Read an eeprom cell

##### Parameters

<i>index</i>	<input type="text"/>
--------------	----------------------

##### Returns

value

Definition at line 30 of file [batt\\_FlashAsEEPROM.cpp](#).

**14.8.3.5 update()**

```
void EEPROMClass::update (
    int address,
    uint8_t value )
```

Update a eeprom cell

**Parameters**

<i>index</i>	
<i>value</i>	

Definition at line 36 of file [batt\\_FlashAsEEPROM.cpp](#).

**14.8.3.6 write()**

```
void EEPROMClass::write (
    int address,
    uint8_t value )
```

Write value to an eeprom cell

**Parameters**

<i>index</i>	
<i>value</i>	

Definition at line 45 of file [batt\\_FlashAsEEPROM.cpp](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/[batt\\_FlashAsEEPROM.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/[batt\\_FlashAsEEPROM.cpp](#)

## 14.9 Element Struct Reference

```
#include <diagnostic.h>
```

### Public Attributes

- `int16_t value = 0`
- `char category [2] = {'X', 'X'}`
- `String name = "NoName"`

#### 14.9.1 Detailed Description

Definition at line 42 of file [diagnostic.h](#).

#### 14.9.2 Member Data Documentation

##### 14.9.2.1 category

```
char Element::category[2] = {'X', 'X'}
```

Definition at line 45 of file [diagnostic.h](#).

#### 14.9.2.2 name

```
String Element::name = "NoName"
Definition at line 46 of file diagnostic.h.
```

#### 14.9.2.3 value

```
int16_t Element::value = 0
Definition at line 44 of file diagnostic.h.
```

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0010-LOGGING/[diagnostic.h](#)

## 14.10 FlashClass Class Reference

```
#include <batt_FlashStorage.h>
```

### Public Member Functions

- [FlashClass](#) (const void \*flash\_addr=NULL, uint32\_t size=0)
- void [write](#) (const void \*[data](#))
- void [erase](#) ()
- void [read](#) (void \*[data](#))
- void [write](#) (const volatile void \*flash\_ptr, const void \*[data](#), uint32\_t size)
- void [erase](#) (const volatile void \*flash\_ptr, uint32\_t size)
- void [read](#) (const volatile void \*flash\_ptr, void \*[data](#), uint32\_t size)

#### 14.10.1 Detailed Description

Definition at line 50 of file [batt\\_FlashStorage.h](#).

#### 14.10.2 Constructor & Destructor Documentation

##### 14.10.2.1 FlashClass()

```
FlashClass::FlashClass (
    const void * flash_addr = NULL,
    uint32_t size = 0 )
```

Definition at line 24 of file [batt\\_FlashStorage.cpp](#).

#### 14.10.3 Member Function Documentation

##### 14.10.3.1 [erase\(\)](#) [1/2]

```
void FlashClass::erase ( ) [inline]
Definition at line 55 of file batt\_FlashStorage.h.
```

**14.10.3.2 erase() [2/2]**

```
void FlashClass::erase (
    const volatile void * flash_ptr,
    uint32_t size )
```

Definition at line 120 of file [batt\\_FlashStorage.cpp](#).

**14.10.3.3 read() [1/2]**

```
void FlashClass::read (
    const volatile void * flash_ptr,
    void * data,
    uint32_t size )
```

Definition at line 145 of file [batt\\_FlashStorage.cpp](#).

**14.10.3.4 read() [2/2]**

```
void FlashClass::read (
    void * data ) [inline]
```

Definition at line 56 of file [batt\\_FlashStorage.h](#).

**14.10.3.5 write() [1/2]**

```
void FlashClass::write (
    const void * data ) [inline]
```

Definition at line 54 of file [batt\\_FlashStorage.h](#).

**14.10.3.6 write() [2/2]**

```
void FlashClass::write (
    const volatile void * flash_ptr,
    const void * data,
    uint32_t size )
```

Definition at line 65 of file [batt\\_FlashStorage.cpp](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/[batt\\_FlashStorage.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/[batt\\_FlashStorage.cpp](#)

## 14.11 FlashStorageClass< T > Class Template Reference

```
#include <batt_FlashStorage.h>
```

### Public Member Functions

- [FlashStorageClass](#) (const void \*flash\_addr)
- void [write](#) (T data)
- void [read](#) (T \*data)
- T [read](#) ()

### 14.11.1 Detailed Description

```
template<class T>
class FlashStorageClass< T >
```

Definition at line 71 of file [batt\\_FlashStorage.h](#).

### 14.11.2 Constructor & Destructor Documentation

#### 14.11.2.1 FlashStorageClass()

```
template<class T >
FlashStorageClass< T >::FlashStorageClass (
    const void * flash_addr ) [inline]
```

Definition at line 73 of file [batt\\_FlashStorage.h](#).

### 14.11.3 Member Function Documentation

#### 14.11.3.1 read() [1/2]

```
template<class T >
T FlashStorageClass< T >::read ( ) [inline]
```

Definition at line 84 of file [batt\\_FlashStorage.h](#).

#### 14.11.3.2 read() [2/2]

```
template<class T >
void FlashStorageClass< T >::read (
    T * data ) [inline]
```

Definition at line 80 of file [batt\\_FlashStorage.h](#).

#### 14.11.3.3 write()

```
template<class T >
void FlashStorageClass< T >::write (
    T data ) [inline]
```

Definition at line 77 of file [batt\\_FlashStorage.h](#).

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

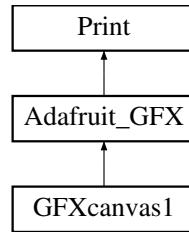
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/[batt\\_FlashStorage.h](#)

## 14.12 GFXcanvas1 Class Reference

A GFX 1-bit canvas context for graphics.

```
#include <batt_Adafruit_GFX.h>
```

Inheritance diagram for GFXcanvas1:



## Public Member Functions

- `GFXcanvas1 (uint16_t w, uint16_t h)`  
*Instantiate a GFX 1-bit canvas context for graphics.*
- `~GFXcanvas1 (void)`  
*Delete the canvas, free memory.*
- `void drawPixel (int16_t x, int16_t y, uint16_t color)`  
*Draw a pixel to the canvas framebuffer.*
- `void fillScreen (uint16_t color)`  
*Fill the framebuffer completely with one color.*
- `void drawFastVLine (int16_t x, int16_t y, int16_t h, uint16_t color)`  
*Speed optimized vertical line drawing.*
- `void drawFastHLine (int16_t x, int16_t y, int16_t w, uint16_t color)`  
*Speed optimized horizontal line drawing.*
- `bool getPixel (int16_t x, int16_t y) const`
- `uint8_t * getBuffer (void) const`  
*Get a pointer to the internal buffer memory.*

## Protected Member Functions

- `bool getRawPixel (int16_t x, int16_t y) const`
- `void drawFastRawVLine (int16_t x, int16_t y, int16_t h, uint16_t color)`  
*Speed optimized vertical line drawing into the raw canvas buffer.*
- `void drawFastRawHLine (int16_t x, int16_t y, int16_t w, uint16_t color)`  
*Speed optimized horizontal line drawing into the raw canvas buffer.*

## Additional Inherited Members

### 14.12.1 Detailed Description

A GFX 1-bit canvas context for graphics.

Definition at line 307 of file [batt\\_Adafruit\\_GFX.h](#).

### 14.12.2 Constructor & Destructor Documentation

#### 14.12.2.1 GFXcanvas1()

```
GFXcanvas1::GFXcanvas1 (
    uint16_t w,
    uint16_t h )
```

Instantiate a GFX 1-bit canvas context for graphics.

#### Parameters

<code>w</code>	Display width, in pixels
<code>h</code>	Display height, in pixels

Definition at line 1761 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.2.2 ~GFXcanvas1()

```
GFXcanvas1::~GFXcanvas1 (
    void )
```

Delete the canvas, free memory.

Definition at line 1773 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.12.3 Member Function Documentation

#### 14.12.3.1 drawFastHLine()

```
void GFXcanvas1::drawFastHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [virtual]
```

Speed optimized horizontal line drawing.

##### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>w</i>	Length of horizontal line to be drawn, including first point
<i>color</i>	Color to fill with

Reimplemented from [Adafruit\\_GFX](#).

Definition at line 1959 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.2 drawFastRawHLine()

```
void GFXcanvas1::drawFastRawHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [protected]
```

Speed optimized horizontal line drawing into the raw canvas buffer.

##### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>w</i>	length of horizontal line to be drawn, including first point
<i>color</i>	Binary (on or off) color to fill with

Definition at line 2053 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.3 drawFastRawVLine()

```
void GFXcanvas1::drawFastRawVLine (
    int16_t x,
```

```
int16_t y,
int16_t h,
uint16_t color ) [protected]
```

Speed optimized vertical line drawing into the raw canvas buffer.

#### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>h</i>	length of vertical line to be drawn, including first point
<i>color</i>	Binary (on or off) color to fill with

Definition at line 2014 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.4 drawFastVLine()

```
void GFXcanvas1::drawFastVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [virtual]
```

Speed optimized vertical line drawing.

#### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>h</i>	Length of vertical line to be drawn, including first point
<i>color</i>	Color to fill with

Reimplemented from [Adafruit\\_GFX](#).

Definition at line 1903 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.5 drawPixel()

```
void GFXcanvas1::drawPixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [virtual]
```

Draw a pixel to the canvas framebuffer.

#### Parameters

<i>x</i>	x coordinate
<i>y</i>	y coordinate
<i>color</i>	Binary (on or off) color to fill with

Implements [Adafruit\\_GFX](#).

Definition at line 1786 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.6 fillScreen()

```
void GFXcanvas1::fillScreen (
```

```
    uint16_t color ) [virtual]
```

Fill the framebuffer completely with one color.

#### Parameters

<i>color</i>	Binary (on or off) color to fill with
--------------	---------------------------------------

Reimplemented from [Adafruit\\_GFX](#).

Definition at line 1887 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.7 `getBuffer()`

```
uint8_t * GFXcanvas1::getBuffer (
    void ) const [inline]
```

Get a pointer to the internal buffer memory.

#### Returns

A pointer to the allocated buffer

Definition at line 322 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.12.3.8 `getPixel()`

```
bool GFXcanvas1::getPixel (
    int16_t x,
    int16_t y ) const

@brief Get the pixel color value at a given coordinate
@param x x coordinate
@param y y coordinate
@returns The desired pixel's binary color value, either 0x1 (on) or 0x0
```

(off)

Definition at line 1833 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.12.3.9 `getRawPixel()`

```
bool GFXcanvas1::getRawPixel (
    int16_t x,
    int16_t y ) const [protected]

@brief Get the pixel color value at a given, unrotated coordinate.
This method is intended for hardware drivers to get pixel value
in physical coordinates.
@param x x coordinate
@param y y coordinate
@returns The desired pixel's binary color value, either 0x1 (on) or 0x0
```

(off)

Definition at line 1865 of file [batt\\_Adafruit\\_GFX.cpp](#).

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

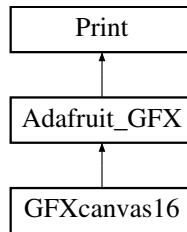
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.cpp](#)

## 14.13 GFXcanvas16 Class Reference

A GFX 16-bit canvas context for graphics.

```
#include <batt_Adafruit_GFX.h>
```

Inheritance diagram for GFXcanvas16:



### Public Member Functions

- `GFXcanvas16 (uint16_t w, uint16_t h)`  
*Instantiate a GFX 16-bit canvas context for graphics.*
- `~GFXcanvas16 (void)`  
*Delete the canvas, free memory.*
- `void drawPixel (int16_t x, int16_t y, uint16_t color)`  
*Draw a pixel to the canvas framebuffer.*
- `void fillScreen (uint16_t color)`  
*Fill the framebuffer completely with one color.*
- `void byteSwap (void)`  
*Reverses the "endian-ness" of each 16-bit pixel within the canvas; little-endian to big-endian, or big-endian to little. Most microcontrollers (such as SAMD) are little-endian, while most displays tend toward big-endianness. All the drawing functions (including RGB bitmap drawing) take care of this automatically, but some specialized code (usually involving DMA) can benefit from having pixel data already in the display-native order. Note that this does NOT convert to a SPECIFIC endian-ness, it just flips the bytes within each word.*
- `void drawFastVLine (int16_t x, int16_t y, int16_t h, uint16_t color)`  
*Speed optimized vertical line drawing.*
- `void drawFastHLine (int16_t x, int16_t y, int16_t w, uint16_t color)`  
*Speed optimized horizontal line drawing.*
- `uint16_t getPixel (int16_t x, int16_t y) const`  
*Get the pixel color value at a given coordinate.*
- `uint16_t * getBuffer (void) const`  
*Get a pointer to the internal buffer memory.*

### Protected Member Functions

- `uint16_t getRawPixel (int16_t x, int16_t y) const`  
*Get the pixel color value at a given, unrotated coordinate. This method is intended for hardware drivers to get pixel value in physical coordinates.*
- `void drawFastRawVLine (int16_t x, int16_t y, int16_t h, uint16_t color)`  
*Speed optimized vertical line drawing into the raw canvas buffer.*
- `void drawFastRawHLine (int16_t x, int16_t y, int16_t w, uint16_t color)`  
*Speed optimized horizontal line drawing into the raw canvas buffer.*

### Additional Inherited Members

#### 14.13.1 Detailed Description

A GFX 16-bit canvas context for graphics.

Definition at line 366 of file [batt\\_Adafruit\\_GFX.h](#).

## 14.13.2 Constructor & Destructor Documentation

### 14.13.2.1 GFXcanvas16()

```
GFXcanvas16::GFXcanvas16 (
    uint16_t w,
    uint16_t h )
```

Instantiate a GFX 16-bit canvas context for graphics.

#### Parameters

w	Display width, in pixels
h	Display height, in pixels

Definition at line [2385](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.13.2.2 ~GFXcanvas16()

```
GFXcanvas16::~GFXcanvas16 (
    void )
```

Delete the canvas, free memory.

Definition at line [2397](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

## 14.13.3 Member Function Documentation

### 14.13.3.1 byteSwap()

```
void GFXcanvas16::byteSwap (
    void )
```

Reverses the "endian-ness" of each 16-bit pixel within the canvas; little-endian to big-endian, or big-endian to little. Most microcontrollers (such as SAMD) are little-endian, while most displays tend toward big-endianness. All the drawing functions (including RGB bitmap drawing) take care of this automatically, but some specialized code (usually involving DMA) can benefit from having pixel data already in the display-native order. Note that this does NOT convert to a SPECIFIC endian-ness, it just flips the bytes within each word.

Definition at line [2517](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.13.3.2 drawFastHLine()

```
void GFXcanvas16::drawFastHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [virtual]
```

Speed optimized horizontal line drawing.

#### Parameters

x	Line horizontal start point
y	Line vertical start point
w	Length of horizontal line to be drawn, including 1st point
color	Color 16-bit 5-6-5 Color to draw line with

Reimplemented from [Adafruit\\_GFX](#).

Definition at line [2589](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.13.3.3 drawFastRawHLine()

```
void GFXcanvas16::drawFastRawHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [protected]
```

Speed optimized horizontal line drawing into the raw canvas buffer.

##### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>w</i>	length of horizontal line to be drawn, including first point
<i>color</i>	color 16-bit 5-6-5 Color to draw line with

Definition at line [2663](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.13.3.4 drawFastRawVLine()

```
void GFXcanvas16::drawFastRawVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [protected]
```

Speed optimized vertical line drawing into the raw canvas buffer.

##### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>h</i>	length of vertical line to be drawn, including first point
<i>color</i>	color 16-bit 5-6-5 Color to draw line with

Definition at line [2644](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.13.3.5 drawFastVLine()

```
void GFXcanvas16::drawFastVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [virtual]
```

Speed optimized vertical line drawing.

##### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>h</i>	length of vertical line to be drawn, including first point
<i>color</i>	color 16-bit 5-6-5 Color to draw line with

Reimplemented from [Adafruit\\_GFX](#).  
 Definition at line 2534 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.13.3.6 drawPixel()

```
void GFXcanvas16::drawPixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [virtual]
```

Draw a pixel to the canvas framebuffer.

##### Parameters

<i>x</i>	x coordinate
<i>y</i>	y coordinate
<i>color</i>	16-bit 5-6-5 Color to fill with

Implements [Adafruit\\_GFX](#).  
 Definition at line 2410 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.13.3.7 fillScreen()

```
void GFXcanvas16::fillScreen (
    uint16_t color ) [virtual]
```

Fill the framebuffer completely with one color.

##### Parameters

<i>color</i>	16-bit 5-6-5 Color to fill with
--------------	---------------------------------

Reimplemented from [Adafruit\\_GFX](#).  
 Definition at line 2491 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.13.3.8 getBuffer()

```
uint16_t * GFXcanvas16::getBuffer (
    void ) const [inline]
```

Get a pointer to the internal buffer memory.

##### Returns

A pointer to the allocated buffer

Definition at line 382 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.13.3.9 getPixel()

```
uint16_t GFXcanvas16::getPixel (
    int16_t x,
    int16_t y ) const
```

Get the pixel color value at a given coordinate.

##### Parameters

<i>x</i>	x coordinate
<i>y</i>	y coordinate

**Returns**

The desired pixel's 16-bit 5-6-5 color value

Definition at line 2445 of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.13.3.10 getRawPixel()**

```
uint16_t GFXcanvas16::getRawPixel (
    int16_t x,
    int16_t y ) const [protected]
```

Get the pixel color value at a given, unrotated coordinate. This method is intended for hardware drivers to get pixel value in physical coordinates.

**Parameters**

x	x coordinate
y	y coordinate

**Returns**

The desired pixel's 16-bit 5-6-5 color value

Definition at line 2476 of file [batt\\_Adafruit\\_GFX.cpp](#).

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

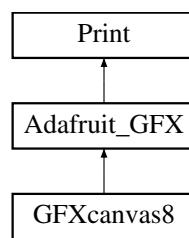
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.cpp](#)

## 14.14 GFXcanvas8 Class Reference

A GFX 8-bit canvas context for graphics.

```
#include <batt_Adafruit_GFX.h>
```

Inheritance diagram for GFXcanvas8:



### Public Member Functions

- [GFXcanvas8](#) (uint16\_t w, uint16\_t h)  
*Instantiate a GFX 8-bit canvas context for graphics.*
- [~GFXcanvas8](#) (void)  
*Delete the canvas, free memory.*
- void [drawPixel](#) (int16\_t x, int16\_t y, uint16\_t color)  
*Draw a pixel to the canvas framebuffer.*
- void [fillScreen](#) (uint16\_t color)  
*Fill the framebuffer completely with one color.*

- void `drawFastVLine` (int16\_t x, int16\_t y, int16\_t h, uint16\_t color)  
*Speed optimized vertical line drawing.*
- void `drawFastHLine` (int16\_t x, int16\_t y, int16\_t w, uint16\_t color)  
*Speed optimized horizontal line drawing.*
- uint8\_t `getPixel` (int16\_t x, int16\_t y) const  
*Get the pixel color value at a given coordinate.*
- uint8\_t \* `getBuffer` (void) const  
*Get a pointer to the internal buffer memory.*

## Protected Member Functions

- uint8\_t `getRawPixel` (int16\_t x, int16\_t y) const  
*Get the pixel color value at a given, unrotated coordinate. This method is intended for hardware drivers to get pixel value in physical coordinates.*
- void `drawFastRawVLine` (int16\_t x, int16\_t y, int16\_t h, uint16\_t color)  
*Speed optimized vertical line drawing into the raw canvas buffer.*
- void `drawFastRawHLine` (int16\_t x, int16\_t y, int16\_t w, uint16\_t color)  
*Speed optimized horizontal line drawing into the raw canvas buffer.*

## Additional Inherited Members

### 14.14.1 Detailed Description

A GFX 8-bit canvas context for graphics.

Definition at line 339 of file [batt\\_Adafruit\\_GFX.h](#).

### 14.14.2 Constructor & Destructor Documentation

#### 14.14.2.1 `GFXcanvas8()`

```
GFXcanvas8::GFXcanvas8 (
    uint16_t w,
    uint16_t h )
```

Instantiate a GFX 8-bit canvas context for graphics.

##### Parameters

<code>w</code>	Display width, in pixels
<code>h</code>	Display height, in pixels

Definition at line 2117 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.14.2.2 `~GFXcanvas8()`

```
GFXcanvas8::~GFXcanvas8 (
    void )
```

Delete the canvas, free memory.

Definition at line 2129 of file [batt\\_Adafruit\\_GFX.cpp](#).

### 14.14.3 Member Function Documentation

**14.14.3.1 drawFastHLine()**

```
void GFXcanvas8::drawFastHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [virtual]
```

Speed optimized horizontal line drawing.

**Parameters**

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>w</i>	Length of horizontal line to be drawn, including 1st point
<i>color</i>	8-bit Color to fill with. Only lower byte of uint16_t is used.

Reimplemented from [Adafruit\\_GFX](#).

Definition at line [2295](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.14.3.2 drawFastRawHLine()**

```
void GFXcanvas8::drawFastRawHLine (
    int16_t x,
    int16_t y,
    int16_t w,
    uint16_t color ) [protected]
```

Speed optimized horizontal line drawing into the raw canvas buffer.

**Parameters**

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>w</i>	length of horizontal line to be drawn, including first point
<i>color</i>	8-bit Color to fill with. Only lower byte of uint16_t is used.

Definition at line [2372](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

**14.14.3.3 drawFastRawVLine()**

```
void GFXcanvas8::drawFastRawVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [protected]
```

Speed optimized vertical line drawing into the raw canvas buffer.

**Parameters**

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>h</i>	length of vertical line to be drawn, including first point
<i>color</i>	8-bit Color to fill with. Only lower byte of uint16_t is used.

Definition at line [2352](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.14.3.4 drawFastVLine()

```
void GFXcanvas8::drawFastVLine (
    int16_t x,
    int16_t y,
    int16_t h,
    uint16_t color ) [virtual]
```

Speed optimized vertical line drawing.

##### Parameters

<i>x</i>	Line horizontal start point
<i>y</i>	Line vertical start point
<i>h</i>	Length of vertical line to be drawn, including first point
<i>color</i>	8-bit Color to fill with. Only lower byte of uint16_t is used.

Reimplemented from [Adafruit\\_GFX](#).

Definition at line [2239](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.14.3.5 drawPixel()

```
void GFXcanvas8::drawPixel (
    int16_t x,
    int16_t y,
    uint16_t color ) [virtual]
```

Draw a pixel to the canvas framebuffer.

##### Parameters

<i>x</i>	x coordinate
<i>y</i>	y coordinate
<i>color</i>	8-bit Color to fill with. Only lower byte of uint16_t is used.

Implements [Adafruit\\_GFX](#).

Definition at line [2142](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.14.3.6 fillScreen()

```
void GFXcanvas8::fillScreen (
    uint16_t color ) [virtual]
```

Fill the framebuffer completely with one color.

##### Parameters

<i>color</i>	8-bit Color to fill with. Only lower byte of uint16_t is used.
--------------	--

Reimplemented from [Adafruit\\_GFX](#).

Definition at line [2223](#) of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.14.3.7 getBuffer()

```
uint8_t * GFXcanvas8::getBuffer (
```

```
void ) const [inline]
```

Get a pointer to the internal buffer memory.

#### Returns

A pointer to the allocated buffer

Definition at line 354 of file [batt\\_Adafruit\\_GFX.h](#).

#### 14.14.3.8 getPixel()

```
uint8_t GFXcanvas8::getPixel (
    int16_t x,
    int16_t y ) const
```

Get the pixel color value at a given coordinate.

#### Parameters

x	x coordinate
y	y coordinate

#### Returns

The desired pixel's 8-bit color value

Definition at line 2177 of file [batt\\_Adafruit\\_GFX.cpp](#).

#### 14.14.3.9 getRawPixel()

```
uint8_t GFXcanvas8::getRawPixel (
    int16_t x,
    int16_t y ) const [protected]
```

Get the pixel color value at a given, unrotated coordinate. This method is intended for hardware drivers to get pixel value in physical coordinates.

#### Parameters

x	x coordinate
y	y coordinate

#### Returns

The desired pixel's 8-bit color value

Definition at line 2208 of file [batt\\_Adafruit\\_GFX.cpp](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_← sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_← sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_Adafruit\\_GFX.cpp](#)

## 14.15 GFXfont Struct Reference

Data stored for FONT AS A WHOLE.

```
#include <batt_gfxfont.h>
```

## Public Attributes

- `uint8_t * bitmap`  
*Glyph bitmaps, concatenated.*
- `GFXglyph * glyph`  
*Glyph array.*
- `uint16_t first`  
*ASCII extents (first char)*
- `uint16_t last`  
*ASCII extents (last char)*
- `uint8_t yAdvance`  
*Newline distance (y axis)*

### 14.15.1 Detailed Description

Data stored for FONT AS A WHOLE.

Definition at line 21 of file [batt\\_gfxfont.h](#).

### 14.15.2 Member Data Documentation

#### 14.15.2.1 `bitmap`

```
uint8_t* GFXfont::bitmap
Glyph bitmaps, concatenated.
Definition at line 22 of file batt\_gfxfont.h.
```

#### 14.15.2.2 `first`

```
uint16_t GFXfont::first
ASCII extents (first char)
Definition at line 24 of file batt\_gfxfont.h.
```

#### 14.15.2.3 `glyph`

```
GFXglyph* GFXfont::glyph
Glyph array.
Definition at line 23 of file batt\_gfxfont.h.
```

#### 14.15.2.4 `last`

```
uint16_t GFXfont::last
ASCII extents (last char)
Definition at line 25 of file batt\_gfxfont.h.
```

#### 14.15.2.5 `yAdvance`

```
uint8_t GFXfont::yAdvance
Newline distance (y axis)
Definition at line 26 of file batt\_gfxfont.h.
```

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_←  
sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_gfxfont.h](#)

## 14.16 GFXglyph Struct Reference

Font data stored PER GLYPH.

```
#include <batt_gfxfont.h>
```

### Public Attributes

- `uint16_t bitmapOffset`  
*Pointer into GFXfont->bitmap.*
- `uint8_t width`  
*Bitmap dimensions in pixels.*
- `uint8_t height`  
*Bitmap dimensions in pixels.*
- `uint8_t xAdvance`  
*Distance to advance cursor (x axis)*
- `int8_t xOffset`  
*X dist from cursor pos to UL corner.*
- `int8_t yOffset`  
*Y dist from cursor pos to UL corner.*

### 14.16.1 Detailed Description

Font data stored PER GLYPH.

Definition at line 11 of file [batt\\_gfxfont.h](#).

### 14.16.2 Member Data Documentation

#### 14.16.2.1 bitmapOffset

```
uint16_t GFXglyph::bitmapOffset
```

Pointer into GFXfont->bitmap.

Definition at line 12 of file [batt\\_gfxfont.h](#).

#### 14.16.2.2 height

```
uint8_t GFXglyph::height
```

Bitmap dimensions in pixels.

Definition at line 14 of file [batt\\_gfxfont.h](#).

#### 14.16.2.3 width

```
uint8_t GFXglyph::width
```

Bitmap dimensions in pixels.

Definition at line 13 of file [batt\\_gfxfont.h](#).

#### 14.16.2.4 xAdvance

```
uint8_t GFXglyph::xAdvance
```

Distance to advance cursor (x axis)

Definition at line 15 of file [batt\\_gfxfont.h](#).

#### 14.16.2.5 xOffset

```
int8_t GFXglyph::xOffset
X dist from cursor pos to UL corner.
Definition at line 16 of file batt\_gfxfont.h.
```

#### 14.16.2.6 yOffset

```
int8_t GFXglyph::yOffset
Y dist from cursor pos to UL corner.
Definition at line 17 of file batt\_gfxfont.h.
The documentation for this struct was generated from the following file:
```

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/[batt\\_gfxfont.h](#)

## 14.17 HealthMonitor Class Reference

```
#include <HealthMonitor.h>
```

### Public Member Functions

- **HealthMonitor** (uint16\_t thres, uint8\_t inc, uint8\_t dec, uint16\_t lim)
 

*Construct a new Health Monitor object.*
- void **setCounter** (uint16\_t value)
 

*Set the Counter object.*
- uint16\_t **getCounter** ()
 

*Get the Counter object.*
- boolean **check** (uint16\_t sample)
 

*Check the input and return an alarm if the counter exceed the count limit..*
- uint16\_t **getSample** (uint16\_t ADCpin)
 

*Get the Sample object from an ADC pin.*

### Public Attributes

- uint16\_t **threshold**
- uint8\_t **rampUPinc**
- uint8\_t **rampDOWNdec**
- uint16\_t **limit**
- bool **alarm**
- int16\_t **sense\_values** [8]

#### 14.17.1 Detailed Description

Definition at line 20 of file [HealthMonitor.h](#).

#### 14.17.2 Constructor & Destructor Documentation

**14.17.2.1 HealthMonitor()**

```
HealthMonitor::HealthMonitor (
    uint16_t thres,
    uint8_t inc,
    uint8_t dec,
    uint16_t lim ) [inline]
```

Construct a new Health Monitor object.

**Parameters**

<i>thres</i>	<input type="text"/>
<i>inc</i>	<input type="text"/>
<i>dec</i>	<input type="text"/>
<i>lim</i>	<input type="text"/>

Definition at line 41 of file [HealthMonitor.h](#).

**14.17.3 Member Function Documentation****14.17.3.1 check()**

```
boolean HealthMonitor::check (
    uint16_t sample ) [inline]
```

Check the input and return an alarm if the counter exceed the count limit..

**Parameters**

<i>sample</i>	<input type="text"/>
---------------	----------------------

**Returns**

boolean

Definition at line 79 of file [HealthMonitor.h](#).

**14.17.3.2 getCounter()**

```
uint16_t HealthMonitor::getCounter ( ) [inline]
```

Get the Counter object.

**Returns**

uint16\_t

Definition at line 64 of file [HealthMonitor.h](#).

**14.17.3.3 getSample()**

```
uint16_t HealthMonitor::getSample (
    uint16_t ADCpin ) [inline]
```

Get the Sample object from an ADC pin.

**Parameters**

<i>ADCpin</i>	<input type="text"/>
---------------	----------------------

**Returns**

```
uint16_t
```

Definition at line 111 of file [HealthMonitor.h](#).

**14.17.3.4 setCounter()**

```
void HealthMonitor::setCounter (
    uint16_t value ) [inline]
```

Set the Counter object.

**Parameters**

value	<input type="text"/>
-------	----------------------

Definition at line 54 of file [HealthMonitor.h](#).

**14.17.4 Member Data Documentation****14.17.4.1 alarm**

```
bool HealthMonitor::alarm
```

Definition at line 30 of file [HealthMonitor.h](#).

**14.17.4.2 limit**

```
uint16_t HealthMonitor::limit
```

Definition at line 29 of file [HealthMonitor.h](#).

**14.17.4.3 rampDOWNdec**

```
uint8_t HealthMonitor::rampDOWNdec
```

Definition at line 28 of file [HealthMonitor.h](#).

**14.17.4.4 rampUPinc**

```
uint8_t HealthMonitor::rampUPinc
```

Definition at line 27 of file [HealthMonitor.h](#).

**14.17.4.5 sense\_values**

```
int16_t HealthMonitor::sense_values[8]
```

Definition at line 31 of file [HealthMonitor.h](#).

**14.17.4.6 threshold**

```
uint16_t HealthMonitor::threshold
```

Definition at line 26 of file [HealthMonitor.h](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_←  
sw/LIBRARIES/0005-SENSING/[HealthMonitor.h](#)

## 14.18 MilliTimer Class Reference

```
#include <MilliTimer.h>
```

### Public Member Functions

- `MilliTimer ()`
- byte `poll (word ms=0)`  
*Return the number of milliseconds before the timer will fire.*
- word `remaining () const`  
*Returns true if the timer is not armed.*
- void `set (word ms)`

### 14.18.1 Detailed Description

Definition at line 6 of file [MilliTimer.h](#).

### 14.18.2 Constructor & Destructor Documentation

#### 14.18.2.1 MilliTimer()

```
MilliTimer::MilliTimer ( ) [inline]
```

Definition at line 12 of file [MilliTimer.h](#).

### 14.18.3 Member Function Documentation

#### 14.18.3.1 idle()

```
byte MilliTimer::idle (
```

void ) const [inline]
-----------------------

Returns true if the timer is not armed.  
Definition at line 20 of file [MilliTimer.h](#).

#### 14.18.3.2 poll()

```
byte MilliTimer::poll (
```

word ms = 0 )
---------------

poll until the timer fires

##### Parameters

<code>ms</code>	Periodic repeat rate of the time, omit for a one-shot timer.
-----------------	--

Definition at line 4 of file [MilliTimer.cpp](#).

#### 14.18.3.3 remaining()

```
word MilliTimer::remaining ( ) const
```

Return the number of milliseconds before the timer will fire.  
Definition at line 24 of file [MilliTimer.cpp](#).

#### 14.18.3.4 set()

```
void MilliTimmer::set (
    word ms )
set the one-shot timeout value
```

##### Parameters

<i>ms</i>	Timeout value. Timer stops once the timer has fired.
-----------	--

Definition at line 30 of file [MilliTimmer.cpp](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0014-MilliTimmer/[MilliTimmer.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0014-MilliTimmer/[MilliTimmer.cpp](#)

## 14.19 Person Struct Reference

### Public Attributes

- boolean [valid](#)
- char [name](#) [100]
- char [surname](#) [100]

#### 14.19.1 Detailed Description

Definition at line 14 of file [StoreNameAndSurname.ino](#).

#### 14.19.2 Member Data Documentation

##### 14.19.2.1 name

```
char Person::name[100]
```

Definition at line 16 of file [StoreNameAndSurname.ino](#).

##### 14.19.2.2 surname

```
char Person::surname[100]
```

Definition at line 17 of file [StoreNameAndSurname.ino](#).

##### 14.19.2.3 valid

```
boolean Person::valid
```

Definition at line 15 of file [StoreNameAndSurname.ino](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/examples/StoreNameAndSurname/[StoreNameAndSurname.ino](#)

## 14.20 RTCZero Class Reference

```
#include <batt_RTCZero.h>
```

## Public Types

- enum `Alarm_Match` : `uint8_t` {  
    `MATCH_OFF` = `RTC_MODE2_MASK_SEL_OFF_Val` , `MATCH_SS` = `RTC_MODE2_MASK_SEL_SS_Val` ,  
    `MATCH_MMSS` = `RTC_MODE2_MASK_SEL_MMSS_Val` , `MATCH_HHMMSS` = `RTC_MODE2_MASK_SEL_HHMMSS_Val` ,  
    `MATCH_DHHMMSS` = `RTC_MODE2_MASK_SEL_DDHHMMSS_Val` , `MATCH_MMDDHHMMSS` = `RTC_MODE2_MASK_SEL_MMDDHHMMSS_Val` ,  
    `MATCH_YYMMDDHHMMSS` = `RTC_MODE2_MASK_SEL_YYMMDDHHMMSS_Val` }

## Public Member Functions

- `RTCZero ()`
- void `begin` (bool `resetTime=false`)
- void `enableAlarm` (`Alarm_Match` `match`)
- void `disableAlarm` ()
- void `attachInterrupt` (`voidFuncPtr` `callback`)
- void `detachInterrupt` ()
- void `standbyMode` ()
- `uint8_t getSeconds ()`
- `uint8_t getMinutes ()`
- `uint8_t getHours ()`
- `uint8_t getDay ()`
- `uint8_t getMonth ()`
- `uint8_t getYear ()`
- `uint8_t getAlarmSeconds ()`
- `uint8_t getAlarmMinutes ()`
- `uint8_t getAlarmHours ()`
- `uint8_t getAlarmDay ()`
- `uint8_t getAlarmMonth ()`
- `uint8_t getAlarmYear ()`
- void `setSeconds` (`uint8_t seconds`)
- void `setMinutes` (`uint8_t minutes`)
- void `setHours` (`uint8_t hours`)
- void `setTime` (`uint8_t hours`, `uint8_t minutes`, `uint8_t seconds`)
- void `setDay` (`uint8_t day`)
- void `setMonth` (`uint8_t month`)
- void `setYear` (`uint8_t year`)
- void `setDate` (`uint8_t day`, `uint8_t month`, `uint8_t year`)
- void `setAlarmSeconds` (`uint8_t seconds`)
- void `setAlarmMinutes` (`uint8_t minutes`)
- void `setAlarmHours` (`uint8_t hours`)
- void `setAlarmTime` (`uint8_t hours`, `uint8_t minutes`, `uint8_t seconds`)
- void `setAlarmDay` (`uint8_t day`)
- void `setAlarmMonth` (`uint8_t month`)
- void `setAlarmYear` (`uint8_t year`)
- void `setAlarmDate` (`uint8_t day`, `uint8_t month`, `uint8_t year`)
- `uint32_t getEpoch ()`
- `uint32_t getY2kEpoch ()`
- void `setEpoch` (`uint32_t ts`)
- void `setY2kEpoch` (`uint32_t ts`)
- void `setAlarmEpoch` (`uint32_t ts`)
- bool `isConfigured ()`

### 14.20.1 Detailed Description

Definition at line 27 of file `batt_RTCZero.h`.

## 14.20.2 Member Enumeration Documentation

### 14.20.2.1 Alarm\_Match

enum `RTCZero::Alarm_Match` : `uint8_t`

Enumerator

<code>MATCH_OFF</code>	
<code>MATCH_SS</code>	
<code>MATCH_MMSS</code>	
<code>MATCH_HHMMSS</code>	
<code>MATCH_DHHMMSS</code>	
<code>MATCH_MMDDHHMMSS</code>	
<code>MATCH_YYMMDDHHMMSS</code>	

Definition at line 30 of file [batt\\_RTCZero.h](#).

## 14.20.3 Constructor & Destructor Documentation

### 14.20.3.1 RTCZero()

`RTCZero::RTCZero( )`

Definition at line 37 of file [batt\\_RTCZero.cpp](#).

## 14.20.4 Member Function Documentation

### 14.20.4.1 attachInterrupt()

```
void RTCZero::attachInterrupt (
    voidFuncPtr callback )
```

Definition at line 136 of file [batt\\_RTCZero.cpp](#).

### 14.20.4.2 begin()

```
void RTCZero::begin (
    bool resetTime = false )
```

Definition at line 42 of file [batt\\_RTCZero.cpp](#).

### 14.20.4.3 detachInterrupt()

```
void RTCZero::detachInterrupt ( )
```

Definition at line 141 of file [batt\\_RTCZero.cpp](#).

### 14.20.4.4 disableAlarm()

```
void RTCZero::disableAlarm ( )
```

Definition at line 127 of file [batt\\_RTCZero.cpp](#).

**14.20.4.5 enableAlarm()**

```
void RTCZero::enableAlarm (
    Alarm_Match match )
```

Definition at line 118 of file [batt\\_RTCZero.cpp](#).

**14.20.4.6 getAlarmDay()**

```
uint8_t RTCZero::getAlarmDay ( )
```

Definition at line 210 of file [batt\\_RTCZero.cpp](#).

**14.20.4.7 getAlarmHours()**

```
uint8_t RTCZero::getAlarmHours ( )
```

Definition at line 205 of file [batt\\_RTCZero.cpp](#).

**14.20.4.8 getAlarmMinutes()**

```
uint8_t RTCZero::getAlarmMinutes ( )
```

Definition at line 200 of file [batt\\_RTCZero.cpp](#).

**14.20.4.9 getAlarmMonth()**

```
uint8_t RTCZero::getAlarmMonth ( )
```

Definition at line 215 of file [batt\\_RTCZero.cpp](#).

**14.20.4.10 getAlarmSeconds()**

```
uint8_t RTCZero::getAlarmSeconds ( )
```

Definition at line 195 of file [batt\\_RTCZero.cpp](#).

**14.20.4.11 getAlarmYear()**

```
uint8_t RTCZero::getAlarmYear ( )
```

Definition at line 220 of file [batt\\_RTCZero.cpp](#).

**14.20.4.12 getDay()**

```
uint8_t RTCZero::getDay ( )
```

Definition at line 177 of file [batt\\_RTCZero.cpp](#).

**14.20.4.13 getEpoch()**

```
uint32_t RTCZero::getEpoch ( )
```

Definition at line 373 of file [batt\\_RTCZero.cpp](#).

**14.20.4.14 getHours()**

```
uint8_t RTCZero::getHours ( )
```

Definition at line 171 of file [batt\\_RTCZero.cpp](#).

**14.20.4.15 getMinutes()**

```
uint8_t RTCZero::getMinutes ( )  
Definition at line 165 of file batt\_RTCZero.cpp.
```

**14.20.4.16 getMonth()**

```
uint8_t RTCZero::getMonth ( )  
Definition at line 183 of file batt\_RTCZero.cpp.
```

**14.20.4.17 getSeconds()**

```
uint8_t RTCZero::getSeconds ( )  
Definition at line 159 of file batt\_RTCZero.cpp.
```

**14.20.4.18 getY2kEpoch()**

```
uint32_t RTCZero::getY2kEpoch ( )  
Definition at line 394 of file batt\_RTCZero.cpp.
```

**14.20.4.19 getYear()**

```
uint8_t RTCZero::getYear ( )  
Definition at line 189 of file batt\_RTCZero.cpp.
```

**14.20.4.20 isConfigured()**

```
bool RTCZero::isConfigured ( ) [inline]  
Definition at line 100 of file batt\_RTCZero.h.
```

**14.20.4.21 setAlarmDate()**

```
void RTCZero::setAlarmDate (   
    uint8_t day,  
    uint8_t month,  
    uint8_t year )  
Definition at line 364 of file batt\_RTCZero.cpp.
```

**14.20.4.22 setAlarmDay()**

```
void RTCZero::setAlarmDay (   
    uint8_t day )  
Definition at line 337 of file batt\_RTCZero.cpp.
```

**14.20.4.23 setAlarmEpoch()**

```
void RTCZero::setAlarmEpoch (   
    uint32_t ts )  
Definition at line 399 of file batt\_RTCZero.cpp.
```

**14.20.4.24 setAlarmHours()**

```
void RTCZero::setAlarmHours (
    uint8_t hours )
```

Definition at line 319 of file [batt\\_RTCZero.cpp](#).

**14.20.4.25 setAlarmMinutes()**

```
void RTCZero::setAlarmMinutes (
    uint8_t minutes )
```

Definition at line 310 of file [batt\\_RTCZero.cpp](#).

**14.20.4.26 setAlarmMonth()**

```
void RTCZero::setAlarmMonth (
    uint8_t month )
```

Definition at line 346 of file [batt\\_RTCZero.cpp](#).

**14.20.4.27 setAlarmSeconds()**

```
void RTCZero::setAlarmSeconds (
    uint8_t seconds )
```

Definition at line 301 of file [batt\\_RTCZero.cpp](#).

**14.20.4.28 setAlarmTime()**

```
void RTCZero::setAlarmTime (
    uint8_t hours,
    uint8_t minutes,
    uint8_t seconds )
```

Definition at line 328 of file [batt\\_RTCZero.cpp](#).

**14.20.4.29 setAlarmYear()**

```
void RTCZero::setAlarmYear (
    uint8_t year )
```

Definition at line 355 of file [batt\\_RTCZero.cpp](#).

**14.20.4.30 setDate()**

```
void RTCZero::setDate (
    uint8_t day,
    uint8_t month,
    uint8_t year )
```

Definition at line 292 of file [batt\\_RTCZero.cpp](#).

**14.20.4.31 setDay()**

```
void RTCZero::setDay (
    uint8_t day )
```

Definition at line 265 of file [batt\\_RTCZero.cpp](#).

**14.20.4.32 setEpoch()**

```
void RTCZero::setEpoch (
    uint32_t ts )
```

Definition at line 414 of file [batt\\_RTCZero.cpp](#).

**14.20.4.33 setHours()**

```
void RTCZero::setHours (
    uint8_t hours )
```

Definition at line 247 of file [batt\\_RTCZero.cpp](#).

**14.20.4.34 setMinutes()**

```
void RTCZero::setMinutes (
    uint8_t minutes )
```

Definition at line 238 of file [batt\\_RTCZero.cpp](#).

**14.20.4.35 setMonth()**

```
void RTCZero::setMonth (
    uint8_t month )
```

Definition at line 274 of file [batt\\_RTCZero.cpp](#).

**14.20.4.36 setSeconds()**

```
void RTCZero::setSeconds (
    uint8_t seconds )
```

Definition at line 229 of file [batt\\_RTCZero.cpp](#).

**14.20.4.37 setTime()**

```
void RTCZero::setTime (
    uint8_t hours,
    uint8_t minutes,
    uint8_t seconds )
```

Definition at line 256 of file [batt\\_RTCZero.cpp](#).

**14.20.4.38 setY2kEpoch()**

```
void RTCZero::setY2kEpoch (
    uint32_t ts )
```

Definition at line 440 of file [batt\\_RTCZero.cpp](#).

**14.20.4.39 setYear()**

```
void RTCZero::setYear (
    uint8_t year )
```

Definition at line 283 of file [batt\\_RTCZero.cpp](#).

#### 14.20.4.40 standbyMode()

```
void RTCZero::standbyMode ( )
```

Definition at line 146 of file [batt\\_RTCZero.cpp](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/src/[batt\\_RTCZero.h](#)
- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/src/[batt\\_RTCZero.cpp](#)

## 14.21 SlowSoftI2CMaster Class Reference

```
#include <batt_SlowSoftI2CMaster.h>
```

### Public Member Functions

- [SlowSoftI2CMaster](#) (uint8\_t sda, uint8\_t scl)
- [SlowSoftI2CMaster](#) (uint8\_t sda, uint8\_t scl, bool internal\_pullup)
- bool [i2c\\_init](#) (void)
- bool [i2c\\_start](#) (uint8\_t addr)
- bool [i2c\\_start\\_wait](#) (uint8\_t addr)
- bool [i2c\\_rep\\_start](#) (uint8\_t addr)
- void [i2c\\_stop](#) (void)
- bool [i2c\\_write](#) (uint8\_t value)
- uint8\_t [i2c\\_read](#) (bool last)

### Public Attributes

- bool [error](#)

#### 14.21.1 Detailed Description

Definition at line 31 of file [batt\\_SlowSoftI2CMaster.h](#).

#### 14.21.2 Constructor & Destructor Documentation

##### 14.21.2.1 SlowSoftI2CMaster() [1/2]

```
SlowSoftI2CMaster::SlowSoftI2CMaster (
    uint8_t sda,
    uint8_t scl )
```

Definition at line 22 of file [batt\\_SlowSoftI2CMaster.cpp](#).

##### 14.21.2.2 SlowSoftI2CMaster() [2/2]

```
SlowSoftI2CMaster::SlowSoftI2CMaster (
    uint8_t sda,
    uint8_t scl,
    bool internal_pullup )
```

Definition at line 28 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3 Member Function Documentation

#### 14.21.3.1 i2c\_init()

```
bool SlowSoftI2CMaster::i2c_init (
    void )
```

Definition at line 36 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3.2 i2c\_read()

```
uint8_t SlowSoftI2CMaster::i2c_read (
    bool last )
```

Definition at line 109 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3.3 i2c\_rep\_start()

```
bool SlowSoftI2CMaster::i2c_rep_start (
    uint8_t addr )
```

Definition at line 69 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3.4 i2c\_start()

```
bool SlowSoftI2CMaster::i2c_start (
    uint8_t addr )
```

Definition at line 48 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3.5 i2c\_start\_wait()

```
bool SlowSoftI2CMaster::i2c_start_wait (
    uint8_t addr )
```

Definition at line 56 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3.6 i2c\_stop()

```
void SlowSoftI2CMaster::i2c_stop (
    void )
```

Definition at line 77 of file [batt\\_SlowSoftI2CMaster.cpp](#).

#### 14.21.3.7 i2c\_write()

```
bool SlowSoftI2CMaster::i2c_write (
    uint8_t value )
```

Definition at line 89 of file [batt\\_SlowSoftI2CMaster.cpp](#).

### 14.21.4 Member Data Documentation

#### 14.21.4.1 error

```
bool SlowSoftI2CMaster::error
```

Definition at line 42 of file [batt\\_SlowSoftI2CMaster.h](#).

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

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0007-SlowSoftI2CMaster/[batt\\_SlowSoftI2CMaster.h](#)

- C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_↪  
sw/LIBRARIES/0007-SlowSoftI2CMaster/batt\_SlowSoftI2CMaster.cpp



# Chapter 15

## File Documentation

### 15.1 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/CODE/B1/B1.ino File Reference

V1: - Gestión y detección del modo Boost/No boost. Cuando la corriente de salida sea superior a un umbral durante un determinado timep, se deberá activar la salida en modo BOOST, mientras que si se encuentra por debajo del umbral durante otro determinado tiempo, se deberá desactivar la salida modo BOOST.

```
#include <MilliTimer.h>
#include <DCDC.h>
#include <HealthMonitor.h>
#include <display.h>
#include <Dpad.h>
#include <Buzzer.h>
#include <diagnostic.h>
#include <power_bar.h>
#include <batt_ArduinoLowPower.h>
```

### Functions

- `HealthMonitor voltage_input_protection (4096 - C_LIMIT_VOLTAGE_INPUT, 10, 10, 200)`  
*HealthMonitor del Vin de entrada.*
- `void setup ()`
- `void loop ()`
- `void IrqCenterButtonHandler ()`
- `void Irq_DeathBattery_Handler ()`
- `void Irq_USB_Handler ()`

### Variables

- `const int16_t C_PIN_I_OUT = A1`
- `const int16_t C_PIN_V_IN = A4`
- `const int16_t C_PIN_V_OUT = A2`
- `const uint16_t C_PIN_EN_DCDC = 2`
- `const uint16_t C_PIN_SHIPPING_MOD = 3`
- `const uint16_t C_PIN_USB_CONNEXION = 7`
- `const uint16_t C_PIN_DEATH_BATTERY_CHECK = 10`
- `const uint16_t C_PIN_TEST_MODE = 11`
- `const int16_t C_PIN_FLAG_CHARG = 0`
- `const int16_t C_PIN_SCL_2 = 1`
- `const int16_t C_PIN_SDA_2 = 6`

- const int16\_t **C\_PIN\_FLAG\_BATTLOW** = 10
- const int16\_t **C\_PIN\_USB\_SEL** = 11
- const bool **C\_MASK\_DEACTIVATE** = false
- const bool **C\_MASK\_ACTIVATE** = true
- const int16\_t **C\_MIN\_VOLT** = 50
- const int16\_t **C\_MAX\_VOLT** = 160
- const uint16\_t **MAX\_NUM\_PROTECTION\_EVENTS** = 6
- const bool **C\_OFF** = false
- const bool **C\_ON** = true
- const int16\_t **C\_TIMER\_ARMED** = 0
- const int16\_t **C\_TIMER\_IDLE** = 1
- const int16\_t **C\_TIMER\_DONE** = 2
- const int16\_t **C\_SW\_ST\_RUN** = 0x00
- const int16\_t **C\_SW\_ST\_SLEEP** = 0x01
- const int16\_t **C\_SW\_ST\_START\_UP** = 0x02
- const int16\_t **C\_SW\_ST\_ERROR** = 0x03
- const int16\_t **C\_SW\_ST\_STOP** = 0x04
- const int16\_t **C\_SW\_ST\_DIAGNOSTIC** = 0x05
- const int16\_t **C\_SW\_ST\_CAPACITY** = 0x06
- const int16\_t **C\_SW\_ST\_USB** = 0x07
- const int16\_t **C\_LIMIT\_CONSUPTION\_PROT** = 450 \* 4096 / 3300
- const int16\_t **C\_LIMIT\_UNDERVOLTAGE\_PROT** = 2000 \* 4096 / 3300
- const int16\_t **C\_LIMIT\_VOLTAGE\_INPUT** = 5000 - 3000
- const int16\_t **C\_LIMIT\_OVERPOWER\_PROT** = 4000
- const int16\_t **C\_RETRY\_750\_COUNT** = 750
- const int32\_t **C\_TIME\_TO\_LIGHT\_DOWN** = 5000
- const uint32\_t **C\_WATCH\_DOG\_TIME\_MS** = 1000
- const String **C\_DIAGNOSTIC\_PASSWORD** = "FPO"
- const bool **C\_USB\_CONNECTED** = true
- const bool **C\_USB\_DISCONNECTED** = false
- const uint8\_t **C\_NUMB\_PRINTS\_STATIC\_DATA** = 5
- const uint8\_t **C\_LOW\_BATTERY\_LEVEL** = 10
- const uint16\_t **C\_IDLE\_TIMER\_COUNT** = 1 \* 2
- const String **C\_CMD\_DISPLAY\_STARTING\_OFF** = "DspStOFF"
- const String **C\_CMD\_DISPLAY\_STARTING\_ON** = "DspStON"
- const String **C\_CMD\_DISPLAY\_ENDING\_OFF** = "DspEndOFF"
- const String **C\_CMD\_DISPLAY\_ENDING\_ON** = "DspEndON"
- const String **C\_CMD\_SOUND\_STARTING\_OFF** = "SndStrOFF"
- const String **C\_CMD\_SOUND\_STARTING\_ON** = "SndSrtON"
- const String **C\_CMD\_SOUND\_ENDING\_OFF** = "SndEndOFF"
- const String **C\_CMD\_SOUND\_ENDING\_ON** = "SndEndON"
- const String **C\_CMD\_SOUND\_FULL\_CHARGE\_OFF** = "SndFChgOFF"
- const String **C\_CMD\_SOUND\_FULL\_CHARGE\_ON** = "SndFChgOFF"
- const String **C\_CMS\_SOUND\_DEATH\_BATTERY\_OFF** = "SndDthBttOFF"
- const String **C\_CMS\_SOUND\_DEATH\_BATTERY\_ON** = "SndDthBttON"
- const String **C\_CMS\_SOUND\_CHARGE\_START\_OFF** = "SndChgStOFF"
- const String **C\_CMS\_SOUND\_CHARGE\_START\_ON** = "SndChgStON"
- const String **C\_CMD\_LIVE\_OFF** = "LiveOFF"
- const String **C\_CMD\_LIVE\_ON** = "LiveON"
- const String **C\_CMD\_DIAGNOSTIC\_STOP** = "GO\_SLEEP"
- const String **C\_LIVE\_CMD\_OUT\_ON** = "EnableOut"
- const String **C\_LIVE\_CMD\_OUT\_OFF** = "DisableOut"
- const String **C\_LIVE\_CMD\_VOLT\_UP\_1** = "VoltUp1"
- const String **C\_LIVE\_CMD\_VOLT\_UP\_10** = "VoltUp10"
- const String **C\_LIVE\_CMD\_VOLT\_DOWN\_1** = "VoltDown1"

- const String **C\_LIVE\_CMD\_VOLT\_DOWN\_10** = "VoltDown10"
- const String **C\_LIVE\_CMD\_OFF\_LIVE\_VIEW** = "LiveViewOff"
- **HealthMonitor boost\_check** (50, 10, 1, 100)  
*HealthMonitor de la corriente de salida para el modo boost.*
- **HealthMonitor over\_consumption\_protection** (**C\_LIMIT\_COMSUPTION\_PROT**, 10, 10, 1500)  
*HealthMonitor del consumo de salida de la bateria.*
- **HealthMonitor over\_power\_protection** (**C\_LIMIT\_OVERPOWER\_PROT**, 10, 10, 10)  
*HealthMonitor de la potencia de salida.*
- **HealthMonitor under\_voltage\_protection** (**C\_LIMIT\_UNDERVOLTAGE\_PROT**, 10, 10, 1000)  
*HealthMonitor del theory\_Vout de salida.*
- **Adafruit\_IS31FL3731\_Wing ledmatrix** = **Adafruit\_IS31FL3731\_Wing()**
- **dcdc\_controller DCDC** (**C\_PIN\_EN\_DCDC**)
- **MilliTimmer program\_tick**
- **MilliTimmer timer\_arranque**
- **MilliTimmer protection\_event\_delay**
- **MilliTimmer timer\_diagnostic\_querist**
- **MilliTimmer timer\_sec\_count**
- **MilliTimmer timer\_print\_diagnostic**
- **MilliTimmer timer\_display\_capacity**
- **MilliTimmer timer\_recover\_voltage**
- **MilliTimmer timer\_wait\_sleep**
- **MilliTimmer timer\_low\_bright**
- **MilliTimmer timer\_irq\_button\_center**
- **MilliTimmer timer\_idle**
- **MilliTimmer timer\_display\_error**
- int32\_t **t0** = 0
- int32\_t **t1** = 0
- int32\_t **cont\_sec** = 0
- uint16\_t **sample\_IOut** = 0
- uint16\_t **sample\_VOut** = 0
- uint16\_t **sample\_POut** = 0
- uint16\_t **sample\_Vin** = 0
- bool **output\_mode** = **C\_NON\_BOOST\_MODE**
- bool **protection\_event\_delay\_flag** = false
- bool **mask\_protection\_state** = **C\_MASK\_DEACTIVATE**
- bool **display\_status** = **C\_DISPLAY\_ST\_NOT\_BUSSY**
- bool **display\_error\_status** = **C\_DISPLAY\_ST\_NOT\_BUSSY**
- bool **buzzer\_state** = **C\_BUZZER\_NOT\_BUSSY**
- bool **buzzer\_error\_state** = **C\_BUZZER\_NOT\_BUSSY**
- int16\_t **consumptn\_event\_protection\_counter** = 0
- int16\_t **OP\_event\_protection\_counter** = 0
- int16\_t **UV\_event\_protection\_counter** = 0
- int16\_t **voltage\_input\_event\_protection\_counter** = 0
- String **start\_string** = "MSTK"
- String **msg\_sleep** = "GOOD JOB"
- String **error\_msg** = "Error!"
- String **diagnostic\_msg** = "DIAG"
- String **capcaity\_ask\_off** = "OFF"
- uint16\_t **theory\_Vout** = 50
- uint16\_t **prev\_volt** = **theory\_Vout**
- uint16\_t **capacity** = 0
- int16\_t **sw\_status** = **C\_SW\_ST\_START\_UP**
- int16\_t **state\_to\_return** = **C\_SW\_ST\_STOP**
- bool **usb\_status** = **C\_USB\_DISCONNECTED**

- bool `sw_output` = C\_OFF
- bool `hw_output` = C\_OFF
- bool `user_output` = C\_OFF
- bool `stage_capacity_1` = true
- bool `stage_capacity_2` = false
- uint8\_t `prev_state` = C\_SW\_ST\_START\_UP
- bool `mode_bright_display` = C\_MODE\_HIGH\_BRIGHT
- uint16\_t `click_events` = 0
- bool `flag_low_battery` = false
- int16\_t `button_event` = C\_NONE\_EVENT
- int16\_t `sound` = C\_SOUND\_UP
- int16\_t `cont_idle_timer` = 0
- bool `low_batt_display` = false
- bool `low_batt_sound` = false
- uint8\_t `counter_prints_static_data` = 0
- uint8\_t `afk_counter` = 0
- String `data`
- bool `enable_starting_text` = false
- bool `enable_starting_sound` = false
- bool `enable_ending_text` = false
- bool `enable_ending_sound` = false
- bool `cmd_go_sleep` = false
- bool `enable_live_view` = false
- bool `new_text_received` = false
- bool `change_text_answered` = false
- MilliTimer `delay_live_view`
- MilliTimer `afk_timer`
- bool `enable_charge_sound` = false
- bool `enable_death_battery_sound` = false
- bool `enable_full_charge_sound` = false
- bool `test_enable` = false
- bool `flag_msg_init` = false
- bool `flag_msg_sleep` = false
- bool `flag_error` = false
- bool `flag_diag_header_printed` = false
- bool `flag_initialize` = false
- bool `flag_sleep` = false
- bool `flag_return` = false
- bool `trigger_Display_volt` = true
- bool `go_sleep` = true
- bool `diag_check` = false
- volatile bool `flag_irq_center_button` = false
- volatile bool `flag_irq_death_battery` = false
- bool `flag_usb_change` = false
- bool `print_diagnostic_static_data` = false
- bool `flag_sound` = false
- bool `flag_display_capacity` = false
- bool `flag_work` = false
- bool `flag_test` = false
- bool `flag_cap_one_shot` = false
- bool `flag_update_eeprom` = false
- bool `flag_irq_singleshot` = false
- int16\_t `flag_arranque` = C\_TIMER\_IDLE
- int16\_t `flag_waiting` = C\_TIMER\_IDLE
- int16\_t `flag_timer_low_bright` = C\_TIMER\_IDLE

- bool `reset_init_text` = true
- bool `reset_capacity_off_text` = true
- bool `reset_sleep_text` = true
- bool `reset_error_text` = true
- bool `reset_diag_text` = true

### 15.1.1 Detailed Description

V1: - Gestion y detección del modo Boost/No boost. Cuando la corriente de salida sea superior a un umbral durante un determinado timep, se deberá activar la salida en modo BOOST, mientras que si se encuentra por debajo del umbral durante otro determinado tiempo, se deberá desactivar la salida modo BOOSt.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

- Umbral: 15 mA
- Tiempo Boost: 100 ms
- Tiempo No Boost: 1s. V2: - Protección de sobreconsumo: La protección va a enfocada a detectar cuando la batería tiene un consumo por encima de lo normal, indicativo de una posible situación de error. Ej: Fallo en el arranque de la máquina de tatuar.
  - Umbral: 450 mA.
  - Tiempo: 1500 ms.
  - Sistema de doble pendiente. V3: - Se crean 2 helathMonitor independientes para cada protección. El funcionamiento de estas protecciones son equivalentes a la protección de sobreconsumo. Protección de UnderVoltage:
    - Umbral: 2 Voltios
    - Tiempo: 1 segundos
    - Sistema de doble pendiente. Protección de SobrePotencia.
    - Umbral: 4 Watos
    - Tiempo: 2 segundos
    - Sistema de doble pendiente. V4: - La parte del display integrará 2 funcionalidades. Funcionalidades Display:
      - Mensaje de Inicio: Al encendido de la batería se mostrará en la pantalla una cadena de caracteres predeterminada.
      - Visualizar Voltaje: En el ciclo de control se mostrará en la pantalla el voltaje con el siguiente formato: State Machine: Diferenciación de distintos estados y las condiciones de cambio correspondientes:
      - Initialization: Mensaje de bienvenida y arranque.
        - \* Run: Voltaje por display y activación de la salida.
        - \* Stop: Voltaje por Display y salida desactivada.
        - \* Sleep: Mensaje de despedida y batería "apagada".
        - \* ERROR: Mensaje de error.
      - Subida y bajada de voltaje con los botones.
      - Activación y desactivación de la salida con los botones. V5: - Se decide añadir el módulo de control del Zumbador/Buzzer. Con este módulo se incorporan sonidos a determinados eventos. Los sonidos se seleccionarán entre alternativas en una función de inicialización, según el atributo de entrada. Los eventos que irán acompañados de sonidos en esta integración son:
        - \* Inicio.
        - \* Apagado.
        - \* Subida de Voltaje.
        - \* Bajada de Voltaje.
        - \* Run/Stop. V6: - Se decide realizar un seguimiento de ciertas variables que se consideran de interés. Para ello se utilizan 2 memorias EEPROM, una situada en la parte del chasis y otra en la parte de la batería. En la batería se guardarán datos relacionados con la identidad y vida de la batería como son:

- \* Modelo.
- \* Numero de serie.
- \* Numero de ciclos/Tiempo de Uso.
- \* Esstadisticas de uso en voltage y potencia.
- \* Log de errores que ha podido sufrir. En el chasis se almacena informacion relacionada con la configuracion del chasis como son:
  - \* Seleccion de los sonidos(Encendido,Run/Stop,Up,Down).
  - \* Voltaje de funcionamiento.
- \* Numero de serie del chasis. V7: - Se incluye el calculo de la capacidad de la bateria partiendo de la lectura del voltaje. Esta lectura de realiza siempre cuando la salida se encuentra desconectada para evitar que la carga afecte al voltaje, realizando la medida en vacio. Se situan los extremos en 4,2V y 3,5 como maximo y minimo respectivamente. Propuesta de muestra de la informacion de la capacidaden 2 lugares:
  - \* En la inicializacion.
  - \* Si se mantiene apretado el boton central, se muestra la capacidad durante un breve tiempo antes de mostrar un mensaje de apagado. Si se mantiene apretado termina entrando en modo sleep. Se incluye la proteccion del voltage de entrada pera prevenir que la bateria baje por debajo de un umbral que no asegure un correcto funcionamiento del sistema. Caracteristicas:
    - \* Umbral: 3300 mV.
    - \* Tiempo: 500 ms.
    - \* Sistema de doble pendiente.
  - \* Fix Error Se ha detectado que si no se levanta el boton central, el sistema se enciende y se apaga constantemente. Como solucion, al apagar se espera a que se levante el boton para permitir volver a pulsar y encender el sistema. V8: - Manejo de la barra de potencia, formada por 14 leds puestos en linea recta. Colocada en la parte inferior del Display. La barra de potencia mostrara de forma grafica el valor de la potencia instantanea, de forma que con un fondo de escala ajustable (incialmente en 2500 mW) y una granularidad de 13 leds, se encienden los leds de manera fluida. En los estados de inicio, apagado, error y diagnostico, la barra de potencia estara completamente apagada. Por otro lado, se ha determinando la necesidad de incluir un modo de brillo tenue que se active a los 5 segundos de que no se produzca ninguna interaccion con la bateria, que disminuya considerablemente el brillo del Display. En el momento que se pulse cualquiera de los botones, la pantalla recupera el brillo. La disminucion del brillo solo afecta al estado de Work y Capacit, deminuyendo el brillo del display y la barra de potencia. V9: - Para poder realizar un perfil de tiempos, se utilizan 6 pines que actualmente no tienen uso para, utilizando un analizador logico, se pueda observar el consumo temporal de las distintas funciones como:
    - \* Duracion del ciclo de control.
    - \* Lectura de la botonera.
    - \* Funciones del Buzzer.
    - \* Funciones del Display.
    - \* Funciones de Diagnostico.
  - \* Funcion de la Power Bar. V10: - Con el fin de aumentar la vida de la bateria, durante el estado de sleep, se introducira al micro en un estado de lowpower o Standby. Para despertar el micro se utilizara una interrupcion hardware provocada por el boton central. Tras el despertar se analiza la rutina de atencion de la interrupcion y se continuara el rrograma justo despues de la interrupcion que duerme al micro. V11: - Cuando se conecte el USB al battery pack el chasis reaccionara en un estado que unicamente mostrara una animacion por el display indicando la conexion, y seguidamente se quedara permanentemente intentando conectarse al puerto serie. Si se consigue conectar al puerto serie, empezara a retransmitir la informacion del modo diagnostico (Aun por concretar esta informacion.) Se volvera al modo sleep tras la desconexion del USB. La detección de la conexión y desconexión se realizará mediante una interrupción hardware.

Version

V11

Date

2021-3-5

Copyright

Copyright (c) 2020

Definition in file [B1.ino](#).

## 15.1.2 Function Documentation

### 15.1.2.1 `Irq_DeathBattery_Handler()`

```
void Irq_DeathBattery_Handler ( )
```

Definition at line [1751](#) of file [B1.ino](#).

### 15.1.2.2 `Irq_USB_Handler()`

```
void Irq_USB_Handler ( )
```

Definition at line [1755](#) of file [B1.ino](#).

### 15.1.2.3 `IrqCenterButtonHandler()`

```
void IrqCenterButtonHandler ( )
```

Definition at line [1747](#) of file [B1.ino](#).

### 15.1.2.4 `loop()`

```
void loop ( )
```

Definition at line [1743](#) of file [B1.ino](#).

### 15.1.2.5 `setup()`

```
void setup ( )  
SETUP  
TEST MODE  
Interaccion Con La Batery.  
CONTROL LOOP  
STATE MACHINE  
Definition at line 335 of file B1.ino.
```

### 15.1.2.6 `voltage_input_protection()`

```
HealthMonitor voltage_input_protection (  
    4096 - C_LIMIT_VOLTAGE_INPUT,  
    10 ,  
    10 ,  
    200 )
```

HealthMonitor del Vin de entrada.

- Umbral : 3200 mV
- Ts = 10 ms
- Time spam = 200 ms

### 15.1.3 Variable Documentation

#### 15.1.3.1 afk\_counter

```
uint8_t afk_counter = 0
```

Definition at line 302 of file [B1.ino](#).

#### 15.1.3.2 afk\_timer

```
MilliTimmer afk_timer
```

Definition at line 313 of file [B1.ino](#).

#### 15.1.3.3 boost\_check

```
HealthMonitor boost_check(50, 10, 1, 100) (
    50 ,
    10 ,
    1 ,
    100 )
```

`HealthMonitor` de la corriente de salida para el modo boost.

FUNCTIONS

- Umbral
- $T_s = 10 \text{ ms}$ .
- $T_{\text{boost}} = 100\text{ms}$ .
- $T_{\text{no boost}} = 1\text{s}$

#### 15.1.3.4 button\_event

```
int16_t button_event = C_NONE_EVENT
```

Definition at line 294 of file [B1.ino](#).

#### 15.1.3.5 buzzer\_error\_state

```
bool buzzer_error_state = C_BUZZER_NOT_BUSSY
```

Definition at line 277 of file [B1.ino](#).

#### 15.1.3.6 buzzer\_state

```
bool buzzer_state = C_BUZZER_NOT_BUSSY
```

Definition at line 276 of file [B1.ino](#).

#### 15.1.3.7 C\_CMD\_DIAGNOSTIC\_STOP

```
const String C_CMD_DIAGNOSTIC_STOP = "GO_SLEEP"
```

Definition at line 203 of file [B1.ino](#).

### 15.1.3.8 C\_CMD\_DISPLAY\_ENDING\_OFF

```
const String C_CMD_DISPLAY_ENDING_OFF = "DspEndOFF"  
Definition at line 189 of file B1.ino.
```

### 15.1.3.9 C\_CMD\_DISPLAY\_ENDING\_ON

```
const String C_CMD_DISPLAY_ENDING_ON = "DspEndON"  
Definition at line 190 of file B1.ino.
```

### 15.1.3.10 C\_CMD\_DISPLAY\_STARTING\_OFF

```
const String C_CMD_DISPLAY_STARTING_OFF = "DspStOFF"  
Definition at line 187 of file B1.ino.
```

### 15.1.3.11 C\_CMD\_DISPLAY\_STARTING\_ON

```
const String C_CMD_DISPLAY_STARTING_ON = "DspStON"  
Definition at line 188 of file B1.ino.
```

### 15.1.3.12 C\_CMD\_LIVE\_OFF

```
const String C_CMD_LIVE_OFF = "LiveOFF"  
Definition at line 201 of file B1.ino.
```

### 15.1.3.13 C\_CMD\_LIVE\_ON

```
const String C_CMD_LIVE_ON = "LiveON"  
Definition at line 202 of file B1.ino.
```

### 15.1.3.14 C\_CMD\_SOUND\_ENDING\_OFF

```
const String C_CMD_SOUND_ENDING_OFF = "SndEndOFF"  
Definition at line 193 of file B1.ino.
```

### 15.1.3.15 C\_CMD\_SOUND\_ENDING\_ON

```
const String C_CMD_SOUND_ENDING_ON = "SndEndON"  
Definition at line 194 of file B1.ino.
```

### 15.1.3.16 C\_CMD\_SOUND\_FULL\_CHARGE\_OFF

```
const String C_CMD_SOUND_FULL_CHARGE_OFF = "SndFChgOFF"  
Definition at line 195 of file B1.ino.
```

### 15.1.3.17 C\_CMD\_SOUND\_FULL\_CHARGE\_ON

```
const String C_CMD_SOUND_FULL_CHARGE_ON = "SndFChgOFF"  
Definition at line 196 of file B1.ino.
```

### 15.1.3.18 C\_CMD\_SOUND\_STARTING\_OFF

```
const String C_CMD_SOUND_STARTING_OFF = "SndStrOFF"  
Definition at line 191 of file B1.ino.
```

### 15.1.3.19 C\_CMD\_SOUND\_STARTING\_ON

```
const String C_CMD_SOUND_STARTING_ON = "SndSrTON"  
Definition at line 192 of file B1.ino.
```

### 15.1.3.20 C\_CMS\_SOUND\_CHARGE\_START\_OFF

```
const String C_CMS_SOUND_CHARGE_START_OFF = "SndChgStOFF"  
Definition at line 199 of file B1.ino.
```

### 15.1.3.21 C\_CMS\_SOUND\_CHARGE\_START\_ON

```
const String C_CMS_SOUND_CHARGE_START_ON = "SndChgStON"  
Definition at line 200 of file B1.ino.
```

### 15.1.3.22 C\_CMS\_SOUND\_DEATH\_BATTERY\_OFF

```
const String C_CMS_SOUND_DEATH_BATTERY_OFF = "SndDthBttOFF"  
Definition at line 197 of file B1.ino.
```

### 15.1.3.23 C\_CMS\_SOUND\_DEATH\_BATTERY\_ON

```
const String C_CMS_SOUND_DEATH_BATTERY_ON = "SndDthBttON"  
Definition at line 198 of file B1.ino.
```

### 15.1.3.24 C\_DIAGNOSTIC\_PASSWORD

```
const String C_DIAGNOSTIC_PASSWORD = "FPO"  
Definition at line 175 of file B1.ino.
```

### 15.1.3.25 C\_IDLE\_TIMER\_COUNT

```
const uint16_t C_IDLE_TIMER_COUNT = 1 * 2  
Definition at line 184 of file B1.ino.
```

### 15.1.3.26 C\_LIMIT\_COMSUPTION\_PROT

```
const int16_t C_LIMIT_COMSUPTION_PROT = 450 * 4096 / 3300  
Definition at line 165 of file B1.ino.
```

### 15.1.3.27 C\_LIMIT\_OVERPOWER\_PROT

```
const int16_t C_LIMIT_OVERPOWER_PROT = 4000  
Definition at line 168 of file B1.ino.
```

### 15.1.3.28 C\_LIMIT\_UNDERVOLTAGE\_PROT

```
const int16_t C_LIMIT_UNDERVOLTAGE_PROT = 2000 * 4096 / 3300
Definition at line 166 of file B1.ino.
```

### 15.1.3.29 C\_LIMIT\_VOLTAGE\_INPUT

```
const int16_t C_LIMIT_VOLTAGE_INPUT = 5000 - 3000
Definition at line 167 of file B1.ino.
```

### 15.1.3.30 C\_LIVE\_CMD\_OFF\_LIVE\_VIEW

```
const String C_LIVE_CMD_OFF_LIVE_VIEW = "LiveViewOff"
Definition at line 211 of file B1.ino.
```

### 15.1.3.31 C\_LIVE\_CMD\_OUT\_OFF

```
const String C_LIVE_CMD_OUT_OFF = "DisableOut"
Definition at line 206 of file B1.ino.
```

### 15.1.3.32 C\_LIVE\_CMD\_OUT\_ON

```
const String C_LIVE_CMD_OUT_ON = "EnableOut"
Definition at line 205 of file B1.ino.
```

### 15.1.3.33 C\_LIVE\_CMD\_VOLT\_DOWN\_1

```
const String C_LIVE_CMD_VOLT_DOWN_1 = "VoltDown1"
Definition at line 209 of file B1.ino.
```

### 15.1.3.34 C\_LIVE\_CMD\_VOLT\_DOWN\_10

```
const String C_LIVE_CMD_VOLT_DOWN_10 = "VoltDown10"
Definition at line 210 of file B1.ino.
```

### 15.1.3.35 C\_LIVE\_CMD\_VOLT\_UP\_1

```
const String C_LIVE_CMD_VOLT_UP_1 = "VoltUp1"
Definition at line 207 of file B1.ino.
```

### 15.1.3.36 C\_LIVE\_CMD\_VOLT\_UP\_10

```
const String C_LIVE_CMD_VOLT_UP_10 = "VoltUp10"
Definition at line 208 of file B1.ino.
```

### 15.1.3.37 C\_LOW\_BATTERY\_LEVEL

```
const uint8_t C_LOW_BATTERY_LEVEL = 10
Definition at line 182 of file B1.ino.
```

**15.1.3.38 C\_MASK\_ACTIVATE**

```
const bool C_MASK_ACTIVATE = true
```

Definition at line 142 of file [B1.ino](#).

**15.1.3.39 C\_MASK\_DEACTIVATE**

```
const bool C_MASK_DEACTIVATE = false
```

Definition at line 141 of file [B1.ino](#).

**15.1.3.40 C\_MAX\_VOLT**

```
const int16_t C_MAX_VOLT = 160
```

Definition at line 145 of file [B1.ino](#).

**15.1.3.41 C\_MIN\_VOLT**

```
const int16_t C_MIN_VOLT = 50
```

Definition at line 144 of file [B1.ino](#).

**15.1.3.42 C\_NUMB\_PRINTS\_STATIC\_DATA**

```
const uint8_t C_NUMB_PRINTS_STATIC_DATA = 5
```

Definition at line 180 of file [B1.ino](#).

**15.1.3.43 C\_OFF**

```
const bool C_OFF = false
```

Definition at line 149 of file [B1.ino](#).

**15.1.3.44 C\_ON**

```
const bool C_ON = true
```

Definition at line 150 of file [B1.ino](#).

**15.1.3.45 C\_PIN\_DEATH\_BATTERY\_CHECK**

```
const uint16_t C_PIN_DEATH_BATTERY_CHECK = 10
```

Definition at line 125 of file [B1.ino](#).

**15.1.3.46 C\_PIN\_EN\_DCDC**

```
const uint16_t C_PIN_EN_DCDC = 2
```

Definition at line 122 of file [B1.ino](#).

**15.1.3.47 C\_PIN\_FLAG\_BATTLOW**

```
const int16_t C_PIN_FLAG_BATTLOW = 10
```

Definition at line 138 of file [B1.ino](#).

#### 15.1.3.48 C\_PIN\_FLAG\_CHARG

```
const int16_t C_PIN_FLAG_CHARG = 0
```

Definition at line 135 of file [B1.ino](#).

#### 15.1.3.49 C\_PIN\_I\_OUT

```
const int16_t C_PIN_I_OUT = A1
```

INCLUDES CONSTANTS  
Definition at line 119 of file [B1.ino](#).

#### 15.1.3.50 C\_PIN\_SCL\_2

```
const int16_t C_PIN_SCL_2 = 1
```

Definition at line 136 of file [B1.ino](#).

#### 15.1.3.51 C\_PIN\_SDA\_2

```
const int16_t C_PIN_SDA_2 = 6
```

Definition at line 137 of file [B1.ino](#).

#### 15.1.3.52 C\_PIN\_SHIPPING\_MOD

```
const uint16_t C_PIN_SHIPPING_MOD = 3
```

Definition at line 123 of file [B1.ino](#).

#### 15.1.3.53 C\_PIN\_TEST\_MODE

```
const uint16_t C_PIN_TEST_MODE = 11
```

Definition at line 126 of file [B1.ino](#).

#### 15.1.3.54 C\_PIN\_USB\_CONNEXION

```
const uint16_t C_PIN_USB_CONNEXION = 7
```

Definition at line 124 of file [B1.ino](#).

#### 15.1.3.55 C\_PIN\_USB\_SEL

```
const int16_t C_PIN_USB_SEL = 11
```

Definition at line 139 of file [B1.ino](#).

#### 15.1.3.56 C\_PIN\_V\_IN

```
const int16_t C_PIN_V_IN = A4
```

Definition at line 120 of file [B1.ino](#).

#### 15.1.3.57 C\_PIN\_V\_OUT

```
const int16_t C_PIN_V_OUT = A2
```

Definition at line 121 of file [B1.ino](#).

**15.1.3.58 C\_RETRY\_750\_COUNT**

```
const int16_t C_RETRY_750_COUNT = 750
Definition at line 169 of file B1.ino.
```

**15.1.3.59 C\_SW\_ST\_CAPACITY**

```
const int16_t C_SW_ST_CAPACITY = 0x06
Definition at line 162 of file B1.ino.
```

**15.1.3.60 C\_SW\_ST\_DIAGNOSTIC**

```
const int16_t C_SW_ST_DIAGNOSTIC = 0x05
Definition at line 161 of file B1.ino.
```

**15.1.3.61 C\_SW\_ST\_ERROR**

```
const int16_t C_SW_ST_ERROR = 0x03
Definition at line 159 of file B1.ino.
```

**15.1.3.62 C\_SW\_ST\_RUN**

```
const int16_t C_SW_ST_RUN = 0x00
Definition at line 156 of file B1.ino.
```

**15.1.3.63 C\_SW\_ST\_SLEEP**

```
const int16_t C_SW_ST_SLEEP = 0x01
Definition at line 157 of file B1.ino.
```

**15.1.3.64 C\_SW\_ST\_START\_UP**

```
const int16_t C_SW_ST_START_UP = 0x02
Definition at line 158 of file B1.ino.
```

**15.1.3.65 C\_SW\_ST\_STOP**

```
const int16_t C_SW_ST_STOP = 0x04
Definition at line 160 of file B1.ino.
```

**15.1.3.66 C\_SW\_ST\_USB**

```
const int16_t C_SW_ST_USB = 0x07
Definition at line 163 of file B1.ino.
```

**15.1.3.67 C\_TIME\_TO\_LIGHT\_DOWN**

```
const int32_t C_TIME_TO_LIGHT_DOWN = 5000
Definition at line 171 of file B1.ino.
```

### 15.1.3.68 C\_TIMER\_ARMED

```
const int16_t C_TIMER_ARMED = 0
Definition at line 152 of file B1.ino.
```

### 15.1.3.69 C\_TIMER\_DONE

```
const int16_t C_TIMER_DONE = 2
Definition at line 154 of file B1.ino.
```

### 15.1.3.70 C\_TIMER\_IDLE

```
const int16_t C_TIMER_IDLE = 1
Definition at line 153 of file B1.ino.
```

### 15.1.3.71 C\_USB\_CONNECTED

```
const bool C_USB_CONNECTED = true
Definition at line 177 of file B1.ino.
```

### 15.1.3.72 C\_USB\_DISCONNECTED

```
const bool C_USB_DISCONNECTED = false
Definition at line 178 of file B1.ino.
```

### 15.1.3.73 C\_WATCH\_DOG\_TIME\_MS

```
const uint32_t C_WATCH_DOG_TIME_MS = 1000
Definition at line 173 of file B1.ino.
```

### 15.1.3.74 capacity

```
uint16_t capacity = 0
Definition at line 282 of file B1.ino.
```

### 15.1.3.75 capcaity\_ask\_off

```
String capcaity_ask_off = "OFF"
Definition at line 280 of file B1.ino.
```

### 15.1.3.76 change\_text\_answered

```
bool change_text_answered = false
Definition at line 311 of file B1.ino.
```

### 15.1.3.77 click\_events

```
uint16_t click_events = 0
Definition at line 292 of file B1.ino.
```

**15.1.3.78 cmd\_go\_sleep**

```
bool cmd_go_sleep = false  
Definition at line 308 of file B1.ino.
```

**15.1.3.79 consmptn\_event\_protection\_counter**

```
int16_t consmptn_event_protection_counter = 0  
Definition at line 278 of file B1.ino.
```

**15.1.3.80 cont\_idle\_timer**

```
int16_t cont_idle_timer = 0  
Definition at line 296 of file B1.ino.
```

**15.1.3.81 cont\_sec**

```
int32_t cont_sec = 0  
Definition at line 269 of file B1.ino.
```

**15.1.3.82 counter\_prints\_static\_data**

```
uint8_t counter_prints_static_data = 0  
Definition at line 301 of file B1.ino.
```

**15.1.3.83 data**

```
String data  
Definition at line 303 of file B1.ino.
```

**15.1.3.84 DCDC**

```
dcdc_controller DCDC(C_PIN_EN_DCDC) (  
    C_PIN_EN_DCDC )
```

**15.1.3.85 delay\_live\_view**

```
MilliTimmer delay_live_view  
Definition at line 312 of file B1.ino.
```

**15.1.3.86 diag\_check**

```
bool diag_check = false  
Definition at line 323 of file B1.ino.
```

**15.1.3.87 diagnostic\_msg**

```
String diagnostic_msg = "DIAG"  
Definition at line 280 of file B1.ino.
```

#### 15.1.3.88 display\_error\_status

```
bool display_error_status = C_DISPLAY_ST_NOT_BUSSY  
Definition at line 275 of file B1.ino.
```

#### 15.1.3.89 display\_status

```
bool display_status = C_DISPLAY_ST_NOT_BUSSY  
Definition at line 274 of file B1.ino.
```

#### 15.1.3.90 enable\_charge\_sound

```
bool enable_charge_sound = false  
Definition at line 314 of file B1.ino.
```

#### 15.1.3.91 enable\_death\_battery\_sound

```
bool enable_death_battery_sound = false  
Definition at line 315 of file B1.ino.
```

#### 15.1.3.92 enable\_ending\_sound

```
bool enable_ending_sound = false  
Definition at line 307 of file B1.ino.
```

#### 15.1.3.93 enable\_ending\_text

```
bool enable_ending_text = false  
Definition at line 306 of file B1.ino.
```

#### 15.1.3.94 enable\_full\_charge\_sound

```
bool enable_full_charge_sound = false  
Definition at line 316 of file B1.ino.
```

#### 15.1.3.95 enable\_live\_view

```
bool enable_live_view = false  
Definition at line 309 of file B1.ino.
```

#### 15.1.3.96 enable\_starting\_sound

```
bool enable_starting_sound = false  
Definition at line 305 of file B1.ino.
```

#### 15.1.3.97 enable\_starting\_text

```
bool enable_starting_text = false  
Definition at line 304 of file B1.ino.
```

**15.1.3.98 error\_msg**

```
String error_msg = "Error!"  
Definition at line 280 of file B1.ino.
```

**15.1.3.99 flag\_arranque**

```
int16_t flag_arranque = C_TIMER_IDLE  
Definition at line 330 of file B1.ino.
```

**15.1.3.100 flag\_cap\_one\_shot**

```
bool flag_cap_one_shot = false  
Definition at line 329 of file B1.ino.
```

**15.1.3.101 flag\_diag\_header\_printed**

```
bool flag_diag_header_printed = false  
Definition at line 320 of file B1.ino.
```

**15.1.3.102 flag\_display\_capacity**

```
bool flag_display_capacity = false  
Definition at line 329 of file B1.ino.
```

**15.1.3.103 flag\_error**

```
bool flag_error = false  
Definition at line 320 of file B1.ino.
```

**15.1.3.104 flag\_initialize**

```
bool flag_initialize = false  
Definition at line 320 of file B1.ino.
```

**15.1.3.105 flag\_irq\_center\_button**

```
volatile bool flag_irq_center_button = false  
Definition at line 324 of file B1.ino.
```

**15.1.3.106 flag\_irq\_death\_battery**

```
volatile bool flag_irq_death_battery = false  
Definition at line 325 of file B1.ino.
```

**15.1.3.107 flag\_irq\_singleshot**

```
bool flag_irq_singleshot = false  
Definition at line 329 of file B1.ino.
```

#### 15.1.3.108 flag\_low\_battery

```
bool flag_low_battery = false  
Definition at line 293 of file B1.ino.
```

#### 15.1.3.109 flag\_msg\_init

```
bool flag_msg_init = false  
Definition at line 320 of file B1.ino.
```

#### 15.1.3.110 flag\_msg\_sleep

```
bool flag_msg_sleep = false  
Definition at line 320 of file B1.ino.
```

#### 15.1.3.111 flag\_return

```
bool flag_return = false  
Definition at line 320 of file B1.ino.
```

#### 15.1.3.112 flag\_sleep

```
bool flag_sleep = false  
Definition at line 320 of file B1.ino.
```

#### 15.1.3.113 flag\_sound

```
bool flag_sound = false  
Definition at line 329 of file B1.ino.
```

#### 15.1.3.114 flag\_test

```
bool flag_test = false  
Definition at line 329 of file B1.ino.
```

#### 15.1.3.115 flag\_timer\_low\_bright

```
int16_t flag_timer_low_bright = C\_TIMER\_IDLE  
Definition at line 330 of file B1.ino.
```

#### 15.1.3.116 flag\_update\_eeprom

```
bool flag_update_eeprom = false  
Definition at line 329 of file B1.ino.
```

#### 15.1.3.117 flag\_usb\_change

```
bool flag_usb_change = false  
Definition at line 326 of file B1.ino.
```

**15.1.3.118 flag\_waiting**

```
int16_t flag_waiting = C_TIMER_IDLE  
Definition at line 330 of file B1.ino.
```

**15.1.3.119 flag\_work**

```
bool flag_work = false  
Definition at line 329 of file B1.ino.
```

**15.1.3.120 go\_sleep**

```
bool go_sleep = true  
Definition at line 322 of file B1.ino.
```

**15.1.3.121 hw\_output**

```
bool hw_output = C_OFF  
Definition at line 287 of file B1.ino.
```

**15.1.3.122 ledmatrix**

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()  
Definition at line 262 of file B1.ino.
```

**15.1.3.123 low\_batt\_display**

```
bool low_batt_display = false  
Definition at line 297 of file B1.ino.
```

**15.1.3.124 low\_batt\_sound**

```
bool low_batt_sound = false  
Definition at line 298 of file B1.ino.
```

**15.1.3.125 mask\_protection\_state**

```
bool mask_protection_state = C_MASK_DEACTIVATE  
Definition at line 273 of file B1.ino.
```

**15.1.3.126 MAX\_NUM\_PROTECTION\_EVENTS**

```
const uint16_t MAX_NUM_PROTECTION_EVENTS = 6  
Definition at line 147 of file B1.ino.
```

**15.1.3.127 mode\_bright\_display**

```
bool mode_bright_display = C_MODE_HIGH_BRIGHT  
Definition at line 291 of file B1.ino.
```

### 15.1.3.128 msg\_sleep

```
String msg_sleep = "GOOD JOB"  
Definition at line 280 of file B1.ino.
```

### 15.1.3.129 new\_text\_received

```
bool new_text_received = false  
Definition at line 310 of file B1.ino.
```

### 15.1.3.130 OP\_event\_protection\_counter

```
int16_t OP_event_protection_counter = 0  
Definition at line 278 of file B1.ino.
```

### 15.1.3.131 output\_mode

```
bool output_mode = C_NON_BOOST_MODE  
Definition at line 271 of file B1.ino.
```

### 15.1.3.132 over\_consumption\_protection

```
HealthMonitor over_consumption_protection(C_LIMIT_COMSUPTION_PROT, 10, 10, 1500) (  
    C_LIMIT_COMSUPTION_PROT ,  
    10 ,  
    10 ,  
    1500 )
```

HealthMonitor del consumo de salida de la bateria.

- Umbral : 450 mah.
- Ts = 10 ms
- Time spam = 1500ms

### 15.1.3.133 over\_power\_protection

```
HealthMonitor over_power_protection(C_LIMIT_OVERPOWER_PROT, 10, 10, 10) (  
    C_LIMIT_OVERPOWER_PROT ,  
    10 ,  
    10 ,  
    10 )
```

HealthMonitor de la potencia de salida.

- Umbral : 4000 mW. (mA x mV / 1000)
- Ts = 10 ms
- Time spam = 2000 ms

### 15.1.3.134 prev\_state

```
uint8_t prev_state = C_SW_ST_START_UP  
Definition at line 290 of file B1.ino.
```

**15.1.3.135 prev\_volt**

```
uint16_t prev_volt = theory_Vout  
Definition at line 282 of file B1.ino.
```

**15.1.3.136 print\_diagnostic\_static\_data**

```
bool print_diagnostic_static_data = false  
Definition at line 327 of file B1.ino.
```

**15.1.3.137 program\_tick**

```
MilliTimmer program_tick  
Definition at line 266 of file B1.ino.
```

**15.1.3.138 protection\_event\_delay**

```
MilliTimmer protection_event_delay  
Definition at line 266 of file B1.ino.
```

**15.1.3.139 protection\_event\_delay\_flag**

```
bool protection_event_delay_flag = false  
Definition at line 272 of file B1.ino.
```

**15.1.3.140 reset\_capacity\_off\_text**

```
bool reset_capacity_off_text = true  
Definition at line 333 of file B1.ino.
```

**15.1.3.141 reset\_diag\_text**

```
bool reset_diag_text = true  
Definition at line 333 of file B1.ino.
```

**15.1.3.142 reset\_error\_text**

```
bool reset_error_text = true  
Definition at line 333 of file B1.ino.
```

**15.1.3.143 reset\_init\_text**

```
bool reset_init_text = true  
Definition at line 333 of file B1.ino.
```

**15.1.3.144 reset\_sleep\_text**

```
bool reset_sleep_text = true  
Definition at line 333 of file B1.ino.
```

#### 15.1.3.145 sample\_IOut

```
uint16_t sample_IOut = 0
Definition at line 270 of file B1.ino.
```

#### 15.1.3.146 sample\_POut

```
uint16_t sample_POut = 0
Definition at line 270 of file B1.ino.
```

#### 15.1.3.147 sample\_Vin

```
uint16_t sample_Vin = 0
Definition at line 270 of file B1.ino.
```

#### 15.1.3.148 sample\_VOut

```
uint16_t sample_VOut = 0
Definition at line 270 of file B1.ino.
```

#### 15.1.3.149 sound

```
int16_t sound = C_SOUND_UP
Definition at line 295 of file B1.ino.
```

#### 15.1.3.150 stage\_capacity\_1

```
bool stage_capacity_1 = true
Definition at line 288 of file B1.ino.
```

#### 15.1.3.151 stage\_capacity\_2

```
bool stage_capacity_2 = false
Definition at line 289 of file B1.ino.
```

#### 15.1.3.152 start\_string

```
String start_string = "MSTK"
Definition at line 280 of file B1.ino.
```

#### 15.1.3.153 state\_to\_return

```
int16_t state_to_return = C_SW_ST_STOP
Definition at line 285 of file B1.ino.
```

#### 15.1.3.154 sw\_output

```
bool sw_output = C_OFF
Definition at line 287 of file B1.ino.
```

**15.1.3.155 sw\_status**

```
int16_t sw_status = C_SW_ST_START_UP  
Definition at line 284 of file B1.ino.
```

**15.1.3.156 t0**

```
int32_t t0 = 0  
Definition at line 268 of file B1.ino.
```

**15.1.3.157 t1**

```
int32_t t1 = 0  
Definition at line 268 of file B1.ino.
```

**15.1.3.158 test\_enable**

```
bool test_enable = false  
Definition at line 317 of file B1.ino.
```

**15.1.3.159 theory\_Vout**

```
uint16_t theory_Vout = 50  
Definition at line 282 of file B1.ino.
```

**15.1.3.160 timer\_arranque**

```
MilliTTimer timer_arranque  
Definition at line 266 of file B1.ino.
```

**15.1.3.161 timer\_diagnostic\_querist**

```
MilliTTimer timer_diagnostic_querist  
Definition at line 266 of file B1.ino.
```

**15.1.3.162 timer\_display\_capacity**

```
MilliTTimer timer_display_capacity  
Definition at line 266 of file B1.ino.
```

**15.1.3.163 timer\_display\_error**

```
MilliTTimer timer_display_error  
Definition at line 266 of file B1.ino.
```

**15.1.3.164 timer\_idle**

```
MilliTTimer timer_idle  
Definition at line 266 of file B1.ino.
```

#### 15.1.3.165 timer\_irq\_button\_center

`MilliTimer` `timer_irq_button_center`  
Definition at line 266 of file [B1.ino](#).

#### 15.1.3.166 timer\_low\_bright

`MilliTimer` `timer_low_bright`  
Definition at line 266 of file [B1.ino](#).

#### 15.1.3.167 timer\_print\_diagnostic

`MilliTimer` `timer_print_diagnostic`  
Definition at line 266 of file [B1.ino](#).

#### 15.1.3.168 timer\_recover\_voltage

`MilliTimer` `timer_recover_voltage`  
Definition at line 266 of file [B1.ino](#).

#### 15.1.3.169 timer\_sec\_count

`MilliTimer` `timer_sec_count`  
Definition at line 266 of file [B1.ino](#).

#### 15.1.3.170 timer\_wait\_sleep

`MilliTimer` `timer_wait_sleep`  
Definition at line 266 of file [B1.ino](#).

#### 15.1.3.171 trigger\_Display\_volt

`bool` `trigger_Display_volt` = true  
Definition at line 321 of file [B1.ino](#).

#### 15.1.3.172 under\_voltage\_protection

```
HealthMonitor under_voltage_protection(C_LIMIT_UNDERVOLTAGE_PROT, 10, 10, 1000) (
```

`C_LIMIT_UNDERVOLTAGE_PROT` ,  
10 ,  
10 ,  
1000 )

`HealthMonitor` del theory\_Vout de salida.

- Umbral : 2000 mV
- Ts = 10 ms
- Time spam = 1000 ms

### 15.1.3.173 `usb_status`

```
bool usb_status = C_USB_DISCONNECTED
Definition at line 286 of file B1.ino.
```

### 15.1.3.174 `user_output`

```
bool user_output = C_OFF
Definition at line 287 of file B1.ino.
```

### 15.1.3.175 `UV_event_protection_counter`

```
int16_t UV_event_protection_counter = 0
Definition at line 278 of file B1.ino.
```

### 15.1.3.176 `voltage_input_event_protection_counter`

```
int16_t voltage_input_event_protection_counter = 0
Definition at line 278 of file B1.ino.
```

## 15.2 B1.ino

[Go to the documentation of this file.](#)

```
00001
00105 #include <MilliTimer.h>
00106 #include <DCDC.h>
00107 #include <HealthMonitor.h>
00108 #include <display.h>
00109 #include <Dpad.h>
00110 #include <Buzzer.h>
00111 #include <diagnostic.h>
00112 #include <power_bar.h>
00113 #include <batt_ArduinoLowPower.h>
00114
00119 const int16_t C_PIN_I_OUT = A1;
00120 const int16_t C_PIN_V_IN = A4;
00121 const int16_t C_PIN_V_OUT = A2;
00122 const uint16_t C_PIN_EN_DCDC = 2;
00123 const uint16_t C_PIN_SHIPPING_MOD = 3;
00124 const uint16_t C_PIN_USB_CONNEXION = 7;
00125 const uint16_t C_PIN_DEATH_BATTERY_CHECK = 10;
00126 const uint16_t C_PIN_TEST_MODE = 11;
00127
00128 //const int16_t C_PIN_PROFILE_CYCLE = 0;
00129 //const int16_t C_PIN_PROFILE_BUTTONS = 1;
00130 //const int16_t C_PIN_PROFILE_BUZZER = 6;
00131 //const int16_t C_PIN_PROFILE_DISPLAY = 10;
00132 //const int16_t C_PIN_PROFILE_EEPROM = 10;
00133 //const int16_t C_PIN_PROFILE_POWER_BAR = 11;
00134
00135 const int16_t C_PIN_FLAG_CHARG = 0;
00136 const int16_t C_PIN_SCL_2 = 1;
00137 const int16_t C_PIN_SDA_2 = 6;
00138 const int16_t C_PIN_FLAG_BATTLOW = 10;
00139 const int16_t C_PIN_USB_SEL = 11;
00140
00141 const bool C_MASK_DEACTIVATE = false;
00142 const bool C_MASK_ACTIVATE = true;
00143
00144 const int16_t C_MIN_VOLT = 50;
00145 const int16_t C_MAX_VOLT = 160;
00146
00147 const uint16_t MAX_NUM_PROTECTION_EVENTS = 6;
00148
00149 const bool C_OFF = false;
00150 const bool C_ON = true;
00151
00152 const int16_t C_TIMER_ARMED = 0;
00153 const int16_t C_TIMER_IDLE = 1;
00154 const int16_t C_TIMER_DONE = 2;
00155
00156 const int16_t C_SW_ST_RUN = 0x00;
```

```

00157 const int16_t C_SW_ST_SLEEP = 0x01;
00158 const int16_t C_SW_ST_START_UP = 0x02;
00159 const int16_t C_SW_ST_ERROR = 0x03;
00160 const int16_t C_SW_ST_STOP = 0x04;
00161 const int16_t C_SW_ST_DIAGNOSTIC = 0x05;
00162 const int16_t C_SW_ST_CAPACITY = 0x06;
00163 const int16_t C_SW_ST_USB = 0x07;
00164
00165 const int16_t C_LIMIT_CONSUPTION_PROT = 450 * 4096 / 3300;
00166 const int16_t C_LIMIT_UNDERVOLTAGE_PROT = 2000 * 4096 / 3300;
00167 const int16_t C_LIMIT_VOLTAGE_INPUT = 5000 - 3000;
00168 const int16_t C_LIMIT_OVERPOWER_PROT = 4000;
00169 const int16_t C_RETRY_750_COUNT = 750;
00170
00171 const int32_t C_TIME_TO_LIGHT_DOWN = 5000; // 5s
00172
00173 const uint32_t C_WATCH_DOG_TIME_MS = 1000;
00174
00175 const String C_DIAGNOSTIC_PASSWORD = "FPO";
00176
00177 const bool C_USB_CONNECTED = true;
00178 const bool C_USB_DISCONNECTED = false;
00179
00180 const uint8_t C_NUMB_PRINTS_STATIC_DATA = 5;
00181
00182 const uint8_t C_LOW_BATTERY_LEVEL = 10; // Porcentaje de bateria partir del cual no se puede asegurar
    un correcto funcionamiento.
00183
00184 const uint16_t C_IDLE_TIMER_COUNT = 1 * 2; // Minutos antes de que se apague la bateria por
    inactividad. NOTE: El contador es de 30seg, por eso se multiplica por 2-
00185
00186 // Diagnostic Commands
00187 const String C_CMD_DISPLAY_STARTING_OFF = "DspStOFF";
00188 const String C_CMD_DISPLAY_STARTING_ON = "DspStON";
00189 const String C_CMD_DISPLAY_ENDING_OFF = "DspEndOFF";
00190 const String C_CMD_DISPLAY_ENDING_ON = "DspEndON";
00191 const String C_CMD_SOUND_STARTING_OFF = "SndStOFF";
00192 const String C_CMD_SOUND_STARTING_ON = "SndStON";
00193 const String C_CMD_SOUND_ENDING_OFF = "SndEndOFF";
00194 const String C_CMD_SOUND_ENDING_ON = "SndEndON";
00195 const String C_CMD_SOUND_FULL_CHARGE_OFF = "SndFChgOFF";
00196 const String C_CMD_SOUND_FULL_CHARGE_ON = "SndFChgON";
00197 const String C_CMS_SOUND_DEATH_BATTERY_OFF = "SndDthBttOFF";
00198 const String C_CMS_SOUND_DEATH_BATTERY_ON = "SndDthBttON";
00199 const String C_CMS_SOUND_CHARGE_START_OFF = "SndChgStOFF";
00200 const String C_CMS_SOUND_CHARGE_START_ON = "SndChgStON";
00201 const String C_CMD_LIVE_OFF = "LiveOff";
00202 const String C_CMD_LIVE_ON = "LiveOn";
00203 const String C_CMD_DIAGNOSTIC_STOP = "GO_SLEEP";
00204
00205 const String C_LIVE_CMD_OUT_ON = "EnableOut";
00206 const String C_LIVE_CMD_OUT_OFF = "DisableOut";
00207 const String C_LIVE_CMD_VOLT_UP_1 = "VoltUp1";
00208 const String C_LIVE_CMD_VOLT_UP_10 = "VoltUp10";
00209 const String C_LIVE_CMD_VOLT_DOWN_1 = "VoltDown1";
00210 const String C_LIVE_CMD_VOLT_DOWN_10 = "VoltDown10";
00211 const String C_LIVE_CMD_OFF_LIVE_VIEW = "LiveViewOff";
00212
00225 HealthMonitor boost_check(50, 10, 1, 100);
00233 HealthMonitor over_consumption_protection(C_LIMIT_CONSUPTION_PROT, 10, 10, 1500);
00234
00242 HealthMonitor over_power_protection(C_LIMIT_OVERPOWER_PROT, 10, 10, 10);
00243
00251 HealthMonitor under_voltage_protection(C_LIMIT_UNDERVOLTAGE_PROT, 10, 10, 1000);
00252
00260 HealthMonitor voltage_input_protection(4096 - C_LIMIT_VOLTAGE_INPUT, 10, 10, 200);
00261
00262 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00263
00264 dcdc_controller DCDC(C_PIN_EN_DCDC);
00265
00266 MilliTimer program_tick, timer_arranque, protection_event_delay, timer_diagnostic_querist,
    timer_sec_count, timer_print_diagnostic, timer_display_capacity, timer_recover_voltage,
    timer_wait_sleep, timer_low_bright, timer_irq_button_center, timer_idle, timer_display_error;
00267
00268 int32_t t0 = 0, t1 = 0;
00269 int32_t cont_sec = 0;
00270 uint16_t sample_IOut = 0, sample_VOut = 0, sample_POut = 0, sample_Vin = 0;
00271 bool output_mode = C_NON_BOOST_MODE;
00272 bool protection_event_delay_flag = false;
00273 bool mask_protection_state = C_MASK_DEACTIVATE;
00274 bool display_status = C_DISPLAY_ST_NOT_BUSSY;
00275 bool display_error_status = C_DISPLAY_ST_NOT_BUSSY;
00276 bool buzzer_state = C_BUZZER_NOT_BUSSY;
00277 bool buzzer_error_state = C_BUZZER_NOT_BUSSY;
00278 int16_t consmptn_event_protection_counter = 0, OP_event_protection_counter = 0,
    UV_event_protection_counter = 0, voltage_input_event_protection_counter = 0;

```

```

00279
00280 String start_string = "MSTK", msg_sleep = "GOOD JOB", error_msg = "Error!", diagnostic_msg = "DIAG",
00281   capcaity_ask_off = "OFF";
00282 uint16_t theory_Vout = 50, prev_volt = theory_Vout, capacity = 0;
00283
00284 int16_t sw_status = C_SW_ST_START_UP;
00285 int16_t state_to_return = C_SW_ST_STOP;
00286 bool usb_status = C_USB_DISCONNECTED;
00287 bool sw_output = C_OFF, hw_output = C_OFF, user_output = C_OFF;
00288 bool stage_capacity_1 = true;
00289 bool stage_capacity_2 = false;
00290 uint8_t prev_state = C_SW_ST_START_UP;
00291 bool mode_bright_display = C_MODE_HIGH_BRIGHT;
00292 uint16_t click_events = 0;
00293 bool flag_low_battery = false;
00294 int16_t button_event = C_NONE_EVENT;
00295 int16_t sound = C_SOUND_UP;
00296 int16_t cont_idle_timer = 0;
00297 bool low_batt_display = false;
00298 bool low_batt_sound = false;
00299
00300 //Diagnostics variables
00301 uint8_t counter_prints_static_data = 0;
00302 uint8_t afk_counter = 0;
00303 String data;
00304 bool enable_starting_text = false;
00305 bool enable_starting_sound = false;
00306 bool enable_ending_text = false;
00307 bool enable_ending_sound = false;
00308 bool cmd_go_sleep = false;
00309 bool enable_live_view = false;
00310 bool new_text_received = false;
00311 bool change_text_answered = false;
00312 MilliTimer delay_live_view;
00313 MilliTimer afk_timer;
00314 bool enable_charge_sound = false;
00315 bool enable_death_battery_sound = false;
00316 bool enable_full_charge_sound = false;
00317 bool test_enable = false;
00318
00319 // FLAGS
00320 bool flag_msg_init = false, flag_msg_sleep = false, flag_error = false, flag_diag_header_printed =
00321   false, flag_initialize = false, flag_sleep = false, flag_return = false;
00321 bool trigger_Display_volt = true;
00322 bool go_sleep = true;
00323 bool diag_check = false;
00324 volatile bool flag_irq_center_button = false;
00325 volatile bool flag_irq_death_battery = false;
00326 bool flag_usb_change = false;
00327 bool print_diagnostic_static_data = false;
00328
00329 bool flag_sound = false, flag_display_capacity = false, flag_work = false, flag_test = false,
00330   flag_cap_one_shot = false, flag_update_eeprom = false, flag_irq_singleshot = false;
00330 int16_t flag_arranque = C_TIMER_IDLE, flag_waiting = C_TIMER_IDLE, flag_timer_low_bright =
00331   C_TIMER_IDLE;
00331
00332 //Resets Display text
00333 bool reset_init_text = true, reset_capacity_off_text = true, reset_sleep_text = true, reset_error_text =
00334   true, reset_diag_text = true;
00334
00335 void setup()
00336 {
00340   Serial.begin(9600);
00341   Wire.begin();
00342   ledmatrix.begin();
00343   // ledmatrix.fillRect(0, 0, C_DISPLAY_WIDTH, 7, 30);
00344   //while (!Serial)
00345   //{
00346   //};
00347   InitBuzzer(C_MODE_DEFAULT);
00348   Init_diagnostic_elements();
00349   theory_Vout = ReadDiagnosticData(C_THEORY_VOLTAGE);
00350   Serial.print("Test Bl_V4.");
00351   pinMode(C_PIN_SHIPPING_MOD, OUTPUT);
00352   pinMode(C_PIN_BUTTON_CENTER, INPUT_PULLUP);
00353   pinMode(C_PIN_BUTTON_UP, INPUT_PULLUP);
00354   pinMode(C_PIN_BUTTON_DOWN, INPUT_PULLUP);
00355   pinMode(C_PIN_USB_CONNEXION, INPUT_PULLUP);
00356   pinMode(C_PIN_DEATH_BATTERY_CHECK, INPUT_PULLUP);
00357
00358   pinMode(C_PIN_FLAG_CHARG, OUTPUT);
00359   pinMode(C_PIN_SCL_2, OUTPUT);
00360   pinMode(C_PIN_SDA_2, OUTPUT);
00361   pinMode(C_PIN_DEATH_BATTERY_CHECK, OUTPUT);
00362   pinMode(C_PIN_FLAG_BATTLOW, OUTPUT);
00363   pinMode(C_PIN_USB_SEL, OUTPUT);

```

```

00364
00365     digitalWrite(C_PIN_SHIPPING_MOD, LOW);
00366     attachInterrupt(C_PIN_USB_CONEXION, Irq_USB_Handler, RISING);
00367 //attachInterrupt(C_PIN_DEATH_BATTERY_CHECK, Irq_DeathBattery_Handler, RISING);
00368
00369     for (int i = 0; i < 5; i++)
00370     {
00371         test_enable = digitalRead(C_PIN_TEST_MODE);
00372         if (test_enable == true)
00373         {
00374             break;
00375         }
00376         delay(500);
00377     }
00378     if (test_enable == true)
00379     {
00380         while (1)
00381         {
00382             for (int i = 0; i < 8; i++)
00383             {
00384                 sample_IOut += analogRead(C_PIN_I_OUT);
00385                 sample_Vin += analogRead(C_PIN_V_IN);
00386                 sample_VOut += analogRead(C_PIN_V_OUT);
00387             }
00388
00389             sample_IOut = sample_IOut / 8;
00390             sample_Vin = sample_Vin / 8;
00391             sample_VOut = sample_VOut / 8;
00392
00393             digitalWrite(C_PIN_FLAG_CHARG);
00394             digitalRead(C_PIN_SCL_2);
00395             digitalRead(C_PIN_SDA_2);
00396             digitalRead(C_PIN_FLAG_BATTLOW);
00397             digitalWrite(C_PIN_USB_SEL, HIGH); // <--- La idea es ir conmutando el multiplexor y
00398             comprobar que el multiplexor funciona, y los pines de la CPU funciona.
00399
00400             Serial.print("\t");
00401             Serial.print(sample_IOut);
00402             Serial.print(";");
00403             Serial.print(sample_Vin);
00404             Serial.print(";");
00405             Serial.print(sample_VOut);
00406             Serial.print(";");
00407             Serial.print(C_PIN_FLAG_CHARG);
00408             Serial.print(";");
00409             Serial.print(C_PIN_SCL_2); // ?
00410             Serial.print(";");
00411             Serial.print(C_PIN_SDA_2); // ?
00412             Serial.print(";");
00413             Serial.print(C_PIN_FLAG_BATTLOW);
00414             Serial.print("\n");
00415         }
00416     }
00417 }
00418 ScreenON(ledmatrix, 1);
00419 delay(1000);
00420 sample_Vin = voltage_input_protection.getSample(C_PIN_V_IN) * 3300 / 4096 * 114 / 75;
00421 capacity = constrain(((sample_Vin - 3500) * 100 / 700), 0, 100);
00422 DisplayCap(capacity, ledmatrix);
00423 delay(1000);
00424 sw_status = C_SW_ST_SLEEP;
00425 while (1)
00426 {
00427     program_tick.set(10);
00428     //digitalWrite(C_PIN_PROFILE_CYCLE, HIGH);
00429     t0 = micros();
00430
00431     under_voltage_protection.threshold = (theory_Vout - 20) * 100;
00432
00433     /* Sense */
00434     sample_IOut = boost_check.getSample(C_PIN_I_OUT);
00435     sample_Vin = voltage_input_protection.getSample(C_PIN_V_IN) * 3300 / 4096 * 114 / 75;
00436     sample_VOut = under_voltage_protection.getSample(C_PIN_V_OUT);
00437     sample_POut = (sample_IOut * 3300 / 4096) * (sample_VOut * 189 / 39 * 3300 / 4096) / 1000;
00438
00439     /* BoostMode Monitor */
00440     if (boost_check.check(sample_IOut) == true)
00441     {
00442         output_mode = C_BOOST_MODE;
00443     }
00444     else if (boost_check.getCounter() == 0)
00445     {
00446         output_mode = C_NON_BOOST_MODE;
00447     }
00448
00449     if (sw_status == C_SW_ST_START_UP) // INITIALITATION
00450     {
00451         mode_bright_display = C_MODE_HIGH_BRIGHT;
00452     }
00453
00454     if (output_mode == C_BOOST_MODE)
00455     {
00456         if (sample_Vin > 3500)
00457         {
00458             if (sample_VOut < 189)
00459             {
00460                 if (sample_POut < 1000)
00461                 {
00462                     if (sample_Vin > 3500)
00463                     {
00464                         if (sample_VOut < 189)
00465                         {
00466                             if (sample_POut < 1000)
00467                             {
00468                                 if (sample_Vin > 3500)
00469                                 {
00470                                     if (sample_VOut < 189)
00471                                     {
00472                                         if (sample_POut < 1000)
00473                                         {
00474                                             if (sample_Vin > 3500)
00475                                             {
00476                                                 if (sample_VOut < 189)
00477                                                 {
00478                                                     if (sample_POut < 1000)
00479                                                     {
00480                                                         if (sample_Vin > 3500)
00481                                                         {
00482                                                             if (sample_VOut < 189)
00483                                                             {
00484                                                                 if (sample_POut < 1000)
00485                                                                 {
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02
```

```

00462     //digitalWrite(C_PIN_PROFILE_POWER_BAR, HIGH);
00463     PowerBar(0, ledmatrix, mode_bright_display);
00464     //digitalWrite(C_PIN_PROFILE_POWER_BAR, LOW);
00465     sw_output = C_OFF;
00466     trigger_Display_volt = true;
00467     if (flag_arranque == C_TIMER_ARMED)
00468     {
00469         if (timer_arranque.poll() != C_TIMER_NOT_EXPIRED)
00470         {
00471             flag_arranque = C_TIMER_DONE;
00472         }
00473     }
00474     else if (flag_arranque == C_TIMER_IDLE)
00475     {
00476         timer_arranque.set(1000);
00477         flag_arranque = C_TIMER_ARMED;
00478     }
00479     if (enable_starting_text == false)
00480     {
00481         flag_msg_init = true;
00482     }
00483     else if (flag_msg_init == false)
00484     {
00485         //digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
00486         display_status = DisplayText(start_string, ledmatrix, reset_init_text,
00487         mode_bright_display);
00488         //digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
00489         reset_init_text = false;
00490         if (display_status == C_DISPLAY_ST_NOT_BUSSY)
00491         {
00492             flag_msg_init = true;
00493             reset_init_text = true;
00494         }
00495         if (enable_starting_sound == false)
00496         {
00497             flag_sound = true;
00498         }
00499         else if (flag_sound == false)
00500         {
00501             if (enable_starting_text == false)
00502             {
00503                 DisplayCap(capacity, ledmatrix);
00504             }
00505             //digitalWrite(C_PIN_PROFILE_BUZZER, HIGH);
00506             buzzer_state = playSound(C_SOUND_START);
00507             //digitalWrite(C_PIN_PROFILE_BUZZER, LOW);
00508             if (buzzer_state == C_BUZZER_NOT_BUSSY)
00509             {
00510                 flag_sound = true;
00511             }
00512         }
00513         if ((flag_arranque == C_TIMER_DONE) && (flag_msg_init == true) && (flag_display_capacity
00514 == false) && (flag_sound == true))
00515         {
00516             flag_display_capacity = true;
00517             capacity = constrain(((sample_Vin - 3500) * 100 / 700), 0, 100);
00518             if (capacity <= C_LOW_BATTERY_LEVEL)
00519             {
00520                 while (flag_low_battery == false)
00521                 {
00522                     if ((DisplayLowBattery(ledmatrix) != C_DISPLAY_ST_BUSSY) && (low_batt_display
00523 == false))
00524                     {
00525                         low_batt_display = true;
00526                     }
00527                     if (low_batt_sound == false)
00528                     {
00529                         if (playSound(C_SOUND_LOW_BATTERY) != C_BUZZER_BUSSY)
00530                         {
00531                             low_batt_sound = true;
00532                         }
00533                         if ((low_batt_sound == true) && (low_batt_display == true))
00534                         {
00535                             flag_low_battery = true;
00536                         }
00537                     }
00538                     low_batt_display = false;
00539                     low_batt_sound = false;
00540                 }
00541             }
00542             flag_low_battery = false;
00543         }
00544     DisplayCap(capacity, ledmatrix);
00545

```

```

00546         // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
00547         timer_display_capacity.set(1000);
00548     }
00549     if (flag_display_capacity == true)
00550     {
00551         if (timer_display_capacity.poll() != C_TIMER_NOT_EXPIRED)
00552         {
00553             if (flag_test == true)
00554             {
00555                 timer_display_capacity.set(1000);
00556                 flag_test = false;
00557             }
00558             else
00559             {
00560                 flag_work = true;
00561             }
00562         }
00563         else
00564         {
00565             if (button_event == C_LP_CENTER)
00566             {
00567                 flag_test = true;
00568             }
00569         }
00570     }
00571     else if ((sw_status == C_SW_ST_RUN) || (sw_status == C_SW_ST_STOP)) // WORK
00572     {
00573         sw_output = C_ON;
00574
00575         if (sw_status == C_SW_ST_RUN)
00576         {
00577             user_output = C_ON;
00578
00579             /* Proteccions */
00580             if (mask_protection_state == C_MASK_DEACTIVATE)
00581             {
00582
00583                 /* Consumption Monitor */
00584                 if (over_consumption_protection.check(sample_IOut) == true)
00585                 {
00586
00587                     consmptn_event_protection_counter++;
00588                     if (consmptn_event_protection_counter == MAX_NUM_PROTECTION_EVENTS)
00589                     {
00590                         flag_error = true;
00591                         //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
00592                         IncrementDiagnosticData(1, C_CONSUMPTION_ERROR);
00593                         //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
00594                         over_consumption_protection.setCounter(0);
00595                     }
00596                 }
00597             }
00598             {
00599                 protection_event_delay.set(100);
00600                 protection_event_delay_flag = true;
00601                 over_consumption_protection.setCounter(C_RETRY_750_COUNT);
00602                 mask_protection_state = C_MASK_ACTIVATE;
00603                 sw_output = C_OFF;
00604             }
00605         }
00606         /* Voltage Input Monitor */
00607         if (voltage_input_protection.check(5000 - sample_Vin) == true)
00608         {
00609
00610             voltage_input_event_protection_counter++;
00611             if (voltage_input_event_protection_counter == MAX_NUM_PROTECTION_EVENTS)
00612             {
00613                 flag_error = true;
00614                 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
00615                 IncrementDiagnosticData(1, C_VOLTAGE_INPUT_ERROR);
00616                 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
00617                 voltage_input_protection.setCounter(0);
00618             }
00619         }
00620         else
00621         {
00622             protection_event_delay.set(100);
00623             protection_event_delay_flag = true;
00624             voltage_input_protection.setCounter(C_RETRY_750_COUNT);
00625             mask_protection_state = C_MASK_ACTIVATE;
00626             sw_output = C_OFF;
00627         }
00628
00629         /* Power Monitor */
00630         if (over_power_protection.check(sample_POut) == true)
00631         {
00632

```

```

00633             OP_event_protection_counter++;
00634             if (OP_event_protection_counter == MAX_NUM_PROTECTION_EVENTS)
00635             {
00636                 flag_error = true;
00637                 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
00638                 IncrementDiagnosticData(1, C_POWER_ERROR);
00639                 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
00640                 over_power_protection.setCounter(0);
00641             }
00642             else
00643             {
00644                 protection_event_delay.set(100);
00645                 protection_event_delay_flag = true;
00646                 over_power_protection.setCounter(over_power_protection.limit / 2);
00647                 mask_protection_state = C_MASK_ACTIVATE;
00648                 sw_output = C_OFF;
00649             }
00650         }
00651
00652         /*      Voltage Monitor      */
00653         if ((under_voltage_protection.check(sample_VOut) == false) &&
00654             (under_voltage_protection.getCounter() == 0))
00655         {
00656             UV_event_protection_counter++;
00657             if (UV_event_protection_counter == MAX_NUM_PROTECTION_EVENTS)
00658             {
00659                 flag_error = true;
00660                 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
00661                 IncrementDiagnosticData(1, C_VOLTAGE_ERROR);
00662                 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
00663                 under_voltage_protection.setCounter(under_voltage_protection.limit);
00664             }
00665             else
00666             {
00667                 protection_event_delay.set(100);
00668                 protection_event_delay_flag = true;
00669                 under_voltage_protection.setCounter(under_voltage_protection.limit / 2);
00670                 mask_protection_state = C_MASK_ACTIVATE;
00671                 sw_output = C_OFF;
00672             }
00673         }
00674         else if (mask_protection_state == C_MASK_ACTIVATE)
00675         {
00676             if ((protection_event_delay_flag == true) && (protection_event_delay.poll() != C_TIMER_NOT_EXPIRED))
00677             {
00678                 mask_protection_state = C_MASK_DEACTIVATE;
00679                 sw_output = C_ON;
00680             }
00681         }
00682     }
00683     else if (sw_status == C_SW_ST_STOP) // STOP
00684     {
00685         user_output = C_OFF;
00686         if (flag_low_battery == false)
00687         {
00688             if (timer_recover_voltage.poll() != C_TIMER_NOT_EXPIRED)
00689             {
00690                 capacity = constrain((sample_Vin - 3500) * 100 / 700, 0, 100);
00691                 if (capacity <= C_LOW_BATTERY_LEVEL)
00692                 {
00693                     if (prev_state == C_SW_ST_ERROR)
00694                     {
00695                         flag_low_battery = true;
00696                     }
00697
00698                     while (flag_low_battery == false)
00699                     {
00700                         if ((DisplayLowBattery(ledmatrix) != C_DISPLAY_ST_BUSSY) &&
00701                             (low_batt_display == false))
00702                         {
00703                             low_batt_display = true;
00704                         }
00705                         if (low_batt_sound == false)
00706                         {
00707                             if (playSound(C_SOUND_LOW_BATTERY) != C_BUZZER_BUSSY)
00708                             {
00709                                 low_batt_sound = true;
00710                             }
00711                         }
00712                         if ((low_batt_sound == true) && (low_batt_display == true))
00713                         {
00714                             flag_low_battery = true;
00715                         }
00716                     }
00717                 }
00718             }
00719         }
00720     }
00721 }
```

```

00717             low_batt_display = false;
00718             low_batt_sound = false;
00719             DisplayCap(capacity, ledmatrix);
00720         }
00721     else
00722     {
00723         flag_low_battery = false;
00724         timer_recover_voltage.set(1000);
00725     }
00726 }
00727 }
00728 if (button_event == C_NONE_EVENT)
00729 {
00730     if (timer_idle.poll() != C_TIMER_NOT_EXPIRED)
00731     {
00732         cont_idle_timer++; // Incrementamos el contador.
00733         if (cont_idle_timer == C_IDLE_TIMER_COUNT) // Si alcanzamos los minutos de
inactividad.
00734     }
00735     flag_sleep = true; // Flag ON para pasar al estado de Sleep
00736     cont_idle_timer = 0; // Reset del contador de minutos.
00737 }
00738 else
00739 {
00740     timer_idle.set(30000); // Inicio del contador de 30 seg.
00741 }
00742 }
00743 }
00744 else
00745 {
00746     timer_idle.set(30000); // Inicio del contador de 30 seg para el Idle Timer.
00747     cont_idle_timer = 0; // Reset del contador de minutos.
00748 }
00749 }
00750 // Buttons
00751 if (button_event == C_NONE_EVENT)
00752 {
00753     if (flag_timer_low_bright == C_TIMER_ARMED)
00754     {
00755         if (timer_low_bright.poll() != C_TIMER_NOT_EXPIRED)
00756         {
00757             mode_brightness_display = C_MODE_LOW_BRIGHT;
00758             flag_timer_low_bright = C_TIMER_DONE;
00759             trigger_Display_volt = true; // Manda actualizar el display con el nuevo
brillo
00760             sound = C_SOUND_MUTE;
00761         }
00762     }
00763     else if (flag_timer_low_bright == C_TIMER_IDLE)
00764     {
00765         timer_low_bright.set(C_TIME_TO_LIGHT_DOWN);
00766         flag_timer_low_bright = C_TIMER_ARMED;
00767     }
00768     else if (flag_timer_low_bright == C_TIMER_DONE)
00769     {
00770         mode_brightness_display = C_MODE_LOW_BRIGHT;
00771     }
00772 }
00773 else
00774 {
00775     sound = C_SOUND_MUTE;
00776     if (flag_timer_low_bright == C_TIMER_ARMED)
00777     {
00778         timer_low_bright.set(C_TIME_TO_LIGHT_DOWN);
00779     }
00780     else if (flag_timer_low_bright == C_TIMER_DONE)
00781     {
00782         flag_timer_low_bright = C_TIMER_IDLE;
00783     }
00784     mode_brightness_display = C_MODE_HIGH_BRIGHT;
00785 }
00786 if (button_event == C_LP_UP)
00787 {
00788     if (theory_Vout < 160)
00789     {
00790         theory_Vout += 10;
00791         sound = C_SOUND_UP;
00792         trigger_Display_volt = true;
00793     }
00794 }
00795 else if (button_event == C_CLICK_UP)
00796 {
00797     if (theory_Vout < 160)
00798     {
00799         theory_Vout += 1;
00800         sound = C_SOUND_UP;
00801         trigger_Display_volt = true;

```

```

00802             }
00803         }
00804     else if (button_event == C_LP_DOWN)
00805     {
00806         if (theory_Vout > 50)
00807         {
00808             theory_Vout -= 10;
00809             sound = C_SOUND_DOWN;
00810             trigger_Display_volt = true;
00811         }
00812     }
00813 else if (button_event == C_CLICK_DOWN)
00814 {
00815     if (theory_Vout > 50)
00816     {
00817         theory_Vout -= 1;
00818         sound = C_SOUND_DOWN;
00819         trigger_Display_volt = true;
00820     }
00821 }
00822 }
00823
00824 theory_Vout = constrain(theory_Vout, C_MIN_VOLT, C_MAX_VOLT);
00825
00826 /*      Display voltage      */
00827
00828 if (trigger_Display_volt == true)
00829 {
00830     if (sound != C_SOUND_MUTE)
00831     {
00832         //digitalWrite(C_PIN_PROFILE_BUZZER, HIGH);
00833         while (playSound(sound) == C_BUZZER_BUSSY)
00834         {
00835         }
00836         //digitalWrite(C_PIN_PROFILE_BUZZER, LOW);
00837         /* code */
00838     }
00839     // digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
00840     DisplayVolt(theory_Vout, ledmatrix, mode_bright_display);
00841     // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
00842     trigger_Display_volt = false;
00843 }
00844 //digitalWrite(C_PIN_PROFILE_POWER_BAR, HIGH);
00845 UpdatePowerBar(sample_POut, ledmatrix, mode_bright_display);
00846 //digitalWrite(C_PIN_PROFILE_POWER_BAR, LOW);
00847 }
00848 else if (sw_status == C_SW_ST_CAPACITY) // CAPACITY
00849 {
00850     mode_bright_display = C_MODE_HIGH_BRIGHT;
00851     if (stage_capacity_1 == true)
00852     {
00853         flag_display_capacity = true;
00854         if (flag_cap_one_shot == false)
00855         {
00856             flag_cap_one_shot = true;
00857             flag_display_capacity = false;
00858             if (prev_state == C_SW_ST_RUN)
00859             {
00860                 digitalWrite(C_PIN_SHIPPING_MOD, LOW);
00861                 delay(1000);
00862             }
00863             capacity = constrain((sample_Vin - 3500) * 100 / 700, 0, 100);
00864             if (capacity <= C_LOW_BATTERY_LEVEL)
00865             {
00866                 flag_low_battery = false;
00867                 while (flag_low_battery == false)
00868                 {
00869                     if ((DisplayLowBattery(ledmatrix) != C_DISPLAY_ST_BUSSY) &&
00870 (low_batt_display == false))
00871                     {
00872                         low_batt_display = true;
00873                     }
00874                     if (low_batt_sound == false)
00875                     {
00876                         if (playSound(C_SOUND_LOW_BATTERY) != C_BUZZER_BUSSY)
00877                         {
00878                             low_batt_sound = true;
00879                         }
00880                         if ((low_batt_sound == true) && (low_batt_display == true))
00881                         {
00882                             flag_low_battery = true;
00883                         }
00884                     }
00885                     low_batt_display = false;
00886                     low_batt_sound = false;
00887                 }
00888 }
00889 }
```

```

00888         // digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
00889         DisplayCap(capacity, ledmatrix, mode_bright_display);
00890
00891         // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
00892     }
00893 }
00894 if (flag_display_capacity == false)
00895 {
00896     timer_display_capacity.set(1100);
00897     flag_display_capacity = true;
00898 }
00899 else if (stage_capacity_2 == true)
00900 {
00901     if ((reset_capacity_off_text == true) || (display_status == C_DISPLAY_ST_BUSSY))
00902     {
00903         // digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
00904         display_status = DisplayText(capacity_ask_off, ledmatrix, reset_capacity_off_text,
00905         mode_bright_display);
00906         // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
00907         reset_capacity_off_text = false;
00908         //digitalWrite(C_PIN_PROFILE_BUZZER, HIGH);
00909         while (playSound(C_SOUND_UP) == C_BUZZER_BUSSY)
00910         {
00911             //digitalWrite(C_PIN_PROFILE_BUZZER, LOW);
00912         }
00913     }
00914     if (timer_display_capacity.poll() != C_TIMER_NOT_EXPIRED)
00915     {
00916         if (click_events >= 2)
00917         {
00918             if (stage_capacity_1 == true)
00919             {
00920                 stage_capacity_1 = false;
00921                 stage_capacity_2 = true;
00922                 click_events = 0;
00923                 flag_display_capacity = false;
00924             }
00925             else if (stage_capacity_2 == true)
00926             {
00927                 click_events = 0;
00928                 flag_sleep = true;
00929                 stage_capacity_1 = true;
00930                 stage_capacity_2 = false;
00931             }
00932         }
00933     }
00934     else
00935     {
00936         flag_return = true;
00937         click_events = 0;
00938         stage_capacity_1 = true;
00939         stage_capacity_2 = false;
00940     }
00941 }
00942 else
00943 {
00944     if (button_event == C_LP_CENTER)
00945     {
00946         click_events++;
00947     }
00948     //digitalWrite(C_PIN_PROFILE_POWER_BAR, HIGH);
00949     UpdatePowerBar(sample_POut, ledmatrix, mode_bright_display);
00950     //digitalWrite(C_PIN_PROFILE_POWER_BAR, LOW);
00951 }
00952 else if (sw_status == C_SW_ST_SLEEP) // SLEEP
00953 {
00954     mode_bright_display = C_MODE_HIGH_BRIGHT;
00955
00956     sw_output = C_OFF;
00957     if (enable_ending_text == false)
00958     {
00959         flag_msg_sleep = true;
00960         SwitchScreenOff(ledmatrix);
00961         PowerBar(0, ledmatrix);
00962     }
00963     else if (flag_msg_sleep == false)
00964     {
00965         //digitalWrite(C_PIN_PROFILE_POWER_BAR, HIGH);
00966         PowerBar(0, ledmatrix, mode_bright_display);
00967         //digitalWrite(C_PIN_PROFILE_POWER_BAR, LOW);
00968         // digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
00969         display_status = DisplayText(msg_sleep, ledmatrix, reset_sleep_text,
00970         mode_bright_display);
00971         // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
00972         reset_sleep_text = false;
00973         if (display_status == C_DISPLAY_ST_NOT_BUSSY)

```

```

00973         {
00974             flag_msg_sleep = true;
00975             SwitchScreenOff(ledmatrix);
00976             PowerBar(0, ledmatrix);
00977         }
00978     }
00979     if (enable Ending Sound == false)
00980     {
00981         flag_sound = true;
00982     }
00983     if (flag_sound == false)
00984     {
00985         //digitalWrite(C_PIN_PROFILE_BUZZER, HIGH);
00986         buzzer_state = playSound(C_SOUND_END);
00987         //digitalWrite(C_PIN_PROFILE_BUZZER, LOW);
00988         if (buzzer_state == C_BUZZER_NOT_BUSSY)
00989         {
00990             flag_sound = true;
00991         }
00992     }
00993     if ((flag_msg_sleep == true) && (flag_waiting == C_TIMER_IDLE) && (flag_sound == true))
00994     {
00995         flag_waiting = C_TIMER_ARMED;
00996         timer_wait_sleep.set(600);
00997     }
00998     if (flag_waiting == C_TIMER_ARMED)
00999     {
01000         if (timer_wait_sleep.poll() != C_TIMER_NOT_EXPIRED)
01001         {
01002             if (flag_test == true)
01003             {
01004                 timer_wait_sleep.set(600);
01005                 flag_test = false;
01006             }
01007             else
01008             {
01009                 flag_waiting = C_TIMER_DONE;
01010             }
01011         }
01012     }
01013     else
01014     {
01015         if (button_event == C_LP_CENTER)
01016         {
01017             flag_test = true;
01018         }
01019     }
01020     else if (flag_waiting == C_TIMER_DONE)
01021     {
01022         if (go_sleep == true)
01023         {
01024             if (flag_irq_center_button == true)
01025             {
01026                 flag_irq_center_button = false;
01027             }
01028             if (flag_irq_death_battery == true)
01029             {
01030                 flag_irq_death_battery = false;
01031             }
01032             SaveEepromChasis();
01033             LowPower.attachInterruptWakeup(C_PIN_BUTT_CENTER, IrqCenterButtonHandler,
01034             FALLING);
01035             //LowPower.attachInterruptWakeup(C_PIN_BUTT_UP, Irq_DeathBattery_Handler,
01036             FALLING);
01037             LowPower.sleep();
01038             detachInterrupt(C_PIN_BUTT_CENTER);
01039             //digitalWrite(C_PIN_PROFILE_BUTTONS, LOW);
01040             if (flag_irq_center_button == true)
01041             {
01042                 go_sleep = false;
01043                 timer_irq_button_center.set(1200);
01044             }
01045             else if (flag_irq_death_battery == true)
01046             {
01047                 go_sleep = false;
01048             }
01049         }
01050         else if (go_sleep == false)
01051         {
01052             if (flag_irq_center_button == true)
01053             {
01054                 if (timer_irq_button_center.poll() != C_TIMER_NOT_EXPIRED)
01055                 {
01056                     go_sleep = true;
01057                 }
01058             }
01059         }
01060     }

```

```

01058             if (button_event == C_LP_CENTER)
01059             {
01060                 flag_initialize = true;
01061                 go_sleep = true;
01062             }
01063         }
01064     }
01065     else if (flag_irq_death_battery == true)
01066     {
01067         while (playSound(C_SOUND_UP) == C_BUZZER_BUSSY)
01068         {
01069         }
01070         go_sleep = true;
01071     }
01072 }
01073 }
01074 }
01075 else if (sw_status == C_SW_ST_ERROR) // ERROR
01076 {
01077     mode_bright_display = C_MODE_HIGH_BRIGHT;
01078     //digitalWrite(C_PIN_PROFILE_POWER_BAR, HIGH);
01079     sw_output = C_OFF;
01080     //digitalWrite(C_PIN_PROFILE_POWER_BAR, LOW);
01081     //digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
01082     if (display_error_status == C_DISPLAY_ST_BUSSY)
01083     {
01084         DisplayError(ledmatrix);
01085
01086         if (timer_display_error.poll() != C_TIMER_NOT_EXPIRED)
01087         {
01088             display_error_status = C_DISPLAY_ST_NOT_BUSSY;
01089         }
01090     }
01091     if (buzzer_error_state == C_BUZZER_BUSSY)
01092     {
01093         while (playSound(C_SOUND_ERROR) == C_BUZZER_BUSSY)
01094         {
01095             /* code */
01096         }
01097         buzzer_error_state = C_BUZZER_NOT_BUSSY;
01098     }
01099
01100     if ((buzzer_error_state == C_BUZZER_NOT_BUSSY) && (display_error_status == C_DISPLAY_ST_NOT_BUSSY))
01101     {
01102         flag_error = false;
01103         voltage_input_event_protection_counter = 0;
01104         consmptn_event_protection_counter = 0;
01105         OP_event_protection_counter = 0;
01106         UV_event_protection_counter = 0;
01107     }
01108     if (flag_update_eeprom == false)
01109     {
01110         //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01111         SaveEepromChasis();
01112         UpdateEepromBateriy();
01113         // digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01114         flag_update_eeprom = true;
01115     }
01116 }
01117 else if (sw_status == C_SW_ST_USB) // USB
01118 {
01119
01120     sw_output = C_ON;
01121
01122     /*      Serial Port Looking For      */
01123     if (diag_check == false)
01124     {
01125         user_output = C_OFF;
01126         if (Serial.available())
01127         {
01128             data = Serial.readString();
01129             data.trim();
01130             Serial.print(data);
01131             if (data == C_DIAGNOSTIC_PASSWORD)
01132             {
01133                 print_diagnostic_static_data = true;
01134                 timer_print_diagnostic.set(500);
01135                 data = "";
01136                 diag_check = true;
01137                 mode_bright_display = C_MODE_LOW_BRIGHT;
01138                 //digitalWrite(C_PIN_PROFILE_POWER_BAR, HIGH);
01139                 PowerBar(0, ledmatrix, mode_bright_display);
01140                 //digitalWrite(C_PIN_PROFILE_POWER_BAR, LOW);
01141                 DisplayDiagnosticMode(ledmatrix);
01142             }
01143         }
01144     }

```

```

01144 }
01145     else if (diag_check == true)
01146 {
01147     if (print_diagnostic_static_data == true)
01148     {
01149         user_output = C_OFF;
01150         if (counter_prints_static_data < C_NUMB_PRINTS_STATIC_DATA)
01151         {
01152             if (timer_print_diagnostic.poll() != C_TIMER_NOT_EXPIRED)
01153             {
01154                 timer_print_diagnostic.set(500);
01155                 PrintStaticData();
01156                 counter_prints_static_data++;
01157             }
01158         }
01159     }
01160     else
01161     {
01162         counter_prints_static_data = 0;
01163         print_diagnostic_static_data = false;
01164     }
01165 }
01166     else if (print_diagnostic_static_data == false)
01167 {
01168     if (enable_live_view == false)
01169     {
01170         user_output = C_OFF;
01171         if (Serial.available())
01172         {
01173             data = Serial.readString();
01174             data.trim();
01175             Serial.println(data);
01176             if (data == C_CMD_DISPLAY_STARTING_OFF)
01177             {
01178                 enable_starting_text = false;
01179                 LogDiagnosticData(enable_starting_text, C_HACK_START_DISPLAY);
01180                 Serial.println("Starting Text Disable.");
01181             }
01182             else if (data == C_CMD_DISPLAY_STARTING_ON)
01183             {
01184                 enable_starting_text = true;
01185                 LogDiagnosticData(enable_starting_text, C_HACK_START_DISPLAY);
01186                 Serial.println("Starting Text Enable");
01187                 Serial.println("Change Text?");
01188                 Serial.print("Actual:");
01189                 Serial.println(start_string);
01190                 while (change_text_answered == false)
01191                 {
01192                     if (Serial.available())
01193                     {
01194                         data = Serial.readString();
01195                         data.trim();
01196                         Serial.println(data);
01197                         if (data == "yes")
01198                         {
01199                             Serial.println("Insert new starting Text:");
01200                             afk_timer.set(30000);
01201                             while (new_text_received == false)
01202                             {
01203                                 if (afk_timer.poll() != C_TIMER_NOT_EXPIRED)
01204                                 {
01205                                     afk_counter++;
01206                                     if (afk_counter == (5 * 2))
01207                                     {
01208                                         new_text_received = true;
01209                                     }
01210                                 }
01211                                 if (Serial.available())
01212                                 {
01213                                     data = Serial.readString();
01214                                     data.trim();
01215                                     Serial.println(data);
01216                                     start_string = data;
01217                                     new_text_received = true;
01218                                 }
01219                                 new_text_received = false;
01220                                 change_text_answered = true;
01221                             }
01222                         else if (data == "no")
01223                         {
01224                             change_text_answered = true;
01225                         }
01226                     }
01227                 }
01228             }
01229         }
01230     }
01231 }
```

```

01231     else if (data == C_CMD_DISPLAY_ENDING_OFF)
01232     {
01233         enable_ending_text = false;
01234         LogDiagnosticData(enable_ending_text, C_HACK_END_DISPLAY);
01235         Serial.println("Ending Text Disable");
01236     }
01237     else if (data == C_CMD_DISPLAY_ENDING_ON)
01238     {
01239         enable_ending_text = true;
01240         LogDiagnosticData(enable_ending_text, C_HACK_END_DISPLAY);
01241         Serial.println("Ending Text Enable");
01242         Serial.println("Change Text?");
01243         Serial.print("Actual:");
01244         Serial.println(msg_sleep);
01245         while (change_text_answered == false)
01246         {
01247             if (Serial.available())
01248             {
01249                 data = Serial.readString();
01250                 data.trim();
01251                 Serial.println(data);
01252                 if (data == "yes")
01253                 {
01254                     Serial.println("Insert new ending Text:");
01255                     afk_timer.set(30000);
01256                     while (new_text_received == false)
01257                     {
01258                         if (afk_timer.poll() != C_TIMER_NOT_EXPIRED)
01259                         {
01260                             afk_counter++;
01261                             if (afk_counter == (5 * 2))
01262                             {
01263                                 new_text_received = true;
01264                             }
01265                         }
01266                     }
01267                     if (Serial.available())
01268                     {
01269                         data = Serial.readString();
01270                         data.trim();
01271                         Serial.println(data);
01272                         msg_sleep = data;
01273                         new_text_received = true;
01274                     }
01275                     new_text_received = false;
01276                     change_text_answered = true;
01277                 }
01278             }
01279             else if (data == "no")
01280             {
01281                 change_text_answered = true;
01282             }
01283         }
01284     }
01285     change_text_answered = false;
01286 }
01287 else if (data == C_CMD_SOUND_STARTING_OFF)
01288 {
01289     enable_starting_sound = false;
01290     LogDiagnosticData(enable_starting_sound, C_HACK_START_SOUND);
01291     Serial.println("Starting Sound Disable");
01292 }
01293 else if (data == C_CMD_SOUND_STARTING_ON)
01294 {
01295     enable_starting_sound = true;
01296     LogDiagnosticData(enable_starting_sound, C_HACK_START_SOUND);
01297     Serial.println("Starting Sound Enable");
01298 }
01299 else if (data == C_CMD_SOUND_ENDING_OFF)
01300 {
01301     enable_ending_sound = false;
01302     LogDiagnosticData(enable_ending_sound, C_HACK_END_SOUND);
01303     Serial.println("Ending Sound Disable");
01304 }
01305 else if (data == C_CMD_SOUND_ENDING_ON)
01306 {
01307     enable_ending_sound = true;
01308     LogDiagnosticData(enable_ending_sound, C_HACK_END_SOUND);
01309     Serial.println("Ending Sound Enable");
01310 }
01311 else if (data == C_CMD_SOUND_FULL_CHARGE_OFF)
01312 {
01313     enable_full_charge_sound = false;
01314     LogDiagnosticData(enable_full_charge_sound, C_HACK_FULL_CHARGE_SOUND);
01315     Serial.println("Received and Applied");
01316 }
01317 else if (data == C_CMD_SOUND_FULL_CHARGE_ON)

```

```
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
{
    enable_full_charge_sound = true;
    LogDiagnosticData(enable_full_charge_sound, C_HACK_FULL_CHARGE_SOUND);
    Serial.println("Received and Applied");
}
else if (data == C_CMS_SOUND_DEATH_BATTERY_OFF)
{
    enable_death_battery_sound = false;
    LogDiagnosticData(enable_death_battery_sound,
C_HACK_DEATH_BATTERY_SOUND);
    Serial.println("Received and Applied");
}
else if (data == C_CMS_SOUND_DEATH_BATTERY_ON)
{
    enable_death_battery_sound = true;
    LogDiagnosticData(enable_death_battery_sound,
C_HACK_DEATH_BATTERY_SOUND);
    Serial.println("Received and Applied");
}
else if (data == C_CMS_SOUND_CHARGE_START_OFF)
{
    enable_charge_sound = true;
    LogDiagnosticData(enable_charge_sound, C_HACK_CHARGE_SOUND);
    Serial.println("Received and Applied");
}
else if (data == C_CMS_SOUND_CHARGE_START_ON)
{
    enable_charge_sound = true;
    LogDiagnosticData(enable_charge_sound, C_HACK_CHARGE_SOUND);
    Serial.println("Received and Applied");
}
else if (data == C_CMD_LIVE_ON)
{
    enable_live_view = true;
    delay_live_view.set(1000);
    Serial.println("Received and Applied");
}
else
{
    Serial.println("Command No identified");
}
}
else if (enable_live_view == true)
{
    if (delay_live_view.poll() != C_TIMER_NOT_EXPIRED)
    {
        delay_live_view.set(1000);
        PrintDiagnosticData();
    }

    if (Serial.available())
    {
        data = Serial.readString();
        data.trim();
        Serial.println(data);
        if (data == C_LIVE_CMD_OUT_ON)
        {
            user_output = C_ON;
        }
        else if (data == C_LIVE_CMD_OUT_OFF)
        {
            user_output = C_OFF;
        }
        else if (data == C_LIVE_CMD_VOLT_UP_1)
        {
            theory_Vout += 1;
        }
        else if (data == C_LIVE_CMD_VOLT_UP_10)
        {
            theory_Vout += 10;
        }
        else if (data == C_LIVE_CMD_VOLT_DOWN_1)
        {
            theory_Vout -= 1;
        }
        else if (data == C_LIVE_CMD_VOLT_DOWN_10)
        {
            theory_Vout -= 10;
        }
        else if (data == C_LIVE_CMD_OFF_LIVE_VIEW)
        {
            enable_live_view = false;
        }
        else
        {
            /* code */
        }
    }
}
```

```
01403                     }
01404             }
01405         }
01406     }
01407 }
01408 }
01409
01410 /*
01411 *      STATUS MANAGE
01412 */
01413 if (sw_status == C_SW_ST_START_UP) // START_UP
{
01415     if (flag_work == true)
01416     {
01417         flag_display_capacity = false;
01418         flag_work = false;
01419         flag_msg_init = false;
01420         flag_aranque = C_TIMER_IDLE;
01421         flag_sound = false;
01422         sw_status = C_SW_ST_STOP;
01423         prev_state = C_SW_ST_START_UP;
01424         // digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
01425         DisplayVolt(theory_Vout, ledmatrix, mode_bright_display);
01426         // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
01427         timer_idle.set(30000);
01428     }
01429     if (usb_status == C_USB_CONNECTED)
01430     {
01431         if (enable_charge_sound == true)
01432         {
01433             while (playSound(C_SOUND_CHARGE_IN) == C_BUZZER_BUSSY)
01434             {
01435             }
01436         }
01437         sw_status = C_SW_ST_USB;
01438         prev_state = C_SW_ST_START_UP;
01439         DisplayUsbIn(ledmatrix);
01440     }
01441 }
01442 else if (sw_status == C_SW_ST_RUN) // RUN
01443 {
01444     if (button_event == C_LP_CENTER)
01445     {
01446         sw_status = C_SW_ST_CAPACITY;
01447         prev_state = C_SW_ST_RUN;
01448         reset_capacity_off_text = true;
01449         state_to_return = C_SW_ST_RUN;
01450     }
01451     if (button_event == C_CLICK_CENTER)
01452     {
01453         sw_status = C_SW_ST_STOP;
01454         prev_state = C_SW_ST_RUN;
01455         //digitalWrite(C_PIN_PROFILE_BUZZER, HIGH);
01456         while (playSound(C_SOUND_OFF) == C_BUZZER_BUSSY)
01457         {
01458         }
01459         //digitalWrite(C_PIN_PROFILE_BUZZER, LOW);
01460         timer_idle.set(30000);
01461         flag_low_battery = false;
01462         timer_recover_voltage.set(1000);
01463     }
01464     if (usb_status == C_USB_CONNECTED)
01465     {
01466         if (enable_charge_sound == true)
01467         {
01468             while (playSound(C_SOUND_CHARGE_IN) == C_BUZZER_BUSSY)
01469             {
01470             }
01471         }
01472         sw_status = C_SW_ST_USB;
01473         prev_state = C_SW_ST_RUN;
01474         DisplayUsbIn(ledmatrix);
01475     }
01476     if (flag_error == true)
01477     {
01478         sw_status = C_SW_ST_ERROR;
01479         prev_state = C_SW_ST_RUN;
01480         flag_timer_low_bright = C_TIMER_IDLE;
01481         display_error_status = C_DISPLAY_ST_BUSSY;
01482         buzzer_error_state = C_BUZZER_BUSSY;
01483         timer_display_error.set(1000);
01484         playSound(C_SOUND_ERROR);
01485         SwitchScreenOff(ledmatrix);
01486         PowerBar(0, ledmatrix, mode_bright_display);
01487         DisplayError(ledmatrix);
01488     }
01489 }
```

```

01490         }
01491     else if (sw_status == C_SW_ST_STOP) // STOP
01492     {
01493         if (button_event == C_LP_CENTER)
01494         {
01495             sw_status = C_SW_ST_CAPACITY;
01496             prev_state = C_SW_ST_STOP;
01497             reset_capacity_off_text = true;
01498             state_to_return = C_SW_ST_STOP;
01499             flag_timer_low_bright = C_TIMER_IDLE;
01500         }
01501     if (button_event == C_CLICK_CENTER)
01502     {
01503         sw_status = C_SW_ST_RUN;
01504         prev_state = C_SW_ST_STOP;
01505         //digitalWrite(C_PIN_PROFILE_BUZZER, HIGH);
01506         while (playSound(C_SOUND_ON) == C_BUZZER_BUSSY)
01507         {
01508         }
01509         //digitalWrite(C_PIN_PROFILE_BUZZER, LOW);
01510     }
01511     if (usb_status == C_USB_CONNECTED)
01512     {
01513         if (enable_charge_sound == true)
01514         {
01515             while (playSound(C_SOUND_CHARGE_IN) == C_BUZZER_BUSSY)
01516             {
01517             }
01518         sw_status = C_SW_ST_USB;
01519         prev_state = C_SW_ST_STOP;
01520         DisplayUsbIn(ledmatrix);
01521     }
01522     if (flag_sleep == true)
01523     {
01524         sw_status = C_SW_ST_SLEEP;
01525         prev_state = C_SW_ST_STOP;
01526         flag_msg_sleep = true;
01527         flag_sound = true;
01528         flag_sleep = false;
01529     }
01530 }
01531 else if (sw_status == C_SW_ST_CAPACITY)
01532 {
01533     if (flag_sleep == true)
01534     {
01535         flag_sleep = false;
01536         sw_status = C_SW_ST_SLEEP;
01537         prev_state = C_SW_ST_CAPACITY;
01538         reset_sleep_text = true;
01539         flag_display_capacity = false;
01540         reset_capacity_off_text = false;
01541         flag_cap_one_shot = false;
01542     }
01543     if (flag_return == true)
01544     {
01545         sw_status = state_to_return;
01546         prev_state = C_SW_ST_CAPACITY;
01547         // digitalWrite(C_PIN_PROFILE_DISPLAY, HIGH);
01548         DisplayVolt(theory_Vout, ledmatrix, mode_bright_display);
01549         // digitalWrite(C_PIN_PROFILE_DISPLAY, LOW);
01550         flag_display_capacity = false;
01551         flag_cap_one_shot = false;
01552         flag_return = false;
01553     }
01554     if (usb_status == C_USB_CONNECTED)
01555     {
01556         if (enable_charge_sound == true)
01557         {
01558             while (playSound(C_SOUND_CHARGE_IN) == C_BUZZER_BUSSY)
01559             {
01560             }
01561         }
01562         sw_status = C_SW_ST_USB;
01563         prev_state = C_SW_ST_CAPACITY;
01564         DisplayUsbIn(ledmatrix);
01565     }
01566 }
01567 else if (sw_status == C_SW_ST_SLEEP) //SLEEP
01568 {
01569     if (flag_initialize == true)
01570     {
01571         flag_waiting = C_TIMER_IDLE;
01572         sw_status = C_SW_ST_START_UP;
01573         prev_state = C_SW_ST_SLEEP;
01574         reset_init_text = true;
01575         flag_msg_sleep = false;
01576     }

```

```

01577         flag_sound = false;
01578         flag_initialize = false;
01579         flag_low_battery = false;
01580         if ((enable_starting_sound == true) && (enable_starting_text == false))
01581     {
01582         capacity = ((sample_Vin - 3500) * 100 / 700);
01583         DisplayCap(capacity, ledmatrix);
01584     }
01585 }
01586 if (usb_status == C_USB_CONNECTED)
01587 {
01588     if (enable_charge_sound == true)
01589     {
01590         while (playSound(C_SOUND_CHARGE_IN) == C_BUZZER_BUSSY)
01591         {
01592         }
01593     }
01594     sw_status = C_SW_ST_USB;
01595     prev_state = C_SW_ST_SLEEP;
01596     DisplayUsbIn(ledmatrix);
01597 }
01598 }
01599 else if (sw_status == C_SW_ST_ERROR) // ERROR
01600 {
01601     if (flag_error == false)
01602     {
01603         sw_status = C_SW_ST_STOP;
01604         prev_state = C_SW_ST_ERROR;
01605         timer_idle.set(30000);
01606         flag_low_battery = false;
01607         timer_recover_voltage.set(1000);
01608     }
01609
01610     if (button_event == C_LP_CENTER)
01611     {
01612         sw_status = C_SW_ST_SLEEP;
01613         prev_state = C_SW_ST_ERROR;
01614         reset_sleep_text = true;
01615         flag_update_eeprom = false;
01616         voltage_input_event_protection_counter = 0;
01617         consmptn_event_protection_counter = 0;
01618         OP_event_protection_counter = 0;
01619         UV_event_protection_counter = 0;
01620     }
01621 if (usb_status == C_USB_CONNECTED)
01622 {
01623     if (enable_charge_sound == true)
01624     {
01625         while (playSound(C_SOUND_CHARGE_IN) == C_BUZZER_BUSSY)
01626         {
01627         }
01628     }
01629     sw_status = C_SW_ST_USB;
01630     prev_state = C_SW_ST_SLEEP;
01631     DisplayUsbIn(ledmatrix);
01632 }
01633 }
01634 else if (sw_status == C_SW_ST_USB)
01635 {
01636     if (usb_status == C_USB_DISCONNECTED)
01637     {
01638         if (enable_charge_sound == true)
01639         {
01640             while (playSound(C_SOUND_CHARGE_OUT) == C_BUZZER_BUSSY)
01641             {
01642             }
01643         }
01644
01645         DisplayUsbOut(ledmatrix);
01646         DisplayCap(capacity, ledmatrix);
01647         delay(1000);
01648         sw_status = C_SW_ST_SLEEP;
01649         prev_state = C_SW_ST_USB;
01650         flag_msg_sleep = true;
01651         flag_sound = true;
01652         diag_check = false;
01653     }
01654 }
01655 if (flag_usb_change == true)
01656 {
01657     delay(100);
01658     if (usb_status == C_USB_DISCONNECTED)
01659     {
01660         usb_status = C_USB_CONNECTED;
01661     }
01662 else if (usb_status == C_USB_CONNECTED)
01663 {

```

```

01664         usb_status = C_USB_DISCONNECTED;
01665     }
01666     flag_usb_change = false;
01667 }
01668
01669 /*      Output Control      */
01670 hw_output = sw_output & user_output;
01671
01672 if (hw_output == C_ON)
01673 {
01674     DCDC.SetVoltage(theory_Vout, output_mode);
01675     digitalWrite(C_PIN_SHIPPING_MOD, HIGH);
01676     LedWork(C_ON, ledmatrix, mode_bright_display);
01677 }
01678 else if (hw_output == C_OFF)
01679 {
01680     digitalWrite(C_PIN_SHIPPING_MOD, LOW);
01681     consmptn_event_protection_counter = 0;
01682     OP_event_protection_counter = 0;
01683     UV_event_protection_counter = 0;
01684     LedWork(C_OFF, ledmatrix, mode_bright_display);
01685 }
01686 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01687 LogDiagnosticData(theory_Vout, C_INSTANT_VOLTAGE);
01688 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01689 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01690 LogDiagnosticData(theory_Vout, C_THEORY_VOLTAGE);
01691 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01692 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01693 LogDiagnosticData(sample_IOut * 3300 / 4096, C_INSTANT_CURRENT);
01694 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01695 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01696 LogDiagnosticData(sample_POut, C_INSTANT_POWER);
01697 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01698 //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01699 LogDiagnosticData(capacity, C_CAPACITY);
01700 //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01701 /*      Timer      */
01702 if (timer_sec_count.poll(1000) != C_TIMER_NOT_EXPIRED)
01703 {
01704     cont_sec++;
01705     if (cont_sec == 3600)
01706     {
01707         IncrementDiagnosticData(1, C_NUMBER_CYCLES);
01708         IncrementDiagnosticData(1, C_WORK_TIME);
01709         //digitalWrite(C_PIN_PROFILE_EEPROM, HIGH);
01710         UpdateEepromBatery();
01711         SaveEepromChasis();
01712         //digitalWrite(C_PIN_PROFILE_EEPROM, LOW);
01713     }
01714 }
01715 /*      Check Battery Death Level      */
01716 if (flag_irq_death_battery == true)
01717 {
01718     delay(2000);
01719     if (enable_death_battery_sound == true)
01720     {
01721         /* code */
01722         while (playSound(C_SOUND_DEATH_BATTERY) == C_BUZZER_BUSSY)
01723         {
01724         }
01725     }
01726 }
01727
01728 /*      Read Buttons      */
01729 //digitalWrite(C_PIN_PROFILE_BUTTONS, HIGH);
01730 button_event = ReadDirPad();
01731 //digitalWrite(C_PIN_PROFILE_BUTTONS, LOW);
01732 /*
01733 *      END
01734 */
01735 t1 = micros();
01736 //digitalWrite(C_PIN_PROFILE_CYCLE, LOW);
01737 //Serial.println(t1 - t0);
01738 // while (program_tick.poll() == C_TIMER_NOT_EXPIRED)
01739 //      ;
01740 }
01741 }
01742
01743 void loop()
01744 {
01745 }
01746
01747 void IrqCenterButtonHandler()
01748 {
01749     flag_irq_center_button = true;
01750 }

```

```
01751 void Irq_DeathBattery_Handler()
01752 {
01753     flag_irq_death_battery = true;
01754 }
01755 void Irq_USB_Handler()
01756 {
01757     flag_usb_change = true;
01758 }
```



- 15.3 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/CODE/README.md File Reference
- 15.4 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0001-BUTTONS/README.md File Reference
- 15.5 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0002-DISPLAY/README.md File Reference
- 15.6 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0003-BUZZER/README.md File Reference
- 15.7 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0004-DCDC/README.md File Reference
- 15.8 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0005-SENSING/README.md File Reference
- 15.9 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0006-POWERBAR/README.md File Reference
- 15.10 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/README.md File Reference
- 15.11 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0010-LOGGING/README.md File Reference
- 15.12 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/readme.md File Reference
- 15.13 C:/Users/Javi/Team Dropbox/Javi

## 15.15 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0001-BUTTONS/Dpad.h File Reference

### Functions

- int16\_t [ReadDirPad \(\)](#)

*Esta funcion lee y detecta las pulsaciones cortas(CLICKs) y largas (LP) de 5 botones. Incluye protecciones antirebotes.*

### Variables

- const int16\_t [C\\_LP\\_CENTER](#) = 0x01
- const int16\_t [C\\_LP\\_UP](#) = 0x04
- const int16\_t [C\\_LP\\_DOWN](#) = 0x05
- const int16\_t [C\\_CLICK\\_CENTER](#) = 0x06
- const int16\_t [C\\_CLICK\\_UP](#) = 0x09
- const int16\_t [C\\_CLICK\\_DOWN](#) = 0x0A
- const int16\_t [C\\_NONE\\_EVENT](#) = 0x0B
- const int16\_t [C\\_PIN\\_BUTT\\_UP](#) = 9
- const int16\_t [C\\_PIN\\_BUTT\\_DOWN](#) = 8
- const int16\_t [C\\_PIN\\_BUTT\\_CENTER](#) = 5
- const int16\_t [C\\_TIMER\\_DEBOUNCE](#) = 50
- const int32\_t [C\\_TIMER\\_LONGPRESS](#) = 1000 + [C\\_TIMER\\_DEBOUNCE](#)
- const int32\_t [C\\_TIMER\\_LONGPRESS\\_LOOP](#) = 500
- const bool [button\\_pressed](#) = LOW
- const bool [button\\_not\\_pressed](#) = HIGH

### 15.15.1 Function Documentation

#### 15.15.1.1 [ReadDirPad\(\)](#)

```
int16_t ReadDirPad ( )
```

Esta funcion lee y detecta las pulsaciones cortas(CLICKs) y largas (LP) de 5 botones. Incluye protecciones antirebotes.

Returns

int16\_t

UP BUTTON

DOWN BUTTON

CENTER BUTTON

Definition at line 42 of file [Dpad.h](#).

### 15.15.2 Variable Documentation

#### 15.15.2.1 [button\\_not\\_pressed](#)

```
const bool button_not_pressed = HIGH
```

Definition at line 35 of file [Dpad.h](#).

### 15.15.2.2 button\_pressed

```
const bool button_pressed = LOW
```

Definition at line 34 of file [Dpad.h](#).

### 15.15.2.3 C\_CLICK\_CENTER

```
const int16_t C_CLICK_CENTER = 0x06
```

Definition at line 17 of file [Dpad.h](#).

### 15.15.2.4 C\_CLICK\_DOWN

```
const int16_t C_CLICK_DOWN = 0x0A
```

Definition at line 21 of file [Dpad.h](#).

### 15.15.2.5 C\_CLICK\_UP

```
const int16_t C_CLICK_UP = 0x09
```

Definition at line 20 of file [Dpad.h](#).

### 15.15.2.6 C\_LP\_CENTER

```
const int16_t C_LP_CENTER = 0x01
```

Definition at line 12 of file [Dpad.h](#).

### 15.15.2.7 C\_LP\_DOWN

```
const int16_t C_LP_DOWN = 0x05
```

Definition at line 16 of file [Dpad.h](#).

### 15.15.2.8 C\_LP\_UP

```
const int16_t C_LP_UP = 0x04
```

Definition at line 15 of file [Dpad.h](#).

### 15.15.2.9 C\_NONE\_EVENT

```
const int16_t C_NONE_EVENT = 0x0B
```

Definition at line 22 of file [Dpad.h](#).

### 15.15.2.10 C\_PIN\_BUTT\_CENTER

```
const int16_t C_PIN_BUTT_CENTER = 5
```

Definition at line 28 of file [Dpad.h](#).

### 15.15.2.11 C\_PIN\_BUTT\_DOWN

```
const int16_t C_PIN_BUTT_DOWN = 8
```

Definition at line 25 of file [Dpad.h](#).

### 15.15.2.12 C\_PIN\_BUTT\_UP

```
const int16_t C_PIN_BUTT_UP = 9
```

Definition at line 24 of file [Dpad.h](#).

### 15.15.2.13 C\_TIMER\_DEBOUNCE

```
const int16_t C_TIMER_DEBOUNCE = 50
```

Definition at line 30 of file [Dpad.h](#).

### 15.15.2.14 C\_TIMER\_LONGPRESS

```
const int32_t C_TIMER_LONGPRESS = 1000 + C_TIMER_DEBOUNCE
```

Definition at line 31 of file [Dpad.h](#).

### 15.15.2.15 C\_TIMER\_LONGPRESS\_LOOP

```
const int32_t C_TIMER_LONGPRESS_LOOP = 500
```

Definition at line 32 of file [Dpad.h](#).

## 15.16 Dpad.h

[Go to the documentation of this file.](#)

```
00001
00012 const int16_t C_LP_CENTER = 0x01;
00013 //const int16_t C_LP_RIGHT = 0x02;
00014 //const int16_t C_LP_LEFT = 0x03;
00015 const int16_t C_LP_UP = 0x04;
00016 const int16_t C_LP_DOWN = 0x05;
00017 const int16_t C_CLICK_CENTER = 0x06;
00018 //const int16_t C_CLICK_RIGHT = 0x07;
00019 //const int16_t C_CLICK_LEFT = 0x08;
00020 const int16_t C_CLICK_UP = 0x09;
00021 const int16_t C_CLICK_DOWN = 0x0A;
00022 const int16_t C_NONE_EVENT = 0x0B;
00023
00024 const int16_t C_PIN_BUTT_UP = 9;
00025 const int16_t C_PIN_BUTT_DOWN = 8;
00026 //const int16_t C_PIN_BUTT_RIGHT = 6;
00027 //const int16_t C_PIN_BUTT_LEFT = 7;
00028 const int16_t C_PIN_BUTT_CENTER = 5;
00029
00030 const int16_t C_TIMER_DEBOUNCE = 50;
00031 const int32_t C_TIMER_LONGPRESS = 1000 + C_TIMER_DEBOUNCE;
00032 const int32_t C_TIMER_LONGPRESS_LOOP = 500;
00033
00034 const bool button_pressed = LOW;
00035 const bool button_not_pressed = HIGH;
00036 //#include <MilliTimmer.h>
00042 int16_t ReadDirPad()
00043 {
00044     int16_t event = C_NONE_EVENT;
00045     int16_t cont_changes = 0;
00046
00047     // Flags debouncing.
00048     static bool up_debouncing = false;
00049     static bool down_debouncing = false;
00050     // static bool left_debouncing = false;
00051     // static bool right_debouncing = false;
00052     static bool center_debouncing = false;
00053     // Flag preEvent.
00054     static bool up_preEvent = false;
00055     static bool down_preEvent = false;
00056     //static bool left_preEvent = false;
00057     //static bool right_preEvent = false;
00058     static bool center_preEvent = false;
00059
00060     // Flags del modo rapido.
00061     static bool up_long_press_flag = false;
00062     static bool down_long_press_flag = false;
00063     //static bool left_long_press_flag = false;
00064     //static bool right_long_press_flag = false;
```

```

00065     static bool center_long_press_flag = false;
00066
00067     // Timers filtro antirebote.
00068     static MilliTimer timer_up_debounce;
00069     static MilliTimer timer_down_debounce;
00070     //static MilliTimer timer_right_debounce;
00071     //static MilliTimer timer_left_debounce;
00072     static MilliTimer timer_center_debounce;
00073
00074     // Timers Long press detect
00075     static MilliTimer timer_up_longpress;
00076     static MilliTimer timer_down_longpress;
00077     //static MilliTimer timer_right_longpress;
00078     //static MilliTimer timer_left_longpress;
00079     static MilliTimer timer_center_longpress;
00080
00081     // Estado actual del boton.
00082     static bool up_butt_state = button_not_pressed;
00083     static bool down_butt_state = button_not_pressed;
00084     //static bool left_butt_state = button_not_pressed;
00085     // static bool right_butt_state = button_not_pressed;
00086     static bool center_butt_state = button_not_pressed;
00087
00088     // Estado anterior del boton.
00089     static bool up_butt_last_state = button_not_pressed;
00090     static bool down_butt_last_state = button_not_pressed;
00091     //static bool left_butt_last_state = button_not_pressed;
00092     //static bool right_butt_last_state = button_not_pressed;
00093     static bool center_butt_last_state = button_not_pressed;
00094
00095     if (up_debouncing == true)
00096     {
00097         if (timer_up_debounce.poll())
00098         {
00099             up_debouncing = false;
00100         }
00101     }
00102     else if (up_debouncing == false)
00103     {
00104         up_butt_state = digitalRead(C_PIN_BUTT_UP);
00105         if (up_butt_state == button_pressed)
00106         {
00107             if (up_butt_last_state == button_not_pressed)
00108             {
00109                 timer_up_debounce.set(C_TIMER_DEBOUNCE);
00110                 up_debouncing = true;
00111                 timer_up_longpress.set(C_TIMER_LONGPRESS);
00112             }
00113             else
00114             {
00115                 if (timer_up_longpress.poll())
00116                 {
00117                     up_long_press_flag = true;
00118                     event = C_LP_UP;
00119                     timer_up_longpress.set(C_TIMER_LONGPRESS_LOOP);
00120                 }
00121             }
00122         }
00123     }
00124     else
00125     {
00126         if (up_butt_last_state == button_pressed)
00127         {
00128             timer_up_debounce.set(C_TIMER_DEBOUNCE);
00129             up_debouncing = true;
00130             up_preEvent = true;
00131         }
00132         else if (up_preEvent == true)
00133         {
00134             up_preEvent = false;
00135             if (up_long_press_flag == false)
00136             {
00137                 event = C_CLICK_UP;
00138             }
00139             else
00140             {
00141                 up_long_press_flag = false;
00142             }
00143         }
00144     }
00145     up_butt_last_state = up_butt_state;
00146 }
00147
00148 if (down_debouncing == true)
00149 {
00150     if (timer_down_debounce.poll())
00151     {
00152         down_debouncing = false;
00153     }

```

```

00159     }
00160     else if (down_debouncing == false)
00161     {
00162         down_butt_state = digitalRead(C_PIN_BUTT_DOWN);
00163         if (down_butt_state == button_pressed)
00164         {
00165             if (down_butt_last_state == button_not_pressed)
00166             {
00167                 timer_down_debounce.set(C_TIMER_DEBOUNCE);
00168                 down_debouncing = true;
00169                 timer_down_longpress.set(C_TIMER_LONGPRESS);
00170             }
00171         }
00172         else
00173         {
00174             if (timer_down_longpress.poll())
00175             {
00176                 down_long_press_flag = true;
00177                 event = C_LP_DOWN;
00178                 timer_down_longpress.set(C_TIMER_LONGPRESS_LOOP);
00179             }
00180         }
00181     }
00182     else
00183     {
00184         if (down_butt_last_state == button_pressed)
00185         {
00186             timer_down_debounce.set(C_TIMER_DEBOUNCE);
00187             down_debouncing = true;
00188             down_preEvent = true;
00189         }
00190         else if (down_preEvent == true)
00191         {
00192             down_preEvent = false;
00193             if (down_long_press_flag == false)
00194             {
00195                 event = C_CLICK_DOWN;
00196             }
00197             else
00198             {
00199                 down_long_press_flag = false;
00200             }
00201         }
00202         down_butt_last_state = down_butt_state;
00203     }
00204     /*
00317     if (center_debouncing == true)
00318     {
00319         if (timer_center_debounce.poll())
00320         {
00321             center_debouncing = false;
00322         }
00323     }
00324     else if (center_debouncing == false)
00325     {
00326         center_butt_state = digitalRead(C_PIN_BUTT_CENTER);
00327         if (center_butt_state == button_pressed)
00328         {
00329             if (center_butt_last_state == button_not_pressed)
00330             {
00331                 timer_center_debounce.set(C_TIMER_DEBOUNCE);
00332                 center_debouncing = true;
00333                 timer_center_longpress.set(C_TIMER_LONGPRESS);
00334             }
00335         }
00336         else
00337         {
00338             if (timer_center_longpress.poll())
00339             {
00340                 center_long_press_flag = true;
00341                 event = C_LP_CENTER;
00342                 timer_center_longpress.set(C_TIMER_LONGPRESS_LOOP);
00343             }
00344         }
00345     }
00346     else
00347     {
00348         if (center_butt_last_state == button_pressed)
00349         {
00350             timer_center_debounce.set(C_TIMER_DEBOUNCE);
00351             center_debouncing = true;
00352             center_preEvent = true;
00353         }
00354         else if (center_preEvent == true)
00355         {
00356             center_preEvent = false;
00357             if (center_long_press_flag == false)
00358             {
00359

```

```
00358         event = C_CLICK_CENTER;
00359     }
00360     else
00361     {
00362         center_long_press_flag = false;
00363     }
00364 }
00365 }
00366 center_butts_last_state = center_butts_state;
00367
00368
00369 return event;
00370 }
```

## 15.17 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0001-BUTTONS/examples/TEST\_0/TEST\_0.ino File Reference

Este test comprobata si se detectan lo eventos de del DPAD.

```
#include <MilliTimmer.h>
#include <Dpad.h>
```

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)

#### 15.17.1 Detailed Description

Este test comprobata si se detectan lo eventos de del DPAD.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

4

##### Date

2021-3-16

##### Copyright

Copyright (c) 2020

Definition in file [TEST\\_0.ino](#).

#### 15.17.2 Function Documentation

##### 15.17.2.1 [loop\(\)](#)

```
void loop ( )
```

Definition at line [126](#) of file [TEST\\_0.ino](#).

### 15.17.2.2 setup()

```
void setup ( )
```

Definition at line 14 of file [TEST\\_0.ino](#).

## 15.18 TEST\_0.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <MilliTTimer.h>
00012 #include <Dpad.h>
00013
00014 void setup()
00015 {
00016     Serial.begin(9600);
00017     while (!Serial)
00018     {
00019         ;
00020     }
00021
00022     int32_t computing_time = 0;
00023     int16_t dpad_event;
00024     int time_0 = 0;
00025     int time_1 = 0;
00026     //pinMode(3, OUTPUT);
00027     pinMode(C_PIN_BUTT_CENTER, INPUT_PULLUP);
00028     pinMode(C_PIN_BUTT_UP, INPUT_PULLUP);
00029     pinMode(C_PIN_BUTT_DOWN, INPUT_PULLUP);
00030     //pinMode(C_PIN_BUTT_RIGHT, INPUT_PULLUP);
00031     //pinMode(C_PIN_BUTT_LEFT, INPUT_PULLUP);
00032
00033     // digitalWrite(3, LOW);
00034     Serial.println("Test 0.Presione los botones del Dpad. En casa activacion se mostrara el evento detectado y el tiempo de computo de la funcion.");
00035
00036     while (1)
00037     {
00038         time_0 = micros();
00039         dpad_event = ReadDirPad();
00040         time_1 = micros();
00041         computing_time = time_1 - time_0;
00042         //digitalWrite(3, LOW);
00043         if (dpad_event == C_CLICK_CENTER)
00044         {
00045             Serial.print("CENTER CLICK Detected.");
00046             Serial.print("Computing Time:");
00047             Serial.print(computing_time);
00048             Serial.println("us.");
00049         }
00050         // else if (dpad_event == C_CLICK_RIGHT)
00051         // {
00052         //     Serial.print("RIGHT CLICK Detected.");
00053         //     Serial.print("Computing Time:");
00054         //     Serial.print(computing_time);
00055         //     Serial.println("us.");
00056         // }
00057         // else if (dpad_event == C_CLICK_LEFT)
00058         // {
00059         //     Serial.print("LEFT CLICK Detected.");
00060         //     Serial.print("Computing Time:");
00061         //     Serial.print(computing_time);
00062         //     Serial.println("us.");
00063         // }
00064         else if (dpad_event == C_CLICK_UP)
00065         {
00066             //digitalWrite(3, HIGH);
00067             Serial.print("UP CLICK Detected.");
00068             Serial.print("Computing Time:");
00069             Serial.print(computing_time);
00070             Serial.println("us.");
00071         }
00072         else if (dpad_event == C_CLICK_DOWN)
00073         {
00074             Serial.print("DOWN CLICK Detected.");
00075             Serial.print("Computing Time:");
00076             Serial.print(computing_time);
00077             Serial.println("us.");
00078         }
00079         else if (dpad_event == C_LP_CENTER)
00080         {
00081             Serial.print("CENTER LONG PRESS Detected.");
00082             Serial.print("Computing Time:");
00083             Serial.print(computing_time);
```

```
00084         Serial.println("us.");
00085     }
00086     // else if (dpad_event == C_LP_RIGHT)
00087     //
00088     //    Serial.print("RIGHT LONG PRESS Detected.");
00089     //    Serial.print("Computing Time:");
00090     //    Serial.print(computing_time);
00091     //    Serial.println("us.");
00092     //
00093     // else if (dpad_event == C_LP_LEFT)
00094     //
00095     //    Serial.print("LEFT LONG PRESS Detected.");
00096     //    Serial.print("Computing Time:");
00097     //    Serial.print(computing_time);
00098     //    Serial.println("us.");
00099     //
00100    else if (dpad_event == C_LP_UP)
00101    {
00102        //digitalWrite(3, HIGH);
00103        Serial.print("UP LONG PRESS Detected.");
00104        Serial.print("Computing Time:");
00105        Serial.print(computing_time);
00106        Serial.println("us.");
00107    }
00108    else if (dpad_event == C_LP_DOWN)
00109    {
00110        Serial.print("DOWN LONG PRESS Detected.");
00111        Serial.print("Computing Time:");
00112        Serial.print(computing_time);
00113        Serial.println("us.");
00114    }
00115    else if (dpad_event == C_NONE_EVENT)
00116    {
00117        //Serial.println("NONE");
00118    }
00119    else
00120    {
00121        Serial.println("Error with the Dpad event Detection.");
00122    }
00123 }
00124 }
00125
00126 void loop()
00127 {
00128 }
```

## 15.19 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_0/TEST\_0.ino File Reference

Testeo de la funcion DisplayTest.

```
#include <display.h>
```

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)

### Variables

- Adafruit\_IS131FL3731\_Wing ledmatrix = Adafruit\_IS131FL3731\_Wing()

#### 15.19.1 Detailed Description

Testeo de la funcion DisplayTest.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

**Version**

2

**Date**

2020-12-03

**Copyright**

Copyright (c) 2020

Definition in file [TEST\\_0.ino](#).

## 15.19.2 Function Documentation

### 15.19.2.1 loop()

```
void loop ()
```

Definition at line 99 of file [TEST\\_0.ino](#).

### 15.19.2.2 setup()

```
void setup ()
```

Definition at line 15 of file [TEST\\_0.ino](#).

## 15.19.3 Variable Documentation

### 15.19.3.1 ledmatrix

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()
```

Definition at line 13 of file [TEST\\_0.ino](#).

## 15.20 TEST\_0.ino

[Go to the documentation of this file.](#)

```
00001
00012 #include <display.h>
00013 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00014
00015 void setup()
00016 {
00017     Serial.begin(9600);
00018     while (!Serial)
00019     {
00020         ;
00021     }
00022
00023     ledmatrix.begin();
00024     int32_t t0 = 0;
00025     int32_t t1 = 0;
00026     bool ok = false;
00027
00028     t0 = micros();
00029     bool reset = true;
00030     while (DisplayText("Hola, que tal?", ledmatrix, reset, C_MODE_HIGH_BRIGHT) == C_DISPLAY_ST_BUSSY)
00031     {
00032         reset = false;
00033         delay(10);
00034     }
00035     t1 = micros();
00036     Serial.print("Tiempo de ejecucion DisplayText:");
00037     Serial.println(t1 - t0);
00038     delay(500);
```

```
00039     while (!ok)
00040     {
00041         t0 = micros();
00042         if (DisplayText("Bien y tu?", ledmatrix) == false)
00043         {
00044             ok = false;
00045             t1 = micros();
00046             Serial.print("Tiempo de ejecucion Bussy DisplayText:");
00047             Serial.println(t1 - t0);
00048         }
00049     else
00050     {
00051         ok = true;
00052         t1 = micros();
00053         Serial.print("Tiempo de ejecucion Not bussy DisplayText:");
00054         Serial.println(t1 - t0);
00055     }
00056 }
00057 delay(500);
00058 ok = false;
00059 while (!ok)
00060 {
00061     t0 = micros();
00062     if (DisplayText("Muy bien, Hasta luego", ledmatrix) == false)
00063     {
00064         ok = false;
00065         t1 = micros();
00066         Serial.print("Tiempo de ejecucion Bussy DisplayText:");
00067         Serial.println(t1 - t0);
00068     }
00069     else
00070     {
00071         ok = true;
00072         t1 = micros();
00073         Serial.print("Tiempo de ejecucion Not bussy DisplayText:");
00074         Serial.println(t1 - t0);
00075     }
00076 }
00077 delay(500);
00078 ok = false;
00079 while (!ok)
00080 {
00081     t0 = micros();
00082     if (DisplayText("Adios! ^~ PD: Hello! (Again)", ledmatrix) == false)
00083     {
00084         ok = false;
00085         t1 = micros();
00086         Serial.print("Tiempo de ejecucion Bussy DisplayText:");
00087         Serial.println(t1 - t0);
00088     }
00089     else
00090     {
00091         ok = true;
00092         t1 = micros();
00093         Serial.print("Tiempo de ejecucion Not bussy DisplayText:");
00094         Serial.println(t1 - t0);
00095     }
00096 }
00097 }
00098 void loop()
00100 {
00101 }
```

## 15.21 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0003-BUZZER/examples/TEST\_0/TEST\_0.ino File Reference

```
#include <MilliTimer.h>
#include <Arduino.h>
#include <Buzzer.h>
```

### Functions

- void **setup ()**

- void `loop()`

## Variables

- `uint32_t t0 = 0`
- `uint32_t t1 = 0`
- `uint32_t i = 0`

### 15.21.1 Detailed Description

Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

Version

1

Date

2021-02-09

Copyright

Copyright (c) 2021

Definition in file [TEST\\_0.ino](#).

### 15.21.2 Function Documentation

#### 15.21.2.1 `loop()`

`void loop ()`

Definition at line [27](#) of file [TEST\\_0.ino](#).

#### 15.21.2.2 `setup()`

`void setup ()`

Definition at line [17](#) of file [TEST\\_0.ino](#).

### 15.21.3 Variable Documentation

#### 15.21.3.1 `i`

`uint32_t i = 0`

Definition at line [15](#) of file [TEST\\_0.ino](#).

#### 15.21.3.2 `t0`

`uint32_t t0 = 0`

Definition at line [15](#) of file [TEST\\_0.ino](#).

### 15.21.3.3 t1

```
uint32_t t1 = 0
Definition at line 15 of file TEST_0.ino.
```

## 15.22 TEST\_0.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <MilliTimer.h>
00012 #include <Arduino.h>
00013 #include <Buzzer.h>
00014
00015 uint32_t t0 = 0, t1 = 0, i = 0;
00016 ;
00017 void setup()
00018 {
00019     Serial.begin(9600);
00020     // while (!Serial)
00021     // {
00022     //     /* code */
00023     // }
00024     InitBuzzer(C_MODE_DEFAULT);
00025 }
00026
00027 void loop()
00028 {
00029     // t0 = micros();
00030     // playSound(C_SOUND_START);
00031     // t1 = micros();
00032     // Serial.println(t1 - t0);
00033     // delay(2000);
00034     // t0 = micros();
00035     // playSound(C_SOUND_END);
00036     // t1 = micros();
00037     // Serial.println(t1 - t0);
00038     // delay(2000);
00039     // t0 = micros();
00040     // playSound(C_SOUND_UP);
00041     // t1 = micros();
00042     // Serial.println(t1 - t0);
00043     // delay(2000);
00044     // t0 = micros();
00045     // playSound(C_SOUND_DOWN);
00046     // t1 = micros();
00047     // Serial.println(t1 - t0);
00048     // delay(2000);
00049
00050     t0 = micros();
00051     if (playSound(i) == C_BUZZER_NOT_BUSSY)
00052     {
00053         delay(2000);
00054         i++;
00055         if (i == 11)
00056         {
00057             i = 0;
00058         }
00059     }
00060
00061     t1 = micros();
00062     //Serial.println(t1 - t0);
00063 }
```

## 15.23 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0004-DCDC/exmples/TEST\_0/TEST\_0.ino File Reference

```
#include <DCDC.h>
```

### Functions

- void [setup \(\)](#)

- void `loop ()`

## Variables

- const int16\_t `C_PIN_I_IN` = 15
- const int16\_t `C_PIN_V_OUT` = 16
- const uint16\_t `C_PIN_EN_DCDC` = 2
- `dcdc_controller DCDC (C_PIN_EN_DCDC)`
- int16\_t `v_sense` = 0
- int16\_t `i_sense` = 0
- float `v_diff` = 0

### 15.23.1 Detailed Description

Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

Version

2

Date

2020-12-15

Copyright

Copyright (c) 2020

Definition in file [TEST\\_0.ino](#).

### 15.23.2 Function Documentation

#### 15.23.2.1 `loop()`

`void loop ()`

Definition at line [59](#) of file [TEST\\_0.ino](#).

#### 15.23.2.2 `setup()`

`void setup ()`

Definition at line [24](#) of file [TEST\\_0.ino](#).

### 15.23.3 Variable Documentation

#### 15.23.3.1 `C_PIN_EN_DCDC`

`const uint16_t C_PIN_EN_DCDC = 2`

Definition at line [16](#) of file [TEST\\_0.ino](#).

#### 15.23.3.2 `C_PIN_I_IN`

`const int16_t C_PIN_I_IN = 15`

Definition at line [14](#) of file [TEST\\_0.ino](#).

### 15.23.3.3 C\_PIN\_V\_OUT

```
const int16_t C_PIN_V_OUT = 16
Definition at line 15 of file TEST_0.ino.
```

### 15.23.3.4 DCDC

```
dcdc_controller DCDC(C_PIN_EN_DCDC) (
    C_PIN_EN_DCDC )
```

### 15.23.3.5 i\_sense

```
int16_t i_sense = 0
Definition at line 21 of file TEST_0.ino.
```

### 15.23.3.6 v\_diff

```
float v_diff = 0
Definition at line 22 of file TEST_0.ino.
```

### 15.23.3.7 v\_sense

```
int16_t v_sense = 0
Definition at line 20 of file TEST_0.ino.
```

## 15.24 TEST\_0.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <DCDC.h>
00012
00013 // Pines de sensado del Makinator.
00014 const int16_t C_PIN_I_IN = 15;
00015 const int16_t C_PIN_V_OUT = 16;
00016 const uint16_t C_PIN_EN_DCDC = 2;
00017
00018 dcdc_controller DCDC(C_PIN_EN_DCDC);
00019
00020 int16_t v_sense = 0;
00021 int16_t i_sense = 0;
00022 float v_diff = 0;
00023
00024 void setup()
00025 {
00026     Wire.begin();
00027     Serial.begin(9600);
00028
00029     String toString;
00030     Serial.println("Theory Volt \t Real Volt \t CURRENT \t Increment");
00031
00032     for (int i = 40; i <= 160; i++)
00033     {
00034         DCDC.SetVoltage(i, C_NON_BOOST_MODE);
00035         v_sense = 0;
00036         i_sense = 0;
00037         delay(1000);
00038         for (int j = 0; j < 8; j++)
00039         {
00040             v_sense += analogRead(C_PIN_V_OUT);
00041             i_sense += analogRead(C_PIN_I_IN);
00042             delay(100);
00043         }
00044         v_sense = (int)((v_sense / 8) * 3300 * 4.84 / 4096);
00045         i_sense = (int)((i_sense / 8) * 3300 / 4096);
00046         Serial.print((float)(i) / 10);
00047         Serial.print("\t\t");
00048         Serial.print((float)(v_sense) / 1000);
00049         Serial.print("\t\t");
```

```

00050     Serial.print((float)(i_sense) / 1000);
00051     v_diff = ((float)(i) / 10) - ((float)(v_sense) / 1000);
00052     Serial.print("\t\t");
00053     Serial.println(v_diff);
00054 }
00055
00056     delay(100);
00057 }
00058
00059 void loop()
00060 {
00061 }
```

## 15.25 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0005-SENSING/examples/TEST\_0/TEST\_0.ino File Reference

```
#include <HealthMonitor.h>
#include <MilliTTimer.h>
```

### Macros

- #define DAC\_PIN A0

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)

### Variables

- [HealthMonitor sensor\\_1](#) (3000, [C\\_RISE\\_COUNT](#), [C\\_FALL\\_COUNT](#), [C\\_COUNTER\\_LIMIT](#))
- [uint16\\_t sample](#) = 0
- [uint16\\_t count](#) = 0
- [uint16\\_t dac\\_output](#)
- [MilliTTimer trigger\\_timer](#)

#### 15.25.1 Detailed Description

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

1

##### Date

2020-12-10

##### Copyright

Copyright (c) 2020

Definition in file [TEST\\_0.ino](#).

#### 15.25.2 Macro Definition Documentation

### 15.25.2.1 DAC\_PIN

```
#define DAC_PIN A0
```

Definition at line 15 of file [TEST\\_0.ino](#).

## 15.25.3 Function Documentation

### 15.25.3.1 loop()

```
void loop ()
```

Definition at line 36 of file [TEST\\_0.ino](#).

### 15.25.3.2 setup()

```
void setup ()
```

Definition at line 25 of file [TEST\\_0.ino](#).

## 15.25.4 Variable Documentation

### 15.25.4.1 count

```
uint16_t count = 0
```

Definition at line 20 of file [TEST\\_0.ino](#).

### 15.25.4.2 dac\_output

```
uint16_t dac_output
```

Definition at line 21 of file [TEST\\_0.ino](#).

### 15.25.4.3 sample

```
uint16_t sample = 0
```

Definition at line 19 of file [TEST\\_0.ino](#).

### 15.25.4.4 sensor\_1

```
HealthMonitor sensor_1(3000, C_RISE_COUNT, C_FALL_COUNT, C_COUNTER_LIMIT) (
```

```
    3000,
```

```
    C_RISE_COUNT,
```

```
    C_FALL_COUNT,
```

```
    C_COUNTER_LIMIT)
```

### 15.25.4.5 trigger\_timer

```
MilliTimer trigger_timer
```

Definition at line 23 of file [TEST\\_0.ino](#).

## 15.26 TEST\_0.ino

[Go to the documentation of this file.](#)

```

00001
00012 #include <HealthMonitor.h>
00013 #include <MilliTimer.h>
00014
00015 #define DAC_PIN A0
00016
00017 HealthMonitor sensor_1(3000, C_RISE_COUNT, C_FALL_COUNT, C_COUNTER_LIMIT);
00018
00019 uint16_t sample = 0;
00020 uint16_t count = 0;
00021 uint16_t dac_output;
00022
00023 MilliTimer trigger_timer;
00024
00025 void setup()
00026 {
00027     pinMode(A2, OUTPUT);
00028     Serial.begin(9600);
00029     while (!Serial)
00030     ;
00031     analogWriteResolution(10); // Set analog out resolution to max, 10-bits
00032
00033     sensor_1.setCounter(0);
00034 }
00035
00036 void loop()
00037 {
00038     sample = sensor_1.getSample(A1);
00039     Serial.print("Sample:");
00040     Serial.print(sample);
00041     if (sensor_1.check(sample) == true)
00042     {
00043         count = sensor_1.getCounter();
00044         Serial.print("    Counter:");
00045         Serial.print(count);
00046         dac_output = (int)(count * 1023.0 / 100.0);
00047         analogWrite(DAC_PIN, dac_output); // 0= 0V, 1023=3.3V, 512=1.65V, etc.
00048         Serial.println("TRIGGERED");
00049         trigger_timer.set(10);
00050         sensor_1.setCounter(0);
00051         digitalWrite(A2, HIGH);
00052     }
00053     else
00054     {
00055         count = sensor_1.getCounter();
00056         Serial.print("    Counter:");
00057         Serial.println(count);
00058         dac_output = (int)(count * 1023.0 / 100.0);
00059         analogWrite(DAC_PIN, dac_output); // 0= 0V, 1023=3.3V, 512=1.65V, etc.
00060     }
00061
00062     if (trigger_timer.poll() != 0)
00063     {
00064         digitalWrite(A2, LOW);
00065     }
00066
00067     delay(10);
00068 }
```

## 15.27 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0006-POWERBAR/examples/TEST\_0/← TEST\_0.ino File Reference

Test para probar la actualización de la barra de potencia. Oscila los valores entre el máximo y el mínimo de potencia que puede mostrar la barra.

```
#include "display.h"
#include "power_bar.h"
```

### Functions

- `void setup ()`

- void `loop()`

## Variables

- `Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()`

### 15.27.1 Detailed Description

Test para probar la actualizacion de la barra de potencia. Oscila los valores entre el maximo y el minimo de potencia que puede mostrar la barra.

#### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

#### Version

0.1

#### Date

2021-03-18

#### Copyright

Copyright (c) 2021

Definition in file [TEST\\_0.ino](#).

### 15.27.2 Function Documentation

#### 15.27.2.1 `loop()`

`void loop ()`

Definition at line 29 of file [TEST\\_0.ino](#).

#### 15.27.2.2 `setup()`

`void setup ()`

Definition at line 16 of file [TEST\\_0.ino](#).

### 15.27.3 Variable Documentation

#### 15.27.3.1 `ledmatrix`

`Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()`

Definition at line 14 of file [TEST\\_0.ino](#).

## 15.28 TEST\_0.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include "display.h"
00012 #include "power_bar.h"
00013
00014 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00015
```

```

00016 void setup()
00017 {
00018     ledmatrix.begin();
00019     while (1)
00020     {
00021         for (int i = 0; i < MAX_POWER_DISPLAYED; i++)
00022         {
00023             UpdatePowerBar(i, ledmatrix, C_MODE_HIGH_BRIGHT);
00024             delay(1);
00025         }
00026     }
00027 }
00028
00029 void loop()
00030 {
00031 }

```

## 15.29 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0010-LOGGING/examples/TEST\_0/TEST\_0.ino File Reference

Test de las funciones de PrintHeader, PrintDiagnosticData y LogDiagnosticData.

```
#include <diagnostic.h>
#include <SEGGER_RTT.h>
```

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)

#### 15.29.1 Detailed Description

Test de las funciones de PrintHeader, PrintDiagnosticData y LogDiagnosticData.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

2

##### Date

2021-03-22

##### Copyright

Copyright (c) 2021

Definition in file [TEST\\_0.ino](#).

#### 15.29.2 Function Documentation

##### 15.29.2.1 [loop\(\)](#)

`void loop ()`

Definition at line 35 of file [TEST\\_0.ino](#).

### 15.29.2.2 setup()

```
void setup ( )
Definition at line 17 of file TEST_0.ino.
```

## 15.30 TEST\_0.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <diagnostic.h>
00012 extern "C"
00013 {
00014 #include <SEGGER_RTT.h>
00015 }
00016
00017 void setup()
00018 {
00019     Serial.begin(9600);
00020     while (!Serial)
00021     {
00022         /* code */
00023     }
00024     Init_diagnostic_elements();
00025
00026     for (int i = 0; i < 50; i++)
00027     {
00028         LogDiagnosticData(i * 3, i);
00029     }
00030     PrintStaticData();
00031     PrintDiagnosticData();
00032     delay(1000);
00033 }
00034
00035 void loop()
00036 {
00037 }
```

## 15.31 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/bitmaps.h File Reference

Conjunto de Bitmaps, funciones y constantes realcionadas con la el display Musotoku 15x7.

### Macros

- `#define DISPLAY_SIZE 6`
- `#define POWERBAR_LOCATION 1`
- `#define C_DISPLAY_WIDTH 15`
- `#define C_DISPLAY_HEIGHT 6`

### Functions

- `void Char2Bitmap (char character, uint8_t *bitmap, int16_t array_len)`

### Variables

- `uint8_t SPACEchar []`
- `uint8_t EXCLMARKchar []`
- `uint8_t QUOTMARKchar []`
- `uint8_t HASTAGchar []`
- `uint8_t DOLLARchar []`
- `uint8_t PERCENTchar []`
- `uint8_t ANPERSANchar []`
- `uint8_t APOSTROPHEchar []`
- `uint8_t OPENPARENTESISchar []`

- `uint8_t CLOSEPARENTESISchar []`
- `uint8_t ASTERISKchar []`
- `uint8_t PLUSchar []`
- `uint8_t COMMAchar []`
- `uint8_t HYPHENchar []`
- `uint8_t DOTchar []`
- `uint8_t SLASHchar []`
- `uint8_t ZEROchar []`
- `uint8_t ONEchar []`
- `uint8_t TWOchar []`
- `uint8_t THREEchar []`
- `uint8_t FOURchar []`
- `uint8_t FIVEchar []`
- `uint8_t SIXchar []`
- `uint8_t SEVENchar []`
- `uint8_t EIGHTchar []`
- `uint8_t NINEchar []`
- `uint8_t COLONchar []`
- `uint8_t SEMICOLONchar []`
- `uint8_t LESSTHANchar []`
- `uint8_t EQUALchar []`
- `uint8_t GREATERTHANchar []`
- `uint8_t QUESTIONchar []`
- `uint8_t ATchar []`
- `uint8_t Achar []`
- `uint8_t Bchar []`
- `uint8_t Cchar []`
- `uint8_t Dchar []`
- `uint8_t Echar []`
- `uint8_t Fchar []`
- `uint8_t Gchar []`
- `uint8_t Hchar []`
- `uint8_t Ichar []`
- `uint8_t Jchar []`
- `uint8_t Kchar []`
- `uint8_t Lchar []`
- `uint8_t Mchar []`
- `uint8_t Nchar []`
- `uint8_t Ochar []`
- `uint8_t Pchar []`
- `uint8_t Qchar []`
- `uint8_t Rchar []`
- `uint8_t Schar []`
- `uint8_t Tchar []`
- `uint8_t Uchar []`
- `uint8_t Vchar []`
- `uint8_t Wchar []`
- `uint8_t Xchar []`
- `uint8_t Ychar []`
- `uint8_t Zchar []`
- `uint8_t SQUAREBRAQUETchar []`
- `uint8_t BACKSLASHchar []`
- `uint8_t CLOSESQUAREBRAQUETchar []`
- `uint8_t CIRCUMFLEXchar []`
- `uint8_t UNDERSCOREchar []`

- `uint8_t ACCENTchar []`
- `uint8_t OPENCURLYBRAchar []`
- `uint8_t CLOSECURLYBRAchar []`
- `uint8_t VERTICALBARchar []`
- `uint8_t SWUNGchar []`
- `const uint16_t width_bitmap [] = {4, 3, 4, 6, 6, 5, 6, 2, 4, 4, 4, 4, 3, 4, 2, 4, 5, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 3, 4, 4, 4, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 6, 5, 4, 4, 6, 4, 5, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 2, 4, 5}`
- `uint8_t Frame_Batt_1 []`
- `uint8_t Frame_Batt_2 []`
- `uint8_t Frame_USB_1 []`
- `uint8_t Frame_USB_2 []`
- `uint8_t EYES_1 []`
- `uint8_t EYES_2 []`
- `uint8_t EYES2_1 []`
- `uint8_t EYES2_2 []`
- `uint8_t FACE_1 []`
- `uint8_t FACE_2 []`
- `uint8_t Frame_Batt_Low_1 []`
- `uint8_t Frame_Batt_Low_2 []`
- `uint8_t Frame_Error_1 []`
- `uint8_t Frame_Error_2 []`

### 15.31.1 Detailed Description

Conjunto de Bitmaps, funciones y constantes realcionadas con la el display Musotoku 15x7.

Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

Version

3

Date

2020-8-4

Copyright

Copyright (c) 2020

Definition in file [bitmaps.h](#).

### 15.31.2 Macro Definition Documentation

#### 15.31.2.1 C\_DISPLAY\_HEIGHT

```
#define C_DISPLAY_HEIGHT 6
```

Definition at line 14 of file [bitmaps.h](#).

#### 15.31.2.2 C\_DISPLAY\_WIDTH

```
#define C_DISPLAY_WIDTH 15
```

Definition at line 13 of file [bitmaps.h](#).

### 15.31.2.3 DISPLAY\_SIZE

```
#define DISPLAY_SIZE 6
```

Definition at line 11 of file [bitmaps.h](#).

### 15.31.2.4 POWERBAR\_LOCATION

```
#define POWERBAR_LOCATION 1
```

Definition at line 12 of file [bitmaps.h](#).

## 15.31.3 Function Documentation

### 15.31.3.1 Char2Bitmap()

```
void Char2Bitmap (
    char character,
    uint8_t * bitmap,
    int16_t array_len )
```

Definition at line 1345 of file [bitmaps.h](#).

## 15.31.4 Variable Documentation

### 15.31.4.1 ACCENTchar

```
uint8_t ACCENTchar[ ]
```

#### Initial value:

```
= {B10000000,  
 B01000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B00000000}
```

Definition at line 1310 of file [bitmaps.h](#).

### 15.31.4.2 Achar

```
uint8_t Achar[ ]
```

#### Initial value:

```
= {B01000000,  
 B10100000,  
 B11100000,  
 B10100000,  
 B10100000,  
 B10100000}
```

Definition at line 1093 of file [bitmaps.h](#).

### 15.31.4.3 ANPERSANchar

```
uint8_t ANPERSANchar[ ]
```

#### Initial value:

```
= {B01000000,  
 B10100000,  
 B01000000,  
 B10101000,  
 B10010000,  
 B01101000}
```

Definition at line 904 of file [bitmaps.h](#).

#### 15.31.4.4 APOSTROPHEchar

```
uint8_t APOSTROPHEchar[ ]
```

**Initial value:**

```
= {B10000000,  
 B10000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B00000000}
```

Definition at line 911 of file [bitmaps.h](#).

#### 15.31.4.5 ASTERISKchar

```
uint8_t ASTERISKchar[ ]
```

**Initial value:**

```
= {B01000000,  
 B11100000,  
 B01000000,  
 B10100000,  
 B00000000,  
 B00000000}
```

Definition at line 932 of file [bitmaps.h](#).

#### 15.31.4.6 ATchar

```
uint8_t ATchar[ ]
```

**Initial value:**

```
= {B01111000,  
 B10000100,  
 B10110100,  
 B10111000,  
 B10000100,  
 B01111000}
```

Definition at line 1086 of file [bitmaps.h](#).

#### 15.31.4.7 BACKSLASHchar

```
uint8_t BACKSLASHchar[ ]
```

**Initial value:**

```
= {B10000000,  
 B10000000,  
 B01000000,  
 B01000000,  
 B00100000,  
 B00100000}
```

Definition at line 1282 of file [bitmaps.h](#).

#### 15.31.4.8 Bchar

```
uint8_t Bchar[ ]
```

**Initial value:**

```
= {B11000000,  
 B10100000,  
 B11000000,  
 B10100000,  
 B10100000,  
 B11000000}
```

Definition at line 1100 of file [bitmaps.h](#).

#### 15.31.4.9 Cchar

```
uint8_t Cchar[ ]
```

**Initial value:**

```
=
{B01000000,
 B10100000,
 B10000000,
 B10000000,
 B10100000,
 B01000000}
```

Definition at line 1107 of file [bitmaps.h](#).

#### 15.31.4.10 CIRCUMFLEXchar

```
uint8_t CIRCUMFLEXchar[ ]
```

**Initial value:**

```
=
{B01000000,
 B10100000,
 B00000000,
 B00000000,
 B00000000,
 B00000000}
```

Definition at line 1296 of file [bitmaps.h](#).

#### 15.31.4.11 CLOSECURLYBRAchar

```
uint8_t CLOSECURLYBRAchar[ ]
```

**Initial value:**

```
=
{B11000000,
 B01000000,
 B01100000,
 B01000000,
 B01000000,
 B11000000}
```

Definition at line 1324 of file [bitmaps.h](#).

#### 15.31.4.12 CLOSEPARENTESISchar

```
uint8_t CLOSEPARENTESISchar[ ]
```

**Initial value:**

```
=
{B10000000,
 B01000000,
 B01000000,
 B01000000,
 B01000000,
 B10000000}
```

Definition at line 925 of file [bitmaps.h](#).

#### 15.31.4.13 CLOSESQUAREBRAQUETchar

```
uint8_t CLOSESQUAREBRAQUETchar[ ]
```

**Initial value:**

```
=
{B11100000,
 B00100000,
 B00100000,
 B00100000,
 B00100000,
 B11100000}
```

Definition at line 1289 of file [bitmaps.h](#).

#### 15.31.4.14 COLONchar

```
uint8_t COLONchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B11000000,  
 B11000000,  
 B00000000,  
 B11000000,  
 B11000000}
```

Definition at line 1044 of file [bitmaps.h](#).

#### 15.31.4.15 COMMAchar

```
uint8_t COMMAchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B01000000,  
 B10000000}
```

Definition at line 946 of file [bitmaps.h](#).

#### 15.31.4.16 Dchar

```
uint8_t Dchar[ ]
```

**Initial value:**

```
=  
{B11000000,  
 B10100000,  
 B10100000,  
 B10100000,  
 B10100000,  
 B11000000}
```

Definition at line 1114 of file [bitmaps.h](#).

#### 15.31.4.17 DOLLARchar

```
uint8_t DOLLARchar[ ]
```

**Initial value:**

```
=  
{B01111000,  
 B10100000,  
 B01110000,  
 B00101000,  
 B00101000,  
 B11110000}
```

Definition at line 890 of file [bitmaps.h](#).

#### 15.31.4.18 DOTchar

```
uint8_t DOTchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B10000000}
```

Definition at line 960 of file [bitmaps.h](#).

#### 15.31.4.19 Echar

```
uint8_t Echar[ ]
```

**Initial value:**

```
=
{B11100000,
B10000000,
B11000000,
B10000000,
B10000000,
B11100000}
```

Definition at line 1121 of file [bitmaps.h](#).

#### 15.31.4.20 EIGHTchar

```
uint8_t EIGHTchar[ ]
```

**Initial value:**

```
=
{B01100000,
B10010000,
B01100000,
B10010000,
B10010000,
B01100000}
```

Definition at line 1030 of file [bitmaps.h](#).

#### 15.31.4.21 EQUALchar

```
uint8_t EQUALchar[ ]
```

**Initial value:**

```
=
{B00000000,
B00000000,
B00000000,
B11100000,
B00000000,
B11100000}
```

Definition at line 1065 of file [bitmaps.h](#).

#### 15.31.4.22 EXCLMARKchar

```
uint8_t EXCLMARKchar[ ]
```

**Initial value:**

```
=
{B11000000,
B11000000,
B11000000,
B00000000,
B11000000,
B11000000}
```

Definition at line 869 of file [bitmaps.h](#).

#### 15.31.4.23 EYES2\_1

```
uint8_t EYES2_1[ ]
```

**Initial value:**

```
=
{B01111100,
B11111110,
B11110010,
B11110010,
B11111110,
B01111100}
```

Definition at line 1686 of file [bitmaps.h](#).

#### 15.31.4.24 EYES2\_2

```
uint8_t EYES2_2[ ]
```

**Initial value:**

```
= {B01111100,  
 B11111110,  
 B10011110,  
 B10011110,  
 B11111110,  
 B01111100}
```

Definition at line 1694 of file [bitmaps.h](#).

#### 15.31.4.25 EYES\_1

```
uint8_t EYES_1[ ]
```

**Initial value:**

```
= {B01111100,  
 B10000010,  
 B10011010,  
 B10011010,  
 B10000010,  
 B01111100}
```

Definition at line 1670 of file [bitmaps.h](#).

#### 15.31.4.26 EYES\_2

```
uint8_t EYES_2[ ]
```

**Initial value:**

```
= {B01111100,  
 B10000010,  
 B10110010,  
 B10110010,  
 B10000010,  
 B01111100}
```

Definition at line 1678 of file [bitmaps.h](#).

#### 15.31.4.27 FACE\_1

```
uint8_t FACE_1[ ]
```

**Initial value:**

```
= {B11100000,  
 B00000001,  
 B01000001,  
 B00000001,  
 B00000100,  
 B00000011}
```

Definition at line 1703 of file [bitmaps.h](#).

#### 15.31.4.28 FACE\_2

```
uint8_t FACE_2[ ]
```

**Initial value:**

```
= {B00001110,  
 B00000000,  
 B00000100,  
 B00000000,  
 B01000000,  
 B10000000}
```

Definition at line 1711 of file [bitmaps.h](#).

#### 15.31.4.29 Fchar

```
uint8_t Fchar[ ]
```

**Initial value:**

```
=  
{B11100000,  
 B10000000,  
 B11000000,  
 B10000000,  
 B10000000,  
 B10000000}
```

Definition at line 1128 of file [bitmaps.h](#).

#### 15.31.4.30 FIVEchar

```
uint8_t FIVEchar[ ]
```

**Initial value:**

```
=  
{B11110000,  
 B10000000,  
 B11100000,  
 B00010000,  
 B10010000,  
 B01100000}
```

Definition at line 1009 of file [bitmaps.h](#).

#### 15.31.4.31 FOURchar

```
uint8_t FOURchar[ ]
```

**Initial value:**

```
=  
{B10000000,  
 B10000000,  
 B10010000,  
 B11100000,  
 B00010000,  
 B00010000}
```

Definition at line 1002 of file [bitmaps.h](#).

#### 15.31.4.32 Frame\_Batt\_1

```
uint8_t Frame_Batt_1[ ]
```

**Initial value:**

```
=  
{B01111111,  
 B01000000,  
 B01000000,  
 B01000000,  
 B01000000,  
 B01111111}
```

Definition at line 1637 of file [bitmaps.h](#).

#### 15.31.4.33 Frame\_Batt\_2

```
uint8_t Frame_Batt_2[ ]
```

**Initial value:**

```
=  
{B11111000,  
 B00001000,  
 B00001100,  
 B00001100,  
 B00001000,  
 B11111000}
```

Definition at line 1645 of file [bitmaps.h](#).

#### 15.31.4.34 Frame\_Batt\_Low\_1

```
uint8_t Frame_Batt_Low_1[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B00111111,  
 B00100001,  
 B00100010,  
 B00111111,  
 B00000100}
```

Definition at line 1719 of file [bitmaps.h](#).

#### 15.31.4.35 Frame\_Batt\_Low\_2

```
uint8_t Frame_Batt_Low_2[ ]
```

**Initial value:**

```
=  
{B10000000,  
 B11110000,  
 B00011000,  
 B00011000,  
 B11110000,  
 B00000000}
```

Definition at line 1727 of file [bitmaps.h](#).

#### 15.31.4.36 Frame\_Error\_1

```
uint8_t Frame_Error_1[ ]
```

**Initial value:**

```
=  
{B00001000,  
 B00010001,  
 B00010001,  
 B00010000,  
 B00010001,  
 B00001000}
```

Definition at line 1735 of file [bitmaps.h](#).

#### 15.31.4.37 Frame\_Error\_2

```
uint8_t Frame_Error_2[ ]
```

**Initial value:**

```
=  
{B00100000,  
 B00010000,  
 B00010000,  
 B00010000,  
 B00010000,  
 B00100000}
```

Definition at line 1743 of file [bitmaps.h](#).

#### 15.31.4.38 Frame\_USB\_1

```
uint8_t Frame_USB_1[ ]
```

**Initial value:**

```
=  
{B00000111,  
 B00001000,  
 B11111000,  
 B00001000,  
 B00000111,  
 B00000000}
```

Definition at line 1653 of file [bitmaps.h](#).

#### 15.31.4.39 Frame\_USB\_2

```
uint8_t Frame_USB_2[ ]
```

**Initial value:**

```
=
{B11100000,
 B00111000,
 B00101000,
 B00111000,
 B11100000,
 B00000000}
```

Definition at line 1661 of file [bitmaps.h](#).

#### 15.31.4.40 Gchar

```
uint8_t Gchar[ ]
```

**Initial value:**

```
=
{B01000000,
 B10100000,
 B10000000,
 B10100000,
 B10100000,
 B01000000}
```

Definition at line 1135 of file [bitmaps.h](#).

#### 15.31.4.41 GREATERTHANchar

```
uint8_t GREATERTHANchar[ ]
```

**Initial value:**

```
=
{B00000000,
 B10000000,
 B01000000,
 B00100000,
 B01000000,
 B10000000}
```

Definition at line 1072 of file [bitmaps.h](#).

#### 15.31.4.42 HASTAGchar

```
uint8_t HASTAGchar[ ]
```

**Initial value:**

```
=
{B01001000,
 B11111100,
 B01001000,
 B01001000,
 B11111100,
 B01001000}
```

Definition at line 883 of file [bitmaps.h](#).

#### 15.31.4.43 Hchar

```
uint8_t Hchar[ ]
```

**Initial value:**

```
=
{B10100000,
 B10100000,
 B11100000,
 B10100000,
 B10100000,
 B10100000}
```

Definition at line 1142 of file [bitmaps.h](#).

#### 15.31.4.44 HYPHENchar

```
uint8_t HYPHENchar[]
```

**Initial value:**

```
=  
{B00000000,  
 B00000000,  
 B00000000,  
 B11100000,  
 B00000000,  
 B00000000}
```

Definition at line 953 of file [bitmaps.h](#).

#### 15.31.4.45 Ichar

```
uint8_t Ichar[]
```

**Initial value:**

```
=  
{B11100000,  
 B01000000,  
 B01000000,  
 B01000000,  
 B01000000,  
 B11100000}
```

Definition at line 1149 of file [bitmaps.h](#).

#### 15.31.4.46 Jchar

```
uint8_t Jchar[]
```

**Initial value:**

```
=  
{B11100000,  
 B00100000,  
 B00100000,  
 B00100000,  
 B10100000,  
 B11100000}
```

Definition at line 1156 of file [bitmaps.h](#).

#### 15.31.4.47 Kchar

```
uint8_t Kchar[]
```

**Initial value:**

```
=  
{B10000000,  
 B10100000,  
 B11000000,  
 B10000000,  
 B11000000,  
 B10100000}
```

Definition at line 1163 of file [bitmaps.h](#).

#### 15.31.4.48 Lchar

```
uint8_t Lchar[]
```

**Initial value:**

```
=  
{B10000000,  
 B10000000,  
 B10000000,  
 B10000000,  
 B10000000,  
 B11000000}
```

Definition at line 1170 of file [bitmaps.h](#).

#### 15.31.4.49 LESSTHANchar

```
uint8_t LESSTHANchar[ ]
```

**Initial value:**

```
= {B00000000,
  B00100000,
  B01000000,
  B10000000,
  B01000000,
  B00100000}
```

Definition at line 1058 of file [bitmaps.h](#).

#### 15.31.4.50 Mchar

```
uint8_t Mchar[ ]
```

**Initial value:**

```
= {B10001000,
  B11011000,
  B10101000,
  B10001000,
  B10001000,
  B10001000}
```

Definition at line 1177 of file [bitmaps.h](#).

#### 15.31.4.51 Nchar

```
uint8_t Nchar[ ]
```

**Initial value:**

```
= {B10010000,
  B11010000,
  B10110000,
  B10010000,
  B10010000,
  B10010000}
```

Definition at line 1184 of file [bitmaps.h](#).

#### 15.31.4.52 NINEchar

```
uint8_t NINEchar[ ]
```

**Initial value:**

```
= {B01100000,
  B10010000,
  B01110000,
  B00010000,
  B10010000,
  B01100000}
```

Definition at line 1037 of file [bitmaps.h](#).

#### 15.31.4.53 Ochar

```
uint8_t Ochar[ ]
```

**Initial value:**

```
= {B01000000,
  B10100000,
  B10100000,
  B10100000,
  B10100000,
  B01000000}
```

Definition at line 1191 of file [bitmaps.h](#).

#### 15.31.4.54 ONEchar

```
uint8_t ONEchar[ ]
```

**Initial value:**

```
=  
{B01000000,  
 B11000000,  
 B01000000,  
 B01000000,  
 B01000000,  
 B01000000}
```

Definition at line 981 of file [bitmaps.h](#).

#### 15.31.4.55 OPENCURLYBRAchar

```
uint8_t OPENCURLYBRAchar[ ]
```

**Initial value:**

```
=  
{B01100000,  
 B01000000,  
 B11000000,  
 B01000000,  
 B01000000,  
 B01100000}
```

Definition at line 1317 of file [bitmaps.h](#).

#### 15.31.4.56 OPENPARENTESISchar

```
uint8_t OPENPARENTESISchar[ ]
```

**Initial value:**

```
=  
{B01000000,  
 B10000000,  
 B10000000,  
 B10000000,  
 B10000000,  
 B01000000}
```

Definition at line 918 of file [bitmaps.h](#).

#### 15.31.4.57 Pchar

```
uint8_t Pchar[ ]
```

**Initial value:**

```
=  
{B11000000,  
 B10100000,  
 B10100000,  
 B11000000,  
 B10000000,  
 B10000000}
```

Definition at line 1198 of file [bitmaps.h](#).

#### 15.31.4.58 PERCENTchar

```
uint8_t PERCENTchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B00000000,  
 B10010000,  
 B00100000,  
 B01000000,  
 B10010000}
```

Definition at line 897 of file [bitmaps.h](#).

#### 15.31.4.59 PLUSchar

```
uint8_t PLUSchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B00000000,  
 B01000000,  
 B11100000,  
 B01000000,  
 B00000000}
```

Definition at line 939 of file [bitmaps.h](#).

#### 15.31.4.60 Qchar

```
uint8_t Qchar[ ]
```

**Initial value:**

```
=  
{B01100000,  
 B10010000,  
 B10010000,  
 B10010000,  
 B10110000,  
 B01101000}
```

Definition at line 1205 of file [bitmaps.h](#).

#### 15.31.4.61 QUESTIONchar

```
uint8_t QUESTIONchar[ ]
```

**Initial value:**

```
=  
{B01100000,  
 B10010000,  
 B00010000,  
 B00100000,  
 B00000000,  
 B00100000}
```

Definition at line 1079 of file [bitmaps.h](#).

#### 15.31.4.62 QUOTMARKchar

```
uint8_t QUOTMARKchar[ ]
```

**Initial value:**

```
=  
{B10100000,  
 B10100000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B00000000}
```

Definition at line 876 of file [bitmaps.h](#).

#### 15.31.4.63 Rchar

```
uint8_t Rchar[ ]
```

**Initial value:**

```
=  
{B11000000,  
 B10100000,  
 B10100000,  
 B11000000,  
 B11000000,  
 B10100000}
```

Definition at line 1212 of file [bitmaps.h](#).

#### 15.31.4.64 Schar

```
uint8_t Schar[ ]
```

**Initial value:**

```
=  
{B01100000,  
 B10010000,  
 B01000000,  
 B00100000,  
 B10010000,  
 B01100000}
```

Definition at line 1219 of file [bitmaps.h](#).

#### 15.31.4.65 SEMICOLONchar

```
uint8_t SEMICOLONchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B11000000,  
 B11000000,  
 B00000000,  
 B01000000,  
 B10000000}
```

Definition at line 1051 of file [bitmaps.h](#).

#### 15.31.4.66 SEVENchar

```
uint8_t SEVENchar[ ]
```

**Initial value:**

```
=  
{B11110000,  
 B00010000,  
 B00100000,  
 B01000000,  
 B10000000,  
 B10000000}
```

Definition at line 1023 of file [bitmaps.h](#).

#### 15.31.4.67 SIXchar

```
uint8_t SIXchar[ ]
```

**Initial value:**

```
=  
{B01100000,  
 B10010000,  
 B10000000,  
 B11100000,  
 B10010000,  
 B01100000}
```

Definition at line 1016 of file [bitmaps.h](#).

#### 15.31.4.68 SLASHchar

```
uint8_t SLASHchar[ ]
```

**Initial value:**

```
=  
{B00100000,  
 B00100000,  
 B01000000,  
 B01000000,  
 B10000000,  
 B10000000}
```

Definition at line 967 of file [bitmaps.h](#).

#### 15.31.4.69 SPACEchar

```
uint8_t SPACEchar[ ]  
Initial value:  
= {B00000000,  
    B00000000,  
    B00000000,  
    B00000000,  
    B00000000,  
    B00000000}
```

Definition at line 862 of file [bitmaps.h](#).

#### 15.31.4.70 SQUAREBRAQUETchar

```
uint8_t SQUAREBRAQUETchar[ ]  
Initial value:  
= {B11100000,  
    B10000000,  
    B10000000,  
    B10000000,  
    B10000000,  
    B11100000}
```

Definition at line 1275 of file [bitmaps.h](#).

#### 15.31.4.71 SWUNGchar

```
uint8_t SWUNGchar[ ]  
Initial value:  
= {B00000000,  
    B00000000,  
    B01010000,  
    B10100000,  
    B00000000,  
    B00000000}
```

Definition at line 1338 of file [bitmaps.h](#).

#### 15.31.4.72 Tchar

```
uint8_t Tchar[ ]  
Initial value:  
= {B11100000,  
    B01000000,  
    B01000000,  
    B01000000,  
    B01000000,  
    B01000000}
```

Definition at line 1226 of file [bitmaps.h](#).

#### 15.31.4.73 THREEchar

```
uint8_t THREEchar[ ]  
Initial value:  
= {B01100000,  
    B10010000,  
    B00100000,  
    B00010000,  
    B10010000,  
    B01100000}
```

Definition at line 995 of file [bitmaps.h](#).

#### 15.31.4.74 TWOchar

```
uint8_t TWOchar[ ]
```

**Initial value:**

```
=  
{B01100000,  
 B10010000,  
 B00100000,  
 B01000000,  
 B10000000,  
 B11100000}
```

Definition at line 988 of file [bitmaps.h](#).

#### 15.31.4.75 Uchar

```
uint8_t Uchar[ ]
```

**Initial value:**

```
=  
{B10100000,  
 B10100000,  
 B10100000,  
 B10100000,  
 B10100000,  
 B11100000}
```

Definition at line 1233 of file [bitmaps.h](#).

#### 15.31.4.76 UNDERSCOREchar

```
uint8_t UNDERSCOREchar[ ]
```

**Initial value:**

```
=  
{B00000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B00000000,  
 B11100000}
```

Definition at line 1303 of file [bitmaps.h](#).

#### 15.31.4.77 Vchar

```
uint8_t Vchar[ ]
```

**Initial value:**

```
=  
{B10100000,  
 B10100000,  
 B10100000,  
 B10100000,  
 B10100000,  
 B01000000}
```

Definition at line 1240 of file [bitmaps.h](#).

#### 15.31.4.78 VERTICALBARchar

```
uint8_t VERTICALBARchar[ ]
```

**Initial value:**

```
=  
{B10000000,  
 B10000000,  
 B10000000,  
 B10000000,  
 B10000000,  
 B10000000}
```

Definition at line 1331 of file [bitmaps.h](#).

#### 15.31.4.79 Wchar

```
uint8_t Wchar[ ]
```

**Initial value:**

```
=
{B10001000,
 B10001000,
 B10001000,
 B10001000,
 B10101000,
 B01010000}
```

Definition at line 1247 of file [bitmaps.h](#).

#### 15.31.4.80 width\_bitmap

```
const uint16_t width_bitmap[] = {4, 3, 4, 6, 6, 5, 6, 2, 4, 4, 4, 4, 3, 4, 2, 4, 5, 3, 5, 5, 5, 5, 5, 5, 3, 3, 4, 4, 4, 4, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 6, 5, 4, 4, 6, 4, 5, 4, 4, 4, 6, 4, 4, 4, 4, 4, 4, 3, 4, 2, 4, 5}
```

Definition at line 1631 of file [bitmaps.h](#).

#### 15.31.4.81 Xchar

```
uint8_t Xchar[ ]
```

**Initial value:**

```
=
{B10100000,
 B10100000,
 B01000000,
 B01000000,
 B10100000,
 B10100000}
```

Definition at line 1254 of file [bitmaps.h](#).

#### 15.31.4.82 Ychar

```
uint8_t Ychar[ ]
```

**Initial value:**

```
=
{B10100000,
 B10100000,
 B10100000,
 B01000000,
 B01000000,
 B01000000}
```

Definition at line 1261 of file [bitmaps.h](#).

#### 15.31.4.83 Zchar

```
uint8_t Zchar[ ]
```

**Initial value:**

```
=
{B11100000,
 B00100000,
 B01000000,
 B10000000,
 B10000000,
 B11100000}
```

Definition at line 1268 of file [bitmaps.h](#).

#### 15.31.4.84 ZEROchar

```
uint8_t ZEROchar[ ]
```

**Initial value:**

```
=
```

```
{B01100000,
B10010000,
B10010000,
B10010000,
B10010000,
B01100000}
```

Definition at line 974 of file [bitmaps.h](#).

## 15.32 bitmaps.h

[Go to the documentation of this file.](#)

```
00011 #define DISPLAY_SIZE 6      //Altura del caracter del display
00012 #define POWERBAR_LOCATION 1 // Posicion de la barra de potencia: 0 = Top, 1 = Bot.
00013 #define C_DISPLAY_WIDTH 15 // Anchura del display.
00014 #define C_DISPLAY_HEIGHT 6 // Altura del display.
00015
00016 #if DISPLAY_SIZE == 7
00017
00018 const uint8_t
00019     SPACEchar[] =
00020         {B00000000,
00021             B00000000,
00022                 B00000000,
00023                     B00000000,
00024                         B00000000,
00025                             B00000000,
00026                                 B00000000},
00027     EXCLMARKchar[] =
00028         {B11000000,
00029             B11000000,
00030                 B11000000,
00031                     B11000000,
00032                         B00000000,
00033                             B11000000,
00034                                 B10000000},
00035     QUOTMARKchar[] =
00036         {B10100000,
00037             B10100000,
00038                 B00000000,
00039                     B00000000,
00040                         B00000000,
00041                             B00000000,
00042                                 B00000000},
00043     HASTAGchar[] =
00044         {B00000000,
00045             B01001000,
00046                 B11111100,
00047                     B01001000,
00048                         B01001000,
00049                             B11111100,
00050                                 B01001000},
00051     DOLLARchar[] =
00052         {B01110000,
00053             B10101000,
00054                 B10100000,
00055                     B01110000,
00056                         B00101000,
00057                             B10101000,
00058                                 B01110000},
00059     PERCENTchar[] =
00060         {B00000000,
00061             B00000000,
00062                 B00000000,
00063                     B10010000,
00064                         B00100000,
00065                             B01000000,
00066                                 B10010000},
00067     ANPERSANchar[] =
00068         {B01000000,
00069             B10100000,
00070                 B10100000,
00071                     B01000000,
00072                         B10101000,
00073                             B10010000,
00074                                 B01101000},
00075     APOSTROPHEchar[] =
00076         {B10000000,
00077             B10000000,
00078                 B00000000,
00079                     B00000000,
00080                         B00000000,
00081                             B00000000,
00082                                 B00000000},
```

```
00083     OPENPARENTESISchar[] =
00084         {B01000000,
00085             B10000000,
00086                 B10000000,
00087                     B10000000,
00088                         B10000000,
00089                             B10000000,
00090                                 B01000000},
00091     CLOSEPARENTESISchar[] =
00092         {B10000000,
00093             B01000000,
00094                 B01000000,
00095                     B01000000,
00096                         B01000000,
00097                             B01000000,
00098                                 B10000000},
00099     ASTERISKchar[] =
00100         {B01000000,
00101             B11100000,
00102                 B01000000,
00103                     B10100000,
00104                         B00000000,
00105                             B00000000,
00106                                 B00000000},
00107     PLUSchar[] =
00108         {B00000000,
00109             B00000000,
00110                 B01000000,
00111                     B11100000,
00112                         B01000000,
00113                             B00000000,
00114                                 B00000000},
00115     COMMAschar[] =
00116         {B00000000,
00117             B00000000,
00118                 B00000000,
00119                     B00000000,
00120                         B00000000,
00121                             B01000000,
00122                                 B10000000},
00123     HYPHENchar[] =
00124         {B00000000,
00125             B00000000,
00126                 B00000000,
00127                     B11100000,
00128                         B00000000,
00129                             B00000000,
00130                                 B00000000},
00131     DOTchar[] =
00132         {B00000000,
00133             B00000000,
00134                 B00000000,
00135                     B00000000,
00136                         B00000000,
00137                             B11000000,
00138                                 B11000000},
00139     SLASHchar[] =
00140         {B00100000,
00141             B00100000,
00142                 B01000000,
00143                     B01000000,
00144                         B01000000,
00145                             B10000000,
00146                                 B10000000},
00147     ZEROchar[] =
00148         {B01100000,
00149             B10010000,
00150                 B10010000,
00151                     B10010000,
00152                         B10010000,
00153                             B10010000,
00154                                 B01100000},
00155     ONEchar[] =
00156         {B01000000,
00157             B11000000,
00158                 B01000000,
00159                     B01000000,
00160                         B01000000,
00161                             B01000000,
00162                                 B01000000},
00163     TWOchar[] =
00164         {B01100000,
00165             B10010000,
00166                 B00010000,
00167                     B01100000,
00168                         B10000000,
00169                                 B10000000,
```

```
00170      B11110000},  
00171  THREEchar[] =  
00172  {B01100000,  
00173  B10010000,  
00174  B00010000,  
00175  B00100000,  
00176  B00010000,  
00177  B10010000,  
00178  B01100000},  
00179  FOURchar[] =  
00180  {B10000000,  
00181  B10000000,  
00182  B10010000,  
00183  B11110000,  
00184  B00010000,  
00185  B00010000,  
00186  B00010000},  
00187  FIVEchar[] =  
00188  {B11110000,  
00189  B10000000,  
00190  B10000000,  
00191  B11100000,  
00192  B00010000,  
00193  B10010000,  
00194  B01100000},  
00195  SIXchar[] =  
00196  {B01100000,  
00197  B10010000,  
00198  B10000000,  
00199  B11100000,  
00200  B10010000,  
00201  B10010000,  
00202  B01100000},  
00203  SEVENchar[] =  
00204  {B11110000,  
00205  B00010000,  
00206  B00010000,  
00207  B00100000,  
00208  B01000000,  
00209  B10000000,  
00210  B10000000},  
00211  EIGHTchar[] =  
00212  {B01100000,  
00213  B10010000,  
00214  B10010000,  
00215  B01100000,  
00216  B10010000,  
00217  B10010000,  
00218  B01100000},  
00219  NINEchar[] =  
00220  {B01100000,  
00221  B10010000,  
00222  B10010000,  
00223  B01110000,  
00224  B00010000,  
00225  B10010000,  
00226  B01100000},  
00227  COLONchar[] =  
00228  {B00000000,  
00229  B00000000,  
00230  B11000000,  
00231  B11000000,  
00232  B00000000,  
00233  B11000000,  
00234  B11000000},  
00235  SEMICOLONchar[] =  
00236  {B00000000,  
00237  B00000000,  
00238  B11000000,  
00239  B11000000,  
00240  B00000000,  
00241  B01000000,  
00242  B10000000},  
00243  LESSTHANchar[] =  
00244  {B00000000,  
00245  B00000000,  
00246  B00100000,  
00247  B01000000,  
00248  B10000000,  
00249  B01000000,  
00250  B00100000},  
00251  EQUALchar[] =  
00252  {B00000000,  
00253  B00000000,  
00254  B00000000,  
00255  B00000000,  
00256  B11100000,
```

```
00257      B00000000,
00258      B11100000},
00259  GREATERTHANchar[] =
00260  {B00000000,
00261  B00000000,
00262  B10000000,
00263  B01000000,
00264  B00100000,
00265  B01000000,
00266  B10000000},
00267  QUESTIONchar[] =
00268  {B01100000,
00269  B10010000,
00270  B00010000,
00271  B00100000,
00272  B00100000,
00273  B00000000,
00274  B00100000},
00275  ATchar[] =
00276  {B01111100,
00277  B10000010,
00278  B10111010,
00279  B10101010,
00280  B10110100,
00281  B10000010,
00282  B01111100},
00283  Achar[] =
00284  {B01000000,
00285  B10100000,
00286  B11100000,
00287  B10100000,
00288  B10100000,
00289  B10100000,
00290  B10100000},
00291  Bchar[] =
00292  {B11000000,
00293  B10100000,
00294  B10100000,
00295  B11000000,
00296  B10100000,
00297  B10100000,
00298  B11000000},
00299  Cchar[] =
00300  {B01000000,
00301  B10100000,
00302  B10000000,
00303  B10000000,
00304  B10000000,
00305  B10100000,
00306  B01000000},
00307  Dchar[] =
00308  {B11000000,
00309  B10100000,
00310  B10100000,
00311  B10100000,
00312  B10100000,
00313  B10100000,
00314  B11000000},
00315  Echar[] =
00316  {B11100000,
00317  B10000000,
00318  B10000000,
00319  B11000000,
00320  B10000000,
00321  B10000000,
00322  B11100000},
00323  Fchar[] =
00324  {B11100000,
00325  B10000000,
00326  B10000000,
00327  B11000000,
00328  B10000000,
00329  B10000000,
00330  B10000000},
00331  Gchar[] =
00332  {B01000000,
00333  B10000000,
00334  B10000000,
00335  B10000000,
00336  B10100000,
00337  B10100000,
00338  B01000000},
00339  Hchar[] =
00340  {B10100000,
00341  B10100000,
00342  B10100000,
00343  B11100000},
```

```
00344     B10100000,  
00345     B10100000,  
00346     B10100000},  
00347     Ichar[] =  
00348     {B11100000,  
00349     B01000000,  
00350     B01000000,  
00351     B01000000,  
00352     B01000000,  
00353     B01000000,  
00354     B11100000},  
00355     Jchar[] =  
00356     {B11100000,  
00357     B00100000,  
00358     B00100000,  
00359     B00100000,  
00360     B00100000,  
00361     B10100000,  
00362     B11100000},  
00363     Kchar[] =  
00364     {B10000000,  
00365     B10000000,  
00366     B10100000,  
00367     B11000000,  
00368     B10000000,  
00369     B11000000,  
00370     B10100000},  
00371     Lchar[] =  
00372     {B10000000,  
00373     B10000000,  
00374     B10000000,  
00375     B10000000,  
00376     B10000000,  
00377     B10000000,  
00378     B11000000},  
00379     Mchar[] =  
00380     {B10001000,  
00381     B11011000,  
00382     B10101000,  
00383     B10001000,  
00384     B10001000,  
00385     B10001000,  
00386     B10001000},  
00387     Nchar[] =  
00388     {B10010000,  
00389     B11010000,  
00390     B10110000,  
00391     B10010000,  
00392     B10010000,  
00393     B10010000,  
00394     B10010000},  
00395     Ochar[] =  
00396     {B01000000,  
00397     B10100000,  
00398     B10100000,  
00399     B10100000,  
00400     B10100000,  
00401     B10100000,  
00402     B01000000},  
00403     Pchar[] =  
00404     {B11000000,  
00405     B10100000,  
00406     B10100000,  
00407     B11000000,  
00408     B10000000,  
00409     B10000000,  
00410     B10000000},  
00411     Qchar[] =  
00412     {B01100000,  
00413     B10010000,  
00414     B10010000,  
00415     B10010000,  
00416     B10010000,  
00417     B10110000,  
00418     B01101000},  
00419     Rchar[] =  
00420     {B11000000,  
00421     B10100000,  
00422     B10100000,  
00423     B10100000,  
00424     B11000000,  
00425     B11000000,  
00426     B10100000},  
00427     Schar[] =  
00428     {B01000000,  
00429     B10100000,  
00430     B10000000,
```

```
00431      B01000000,
00432      B00100000,
00433      B10100000,
00434      B01000000},
00435  Tchar[] =
00436  {B11100000,
00437  B01000000,
00438  B01000000,
00439  B01000000,
00440  B01000000,
00441  B01000000,
00442  B01000000},
00443  Uchar[] =
00444  {B10100000,
00445  B10100000,
00446  B10100000,
00447  B10100000,
00448  B10100000,
00449  B10100000,
00450  B11100000},
00451  Vchar[] =
00452  {B10100000,
00453  B10100000,
00454  B10100000,
00455  B10100000,
00456  B10100000,
00457  B10100000,
00458  B01000000},
00459  Wchar[] =
00460  {B10001000,
00461  B10001000,
00462  B10001000,
00463  B10001000,
00464  B10001000,
00465  B10101000,
00466  B01010000},
00467  Xchar[] =
00468  {B00000000,
00469  B00000000,
00470  B10001000,
00471  B01010000,
00472  B00100000,
00473  B01010000,
00474  B10001000},
00475  Ychar[] =
00476  {B10001000,
00477  B10001000,
00478  B10001000,
00479  B01010000,
00480  B00100000,
00481  B00100000,
00482  B00100000},
00483  Zchar[] =
00484  {B00000000,
00485  B11110000,
00486  B00010000,
00487  B00100000,
00488  B01000000,
00489  B10000000,
00490  B11110000},
00491  SQUAREBRAQUETchar[] =
00492  {B11100000,
00493  B10000000,
00494  B10000000,
00495  B10000000,
00496  B10000000,
00497  B10000000,
00498  B11100000},
00499  BACKSLASHchar[] =
00500  {B10000000,
00501  B10000000,
00502  B01000000,
00503  B01000000,
00504  B01000000,
00505  B00100000,
00506  B00100000},
00507  CLOSESQUAREBRAQUETchar[] =
00508  {B11100000,
00509  B00100000,
00510  B00100000,
00511  B00100000,
00512  B00100000,
00513  B00100000,
00514  B11100000},
00515  CIRCUMFLEXchar[] =
00516  {B01000000,
00517  B10100000,
```

```
00518     B00000000,
00519     B00000000,
00520     B00000000,
00521     B00000000,
00522     B00000000},
00523     UNDERSCOREchar[] =
00524     {B00000000,
00525     B00000000,
00526     B00000000,
00527     B00000000,
00528     B00000000,
00529     B00000000,
00530     B11100000},
00531     ACCENTchar[] =
00532     {B10000000,
00533     B01000000,
00534     B00000000,
00535     B00000000,
00536     B00000000,
00537     B00000000,
00538     B00000000},
00539     OPENCURLYBRACchar[] =
00540     {B01100000,
00541     B01000000,
00542     B01000000,
00543     B11000000,
00544     B01000000,
00545     B01000000,
00546     B01100000},
00547     CLOSECURLYBRACchar[] =
00548     {B11000000,
00549     B01000000,
00550     B01100000,
00551     B01000000,
00552     B01000000,
00553     B01000000,
00554     B11000000},
00555     VERTICALBARchar[] =
00556     {B10000000,
00557     B10000000,
00558     B10000000,
00559     B10000000,
00560     B10000000,
00561     B10000000,
00562     B10000000},
00563     SWUNGchar[] =
00564     {B00000000,
00565     B00000000,
00566     B00000000,
00567     B01010000,
00568     B10100000,
00569     B00000000,
00570     B00000000};
00571
00572 void Char2Bitmap(char character, uint8_t *bitmap, int16_t array_len)
00573 {
00574     if (array_len < sizeof(SPACEchar))
00575     {
00576         for (int i = 0; i < array_len; i++)
00577         {
00578             bitmap[i] = 0xFF;
00579         }
00580     }
00581     else if (character == 32)
00582     {
00583         memcpy(bitmap, SPACEchar, array_len);
00584     }
00585     else if (character == 33)
00586     {
00587         memcpy(bitmap, EXCLMARKchar, array_len);
00588     }
00589     else if (character == 34)
00590     {
00591         memcpy(bitmap, QUOTMARKchar, array_len);
00592     }
00593     else if (character == 35)
00594     {
00595         memcpy(bitmap, HASTAGchar, array_len);
00596     }
00597     else if (character == 36)
00598     {
00599         memcpy(bitmap, DOLLARchar, array_len);
00600     }
00601     else if (character == 37)
00602     {
00603         memcpy(bitmap, PERCENTchar, array_len);
00604     }
}
```

```
00605     else if (character == 38)
00606     {
00607         memcpy(bitmap, ANPERSANchar, array_len);
00608     }
00609     else if (character == 39)
00610     {
00611         memcpy(bitmap, APOSTROPHEchar, array_len);
00612     }
00613     else if (character == 40)
00614     {
00615         memcpy(bitmap, OPENPARENTESISchar, array_len);
00616     }
00617     else if (character == 41)
00618     {
00619         memcpy(bitmap, CLOSEPARENTESISchar, array_len);
00620     }
00621     else if (character == 42)
00622     {
00623         memcpy(bitmap, ASTERISKchar, array_len);
00624     }
00625     else if (character == 43)
00626     {
00627         memcpy(bitmap, PLUSchar, array_len);
00628     }
00629     else if (character == 44)
00630     {
00631         memcpy(bitmap, COMMAschar, array_len);
00632     }
00633     else if (character == 45)
00634     {
00635         memcpy(bitmap, HYPHENchar, array_len);
00636     }
00637     else if (character == 46)
00638     {
00639         memcpy(bitmap, DOTchar, array_len);
00640     }
00641     else if (character == 47)
00642     {
00643         memcpy(bitmap, SLASHchar, array_len);
00644     }
00645     else if (character == 48)
00646     {
00647         memcpy(bitmap, ZEROchar, array_len);
00648     }
00649     else if (character == 49)
00650     {
00651         memcpy(bitmap, ONEchar, array_len);
00652     }
00653     else if (character == 50)
00654     {
00655         memcpy(bitmap, TWOchar, array_len);
00656     }
00657     else if (character == 51)
00658     {
00659         memcpy(bitmap, THREEchar, array_len);
00660     }
00661     else if (character == 52)
00662     {
00663         memcpy(bitmap, FOURchar, array_len);
00664     }
00665     else if (character == 53)
00666     {
00667         memcpy(bitmap, FIVEchar, array_len);
00668     }
00669     else if (character == 54)
00670     {
00671         memcpy(bitmap, SIXchar, array_len);
00672     }
00673     else if (character == 55)
00674     {
00675         memcpy(bitmap, SEVENchar, array_len);
00676     }
00677     else if (character == 56)
00678     {
00679         memcpy(bitmap, EIGHTchar, array_len);
00680     }
00681     else if (character == 57)
00682     {
00683         memcpy(bitmap, NINEchar, array_len);
00684     }
00685     else if (character == 58)
00686     {
00687         memcpy(bitmap, COLONchar, array_len);
00688     }
00689     else if (character == 59)
00690     {
00691         memcpy(bitmap, SEMICOLONchar, array_len);
```

```
00692     }
00693     else if (character == 60)
00694     {
00695         memcpy(bitmap, LESSTHANchar, array_len);
00696     }
00697     else if (character == 61)
00698     {
00699         memcpy(bitmap, EQUALchar, array_len);
00700     }
00701     else if (character == 62)
00702     {
00703         memcpy(bitmap, GREATERTHANchar, array_len);
00704     }
00705     else if (character == 63)
00706     {
00707         memcpy(bitmap, QUESTIONchar, array_len);
00708     }
00709     else if (character == 64)
00710     {
00711         memcpy(bitmap, ATchar, array_len);
00712     }
00713     else if (character == 65)
00714     {
00715         memcpy(bitmap, Achar, array_len);
00716     }
00717     else if (character == 66)
00718     {
00719         memcpy(bitmap, Bchar, array_len);
00720     }
00721     else if (character == 67)
00722     {
00723         memcpy(bitmap, Cchar, array_len);
00724     }
00725     else if (character == 68)
00726     {
00727         memcpy(bitmap, Dchar, array_len);
00728     }
00729     else if (character == 69)
00730     {
00731         memcpy(bitmap, Echar, array_len);
00732     }
00733     else if (character == 70)
00734     {
00735         memcpy(bitmap, Fchar, array_len);
00736     }
00737     else if (character == 71)
00738     {
00739         memcpy(bitmap, Gchar, array_len);
00740     }
00741     else if (character == 72)
00742     {
00743         memcpy(bitmap, Hchar, array_len);
00744     }
00745     else if (character == 73)
00746     {
00747         memcpy(bitmap, Ichar, array_len);
00748     }
00749     else if (character == 74)
00750     {
00751         memcpy(bitmap, Jchar, array_len);
00752     }
00753     else if (character == 75)
00754     {
00755         memcpy(bitmap, Kchar, array_len);
00756     }
00757     else if (character == 76)
00758     {
00759         memcpy(bitmap, Lchar, array_len);
00760     }
00761     else if (character == 77)
00762     {
00763         memcpy(bitmap, Mchar, array_len);
00764     }
00765     else if (character == 78)
00766     {
00767         memcpy(bitmap, Nchar, array_len);
00768     }
00769     else if (character == 79)
00770     {
00771         memcpy(bitmap, Ochar, array_len);
00772     }
00773     else if (character == 80)
00774     {
00775         memcpy(bitmap, Pchar, array_len);
00776     }
00777     else if (character == 81)
00778     {
```

```

00779     memcpy(bitmap, Qchar, array_len);
00780 }
00781 else if (character == 82)
00782 {
00783     memcpy(bitmap, Rchar, array_len);
00784 }
00785 else if (character == 83)
00786 {
00787     memcpy(bitmap, Schar, array_len);
00788 }
00789 else if (character == 84)
00790 {
00791     memcpy(bitmap, Tchar, array_len);
00792 }
00793 else if (character == 85)
00794 {
00795     memcpy(bitmap, Uchar, array_len);
00796 }
00797 else if (character == 86)
00798 {
00799     memcpy(bitmap, Vchar, array_len);
00800 }
00801 else if (character == 87)
00802 {
00803     memcpy(bitmap, Wchar, array_len);
00804 }
00805 else if (character == 88)
00806 {
00807     memcpy(bitmap, Xchar, array_len);
00808 }
00809 else if (character == 89)
00810 {
00811     memcpy(bitmap, Ychar, array_len);
00812 }
00813 else if (character == 90)
00814 {
00815     memcpy(bitmap, Zchar, array_len);
00816 }
00817 else if (character == 91)
00818 {
00819     memcpy(bitmap, SQUAREBRAQUETchar, array_len);
00820 }
00821 else if (character == 92)
00822 {
00823     memcpy(bitmap, BACKSLASHchar, array_len);
00824 }
00825 else if (character == 93)
00826 {
00827     memcpy(bitmap, CLOSESQUAREBRAQUETchar, array_len);
00828 }
00829 else if (character == 94)
00830 {
00831     memcpy(bitmap, CIRCUMFLEXchar, array_len);
00832 }
00833 else if (character == 95)
00834 {
00835     memcpy(bitmap, UNDERSCOREchar, array_len);
00836 }
00837 else if (character == 96)
00838 {
00839     memcpy(bitmap, ACCENTchar, array_len);
00840 }
00841 else if (character == 123)
00842 {
00843     memcpy(bitmap, OPENCURLYBRAchar, array_len);
00844 }
00845 else if (character == 124)
00846 {
00847     memcpy(bitmap, VERTICALBARchar, array_len);
00848 }
00849 else if (character == 125)
00850 {
00851     memcpy(bitmap, CLOSECURLYBRACHAR, array_len);
00852 }
00853 else if (character == 126)
00854 {
00855     memcpy(bitmap, SWUNGchar, array_len);
00856 }
00857 }
00858 const uint16_t width_bitmap[] = {4, 3, 4, 6, 6, 5, 6, 2, 4, 4, 4, 4, 4, 3, 4, 5, 3, 5, 5, 5, 5, 5, 5, 5, 3, 3, 4, 4, 4, 8, 4, 4, 4, 4, 4, 4, 4, 4, 3, 6, 5, 4, 4, 6, 4, 4, 4, 4, 4, 6, 6, 6, 5, 4, 4, 4, 4, 3, 4, 2, 4, 5};
00859
00860 #elif DISPLAY_SIZE == 6
00861 uint8_t
00862     SPACEchar[] =
00863     {B00000000,

```

```
00864      B00000000,  
00865      B00000000,  
00866      B00000000,  
00867      B00000000,  
00868      B00000000},  
00869      EXCLMARKchar[] =  
00870      {B1000000,  
00871      B1000000,  
00872      B1000000,  
00873      B0000000,  
00874      B1000000,  
00875      B1000000},  
00876      QUOTMARKchar[] =  
00877      {B10100000,  
00878      B10100000,  
00879      B00000000,  
00880      B00000000,  
00881      B00000000,  
00882      B00000000},  
00883      HASTAGchar[] =  
00884      {B01001000,  
00885      B11111100,  
00886      B01001000,  
00887      B01001000,  
00888      B11111100,  
00889      B01001000},  
00890      DOLLARchar[] =  
00891      {B01111000,  
00892      B10100000,  
00893      B01110000,  
00894      B00101000,  
00895      B00101000,  
00896      B11110000},  
00897      PERCENTchar[] =  
00898      {B00000000,  
00899      B00000000,  
00900      B10010000,  
00901      B00100000,  
00902      B01000000,  
00903      B10010000},  
00904      ANPERSANchar[] =  
00905      {B01000000,  
00906      B10100000,  
00907      B01000000,  
00908      B10101000,  
00909      B10010000,  
00910      B01101000},  
00911      APOSTROPHEchar[] =  
00912      {B10000000,  
00913      B10000000,  
00914      B00000000,  
00915      B00000000,  
00916      B00000000,  
00917      B00000000},  
00918      OPENPARENTESISchar[] =  
00919      {B01000000,  
00920      B10000000,  
00921      B10000000,  
00922      B10000000,  
00923      B10000000,  
00924      B01000000},  
00925      CLOSEPARENTESISchar[] =  
00926      {B10000000,  
00927      B01000000,  
00928      B01000000,  
00929      B01000000,  
00930      B01000000,  
00931      B10000000},  
00932      ASTERISKchar[] =  
00933      {B01000000,  
00934      B11100000,  
00935      B01000000,  
00936      B10100000,  
00937      B00000000,  
00938      B00000000},  
00939      PLUSchar[] =  
00940      {B00000000,  
00941      B00000000,  
00942      B01000000,  
00943      B11100000,  
00944      B01000000,  
00945      B00000000},  
00946      COMMAchar[] =  
00947      {B00000000,  
00948      B00000000,  
00949      B00000000,  
00950      B00000000,
```

```
00951      B01000000,  
00952      B10000000},  
00953  HYPHENchar[] =  
00954  {B00000000,  
00955  B00000000,  
00956  B00000000,  
00957  B11100000,  
00958  B00000000,  
00959  B00000000},  
00960  DOTchar[] =  
00961  {B00000000,  
00962  B00000000,  
00963  B00000000,  
00964  B00000000,  
00965  B00000000,  
00966  B10000000},  
00967  SLASHchar[] =  
00968  {B00100000,  
00969  B00100000,  
00970  B01000000,  
00971  B01000000,  
00972  B10000000,  
00973  B10000000},  
00974  ZEROchar[] =  
00975  {B01100000,  
00976  B10010000,  
00977  B10010000,  
00978  B10010000,  
00979  B10010000,  
00980  B01100000},  
00981  ONEchar[] =  
00982  {B01000000,  
00983  B11000000,  
00984  B01000000,  
00985  B01000000,  
00986  B01000000,  
00987  B01000000},  
00988  TWOchar[] =  
00989  {B01100000,  
00990  B10010000,  
00991  B00100000,  
00992  B01000000,  
00993  B10000000,  
00994  B11110000},  
00995  THREEchar[] =  
00996  {B01100000,  
00997  B10010000,  
00998  B00100000,  
00999  B00010000,  
01000  B10010000,  
01001  B01100000},  
01002  FOURchar[] =  
01003  {B10000000,  
01004  B10000000,  
01005  B10010000,  
01006  B11110000,  
01007  B00010000,  
01008  B00010000},  
01009  FIVEchar[] =  
01010  {B11110000,  
01011  B10000000,  
01012  B11100000,  
01013  B00010000,  
01014  B10010000,  
01015  B01100000},  
01016  SIXchar[] =  
01017  {B01100000,  
01018  B10010000,  
01019  B10000000,  
01020  B11100000,  
01021  B10010000,  
01022  B01100000},  
01023  SEVENchar[] =  
01024  {B11110000,  
01025  B00010000,  
01026  B00100000,  
01027  B01000000,  
01028  B10000000,  
01029  B10000000},  
01030  EIGHTchar[] =  
01031  {B01100000,  
01032  B10010000,  
01033  B01100000,  
01034  B10010000,  
01035  B10010000,  
01036  B01100000},  
01037  NINEchar[] =
```

```
01038      {B01100000,
01039      B10010000,
01040      B01110000,
01041      B00010000,
01042      B10010000,
01043      B01100000},
01044  COLONchar[] =
01045  {B00000000,
01046  B11000000,
01047  B11000000,
01048  B00000000,
01049  B11000000,
01050  B11000000},
01051  SEMICOLONchar[] =
01052  {B00000000,
01053  B11000000,
01054  B11000000,
01055  B00000000,
01056  B01000000,
01057  B10000000},
01058  LESSTHANchar[] =
01059  {B00000000,
01060  B00100000,
01061  B01000000,
01062  B10000000,
01063  B01000000,
01064  B00100000},
01065  EQUALchar[] =
01066  {B00000000,
01067  B00000000,
01068  B00000000,
01069  B11100000,
01070  B00000000,
01071  B11100000},
01072  GREATERTHANchar[] =
01073  {B00000000,
01074  B10000000,
01075  B01000000,
01076  B00100000,
01077  B01000000,
01078  B10000000},
01079  QUESTIONchar[] =
01080  {B01100000,
01081  B10010000,
01082  B00010000,
01083  B00100000,
01084  B00000000,
01085  B00100000},
01086  ATchar[] =
01087  {B01111000,
01088  B10000100,
01089  B10110100,
01090  B10111000,
01091  B10000100,
01092  B01111000},
01093  Achar[] =
01094  {B01000000,
01095  B10100000,
01096  B11100000,
01097  B10100000,
01098  B10100000,
01099  B10100000},
01100  Bchar[] =
01101  {B11000000,
01102  B10100000,
01103  B11000000,
01104  B10100000,
01105  B10100000,
01106  B11000000},
01107  Cchar[] =
01108  {B01000000,
01109  B10100000,
01110  B10000000,
01111  B10000000,
01112  B10100000,
01113  B01000000},
01114  Dchar[] =
01115  {B11000000,
01116  B10100000,
01117  B10100000,
01118  B10100000,
01119  B10100000,
01120  B11000000},
01121  Echar[] =
01122  {B11100000,
01123  B10000000,
01124  B11000000},
```

```
01125      B10000000,
01126      B10000000,
01127      B11100000},
01128  Fchar[] =
01129  {B11100000,
01130  B10000000,
01131  B11000000,
01132  B10000000,
01133  B10000000,
01134  B10000000},
01135  Gchar[] =
01136  {B01000000,
01137  B10100000,
01138  B10000000,
01139  B10100000,
01140  B10100000,
01141  B01000000},
01142  Hchar[] =
01143  {B10100000,
01144  B10100000,
01145  B11100000,
01146  B10100000,
01147  B10100000,
01148  B10100000},
01149  Ichar[] =
01150  {B11100000,
01151  B01000000,
01152  B01000000,
01153  B01000000,
01154  B01000000,
01155  B11100000},
01156  Jchar[] =
01157  {B11100000,
01158  B00100000,
01159  B00100000,
01160  B00100000,
01161  B10100000,
01162  B11100000},
01163  Kchar[] =
01164  {B10000000,
01165  B10100000,
01166  B11000000,
01167  B10000000,
01168  B11000000,
01169  B10100000},
01170  Lchar[] =
01171  {B10000000,
01172  B10000000,
01173  B10000000,
01174  B10000000,
01175  B10000000,
01176  B11000000},
01177  Mchar[] =
01178  {B10001000,
01179  B11011000,
01180  B10101000,
01181  B10001000,
01182  B10001000,
01183  B10001000},
01184  Nchar[] =
01185  {B10010000,
01186  B11010000,
01187  B10110000,
01188  B10010000,
01189  B10010000,
01190  B10010000},
01191  Ochar[] =
01192  {B01000000,
01193  B10100000,
01194  B10100000,
01195  B10100000,
01196  B10100000,
01197  B01000000},
01198  Pchar[] =
01199  {B11000000,
01200  B10100000,
01201  B10100000,
01202  B11000000,
01203  B10000000,
01204  B10000000},
01205  Qchar[] =
01206  {B01100000,
01207  B10010000,
01208  B10010000,
01209  B10010000,
01210  B10110000,
01211  B01101000},
```

```
01212     Rchar[] =
01213         {B11000000,
01214             B10100000,
01215                 B10100000,
01216                     B11000000,
01217                         B11000000,
01218                             B10100000},
01219     Schar[] =
01220         {B01100000,
01221             B10010000,
01222                 B01000000,
01223                     B00100000,
01224                         B10010000,
01225                             B01100000},
01226     Tchar[] =
01227         {B11100000,
01228             B01000000,
01229                 B01000000,
01230                     B01000000,
01231                         B01000000,
01232                             B01000000},
01233     Uchar[] =
01234         {B10100000,
01235             B10100000,
01236                 B10100000,
01237                     B10100000,
01238                         B10100000,
01239                             B11100000},
01240     Vchar[] =
01241         {B10100000,
01242             B10100000,
01243                 B10100000,
01244                     B10100000,
01245                         B10100000,
01246                             B01000000},
01247     Wchar[] =
01248         {B10001000,
01249             B10001000,
01250                 B10001000,
01251                     B10001000,
01252                         B10101000,
01253                             B01010000},
01254     Xchar[] =
01255         {B10100000,
01256             B10100000,
01257                 B01000000,
01258                     B01000000,
01259                         B10100000,
01260                             B10100000},
01261     Ychar[] =
01262         {B10100000,
01263             B10100000,
01264                 B10100000,
01265                     B01000000,
01266                         B01000000,
01267                             B01000000},
01268     Zchar[] =
01269         {B11100000,
01270             B00100000,
01271                 B01000000,
01272                     B10000000,
01273                         B10000000,
01274                             B11100000},
01275     SQUAREBRAQUETchar[] =
01276         {B11100000,
01277             B10000000,
01278                 B10000000,
01279                     B10000000,
01280                         B10000000,
01281                             B11100000},
01282     BACKSLASHchar[] =
01283         {B10000000,
01284             B10000000,
01285                 B01000000,
01286                     B01000000,
01287                         B00100000,
01288                             B00100000},
01289     CLOSESQUAREBRAQUETchar[] =
01290         {B11100000,
01291             B00100000,
01292                 B00100000,
01293                     B00100000,
01294                         B00100000,
01295                             B11100000},
01296     CIRCUMFLEXchar[] =
01297         {B01000000,
01298             B10100000,
```

```
01299      B00000000,
01300      B00000000,
01301      B00000000,
01302      B00000000},
01303  UNDERSCOREchar[] =
01304  {B00000000,
01305  B00000000,
01306  B00000000,
01307  B00000000,
01308  B00000000,
01309  B11100000},
01310  ACCENTchar[] =
01311  {B10000000,
01312  B01000000,
01313  B00000000,
01314  B00000000,
01315  B00000000,
01316  B00000000},
01317  OPENCURLYBRAchar[] =
01318  {B01100000,
01319  B01000000,
01320  B11000000,
01321  B01000000,
01322  B01000000,
01323  B01100000},
01324  CLOSECURLYBRAchar[] =
01325  {B11000000,
01326  B01000000,
01327  B01100000,
01328  B01000000,
01329  B01000000,
01330  B11000000},
01331  VERTICALBARchar[] =
01332  {B10000000,
01333  B10000000,
01334  B10000000,
01335  B10000000,
01336  B10000000,
01337  B10000000},
01338  SWUNGchar[] =
01339  {B00000000,
01340  B00000000,
01341  B01010000,
01342  B10100000,
01343  B00000000,
01344  B00000000};
01345 void Char2Bitmap(char character, uint8_t *bitmap, int16_t array_len)
01346 {
01347  if (array_len < sizeof(SPACEchar))
01348  {
01349    for (int i = 0; i < array_len; i++)
01350    {
01351      bitmap[i] = 0xFF;
01352    }
01353  }
01354  else if (character == 32)
01355  {
01356    memcpy(bitmap, SPACEchar, array_len);
01357  }
01358  else if (character == 33)
01359  {
01360    memcpy(bitmap, EXCLMARKchar, array_len);
01361  }
01362  else if (character == 34)
01363  {
01364    memcpy(bitmap, QUOTMARKchar, array_len);
01365  }
01366  else if (character == 35)
01367  {
01368    memcpy(bitmap, HASTAGchar, array_len);
01369  }
01370  else if (character == 36)
01371  {
01372    memcpy(bitmap, DOLLARchar, array_len);
01373  }
01374  else if (character == 37)
01375  {
01376    memcpy(bitmap, PERCENTchar, array_len);
01377  }
01378  else if (character == 38)
01379  {
01380    memcpy(bitmap, ANPERSANchar, array_len);
01381  }
01382  else if (character == 39)
01383  {
01384    memcpy(bitmap, APOSTROPHEchar, array_len);
01385  }
```

```
01386     else if (character == 40)
01387     {
01388         memcpy(bitmap, OPENPARENTESISchar, array_len);
01389     }
01390     else if (character == 41)
01391     {
01392         memcpy(bitmap, CLOSEPARENTESISchar, array_len);
01393     }
01394     else if (character == 42)
01395     {
01396         memcpy(bitmap, ASTERISKchar, array_len);
01397     }
01398     else if (character == 43)
01399     {
01400         memcpy(bitmap, PLUSchar, array_len);
01401     }
01402     else if (character == 44)
01403     {
01404         memcpy(bitmap, COMMAschar, array_len);
01405     }
01406     else if (character == 45)
01407     {
01408         memcpy(bitmap, HYPHENchar, array_len);
01409     }
01410     else if (character == 46)
01411     {
01412         memcpy(bitmap, DOTchar, array_len);
01413     }
01414     else if (character == 47)
01415     {
01416         memcpy(bitmap, SLASHchar, array_len);
01417     }
01418     else if (character == 48)
01419     {
01420         memcpy(bitmap, ZEROchar, array_len);
01421     }
01422     else if (character == 49)
01423     {
01424         memcpy(bitmap, ONEchar, array_len);
01425     }
01426     else if (character == 50)
01427     {
01428         memcpy(bitmap, TWOchar, array_len);
01429     }
01430     else if (character == 51)
01431     {
01432         memcpy(bitmap, THREEchar, array_len);
01433     }
01434     else if (character == 52)
01435     {
01436         memcpy(bitmap, FOURchar, array_len);
01437     }
01438     else if (character == 53)
01439     {
01440         memcpy(bitmap, FIVEchar, array_len);
01441     }
01442     else if (character == 54)
01443     {
01444         memcpy(bitmap, SIXchar, array_len);
01445     }
01446     else if (character == 55)
01447     {
01448         memcpy(bitmap, SEVENchar, array_len);
01449     }
01450     else if (character == 56)
01451     {
01452         memcpy(bitmap, EIGHTchar, array_len);
01453     }
01454     else if (character == 57)
01455     {
01456         memcpy(bitmap, NINEchar, array_len);
01457     }
01458     else if (character == 58)
01459     {
01460         memcpy(bitmap, COLONchar, array_len);
01461     }
01462     else if (character == 59)
01463     {
01464         memcpy(bitmap, SEMICOLONchar, array_len);
01465     }
01466     else if (character == 60)
01467     {
01468         memcpy(bitmap, LESSTHANchar, array_len);
01469     }
01470     else if (character == 61)
01471     {
01472         memcpy(bitmap, EQUALchar, array_len);
```

```
01473     }
01474     else if (character == 62)
01475     {
01476         memcpy(bitmap, GREATERTHANchar, array_len);
01477     }
01478     else if (character == 63)
01479     {
01480         memcpy(bitmap, QUESTIONchar, array_len);
01481     }
01482     else if (character == 64)
01483     {
01484         memcpy(bitmap, ATchar, array_len);
01485     }
01486     else if (character == 65)
01487     {
01488         memcpy(bitmap, Achar, array_len);
01489     }
01490     else if (character == 66)
01491     {
01492         memcpy(bitmap, Bchar, array_len);
01493     }
01494     else if (character == 67)
01495     {
01496         memcpy(bitmap, Cchar, array_len);
01497     }
01498     else if (character == 68)
01499     {
01500         memcpy(bitmap, Dchar, array_len);
01501     }
01502     else if (character == 69)
01503     {
01504         memcpy(bitmap, Echar, array_len);
01505     }
01506     else if (character == 70)
01507     {
01508         memcpy(bitmap, Fchar, array_len);
01509     }
01510     else if (character == 71)
01511     {
01512         memcpy(bitmap, Gchar, array_len);
01513     }
01514     else if (character == 72)
01515     {
01516         memcpy(bitmap, Hchar, array_len);
01517     }
01518     else if (character == 73)
01519     {
01520         memcpy(bitmap, Ichar, array_len);
01521     }
01522     else if (character == 74)
01523     {
01524         memcpy(bitmap, Jchar, array_len);
01525     }
01526     else if (character == 75)
01527     {
01528         memcpy(bitmap, Kchar, array_len);
01529     }
01530     else if (character == 76)
01531     {
01532         memcpy(bitmap, Lchar, array_len);
01533     }
01534     else if (character == 77)
01535     {
01536         memcpy(bitmap, Mchar, array_len);
01537     }
01538     else if (character == 78)
01539     {
01540         memcpy(bitmap, Nchar, array_len);
01541     }
01542     else if (character == 79)
01543     {
01544         memcpy(bitmap, Ochar, array_len);
01545     }
01546     else if (character == 80)
01547     {
01548         memcpy(bitmap, Pchar, array_len);
01549     }
01550     else if (character == 81)
01551     {
01552         memcpy(bitmap, Qchar, array_len);
01553     }
01554     else if (character == 82)
01555     {
01556         memcpy(bitmap, Rchar, array_len);
01557     }
01558     else if (character == 83)
01559     {
```

```
01560     memcpy(bitmap, Schar, array_len);
01561 }
01562 else if (character == 84)
01563 {
01564     memcpy(bitmap, Tchar, array_len);
01565 }
01566 else if (character == 85)
01567 {
01568     memcpy(bitmap, Uchar, array_len);
01569 }
01570 else if (character == 86)
01571 {
01572     memcpy(bitmap, Vchar, array_len);
01573 }
01574 else if (character == 87)
01575 {
01576     memcpy(bitmap, Wchar, array_len);
01577 }
01578 else if (character == 88)
01579 {
01580     memcpy(bitmap, Xchar, array_len);
01581 }
01582 else if (character == 89)
01583 {
01584     memcpy(bitmap, Ychar, array_len);
01585 }
01586 else if (character == 90)
01587 {
01588     memcpy(bitmap, Zchar, array_len);
01589 }
01590 else if (character == 91)
01591 {
01592     memcpy(bitmap, SQUAREBRAQUETchar, array_len);
01593 }
01594 else if (character == 92)
01595 {
01596     memcpy(bitmap, BACKSLASHchar, array_len);
01597 }
01598 else if (character == 93)
01599 {
01600     memcpy(bitmap, CLOSESSQUAREBRAQUETchar, array_len);
01601 }
01602 else if (character == 94)
01603 {
01604     memcpy(bitmap, CIRCUMFLEXchar, array_len);
01605 }
01606 else if (character == 95)
01607 {
01608     memcpy(bitmap, UNDERSCOREchar, array_len);
01609 }
01610 else if (character == 96)
01611 {
01612     memcpy(bitmap, ACCENTchar, array_len);
01613 }
01614 else if (character == 123)
01615 {
01616     memcpy(bitmap, OPENCURLYBRAchar, array_len);
01617 }
01618 else if (character == 124)
01619 {
01620     memcpy(bitmap, VERTICALBARchar, array_len);
01621 }
01622 else if (character == 125)
01623 {
01624     memcpy(bitmap, CLOSECURLYBRAchar, array_len);
01625 }
01626 else if (character == 126)
01627 {
01628     memcpy(bitmap, SWUNGchar, array_len);
01629 }
01630 }
01631 const uint16_t width_bitmap[] = {4, 3, 4, 6, 6, 5, 6, 2, 4, 4, 4, 4, 3, 4, 2, 4, 5, 3, 5, 5, 5, 5,
5, 5, 3, 3, 4, 4, 4, 4, 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 6, 5, 4, 4, 6, 4, 5, 4, 4, 4, 6, 4,
4, 4, 4, 4, 4, 3, 4, 2, 4, 5};
01632
01633 #else "Size of character not supported."
01634 #endif
01635
01636 uint8_t
01637     Frame_Batt_1[] =
01638     {B01111111,
01639      B01000000,
01640      B01000000,
01641      B01000000,
01642      B01000000,
01643      B01111111},
01644
```

```
01645     Frame_Batt_2[] =
01646         {B11111000,
01647             B00001000,
01648                 B00001100,
01649                     B00001100,
01650                         B00001000,
01651                             B11111000};
01652 uint8_t
01653     Frame_USB_1[] =
01654         {B00000111,
01655             B00001000,
01656                 B11111000,
01657                     B00001000,
01658                         B00000111,
01659                             B00000000},
01660
01661     Frame_USB_2[] =
01662         {B11100000,
01663             B00111000,
01664                 B00101000,
01665                     B00111000,
01666                         B11100000,
01667                             B00000000};
01668
01669 uint8_t
01670     EYES_1[] =
01671         {B01111100,
01672             B10000010,
01673                 B10011010,
01674                     B10011010,
01675                         B10000010,
01676                             B01111100},
01677
01678     EYES_2[] =
01679         {B01111100,
01680             B10000010,
01681                 B10110010,
01682                     B10110010,
01683                         B10000010,
01684                             B01111100},
01685
01686     EYES2_1[] =
01687         {B01111100,
01688             B11111110,
01689                 B11110010,
01690                     B11110010,
01691                         B11111110,
01692                             B01111100},
01693
01694     EYES2_2[] =
01695         {B01111100,
01696             B11111110,
01697                 B10011110,
01698                     B10011110,
01699                         B11111110,
01700                             B01111100};
01701
01702 uint8_t
01703     FACE_1[] =
01704         {B11100000,
01705             B00000001,
01706                 B01000001,
01707                     B00000001,
01708                         B00000100,
01709                             B00000011},
01710
01711     FACE_2[] =
01712         {B00001110,
01713             B00000000,
01714                 B00000100,
01715                     B00000000,
01716                         B01000000,
01717                             B10000000};
01718 uint8_t
01719     Frame_Batt_Low_1[] =
01720         {B00000000,
01721             B00111111,
01722                 B00100001,
01723                     B00100010,
01724                         B00111111,
01725                             B00000100},
01726
01727     Frame_Batt_Low_2[] =
01728         {B10000000,
01729             B11110000,
01730                 B00011000,
01731                     B00011000,
```

```

01732     B11110000,
01733     B00000000};
01734 uint8_t
01735     Frame_Error_1[] =
01736     {B00001000,
01737     B00010001,
01738     B00010001,
01739     B00010000,
01740     B00010001,
01741     B00001000},
01742
01743     Frame_Error_2[] =
01744     {B00100000,
01745     B00010000,
01746     B00010000,
01747     B00010000,
01748     B00010000,
01749     B00100000};

```

## 15.33 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/display.h File Reference

Libreria MUSOTOKU para display 15x7.

```

#include "bitmaps.h"
#include <Wire.h>
#include <batt_Adafruit_GFX.h>
#include <batt_Adafruit_IS31FL3731.h>

```

### Functions

- void [SwitchScreenOff \(Adafruit\\_IS31FL3731\\_Wing ledmatrix\)](#)  
*Apagado de la pantalla.*
- bool [DisplayText \(String cadena, Adafruit\\_IS31FL3731\\_Wing ledmatrix, bool reset=false, bool mode\\_bright=C\\_MODE\\_HIGH\\_BRIGHT\)](#)  
*Esta funcion muestra en el display una cadena de caracteres en un scroll lateral der-izq.*
- void [DisplayVolt \(int16\\_t number, Adafruit\\_IS31FL3731\\_Wing ledmatrix, bool mode\\_bright=C\\_MODE\\_HIGH\\_BRIGHT\)](#)  
*Esta funcion imprime la pantalla de voltaje, que muestra le voltaje al que se encuentra la salida de la bateria, con un decimal.*
- void [DisplayCap \(int16\\_t capacity, Adafruit\\_IS31FL3731\\_Wing ledmatrix, bool mode\\_bright=C\\_MODE\\_HIGH\\_BRIGHT\)](#)  
*Esta funcion muestra la pantalla de porcentaje de capacidad de bateria. si la capacidad es 100% muestra la pantalla FULL.*
- void [DisplayBattCharging \(int16\\_t capacity, Adafruit\\_IS31FL3731\\_Wing ledmatrix, bool mode\\_bright=C\\_MODE\\_HIGH\\_BRIGHT\)](#)  
*Esta funcion muestra el marco de una bateria, se encarga de ir rellenando e manera solida la capacidad recibida, y deja parpadeando la que seria el proximo 10% que se encuentra cargando.*
- void [LedWork \(bool state\\_led, Adafruit\\_IS31FL3731\\_Wing ledmatrix, bool mode\\_bright=C\\_MODE\\_HIGH\\_BRIGHT\)](#)  
*Funcion que permite encender y apagar el boton que representa el estado de la salida del voltage.*
- void [DisplayUsbIn \(Adafruit\\_IS31FL3731\\_Wing ledmatrix\)](#)  
*Animacion de Entrada del USB. A los 2 segundos disminuye su brillo.*
- void [DisplayUsbOut \(Adafruit\\_IS31FL3731\\_Wing ledmatrix\)](#)  
*Anuimacion de salida del USB.*
- void [DisplayDiagnosticMode \(Adafruit\\_IS31FL3731\\_Wing ledmatrix\)](#)  
*Animacion del modo Diagnostico.*
- bool [DisplayLowBattery \(Adafruit\\_IS31FL3731\\_Wing ledmatrix\)](#)  
*Animacion Bateria Baja.*
- void [DisplayError \(Adafruit\\_IS31FL3731\\_Wing ledmatrix\)](#)  
*Animacion de Error.*

- void [ScreenON](#) ([Adafruit\\_IS31FL3731\\_Wing ledmatrix](#), int16\_t bright)  
*Encender toda la pantalla con una cantidad de brillo.*

## Variables

- const int16\_t [C\\_TEXT\\_CHAR\\_HEIGHT](#) = 6
- const int16\_t [C\\_PWBAR\\_Y\\_AXE](#) = 6
- const int16\_t [C\\_TEXT\\_CHAR\\_OFFSET\\_Y](#) = 0
- const bool [C\\_MODE\\_HIGH\\_BRIGHT](#) = true
- const bool [C\\_MODE\\_LOW\\_BRIGHT](#) = false
- const int16\_t [C\\_HIGH\\_BRIGHTNESS](#) = 20
- const int16\_t [C\\_LOW\\_BRIGHTNESS](#) = 5
- int16\_t [C\\_DISPLAY\\_ON](#) = 0
- const int16\_t [C\\_DISPLAY\\_OFF](#) = 0
- const bool [C\\_DISPLAY\\_ST\\_BUSSY](#) = false
- const bool [C\\_DISPLAY\\_ST\\_NOT\\_BUSSY](#) = true
- const uint8\_t [C\\_NUMBER\\_BLINK\\_LOW\\_BATTERY](#) = 4
- const uint16\_t [C\\_DELAY\\_BLINK\\_LOW\\_BATTERY](#) = 300
- const int32\_t [C\\_TIMER\\_BTW\\_FRAMES](#) = 75
- const int16\_t [C\\_TEXT\\_BITMAP\\_WEIGHT](#) = 8
- const bool [blink\\_OFF](#) = false
- const bool [blink\\_ON](#) = true
- const int16\_t [C\\_PIXEL\\_START\\_STRING](#) = 7
- const int16\_t [C\\_PIXEL\\_STOP\\_STRING](#) = 0

### 15.33.1 Detailed Description

Libreria MUSOTOKU para display 15x7.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

#### Version

4

#### Date

2020-8-4

#### Copyright

Copyright (c) 2020

Definition in file [display.h](#).

### 15.33.2 Function Documentation

#### 15.33.2.1 DisplayBattCharging()

```
void DisplayBattCharging (
    int16_t capacity,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool mode_bright = C\_MODE\_HIGH\_BRIGHT )
```

Esta funcion muestra el marco de una bateria, se encarga de ir rellenando de manera solida la capacidad recibida, y deja parpadeando la que seria el proximo 10% que se encuentra cargando.

Parameters

<i>capacity</i>	
<i>ledmatrix</i>	
<i>mode_bright</i>	

Definition at line 311 of file [display.h](#).

### 15.33.2.2 DisplayCap()

```
void DisplayCap (
    int16_t capacity,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool mode_bright = C_MODE_HIGH_BRIGHT )
```

Esta funcion muestra la pantalla de porcentaje de capacidad de bateria. si la capacidad es 100% muestra la pantalla FULL.

Parameters

<i>capacity</i>	
<i>ledmatrix</i>	
<i>mode_bright</i>	

Definition at line 248 of file [display.h](#).

### 15.33.2.3 DisplayDiagnosticMode()

```
void DisplayDiagnosticMode (
    Adafruit_IS31FL3731_Wing ledmatrix )
```

Animacion del modo Diagnostico.

Definition at line 456 of file [display.h](#).

### 15.33.2.4 DisplayError()

```
void DisplayError (
    Adafruit_IS31FL3731_Wing ledmatrix )
```

Animacion de Error.

Definition at line 524 of file [display.h](#).

### 15.33.2.5 DisplayLowBattery()

```
bool DisplayLowBattery (
    Adafruit_IS31FL3731_Wing ledmatrix )
```

Animacion Bateria Baja.

Definition at line 476 of file [display.h](#).

### 15.33.2.6 DisplayText()

```
bool DisplayText (
    String cadena,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool reset = false,
    bool mode_bright = C_MODE_HIGH_BRIGHT )
```

Esta funcion muestra en el display una cadena de caracteres en un scroll lateral der-izq.

#### Parameters

<i>cadena</i>	<input type="text"/>
<i>ledmatrix</i>	<input type="text"/>
<i>reset</i>	<input type="text"/>
<i>mode_bright</i>	<input type="text"/>

#### Returns

true

false

Definition at line 72 of file [display.h](#).

### 15.33.2.7 DisplayUsbIn()

```
void DisplayUsbIn (
    Adafruit_IS31FL3731_Wing ledmatrix )
```

Animacion de Entrada del USB. A los 2 segundos disminuye su brillo.

#### Parameters

<i>ledmatrix</i>	<input type="text"/>
------------------	----------------------

Definition at line 391 of file [display.h](#).

### 15.33.2.8 DisplayUsbOut()

```
void DisplayUsbOut (
    Adafruit_IS31FL3731_Wing ledmatrix )
```

Anuimacion de salida del USB.

#### Parameters

<i>ledmatrix</i>	<input type="text"/>
------------------	----------------------

Definition at line 427 of file [display.h](#).

### 15.33.2.9 DisplayVolt()

```
void DisplayVolt (
    int16_t number,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool mode_bright = C_MODE_HIGH_BRIGHT )
```

Esta funcion imprime la pantalla de voltaje, que muestra le voltaje al que se encuentra la salida de la bateria, con un decimal.

#### Parameters

<i>number</i>	<input type="text"/>
<i>ledmatrix</i>	<input type="text"/>
<i>mode_bright</i>	<input type="text"/>

Definition at line 188 of file [display.h](#).

### 15.33.2.10 LedWork()

```
void LedWork (
    bool state_led,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool mode_bright = C_MODE_HIGH_BRIGHT )
```

Funcion que permite encender y apagar el boton que representa el estado de la salida del voltage.

#### Parameters

state_led	
ledmatrix	
mode_bright	

Definition at line 358 of file [display.h](#).

### 15.33.2.11 ScreenON()

```
void ScreenON (
    Adafruit_IS31FL3731_Wing ledmatrix,
    int16_t bright )
```

Encender toda la pantalla con una cantidad de brillo.

Definition at line 535 of file [display.h](#).

### 15.33.2.12 SwitchScreenOff()

```
void SwitchScreenOff (
    Adafruit_IS31FL3731_Wing ledmatrix )
```

Apagado de la pantalla.

Definition at line 58 of file [display.h](#).

## 15.33.3 Variable Documentation

### 15.33.3.1 blink\_OFF

```
const bool blink_OFF = false
```

Definition at line 49 of file [display.h](#).

### 15.33.3.2 blink\_ON

```
const bool blink_ON = true
```

Definition at line 50 of file [display.h](#).

### 15.33.3.3 C\_DELAY\_BLINK\_LOW\_BATTERY

```
const uint16_t C_DELAY_BLINK_LOW_BATTERY = 300
```

Definition at line 44 of file [display.h](#).

#### 15.33.3.4 C\_DISPLAY\_OFF

```
const int16_t C_DISPLAY_OFF = 0
Definition at line 38 of file display.h.
```

#### 15.33.3.5 C\_DISPLAY\_ON

```
int16_t C_DISPLAY_ON = 0
Definition at line 37 of file display.h.
```

#### 15.33.3.6 C\_DISPLAY\_ST\_BUSSY

```
const bool C_DISPLAY_ST_BUSSY = false
Definition at line 40 of file display.h.
```

#### 15.33.3.7 C\_DISPLAY\_ST\_NOT\_BUSSY

```
const bool C_DISPLAY_ST_NOT_BUSSY = true
Definition at line 41 of file display.h.
```

#### 15.33.3.8 C\_HIGH\_BRIGHTNESS

```
const int16_t C_HIGH_BRIGHTNESS = 20
Definition at line 35 of file display.h.
```

#### 15.33.3.9 C\_LOW\_BRIGHTNESS

```
const int16_t C_LOW_BRIGHTNESS = 5
Definition at line 36 of file display.h.
```

#### 15.33.3.10 C\_MODE\_HIGH\_BRIGHT

```
const bool C_MODE_HIGH_BRIGHT = true
Definition at line 32 of file display.h.
```

#### 15.33.3.11 C\_MODE\_LOW\_BRIGHT

```
const bool C_MODE_LOW_BRIGHT = false
Definition at line 33 of file display.h.
```

#### 15.33.3.12 C\_NUMBER\_BLINK\_LOW\_BATTERY

```
const uint8_t C_NUMBER_BLINK_LOW_BATTERY = 4
Definition at line 43 of file display.h.
```

#### 15.33.3.13 C\_PIXEL\_START\_STRING

```
const int16_t C_PIXEL_START_STRING = 7
Definition at line 52 of file display.h.
```

### 15.33.3.14 C\_PIXEL\_STOP\_STRING

```
const int16_t C_PIXEL_STOP_STRING = 0
Definition at line 53 of file display.h.
```

### 15.33.3.15 C\_PWBAR\_Y\_AXE

```
const int16_t C_PWBAR_Y_AXE = 6
Definition at line 25 of file display.h.
```

### 15.33.3.16 C\_TEXT\_BITMAP\_WEIGHT

```
const int16_t C_TEXT_BITMAP_WEIGHT = 8
Definition at line 47 of file display.h.
```

### 15.33.3.17 C\_TEXT\_CHAR\_HEIGHT

```
const int16_t C_TEXT_CHAR_HEIGHT = 6
Definition at line 21 of file display.h.
```

### 15.33.3.18 C\_TEXT\_CHAR\_OFFSET\_Y

```
const int16_t C_TEXT_CHAR_OFFSET_Y = 0
Definition at line 26 of file display.h.
```

### 15.33.3.19 C\_TIMER\_BTW\_FRAMES

```
const int32_t C_TIMER_BTW_FRAMES = 75
Definition at line 46 of file display.h.
```

## 15.34 display.h

[Go to the documentation of this file.](#)

```
00001
00011 #include "bitmaps.h"
00012 #include <Wire.h>
00013 #include <batt_Adafruit_GFX.h>
00014 #include <batt_Adafruit_IS31FL3731.h>
00015
00016 // #include <MilliTimer.h>
00017
00018 #if DISPLAY_SIZE == 7
00019 const int16_t C_TEXT_CHAR_HEIGHT = 7; // Altura del caracter.
00020 #elif DISPLAY_SIZE == 6
00021 const int16_t C_TEXT_CHAR_HEIGHT = 6; // Altura del caracter.
00022 #endif
00023
00024 #if POWERBAR_LOCATION == 1
00025 const int16_t C_PWBAR_Y_AXE = 6; // Posicion de la barra de potencia.
00026 const int16_t C_TEXT_CHAR_OFFSET_Y = 0; // Altura de inicio de escritura de los caracteres.
00027 #elif POWERBAR_LOCATION == 0
00028 const int16_t C_PWBAR_Y_AXE = 0; // Posicion de la barra de potencia.
00029 const int16_t C_TEXT_CHAR_OFFSET_Y = 1; // Altura de inicio de escritura de los caracteres.
00030 #endif
00031
00032 const bool C_MODE_HIGH_BRIGHT = true; //
00033 const bool C_MODE_LOW_BRIGHT = false; //
00034
00035 const int16_t C_HIGH_BRIGHTNESS = 20; // Nivel de brillo alto de los Leds.
00036 const int16_t C_LOW_BRIGHTNESS = 5; // Nivel de brillo bajo de los Leds.
00037 int16_t C_DISPLAY_ON = 0; // Encendido del led.
00038 const int16_t C_DISPLAY_OFF = 0; // Apagado del LED.
00039
00040 const bool C_DISPLAY_ST_BUSSY = false;
```

```

00041 const bool C_DISPLAY_ST_NOT_BUSSY = true;
00042
00043 const uint8_t C_NUMBER_BLINK_LOW_BATTERY = 4; // Numero de veces que parpadea la pantalla de Low
00044 const uint16_t C_DELAY_BLINK_LOW_BATTERY = 300; // Duracion del parpadeo del Low Battery.
00045
00046 const int32_t C_TIMER_BTW_FRAMES = 75; // Tiempo de refresco de la pantalla.
00047 const int16_t C_TEXT_BITMAP_WEIGHT = 8; // Tamaño del bitmap.
00048
00049 const bool blink_OFF = false;
00050 const bool blink_ON = true;
00051
00052 const int16_t C_PIXEL_START_STRING = 7; // Pixel en que empieza la rotacion de la cadena.
00053 const int16_t C_PIXEL_STOP_STRING = 0; // Pixel en que termina la rotacion de la cadena.
00058 void SwitchScreenOff(Adafruit_IS31FL3731_Wing ledmatrix)
00059 {
00060     ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00061                         C_DISPLAY_OFF);
00061 }
00072 bool DisplayText(String cadena, Adafruit_IS31FL3731_Wing ledmatrix, bool reset = false, bool
00073 mode_bright = C_MODE_HIGH_BRIGHT)
00073 {
00074     uint8_t bitmap_char[] = {B00000000,
00075                             B00000000,
00076                             B00000000,
00077                             B00000000,
00078                             B00000000,
00079                             B00000000,
00080                             B00000000};
00081
00082     static bool display_status = C_DISPLAY_ST_NOT_BUSSY;
00083     static int16_t length_bit = 0; // Tamaño del bitmap de la cadena.
00084     char char_of_string;
00085     static int16_t pos_x_display = 0;
00086     static int16_t x_axis_index = 0;
00087     static String cadena_local;
00088     static MilliTimer frame_delay_timer; // Timer de refresco de la pantalla.
00089     if (mode_bright == C_MODE_HIGH_BRIGHT)
00090     {
00091         C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00092     }
00093     else if (mode_bright == C_MODE_LOW_BRIGHT)
00094     {
00095         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00096     }
00097     if ((display_status == C_DISPLAY_ST_NOT_BUSSY) || (reset == true))
00098     {
00099         length_bit = 0;
00100         cadena_local = cadena;
00101         for (uint16_t i = 0; i < cadena_local.length(); i++)
00102         {
00103             char_of_string = toupper(cadena_local.charAt(i));
00104             if (char_of_string >= 32 && char_of_string <= 126)
00105             {
00106                 length_bit += width_bitmap[char_of_string - 32];
00107             }
00108         }
00109         display_status = C_DISPLAY_ST_BUSSY;
00110         if (length_bit <= 15)
00111         {
00112             x_axis_index = (15 - length_bit + 1) / 2;
00113         }
00114         else
00115         {
00116             x_axis_index = C_PIXEL_START_STRING;
00117             frame_delay_timer.set(C_TIMER_BTW_FRAMES);
00118         }
00119     }
00120     else if (display_status == C_DISPLAY_ST_BUSSY)
00121     {
00122         if (length_bit <= 15)
00123         {
00124             ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00125                         C_DISPLAY_OFF);
00126             pos_x_display = x_axis_index;
00127             for (uint16_t i = 0; i < cadena_local.length(); i++)
00128             {
00129                 char_of_string = toupper(cadena_local.charAt(i));
00130                 if (char_of_string >= 32 && char_of_string <= 126)
00131                 {
00132                     Char2Bitmap(char_of_string, bitmap_char, sizeof(bitmap_char));
00133                     ledmatrix.drawBitmap(pos_x_display, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON,
00134                                         C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00135                     pos_x_display += width_bitmap[char_of_string - 32];
00136                 }
00137             }
00138         }
00139     }
}

```

```

00137             Char2Bitmap(' ', bitmap_char, sizeof(bitmap_char));
00138             ledmatrix.drawBitmap(pos_x_display, C_TEXT_CHAR_OFFSET_Y, bitmap_char,
00139             C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00140             pos_x_display += width_bitmap[' ' - 32];
00141         }
00142     }
00143     display_status = C_DISPLAY_ST_NOT_BUSSY;
00144 } else
00145 {
00146     if (frame_delay_timer.poll())
00147     {
00148         ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00149         C_DISPLAY_OFF);
00150         pos_x_display = x_axis_index;
00151         for (uint16_t i = 0; i < cadena_local.length(); i++)
00152         {
00153             char_of_string = toupper(cadena_local.charAt(i));
00154             if (char_of_string >= 32 && char_of_string <= 126)
00155             {
00156                 Char2Bitmap(char_of_string, bitmap_char, sizeof(bitmap_char));
00157                 ledmatrix.drawBitmap(pos_x_display, C_TEXT_CHAR_OFFSET_Y, bitmap_char,
00158                 C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00159                 pos_x_display += width_bitmap[char_of_string - 32];
00160             }
00161             else
00162             {
00163                 Char2Bitmap(' ', bitmap_char, sizeof(bitmap_char));
00164                 ledmatrix.drawBitmap(pos_x_display, C_TEXT_CHAR_OFFSET_Y, bitmap_char,
00165                 C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00166                 pos_x_display += width_bitmap[' ' - 32];
00167             }
00168             if (x_axis_index == -length_bit + C_PIXEL_STOP_STRING)
00169             {
00170                 display_status = C_DISPLAY_ST_NOT_BUSSY;
00171             }
00172             else
00173             {
00174                 x_axis_index--;
00175                 frame_delay_timer.set(C_TIMER_BTW_FRAMES);
00176             }
00177         }
00178     }
00179     return display_status;
00180 }
00181
00182 void DisplayVolt(int16_t number, Adafruit_IS31FL3731_Wing ledmatrix, bool mode_bright =
00183 C_MODE_HIGH_BRIGHT)
00184 {
00185     uint8_t dec_number[] = {B00000000,
00186                             B00000000,
00187                             B00000000,
00188                             B00000000,
00189                             B00000000,
00190                             B00000000,
00191                             B00000000};
00192     uint8_t cent_number[] = {B00000000,
00193                             B00000000,
00194                             B00000000,
00195                             B00000000,
00196                             B00000000,
00197                             B00000000};
00198     uint8_t unit_number[] = {B00000000,
00199                             B00000000,
00200                             B00000000,
00201                             B00000000,
00202                             B00000000,
00203                             B00000000,
00204                             B00000000};
00205     String str_number = String(number);
00206     if (str_number.length() == 2)
00207     {
00208         Char2Bitmap(' ', cent_number, sizeof(cent_number));
00209         Char2Bitmap(str_number.charAt(0), dec_number, sizeof(dec_number));
00210         Char2Bitmap(str_number.charAt(1), unit_number, sizeof(unit_number));
00211     }
00212     else if (str_number.length() == 3)
00213     {
00214         Char2Bitmap(str_number.charAt(0), cent_number, sizeof(cent_number));
00215         Char2Bitmap(str_number.charAt(1), dec_number, sizeof(dec_number));
00216         Char2Bitmap(str_number.charAt(2), unit_number, sizeof(unit_number));
00217     }
00218     if (mode_bright == C_MODE_HIGH_BRIGHT)
00219 
```

```

00226     {
00227         C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00228     }
00229     else if (mode_bright == C_MODE_LOW_BRIGHT)
00230     {
00231         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00232     }
00233
00234     ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00235     C_DISPLAY_OFF);
00236     ledmatrix.drawBitmap(1, C_TEXT_CHAR_OFFSET_Y, cent_number, C_TEXT_BITMAP_WEIGHT,
00237     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00238     ledmatrix.drawBitmap(4, C_TEXT_CHAR_OFFSET_Y, dec_number, C_TEXT_BITMAP_WEIGHT,
00239     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00240     ledmatrix.drawPixel(9, C_TEXT_CHAR_OFFSET_Y + 5, C_DISPLAY_ON);
00241     ledmatrix.drawBitmap(11, C_TEXT_CHAR_OFFSET_Y, unit_number, C_TEXT_BITMAP_WEIGHT,
00242     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00243 }
00244
00245 void DisplayCap(int16_t capacity, Adafruit_IS31FL3731_Wing ledmatrix, bool mode_bright =
00246 C_MODE_HIGH_BRIGHT)
00247 {
00248     uint8_t dec_number[] = {B00000000,
00249     B00000000,
00250     B00000000,
00251     B00000000,
00252     B00000000,
00253     B00000000,
00254     B00000000,
00255     B00000000,
00256     B00000000};
00257     uint8_t unit_number[] = {B00000000,
00258     B00000000,
00259     B00000000,
00260     B00000000,
00261     B00000000,
00262     B00000000,
00263     B00000000};
00264
00265     capacity = constrain(capacity, 0, 100);
00266     if (mode_bright == C_MODE_HIGH_BRIGHT)
00267     {
00268         C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00269     }
00270     else if (mode_bright == C_MODE_LOW_BRIGHT)
00271     {
00272         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00273     }
00274
00275     String str_capacity = String(capacity);
00276     if (capacity == 100)
00277     {
00278         ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00279     C_DISPLAY_OFF);
00280         ledmatrix.drawBitmap(1, C_TEXT_CHAR_OFFSET_Y, Fchar, C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT,
00281     C_DISPLAY_ON);
00282         ledmatrix.drawBitmap(5, C_TEXT_CHAR_OFFSET_Y, Uchar, C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT,
00283     C_DISPLAY_ON);
00284         ledmatrix.drawBitmap(9, C_TEXT_CHAR_OFFSET_Y, Lchar, C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT,
00285     C_DISPLAY_ON);
00286         ledmatrix.drawBitmap(12, C_TEXT_CHAR_OFFSET_Y, Lchar, C_TEXT_BITMAP_WEIGHT,
00287     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00288     }
00289     else
00290     {
00291         if (str_capacity.length() == 1)
00292         {
00293             Char2Bitmap(' ', dec_number, sizeof(dec_number));
00294             Char2Bitmap(str_capacity.charAt(0), unit_number, sizeof(unit_number));
00295         }
00296         else if (str_capacity.length() == 2)
00297         {
00298             Char2Bitmap(str_capacity.charAt(0), dec_number, sizeof(dec_number));
00299             Char2Bitmap(str_capacity.charAt(1), unit_number, sizeof(unit_number));
00300         }
00301
00302     }
00303
00304     ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00305     C_DISPLAY_OFF);
00306     ledmatrix.drawBitmap(11, C_TEXT_CHAR_OFFSET_Y, PERCENTchar, C_TEXT_BITMAP_WEIGHT,
00307     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00308     ledmatrix.drawBitmap(1, C_TEXT_CHAR_OFFSET_Y, dec_number, C_TEXT_BITMAP_WEIGHT,
00309     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00310     ledmatrix.drawBitmap(6, C_TEXT_CHAR_OFFSET_Y, unit_number, C_TEXT_BITMAP_WEIGHT,
00311     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00312 }
00313
00314 void DisplayBattCharging(int16_t capacity, Adafruit_IS31FL3731_Wing ledmatrix, bool mode_bright =
00315 C_MODE_HIGH_BRIGHT)

```

```
00312 {
00313     capacity = constrain(capacity, 10, 100);
00314     int16_t fill_capacity_column;
00315     static MilliTimer timer_blinking_column;
00316     int16_t blinking_column = 0;
00317     static bool blink_state = blink_OFF;
00318     if (mode_bright == C_MODE_HIGH_BRIGHT)
00319     {
00320         C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00321     }
00322     else if (mode_bright == C_MODE_LOW_BRIGHT)
00323     {
00324         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00325     }
00326
00327     ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, Frame_Batt_1, C_TEXT_BITMAP_WEIGHT,
00328                           C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00329     ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, Frame_Batt_2, C_TEXT_BITMAP_WEIGHT,
00330                           C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00331
00332     fill_capacity_column = capacity / 10;
00333
00334     ledmatrix.fillRect(1, C_TEXT_CHAR_OFFSET_Y + 1, fill_capacity_column, C_TEXT_CHAR_OFFSET_Y + 5,
00335                        C_DISPLAY_ON);
00336
00337     blinking_column = fill_capacity_column + 1;
00338
00339     if (timer_blinking_column.poll(500))
00340     {
00341         if (blink_state == blink_ON)
00342         {
00343             ledmatrix.writeFastVLine(blinking_column, C_TEXT_CHAR_OFFSET_Y + 1, DISPLAY_SIZE - 2,
00344                                       C_DISPLAY_ON);
00345             blink_state = blink_OFF;
00346         }
00347         else if (blink_state == blink_OFF)
00348         {
00349             ledmatrix.writeFastVLine(blinking_column, C_TEXT_CHAR_OFFSET_Y + 1, DISPLAY_SIZE - 2,
00350                                       C_DISPLAY_OFF);
00351             blink_state = blink_ON;
00352         }
00353     }
00354 }
00355
00356 void LedWork(bool state_led, Adafruit_IS31FL3731_Wing ledmatrix, bool mode_bright =
00357             C_MODE_HIGH_BRIGHT)
00358 {
00359     uint16_t x_LED_WORK = 2;
00360     uint16_t y_LED_WORK = C_PWBAR_Y_AXE;
00361     if (mode_bright == C_MODE_HIGH_BRIGHT)
00362     {
00363         C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00364     }
00365     else if (mode_bright == C_MODE_LOW_BRIGHT)
00366     {
00367         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00368     }
00369
00370     if (state_led == true)
00371     {
00372         for (int i = 0; i < x_LED_WORK; i++)
00373         {
00374             ledmatrix.drawPixel(i, C_PWBAR_Y_AXE, C_DISPLAY_ON);
00375         }
00376     }
00377     else if (state_led == false)
00378     {
00379         for (int i = 0; i < x_LED_WORK; i++)
00380         {
00381             ledmatrix.drawPixel(i, C_PWBAR_Y_AXE, C_DISPLAY_OFF);
00382         }
00383     }
00384 }
00385
00386 void DisplayUsbIn(Adafruit_IS31FL3731_Wing ledmatrix)
00387 {
00388     bool flag_initialize = true;
00389     bool flag_set_low_bright = false;
00390     int16_t index = 15;
00391
00392     MilliTimer timer_set_bright_low;
00393     MilliTimer timer_frames_spacer;
00394     while (flag_initialize == true)
00395     {
00396         if (timer_frames_spacer.poll(75) != C_TIMER_NOT_EXPIRED)
00397         {
00398             C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00399             index--;
00400         }
00401     }
00402 }
```

```

00405     ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_DISPLAY_HEIGHT + 1,
00406     C_DISPLAY_OFF);
00407     ledmatrix.drawBitmap(8 - index, C_TEXT_CHAR_OFFSET_Y, Frame_USB_2, C_TEXT_BITMAP_WEIGHT,
00408     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00409     ledmatrix.drawBitmap(0 - index, C_TEXT_CHAR_OFFSET_Y, Frame_USB_1, C_TEXT_BITMAP_WEIGHT,
00410     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00411     if (index == 0)
00412     {
00413         ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, Frame_USB_2, C_TEXT_BITMAP_WEIGHT,
00414     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00415         ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, Frame_USB_1, C_TEXT_BITMAP_WEIGHT,
00416     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00417         delay(2000);
00418         flag_initialize = false;
00419         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00420         ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, Frame_USB_2, C_TEXT_BITMAP_WEIGHT,
00421     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00422         ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, Frame_USB_1, C_TEXT_BITMAP_WEIGHT,
00423     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00424         index = 11;
00425     }
00426 }
00427 void DisplayUsbOut(Adafruit_IS31FL3731_Wing ledmatrix)
00428 {
00429     bool flag_initialize = true;
00430     int16_t index = 0;
00431
00432     MilliTimer timer_frames_spacer;
00433
00434     while (flag_initialize == true)
00435     {
00436         if (timer_frames_spacer.poll(50) != C_TIMER_NOT_EXPIRED)
00437         {
00438             C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00439             index++;
00440             ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00441     C_DISPLAY_OFF);
00442             ledmatrix.drawBitmap(8 - index, C_TEXT_CHAR_OFFSET_Y, Frame_USB_2, C_TEXT_BITMAP_WEIGHT,
00443     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00444             ledmatrix.drawBitmap(0 - index, C_TEXT_CHAR_OFFSET_Y, Frame_USB_1, C_TEXT_BITMAP_WEIGHT,
00445     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00446             if (index == 15)
00447             {
00448                 flag_initialize = false;
00449                 index = 11;
00450             }
00451 }
00452 }
00453 void DisplayDiagnosticMode(Adafruit_IS31FL3731_Wing ledmatrix)
00454 {
00455     MilliTimer timer_fr_diagnostic;
00456     int i = 0;
00457     ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6,
00458     C_DISPLAY_OFF);
00459     while (i <= 30)
00460     {
00461         if (timer_fr_diagnostic.poll(200))
00462         {
00463             ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, FACE_2, C_TEXT_BITMAP_WEIGHT,
00464     C_TEXT_CHAR_HEIGHT, i);
00465             ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, FACE_1, C_TEXT_BITMAP_WEIGHT,
00466     C_TEXT_CHAR_HEIGHT, i);
00467             i += 6;
00468         }
00469     }
00470 }
00471 bool DisplayLowBattery(Adafruit_IS31FL3731_Wing ledmatrix)
00472 {
00473     static MilliTimer timer_blink;
00474     static int i = 0;
00475     static bool display_status = C_DISPLAY_ST_NOT_BUSSY;
00476     static bool blink_state = blink_ON;
00477     if (display_status == C_DISPLAY_ST_NOT_BUSSY)
00478     {
00479         SwitchScreenOff(ledmatrix);
00480         ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, Frame_Batt_Low_1, C_TEXT_BITMAP_WEIGHT,
00481     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00482         ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, Frame_Batt_Low_2, C_TEXT_BITMAP_WEIGHT,
00483     C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00484         timer_blink.set(C_DELAY_BLINK_LOW_BATTERY);
00485         display_status = C_DISPLAY_ST_BUSSY;
00486         i = 0;
00487     }
00488 }
```

```

00491     else if (display_status == C_DISPLAY_ST_BUSSY)
00492     {
00493         if (timer_blink.poll() != C_TIMER_NOT_EXPIRED)
00494         {
00495             if (blink_state == blink_OFF)
00496             {
00497                 i++;
00498                 if (i == C_NUMBER_BLINK_LOW_BATTERY)
00499                 {
00500                     display_status = C_DISPLAY_ST_NOT_BUSSY;
00501                 }
00502                 else
00503                 {
00504                     blink_state = blink_ON;
00505                     ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, Frame_Batt_Low_1,
00506 C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00507                     ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, Frame_Batt_Low_2,
00508 C_TEXT_BITMAP_WEIGHT, C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00509                     timer_blink.set(C_DELAY_BLINK_LOW_BATTERY);
00510                 }
00511             else if (blink_state == blink_ON)
00512             {
00513                 blink_state = blink_OFF;
00514                 SwitchScreenOff(ledmatrix);
00515                 timer_blink.set(C_DELAY_BLINK_LOW_BATTERY);
00516             }
00517         }
00518         return display_status;
00519     }
00524 void DisplayError(Adafruit_IS31FL3731_Wing ledmatrix)
00525 {
00526     C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00527     ledmatrix.drawBitmap(0, C_TEXT_CHAR_OFFSET_Y, Frame_Error_1, C_TEXT_BITMAP_WEIGHT,
00528 C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00529     ledmatrix.drawBitmap(8, C_TEXT_CHAR_OFFSET_Y, Frame_Error_2, C_TEXT_BITMAP_WEIGHT,
00530 C_TEXT_CHAR_HEIGHT, C_DISPLAY_ON);
00535 void ScreenON(Adafruit_IS31FL3731_Wing ledmatrix, int16_t bright)
00536 {
00537     ledmatrix.fillRect(0, C_TEXT_CHAR_OFFSET_Y, C_DISPLAY_WIDTH, C_TEXT_CHAR_OFFSET_Y + 6, bright);
00538 }

```

## 15.35 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_1/TEST\_1.ino File Reference

Test de la funcion DisplayVolts.

```
#include <display.h>
```

### Functions

- void `setup ()`
- void `loop ()`

### Variables

- Adafruit\_IS31FL3731\_Wing ledmatrix = Adafruit\_IS31FL3731\_Wing()

#### 15.35.1 Detailed Description

Test de la funcion DisplayVolts.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

**Version**

3

**Date**

2021-03-16

**Copyright**

Copyright (c) 2020

Definition in file [TEST\\_1.ino](#).

## 15.35.2 Function Documentation

### 15.35.2.1 loop()

```
void loop ()
```

Definition at line 47 of file [TEST\\_1.ino](#).

### 15.35.2.2 setup()

```
void setup ()
```

Definition at line 15 of file [TEST\\_1.ino](#).

## 15.35.3 Variable Documentation

### 15.35.3.1 ledmatrix

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()
```

Definition at line 13 of file [TEST\\_1.ino](#).

## 15.36 TEST\_1.ino

[Go to the documentation of this file.](#)

```
00001
00012 #include <display.h>
00013 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00014
00015 void setup()
00016 {
00017     Serial.begin(9600);
00018     while (!Serial)
00019     {
00020         ;
00021     }
00022     ledmatrix.begin();
00023
00024     Serial.println("IS31 Found!");
00025     int32_t t0 = 0;
00026     int32_t t1 = 0;
00027     for (int16_t i = 40; i <= 160; i++)
00028     {
00029         t0 = micros();
00030         DisplayVolt(i, ledmatrix);
00031         t1 = micros();
00032         Serial.print("Tiempo de ejecucion DisplayVolt:");
00033         Serial.println(t1 - t0);
00034         delay(100);
00035     }
00036     for (int16_t i = 160; i >= 40; i--)
00037     {
00038         t0 = micros();
```

```
00039     DisplayVolt(i, ledmatrix);
00040     t1 = micros();
00041     Serial.print("Tiempo de ejecucion DisplayVolt:");
00042     Serial.println(t1 - t0);
00043     delay(100);
00044 }
00045 }
00046
00047 void loop()
00048 {
00049 }
```

## 15.37 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0004-DCDC/exmples/TEST\_1/TEST\_1.ino File Reference

```
#include <DCDC.h>
```

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)

### Variables

- const uint16\_t [C\\_PIN\\_EN\\_DCDC](#) = 2
- [dcdc\\_controler DCDC \(C\\_PIN\\_EN\\_DCDC\)](#)

#### 15.37.1 Detailed Description

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

0.1

##### Date

2021-09-03

##### Copyright

Copyright (c) 2021

Definition in file [TEST\\_1.ino](#).

#### 15.37.2 Function Documentation

##### 15.37.2.1 [loop\(\)](#)

```
void loop ()  
Definition at line 19 of file TEST\_1.ino.
```

### 15.37.2.2 `setup()`

```
void setup ( )
```

Definition at line 14 of file [TEST\\_1.ino](#).

## 15.37.3 Variable Documentation

### 15.37.3.1 `C_PIN_EN_DCDC`

```
const uint16_t C_PIN_EN_DCDC = 2
```

Definition at line 12 of file [TEST\\_1.ino](#).

### 15.37.3.2 `DCDC`

```
dcdc_controller DCDC(C_PIN_EN_DCDC) (
    C_PIN_EN_DCDC )
```

## 15.38 TEST\_1.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <DCDC.h>
00012 const uint16_t C_PIN_EN_DCDC = 2;
00013 dcdc_controller DCDC(C_PIN_EN_DCDC);
00014 void setup()
00015 {
00016     Wire.begin();
00017 }
00018
00019 void loop()
00020 {
00021     DCDC.SetVoltage(100, C_NON_BOOST_MODE);
00022     delay(500);
00023 }
```

## 15.39 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0010-LOGGING/examples/TEST\_1/TEST\_1.ino File Reference

Testeo de la funcionalidad de guardado en EEMPROM tanto Chasis como Batt Pack. Prueba de las funciones `isValid()`, `Incrementeent DiagnosticData`.

```
#include <diagnostic.h>
```

## Functions

- `void setup ()`
- `void loop ()`

### 15.39.1 Detailed Description

Testeo de la funcionalidad de guardado en EEMPROM tanto Chasis como Batt Pack. Prueba de las funciones `isValid()`, `Incrementeent DiagnosticData`.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com) )

Version

1

Date

2021-03-22

Copyright

Copyright (c) 2021

Definition in file [TEST\\_1.ino](#).

## 15.39.2 Function Documentation

### 15.39.2.1 loop()

```
void loop ()
```

Definition at line [50](#) of file [TEST\\_1.ino](#).

### 15.39.2.2 setup()

```
void setup ()
```

Definition at line [12](#) of file [TEST\\_1.ino](#).

## 15.40 TEST\_1.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <diagnostic.h>
00012 void setup()
00013 {
00014     Serial.begin(9600);
00015     while (!Serial)
00016     {
00017         /* code */
00018     }
00019     Init_diagnostic_elements();
00020     if (isValid() == true)
00021     {
00022         Serial.println("Valid!");
00023         for (int i = 0; i < C_NUM_ELEMENTS; i++)
00024         {
00025             Serial.print(ReadDiagnosticData(i));
00026             Serial.println(";");
00027         }
00028     }
00029     else
00030     {
00031         Serial.println("Not Valid");
00032         for (int i = 0; i < C_NUM_ELEMENTS; i++)
00033         {
00034             LogDiagnosticData(i, i);
00035         }
00036         for (int i = 0; i < 5; i++)
00037         {
00038             IncrementDiagnosticData(i * 10, 3 + i);
00039         }
00040         SaveEepromChasis();
00041         UpdateEepromBatory();
00042         for (int i = 0; i < C_NUM_ELEMENTS; i++)
00043         {
00044             Serial.print(ReadDiagnosticData(i));
00045             Serial.println(";");
00046         }
00047     }
00048 }
00049
00050 void loop()
00051 {
00052 }
```

## 15.41 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_2/TEST\_2.ino File Reference

Este test simula el funcionamiento de la barra de potencia creando un movimiento senoidal de manera ciclica.

```
#include <display.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

### Variables

- Adafruit\_IS31FL3731\_Wing ledmatrix = Adafruit\_IS31FL3731\_Wing()

#### 15.41.1 Detailed Description

Este test simula el funcionamiento de la barra de potencia creando un movimiento senoidal de manera ciclica.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

2

##### Date

2021-03-16

##### Copyright

Copyright (c) 2020

Definition in file [TEST\\_2.ino](#).

#### 15.41.2 Function Documentation

##### 15.41.2.1 loop()

```
void loop ()
```

Definition at line [44](#) of file [TEST\\_2.ino](#).

##### 15.41.2.2 setup()

```
void setup ()
```

Definition at line [16](#) of file [TEST\\_2.ino](#).

#### 15.41.3 Variable Documentation

### 15.41.3.1 ledmatrix

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()
Definition at line 14 of file TEST_2.ino.
```

## 15.42 TEST\_2.ino

[Go to the documentation of this file.](#)

```
00012 #include <display.h>
00013
00014 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00015
00016 void setup()
00017 {
00018     Serial.begin(9600);
00019     while (!Serial)
00020     {
00021         ;
00022     }
00023
00024     ledmatrix.begin();
00025
00026     int x;
00027     int32_t t0 = 0;
00028     int32_t t1 = 0;
00029     while (1)
00030     {
00031         for (int16_t i = 0; i <= 360; i++)
00032         {
00033             x = abs(sin(radians(i))) * 14;
00034             t0 = micros();
00035             PowerBar(x, ledmatrix);
00036             t1 = micros();
00037             Serial.print("Tiempo de ejecucion PowerBar:");
00038             Serial.println(t1 - t0);
00039             delay(2);
00040         }
00041     }
00042 }
00043
00044 void loop()
00045 {
00046 }
```

## 15.43 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0010-LOGGING/examples/TEST\_2/TEST\_2.ino File Reference

Test del guardado y lectura de la eeprom chasis. Para ello se utiliza y prueba la funcion de estadisticas.

```
#include <diagnostic.h>
```

### Functions

- void **setup** ()
- void **loop** ()

### 15.43.1 Detailed Description

Test del guardado y lectura de la eeprom chasis. Para ello se utiliza y prueba la funcion de estadisticas.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

**Version**

1

**Date**

2021-03-22

**Copyright**

Copyright (c) 2021

Definition in file [TEST\\_2.ino](#).

## 15.43.2 Function Documentation

### 15.43.2.1 loop()

void loop ()

Definition at line [39](#) of file [TEST\\_2.ino](#).

### 15.43.2.2 setup()

void setup ()

Definition at line [12](#) of file [TEST\\_2.ino](#).

## 15.44 TEST\_2.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include <diagnostic.h>
00012 void setup()
00013 {
00014     Serial.begin(9600);
00015     while (!Serial)
00016     {
00017         /* code */
00018     }
00019     uint16_t voltage = 0;
00020     uint16_t power = 0;
00021     Init_diagnostic_elements();
00022     for (int i = 0; i < 30; i++)
00023     {
00024         voltage = random(5000, 16000);
00025         power = random(500, 5000);
00026         Serial.print(voltage);
00027         Serial.print(";");
00028         Serial.println(power);
00029         Stats(voltage, power);
00030     }
00031     PrintStats();
00032     Serial.println();
00033     int t0 = millis();
00034     UpdateEepromBatery();
00035     int t1 = millis();
00036     Serial.println(t1 - t0);
00037 }
00038
00039 void loop()
00040 {
00041 }
```

## 15.45 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_3/TEST\_3.ino File Reference

Este test pone a prueba la compatibilidad y convivencia de la barra de potencia y el display de informacion, en este caso mostrando cambios de voltaje.

```
#include <MilliTimer.h>
#include <display.h>
#include <power_bar.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

### Variables

- Adafruit\_IS31FL3731\_Wing ledmatrix = Adafruit\_IS31FL3731\_Wing()

#### 15.45.1 Detailed Description

Este test pone a prueba la compatibilidad y convivencia de la barra de potencia y el display de informacion, en este caso mostrando cambios de voltaje.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

2

##### Date

2021-03-16

##### Copyright

Copyright (c) 2020

Definition in file [TEST\\_3.ino](#).

#### 15.45.2 Function Documentation

##### 15.45.2.1 [loop\(\)](#)

```
void loop ()
```

Definition at line [65](#) of file [TEST\\_3.ino](#).

##### 15.45.2.2 [setup\(\)](#)

```
void setup ()
```

Definition at line [17](#) of file [TEST\\_3.ino](#).

### 15.45.3 Variable Documentation

#### 15.45.3.1 ledmatrix

`Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()`  
 Definition at line 15 of file [TEST\\_3.ino](#).

## 15.46 TEST\_3.ino

[Go to the documentation of this file.](#)

```
00001
00012 #include <MilliTimer.h>
00013 #include <display.h>
00014 #include <power_bar.h>
00015 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00016
00017 void setup()
00018 {
00019     // Serial.begin(9600);
00020     // while (!Serial)
00021     // {
00022     //     ;
00023     // }
00024     ledmatrix.begin();
00025
00026     int x, j = 0;
00027     MilliTimer timer;
00028     while (1)
00029     {
00030         for (int16_t i = 40; i <= 160; i++)
00031         {
00032             DisplayVolt(i, ledmatrix);
00033             if (timer.poll(10))
00034             {
00035                 j++;
00036                 if (j == 360)
00037                 {
00038                     j = 0;
00039                 }
00040                 x = abs(sin(radians(j++))) * 16;
00041                 Serial.println(x);
00042                 PowerBar(x, ledmatrix);
00043                 delay(2);
00044             }
00045         }
00046         for (int16_t i = 160; i >= 40; i--)
00047         {
00048             DisplayVolt(i, ledmatrix);
00049             if (timer.poll(10))
00050             {
00051                 j++;
00052                 if (j == 360)
00053                 {
00054                     j = 0;
00055                 }
00056                 x = abs(sin(radians(j++))) * 16;
00057                 Serial.println(x);
00058                 PowerBar(x, ledmatrix);
00059                 delay(2);
00060             }
00061         }
00062     }
00063 }
00064
00065 void loop()
00066 {
00067 }
```

## 15.47 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0010-LOGGING/examples/TEST\_3/TEST\_3.ino File Reference

Test que mide los tiempos de lectura y escritura de distintas cantidades de bytes en la memoria EEPROM del Battery Pack.

```
#include <diagnostic.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

#### 15.47.1 Detailed Description

Test que mide los tiempos de lectura y escritura de distintas cantidades de bytes en la memoria EEPROM del Battery Pack.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

0.1

##### Date

2021-02-17

##### Copyright

Copyright (c) 2021

Definition in file [TEST\\_3.ino](#).

#### 15.47.2 Function Documentation

##### 15.47.2.1 [loop\(\)](#)

```
void loop ()
```

Definition at line [81](#) of file [TEST\\_3.ino](#).

##### 15.47.2.2 [setup\(\)](#)

```
void setup ()
```

Definition at line [13](#) of file [TEST\\_3.ino](#).

## 15.48 TEST\_3.ino

[Go to the documentation of this file.](#)

```
00001
00012 #include <diagnostic.h>
00013 void setup()
```

```

00014 {
00015     Serial.begin(9600);
00016     while (!Serial)
00017     {
00018         /* code */
00019     }
00020     Init_diagnostic_elements();
00021     uint16_t init_address = 0x1000;
00022     uint8_t data_eeprom;
00023     uint32_t t0 = 0, t1 = 0;
00024     Serial.print("Tiempo de escritura de 1 byte: ");
00025     t0 = micros();
00026     writeEEPROM(init_address, 0);
00027     t1 = micros();
00028     Serial.print(t1 - t0);
00029     Serial.println(" us");
00030     Serial.print("Tiempo de lectura de 1 byte: ");
00031     t0 = micros();
00032     readEEPROM(init_address, &data_eeprom);
00033     t1 = micros();
00034     Serial.print(t1 - t0);
00035     Serial.println(" us");
00036 //Serial.println(data_eeprom);

00037     Serial.print("Tiempo de escritura de 10 bytes: ");
00038     t0 = micros();
00039     for (int i = 1; i < 11; i++)
00040     {
00041         writeEEPROM(init_address + i, i);
00042     }
00043     t1 = micros();
00044     Serial.print(t1 - t0);
00045     Serial.println(" us");
00046     Serial.print("Tiempo de lectura de 10 bytes: ");
00047     t0 = micros();
00048     for (int i = 1; i < 11; i++)
00049     {
00050         readEEPROM(init_address + i, &data_eeprom);
00051         //Serial.print(data_eeprom);
00052         //Serial.print(";");
00053     }
00054     t1 = micros();
00055     Serial.print(t1 - t0);
00056     Serial.println(" us");

00057     Serial.print("Tiempo de escritura de 20 bytes: ");
00058     t0 = micros();
00059     for (int i = 11; i < 30; i++)
00060     {
00061         writeEEPROM(init_address + i, i);
00062     }
00063     t1 = micros();
00064     Serial.print(t1 - t0);
00065     Serial.println(" us");
00066     Serial.print("Tiempo de lectura de 20 bytes: ");
00067     t0 = micros();
00068     for (int i = 11; i < 31; i++)
00069     {
00070         readEEPROM(init_address + i, &data_eeprom);
00071         //Serial.print(data_eeprom);
00072         //Serial.print(";");
00073     }
00074     t1 = micros();
00075     Serial.print(t1 - t0);
00076     Serial.println(" us");
00077 }
00078 }

00081 void loop()
00082 {
00083 }

```

## 15.49 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_4/TEST\_← 4.ino File Reference

Este test imprime toda la capacidad posible de la bateria ( 0-100 ). Ascendente y descendente, de manera ciclica.  
`#include <display.h>`

## Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

## Variables

- Adafruit\_IS31FL3731\_Wing ledmatrix = Adafruit\_IS31FL3731\_Wing()

### 15.49.1 Detailed Description

Este test imprime toda la capacidad posible de la bateria ( 0-100 ). Ascendente y descendente, de manera ciclica.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

#### Version

2

#### Date

2021-03-16

#### Copyright

Copyright (c) 2020

Definition in file [TEST\\_4.ino](#).

### 15.49.2 Function Documentation

#### 15.49.2.1 loop()

```
void loop ()  
Definition at line 53 of file TEST\_4.ino.
```

#### 15.49.2.2 setup()

```
void setup ()  
Definition at line 17 of file TEST\_4.ino.
```

### 15.49.3 Variable Documentation

#### 15.49.3.1 ledmatrix

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()  
Definition at line 15 of file TEST\_4.ino.
```

## 15.50 TEST\_4.ino

[Go to the documentation of this file.](#)

```

00001
00013 #include <display.h>
00014
00015 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00016
00017 void setup()
00018 {
00019     Serial.begin(9600);
00020     while (!Serial)
00021     {
00022         ;
00023     }
00024     ledmatrix.begin();
00025
00026     int x, j = 0;
00027     int32_t t0 = 0;
00028     int32_t t1 = 0;
00029     MilliTimer timer;
00030     while (1)
00031     {
00032         for (int16_t i = 0; i <= 100; i++)
00033         {
00034             t0 = micros();
00035             DisplayCap(i, ledmatrix);
00036             t1 = micros();
00037             Serial.print("Tiempo de ejecucion PowerBar:");
00038             Serial.println(t1 - t0);
00039             delay(100);
00040         }
00041         for (int16_t i = 100; i >= 0; i--)
00042         {
00043             t0 = micros();
00044             DisplayCap(i, ledmatrix);
00045             t1 = micros();
00046             Serial.print("Tiempo de ejecucion PowerBar:");
00047             Serial.println(t1 - t0);
00048             delay(100);
00049         }
00050     }
00051 }
00052
00053 void loop()
00054 {
00055 }
```

## 15.51 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_5/TEST\_5.ino File Reference

Test de la funcion DisplayBattCharging. Este test muestra la evolucion de la carga de la bateria, desde el 0% al 100%.

```
#include <display.h>
```

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)

### Variables

- Adafruit\_IS31FL3731\_Wing [ledmatrix = Adafruit\\_IS31FL3731\\_Wing\(\)](#)

#### 15.51.1 Detailed Description

Test de la funcion DisplayBattCharging. Este test muestra la evolucion de la carga de la bateria, desde el 0% al 100%.

**Author**

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

**Version**

2

**Date**

2021-03-16

**Copyright**

Copyright (c) 2020

Definition in file [TEST\\_5.ino](#).

## 15.51.2 Function Documentation

### 15.51.2.1 loop()

void loop ()

Definition at line 40 of file [TEST\\_5.ino](#).

### 15.51.2.2 setup()

void setup ()

Definition at line 16 of file [TEST\\_5.ino](#).

## 15.51.3 Variable Documentation

### 15.51.3.1 ledmatrix

[Adafruit\\_IS31FL3731\\_Wing](#) ledmatrix = [Adafruit\\_IS31FL3731\\_Wing\(\)](#)

Definition at line 14 of file [TEST\\_5.ino](#).

## 15.52 TEST\_5.ino

[Go to the documentation of this file.](#)

```
00001
00012 #include <display.h>
00013
00014 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00015
00016 void setup()
00017 {
00018     Serial.begin(9600);
00019     while (!Serial)
00020     {
00021         ;
00022     }
00023     int32_t t0 = 0;
00024     int32_t t1 = 0;
00025     ledmatrix.begin();
00026     Serial.println("IS31 Found!");
00027
00028     for (int16_t i = 0; i <= 100; i++)
00029     {
00030         t0 = micros();
00031         DisplayBattCharging(i, ledmatrix);
```

```

00033     t1 = micros();
00034     Serial.print("Tiempo de ejecucion DisplayBattCharging:");
00035     Serial.println(t1 - t0);
00036     delay(100);
00037 }
00038 }
00039
00040 void loop()
00041 {
00042 }
```

## 15.53 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_6/TEST\_6.ino File Reference

Testeo de la funcion DisplayTest mostrando todos los caracteres imprimibles posibles.

```
#include <display.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

### Variables

- Adafruit\_IS31FL3731\_Wing ledmatrix = Adafruit\_IS31FL3731\_Wing()
- bool display\_status = C\_DISPLAY\_ST\_NOT\_BUSSY
- bool reset\_init\_text = true
- bool mode\_bright\_display = C\_MODE\_HIGH\_BRIGHT

#### 15.53.1 Detailed Description

Testeo de la funcion DisplayTest mostrando todos los caracteres imprimibles posibles.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

2

##### Date

2020-12-03

##### Copyright

Copyright (c) 2020

Definition in file [TEST\\_6.ino](#).

#### 15.53.2 Function Documentation

##### 15.53.2.1 [loop\(\)](#)

`void loop ()`

Definition at line 41 of file [TEST\\_6.ino](#).

### 15.53.2.2 setup()

```
void setup ( )  
Definition at line 18 of file TEST_6.ino.
```

## 15.53.3 Variable Documentation

### 15.53.3.1 display\_status

```
bool display_status = C_DISPLAY_ST_NOT_BUSSY  
Definition at line 15 of file TEST_6.ino.
```

### 15.53.3.2 ledmatrix

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()  
Definition at line 14 of file TEST_6.ino.
```

### 15.53.3.3 mode\_bright\_display

```
bool mode_bright_display = C_MODE_HIGH_BRIGHT  
Definition at line 17 of file TEST_6.ino.
```

### 15.53.3.4 reset\_init\_text

```
bool reset_init_text = true  
Definition at line 16 of file TEST_6.ino.
```

## 15.54 TEST\_6.ino

[Go to the documentation of this file.](#)

```
00001  
00012 #include <display.h>  
00013  
00014 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();  
00015 bool display_status = C_DISPLAY_ST_NOT_BUSSY;  
00016 bool reset_init_text = true;  
00017 bool mode_bright_display = C_MODE_HIGH_BRIGHT;  
00018 void setup()  
00019 {  
00020     Serial.begin(9600);  
00021     ledmatrix.begin();  
00022  
00023     Serial.println("IS31 Found!");  
00024  
00025     char c;  
00026     String toString = "";  
00027     for (int i = 32; i <= 126; i++)  
00028     {  
00029         c = i;  
00030         toString += String(c) + " ";  
00031     }  
00032     Serial.println(toString);  
00033     display_status = DisplayText(toString, ledmatrix, reset_init_text, mode_bright_display);  
00034     reset_init_text = false;  
00035     while (display_status != C_DISPLAY_ST_NOT_BUSSY)  
00036     {  
00037         display_status = DisplayText(toString, ledmatrix, reset_init_text, mode_bright_display);  
00038     }  
00039 }  
00040  
00041 void loop()  
00042 {  
00043 }
```

## 15.55 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_7/TEST\_7.ino File Reference

Testeo del boton que marca el estado de la salida de voltage que forma parte de la pantalla.

```
#include <display.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

### Variables

- Adafruit\_IS31FL3731\_Wing ledmatrix = Adafruit\_IS31FL3731\_Wing()
- bool display\_status = C\_DISPLAY\_ST\_NOT\_BUSSY
- bool reset\_init\_text = true
- bool mode\_bright\_display = C\_MODE\_HIGH\_BRIGHT

#### 15.55.1 Detailed Description

Testeo del boton que marca el estado de la salida de voltage que forma parte de la pantalla.

##### Author

Javi ([Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

2

##### Date

2020-12-03

##### Copyright

Copyright (c) 2020

Definition in file [TEST\\_7.ino](#).

#### 15.55.2 Function Documentation

##### 15.55.2.1 loop()

```
void loop ()  
Definition at line 34 of file TEST\_7.ino.
```

##### 15.55.2.2 setup()

```
void setup ()  
Definition at line 18 of file TEST\_7.ino.
```

### 15.55.3 Variable Documentation

#### 15.55.3.1 display\_status

```
bool display_status = C_DISPLAY_ST_NOT_BUSSY
Definition at line 15 of file TEST_7.ino.
```

#### 15.55.3.2 ledmatrix

```
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()
Definition at line 14 of file TEST_7.ino.
```

#### 15.55.3.3 mode\_bright\_display

```
bool mode_bright_display = C_MODE_HIGH_BRIGHT
Definition at line 17 of file TEST_7.ino.
```

#### 15.55.3.4 reset\_init\_text

```
bool reset_init_text = true
Definition at line 16 of file TEST_7.ino.
```

## 15.56 TEST\_7.ino

[Go to the documentation of this file.](#)

```
00001
00012 #include <display.h>
00013
00014 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00015 bool display_status = C_DISPLAY_ST_NOT_BUSSY;
00016 bool reset_init_text = true;
00017 bool mode_bright_display = C_MODE_HIGH_BRIGHT;
00018 void setup()
00019 {
00020     Serial.begin(9600);
00021     ledmatrix.begin();
00022
00023     Serial.println("IS31 Found!");
00024     bool led_state = false;
00025     for (int i = 32; i <= 126; i++)
00026     {
00027         LedWork(true, ledmatrix, mode_bright_display);
00028         delay(1000);
00029         LedWork(false, ledmatrix, mode_bright_display);
00030         delay(1000);
00031     }
00032 }
00033
00034 void loop()
00035 {
00036 }
```

## 15.57 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0002-DISPLAY/examples/TEST\_8/TEST\_←  
 8.ino File Reference

Test de la funcion DisplayUsbIn y DisplayUsbOut.

```
#include "display.h"
```

## Functions

- void `setup()`
- void `loop()`

## Variables

- `Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()`
- `MilliTimer timer`

### 15.57.1 Detailed Description

Test de la funcion DisplayUsbIn y DisplayUsbOut.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

#### Version

1

#### Date

2021-03-26

#### Copyright

Copyright (c) 2021

Definition in file [TEST\\_8.ino](#).

### 15.57.2 Function Documentation

#### 15.57.2.1 `loop()`

`void loop ()`

Definition at line [31](#) of file [TEST\\_8.ino](#).

#### 15.57.2.2 `setup()`

`void setup ()`

Definition at line [17](#) of file [TEST\\_8.ino](#).

### 15.57.3 Variable Documentation

#### 15.57.3.1 `ledmatrix`

`Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing()`

Definition at line [14](#) of file [TEST\\_8.ino](#).

#### 15.57.3.2 `timer`

`MilliTimer timer`

Definition at line [15](#) of file [TEST\\_8.ino](#).

## 15.58 TEST\_8.ino

[Go to the documentation of this file.](#)

```
00001
00011 #include "display.h"
00012 //#include "MilliTimmer.h"
00013
00014 Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00015 MilliTimmer timer;
00016
00017 void setup()
00018 {
00019     ledmatrix.begin();
00020     timer.set(10000);
00021     while (timer.poll() == C_TIMER_NOT_EXPIRED)
00022     {
00023         DisplayUsbIn(ledmatrix);
00024     }
00025     while (true)
00026     {
00027         DisplayUsbOut(ledmatrix);
00028     }
00029 }
00030
00031 void loop()
00032 {
00033 }
```

## 15.59 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0003-BUZZER/Buzzer.h File Reference

```
#include "notes.h"
```

### Macros

- `#define NOTE_B9 15804`

### Functions

- `void InitBuzzer (uint16_t mode=C_MODE_DEFAULT)`  
*Inicializacion del buzzer con la seleccion del sonido de cada accion.*
- `uint16_t getSound (uint16_t *local, uint16_t sound_id)`  
*Get the Sound object.*
- `bool playSound (int16_t num)`  
*Funcion que reproduce los sonidos.*

### Variables

- `const uint16_t C_PIN_BUZZER = 4`
- `const uint16_t C_SOUND_MUTE = 0`
- `const uint16_t C_SOUND_START = 1`
- `const uint16_t C_SOUND_END = 2`
- `const uint16_t C_SOUND_UP = 3`
- `const uint16_t C_SOUND_DOWN = 4`
- `const uint16_t C_SOUND_ON = 5`
- `const uint16_t C_SOUND_OFF = 6`
- `const uint16_t C_SOUND_CHARGE_IN = 7`
- `const uint16_t C_SOUND_CHARGE_OUT = 8`
- `const uint16_t C_SOUND_DEATH_BATTERY = 9`
- `const uint16_t C_SOUND_FULL_CHARGE = 10`
- `const uint16_t C_SOUND_LOW_BATTERY = 11`

- const uint16\_t **C\_SOUND\_ERROR** = 12
- const uint16\_t **C\_NOTES\_START\_1** [] = {**NOTE\_E6**, 125, 5, **NOTE\_G6**, 125, 5, **NOTE\_E7**, 125, 5, **NOTE\_C7**, 125, 5, **NOTE\_D7**, 125, 5, **NOTE\_G7**, 125, 5}
- const uint16\_t **C\_NOTES\_END\_1** [] = {**NOTE\_G7**, 125, 5, **NOTE\_D7**, 125, 5, **NOTE\_C7**, 125, 5, **NOTE\_E7**, 125, 5, **NOTE\_G6**, 125, 5, **NOTE\_E6**, 125, 5}
- const uint16\_t **C\_NOTES\_DOWN\_1** [] = {**NOTE\_E6**, 10, 10, **NOTE\_B5**, 10, 10}
- const uint16\_t **C\_NOTES\_UP\_1** [] = {**NOTE\_B5**, 10, 10, **NOTE\_E6**, 10, 10}
- const uint16\_t **C\_NOTES\_ON\_1** [] = {**NOTE\_B5**, 100, 0, **NOTE\_E6**, 200, 0}
- const uint16\_t **C\_NOTES\_OFF\_1** [] = {**NOTE\_E6**, 100, 0, **NOTE\_B5**, 200, 0}
- const uint16\_t **C\_NOTES\_CHARGE\_IN** [] = {**NOTE\_C5**, 125, 10, **NOTE\_C7**, 125, 10, **NOTE\_C6**, 125, 10}
- const uint16\_t **C\_NOTES\_CHARGE\_OUT** [] = {**NOTE\_C6**, 125, 10, **NOTE\_C7**, 125, 10, **NOTE\_C5**, 125, 10}
- const uint16\_t **C\_NOTES\_LOW\_BATTERY** [] = {**NOTE\_B3**, 200, 10, **NOTE\_F3**, 200, 10, **NOTE\_B3**, 200, 10, **NOTE\_F3**, 200, 10, **NOTE\_B3**, 200, 10, **NOTE\_F3**, 200, 10, **NOTE\_B3**, 200, 10, **NOTE\_F3**, 200, 10}
- const uint16\_t **C\_NOTES\_FULL\_CHARGE** [] = {**NOTE\_DS5**, 150, 10, **NOTE\_DS5**, 150, 10, **NOTE\_DS5**, 150, 10, **NOTE\_DS5**, 400, 5, **NOTE\_B4**, 400, 5, **NOTE\_CS5**, 400, 5, **NOTE\_DS5**, 150, 150, **NOTE\_CS5**, 150, 5, **NOTE\_DS5**, 500, 5}
- const uint16\_t **C\_NOTES\_DEATHBATTERY** [] = {**NOTE\_G5**, 100, 10, **NOTE\_G5**, 100, 10, **NOTE\_G5**, 100, 10}
- const uint16\_t **C\_NOTES\_ERROR** [] = {**NOTE\_C8**, 100, 50, **NOTE\_C8**, 100, 50, **NOTE\_C8**, 100, 50, **NOTE\_C8**, 100, 50}
- MilliTimer timer\_buzzer
- const bool **C\_BUZZER\_BUSSY** = true
- const bool **C\_BUZZER\_NOT\_BUSSY** = false
- uint16\_t **SOUND\_START** [100]
- uint16\_t **size\_sound\_start**
- uint16\_t **SOUND\_END** [100]
- uint16\_t **size\_sound\_end**
- uint16\_t **SOUND\_DOWN** [100]
- uint16\_t **size\_sound\_down**
- uint16\_t **SOUND\_UP** [100]
- uint16\_t **size\_sound\_up**
- uint16\_t **SOUND\_ON** [100]
- uint16\_t **size\_sound\_on**
- uint16\_t **SOUND\_OFF** [100]
- uint16\_t **size\_sound\_off**
- uint16\_t **SOUND\_CHARGE\_IN** [100]
- uint16\_t **size\_sound\_charge\_in**
- uint16\_t **SOUND\_CHARGE\_OUT** [100]
- uint16\_t **size\_sound\_charge\_out**
- uint16\_t **SOUND\_DEATH\_BATTERY** [100]
- uint16\_t **size\_sound\_death\_battery**
- uint16\_t **SOUND\_FULL\_CHARGE** [100]
- uint16\_t **size\_sound\_full\_charge**
- uint16\_t **SOUND\_LOW\_BATTERY** [100]
- uint16\_t **size\_sound\_low\_battery**
- uint16\_t **SOUND\_ERROR** [100]
- uint16\_t **size\_sound\_error**
- const uint16\_t **C\_MODE\_HACK\_ALTERNATIVE** = 0x10
- const uint16\_t **C\_MODE\_DEFAULT** = 0x11

## 15.59.1 Detailed Description

### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

### Version

2

### Date

2021-02-11

### Copyright

Copyright (c) 2021

Definition in file [Buzzer.h](#).

## 15.59.2 Macro Definition Documentation

### 15.59.2.1 NOTE\_B9

```
#define NOTE_B9 15804
```

Definition at line [120](#) of file [notes.h](#).

## 15.59.3 Function Documentation

### 15.59.3.1 getSound()

```
uint16_t getSound (  
    uint16_t * local,  
    uint16_t sound_id )
```

Get the Sound object.

#### Parameters

<i>local</i>	
<i>sound_id</i>	

#### Returns

uint16\_t

Definition at line [137](#) of file [Buzzer.h](#).

### 15.59.3.2 InitBuzzer()

```
void InitBuzzer (   
    uint16_t mode = C\_MODE\_DEFAULT )
```

Inicializacion del buzzer con la seleccion del sonido de cada accion.

**Parameters**

<i>mode</i>	<input type="text"/>
-------------	----------------------

Definition at line 81 of file [Buzzer.h](#).

### 15.59.3.3 playSound()

```
bool playSound (
    int16_t num )
```

Funcion que reproduce los sonidos.

**Parameters**

<i>num</i>	<input type="text"/>
------------	----------------------

**Returns**

true

false

Definition at line 211 of file [Buzzer.h](#).

## 15.59.4 Variable Documentation

### 15.59.4.1 C\_BUZZER\_BUSSY

```
const bool C_BUZZER_BUSSY = true
```

Definition at line 46 of file [Buzzer.h](#).

### 15.59.4.2 C\_BUZZER\_NOT\_BUSSY

```
const bool C_BUZZER_NOT_BUSSY = false
```

Definition at line 47 of file [Buzzer.h](#).

### 15.59.4.3 C\_MODE\_DEFAULT

```
const uint16_t C_MODE_DEFAULT = 0x11
```

Definition at line 74 of file [Buzzer.h](#).

### 15.59.4.4 C\_MODE\_HACK\_ALTERNATIVE

```
const uint16_t C_MODE_HACK_ALTERNATIVE = 0x10
```

Definition at line 73 of file [Buzzer.h](#).

### 15.59.4.5 C\_NOTES\_CHARGE\_IN

```
const uint16_t C_NOTES_CHARGE_IN[] = {NOTE_C5, 125, 10, NOTE_C7, 125, 10, NOTE_C6, 125, 10}
```

Definition at line 37 of file [Buzzer.h](#).

#### 15.59.4.6 C\_NOTES\_CHARGE\_OUT

```
const uint16_t C_NOTES_CHARGE_OUT[ ] = {NOTE_C6, 125, 10, NOTE_C7, 125, 10, NOTE_C5, 125, 10}
```

Definition at line 38 of file [Buzzer.h](#).

#### 15.59.4.7 C\_NOTES\_DEATHBATTERY

```
const uint16_t C_NOTES_DEATHBATTERY[ ] = {NOTE_G5, 100, 10, NOTE_G5, 100, 10, NOTE_G5, 100, 10}
```

Definition at line 41 of file [Buzzer.h](#).

#### 15.59.4.8 C\_NOTES\_DOWN\_1

```
const uint16_t C_NOTES_DOWN_1[ ] = {NOTE_E6, 10, 10, NOTE_B5, 10, 10}
```

Definition at line 33 of file [Buzzer.h](#).

#### 15.59.4.9 C\_NOTES\_END\_1

```
const uint16_t C_NOTES_END_1[ ] = {NOTE_G7, 125, 5, NOTE_D7, 125, 5, NOTE_C7, 125, 5, NOTE_E7,
```

125, 5, NOTE\_G6, 125, 5, NOTE\_E6, 125, 5}

Definition at line 32 of file [Buzzer.h](#).

#### 15.59.4.10 C\_NOTES\_ERROR

```
const uint16_t C_NOTES_ERROR[ ] = {NOTE_C8, 100, 50, NOTE_C8, 100, 50, NOTE_C8, 100, 50, NOTE_C8, 100, 50}
```

Definition at line 42 of file [Buzzer.h](#).

#### 15.59.4.11 C\_NOTES\_FULL\_CHARGE

```
const uint16_t C_NOTES_FULL_CHARGE[ ] = {NOTE_DS5, 150, 10, NOTE_DS5, 150, 10, NOTE_DS5, 150,
```

10, NOTE\_DS5, 400, 5, NOTE\_B4, 400, 5, NOTE\_CS5, 400, 5, NOTE\_DS5, 150, 150, NOTE\_CS5, 150, 5,

NOTE\_DS5, 500, 5}

Definition at line 40 of file [Buzzer.h](#).

#### 15.59.4.12 C\_NOTES\_LOW\_BATTERY

```
const uint16_t C_NOTES_LOW_BATTERY[ ] = {NOTE_B3, 200, 10, NOTE_F3, 200, 10, NOTE_B3, 200, 10,
```

NOTE\_F3, 200, 10, NOTE\_B3, 200, 10, NOTE\_F3, 200, 10, NOTE\_B3, 200, 10, NOTE\_F3, 200, 10}

Definition at line 39 of file [Buzzer.h](#).

#### 15.59.4.13 C\_NOTES\_OFF\_1

```
const uint16_t C_NOTES_OFF_1[ ] = {NOTE_E6, 100, 0, NOTE_B5, 200, 0}
```

Definition at line 36 of file [Buzzer.h](#).

#### 15.59.4.14 C\_NOTES\_ON\_1

```
const uint16_t C_NOTES_ON_1[ ] = {NOTE_B5, 100, 0, NOTE_E6, 200, 0}
```

Definition at line 35 of file [Buzzer.h](#).

#### 15.59.4.15 C\_NOTES\_START\_1

```
const uint16_t C_NOTES_START_1[] = {NOTE_E6, 125, 5, NOTE_G6, 125, 5, NOTE_E7, 125, 5, NOTE_C7,  
125, 5, NOTE_D7, 125, 5, NOTE_G7, 125, 5}
```

Definition at line 31 of file [Buzzer.h](#).

#### 15.59.4.16 C\_NOTES\_UP\_1

```
const uint16_t C_NOTES_UP_1[] = {NOTE_B5, 10, 10, NOTE_E6, 10, 10}
```

Definition at line 34 of file [Buzzer.h](#).

#### 15.59.4.17 C\_PIN\_BUZZER

```
const uint16_t C_PIN_BUZZER = 4
```

Definition at line 14 of file [Buzzer.h](#).

#### 15.59.4.18 C\_SOUND\_CHARGE\_IN

```
const uint16_t C_SOUND_CHARGE_IN = 7
```

Definition at line 23 of file [Buzzer.h](#).

#### 15.59.4.19 C\_SOUND\_CHARGE\_OUT

```
const uint16_t C_SOUND_CHARGE_OUT = 8
```

Definition at line 24 of file [Buzzer.h](#).

#### 15.59.4.20 C\_SOUND\_DEATH\_BATTERY

```
const uint16_t C_SOUND_DEATH_BATTERY = 9
```

Definition at line 25 of file [Buzzer.h](#).

#### 15.59.4.21 C\_SOUND\_DOWN

```
const uint16_t C_SOUND_DOWN = 4
```

Definition at line 20 of file [Buzzer.h](#).

#### 15.59.4.22 C\_SOUND\_END

```
const uint16_t C_SOUND_END = 2
```

Definition at line 18 of file [Buzzer.h](#).

#### 15.59.4.23 C\_SOUND\_ERROR

```
const uint16_t C_SOUND_ERROR = 12
```

Definition at line 28 of file [Buzzer.h](#).

#### 15.59.4.24 C\_SOUND\_FULL\_CHARGE

```
const uint16_t C_SOUND_FULL_CHARGE = 10
```

Definition at line 26 of file [Buzzer.h](#).

#### 15.59.4.25 C\_SOUND\_LOW\_BATTERY

```
const uint16_t C_SOUND_LOW_BATTERY = 11
Definition at line 27 of file Buzzer.h.
```

#### 15.59.4.26 C\_SOUND\_MUTE

```
const uint16_t C_SOUND_MUTE = 0
Definition at line 16 of file Buzzer.h.
```

#### 15.59.4.27 C\_SOUND\_OFF

```
const uint16_t C_SOUND_OFF = 6
Definition at line 22 of file Buzzer.h.
```

#### 15.59.4.28 C\_SOUND\_ON

```
const uint16_t C_SOUND_ON = 5
Definition at line 21 of file Buzzer.h.
```

#### 15.59.4.29 C\_SOUND\_START

```
const uint16_t C_SOUND_START = 1
Definition at line 17 of file Buzzer.h.
```

#### 15.59.4.30 C\_SOUND\_UP

```
const uint16_t C_SOUND_UP = 3
Definition at line 19 of file Buzzer.h.
```

#### 15.59.4.31 size\_sound\_charge\_in

```
uint16_t size_sound_charge_in
Definition at line 61 of file Buzzer.h.
```

#### 15.59.4.32 size\_sound\_charge\_out

```
uint16_t size_sound_charge_out
Definition at line 63 of file Buzzer.h.
```

#### 15.59.4.33 size\_sound\_death\_battery

```
uint16_t size_sound_death_battery
Definition at line 65 of file Buzzer.h.
```

#### 15.59.4.34 size\_sound\_down

```
uint16_t size_sound_down
Definition at line 53 of file Buzzer.h.
```

**15.59.4.35 size\_sound\_end**

```
uint16_t size_sound_end
```

Definition at line 51 of file [Buzzer.h](#).

**15.59.4.36 size\_sound\_error**

```
uint16_t size_sound_error
```

Definition at line 71 of file [Buzzer.h](#).

**15.59.4.37 size\_sound\_full\_charge**

```
uint16_t size_sound_full_charge
```

Definition at line 67 of file [Buzzer.h](#).

**15.59.4.38 size\_sound\_low\_battery**

```
uint16_t size_sound_low_battery
```

Definition at line 69 of file [Buzzer.h](#).

**15.59.4.39 size\_sound\_off**

```
uint16_t size_sound_off
```

Definition at line 59 of file [Buzzer.h](#).

**15.59.4.40 size\_sound\_on**

```
uint16_t size_sound_on
```

Definition at line 57 of file [Buzzer.h](#).

**15.59.4.41 size\_sound\_start**

```
uint16_t size_sound_start
```

Definition at line 49 of file [Buzzer.h](#).

**15.59.4.42 size\_sound\_up**

```
uint16_t size_sound_up
```

Definition at line 55 of file [Buzzer.h](#).

**15.59.4.43 SOUND\_CHARGE\_IN**

```
uint16_t SOUND_CHARGE_IN[100]
```

Definition at line 60 of file [Buzzer.h](#).

**15.59.4.44 SOUND\_CHARGE\_OUT**

```
uint16_t SOUND_CHARGE_OUT[100]
```

Definition at line 62 of file [Buzzer.h](#).

#### 15.59.4.45 SOUND\_DEATH\_BATTERY

```
uint16_t SOUND_DEATH_BATTERY[100]
```

Definition at line [64](#) of file [Buzzer.h](#).

#### 15.59.4.46 SOUND\_DOWN

```
uint16_t SOUND_DOWN[100]
```

Definition at line [52](#) of file [Buzzer.h](#).

#### 15.59.4.47 SOUND\_END

```
uint16_t SOUND_END[100]
```

Definition at line [50](#) of file [Buzzer.h](#).

#### 15.59.4.48 SOUND\_ERROR

```
uint16_t SOUND_ERROR[100]
```

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

#### 15.59.4.49 SOUND\_FULL\_CHARGE

```
uint16_t SOUND_FULL_CHARGE[100]
```

Definition at line [66](#) of file [Buzzer.h](#).

#### 15.59.4.50 SOUND\_LOW\_BATTERY

```
uint16_t SOUND_LOW_BATTERY[100]
```

Definition at line [68](#) of file [Buzzer.h](#).

#### 15.59.4.51 SOUND\_OFF

```
uint16_t SOUND_OFF[100]
```

Definition at line [58](#) of file [Buzzer.h](#).

#### 15.59.4.52 SOUND\_ON

```
uint16_t SOUND_ON[100]
```

Definition at line [56](#) of file [Buzzer.h](#).

#### 15.59.4.53 SOUND\_START

```
uint16_t SOUND_START[100]
```

Definition at line [48](#) of file [Buzzer.h](#).

#### 15.59.4.54 SOUND\_UP

```
uint16_t SOUND_UP[100]
```

Definition at line [54](#) of file [Buzzer.h](#).

### 15.59.4.55 timer\_buzzer

`MilliTimer timer_buzzer`  
 Definition at line 44 of file [Buzzer.h](#).

## 15.60 Buzzer.h

[Go to the documentation of this file.](#)

```

00001
00011 #include "notes.h"
00012 //##include <MilliTimer.h>
00013
00014 const uint16_t C_PIN_BUZZER = 4;
00015
00016 const uint16_t C_SOUND_MUTE = 0;
00017 const uint16_t C_SOUND_START = 1;
00018 const uint16_t C_SOUND_END = 2;
00019 const uint16_t C_SOUND_UP = 3;
00020 const uint16_t C_SOUND_DOWN = 4;
00021 const uint16_t C_SOUND_ON = 5;
00022 const uint16_t C_SOUND_OFF = 6;
00023 const uint16_t C_SOUND_CHARGE_IN = 7;
00024 const uint16_t C_SOUND_CHARGE_OUT = 8;
00025 const uint16_t C_SOUND_DEATH_BATTERY = 9;
00026 const uint16_t C_SOUND_FULL_CHARGE = 10;
00027 const uint16_t C_SOUND_LOW_BATTERY = 11;
00028 const uint16_t C_SOUND_ERROR = 12;
00029
00030
00031 const uint16_t C_NOTES_START_1[] = {NOTE_E6, 125, 5, NOTE_G6, 125, 5, NOTE_E7, 125, 5, NOTE_C7, 125,
00032   5, NOTE_D7, 125, 5, NOTE_G7, 125, 5};
00033 const uint16_t C_NOTES_END_1[] = {NOTE_G7, 125, 5, NOTE_D7, 125, 5, NOTE_C7, 125, 5, NOTE_E7, 125, 5,
00034   NOTE_G6, 125, 5, NOTE_E6, 125, 5};
00035 const uint16_t C_NOTES_DOWN_1[] = {NOTE_E6, 10, 10, NOTE_B5, 10, 10};
00036 const uint16_t C_NOTES_UP_1[] = {NOTE_B5, 10, 10, NOTE_E6, 10, 10};
00037 const uint16_t C_NOTES_ON_1[] = {NOTE_B5, 100, 0, NOTE_E6, 200, 0};
00038 const uint16_t C_NOTES_OFF_1[] = {NOTE_E6, 100, 0, NOTE_B5, 200, 0};
00039 const uint16_t C_NOTES_CHARGE_IN[] = {NOTE_C5, 125, 10, NOTE_C7, 125, 10, NOTE_C6, 125, 10};
00040 const uint16_t C_NOTES_CHARGE_OUT[] = {NOTE_C6, 125, 10, NOTE_C7, 125, 10, NOTE_C5, 125, 10};
00041 const uint16_t C_NOTES_LOW_BATTERY[] = {NOTE_B3, 200, 10, NOTE_F3, 200, 10, NOTE_B3, 200, 10, NOTE_F3,
00042   200, 10, NOTE_B3, 200, 10, NOTE_F3, 200, 10, NOTE_B3, 200, 10, NOTE_F3, 200, 10};
00043 const uint16_t C_NOTES_FULL_CHARGE[] = {NOTE_DS5, 150, 10, NOTE_DS5, 150, 10, NOTE_DS5, 150, 10,
00044   NOTE_DS5, 400, 5, NOTE_B4, 400, 5, NOTE_CS5, 400, 5, NOTE_DS5, 150, 150, NOTE_CS5, 150, 5, NOTE_DS5,
00045   500, 5};
00046 const uint16_t C_NOTES_DEATHBATTERY[] = {NOTE_G5, 100, 10, NOTE_G5, 100, 10, NOTE_G5, 100, 10};
00047 const uint16_t C_NOTES_ERROR[] = {NOTE_C8, 100, 50, NOTE_C8, 100, 50, NOTE_C8, 100, 50, NOTE_C8, 100, 50};
00048 MilliTimer timer_buzzer;
00049
00050 const bool C_BUZZER_BUSSY = true;
00051 const bool C_BUZZER_NOT_BUSSY = false;
00052 uint16_t SOUND_START[100];
00053 uint16_t size_sound_start;
00054 uint16_t SOUND_END[100];
00055 uint16_t size_sound_end;
00056 uint16_t SOUND_DOWN[100];
00057 uint16_t size_sound_down;
00058 uint16_t SOUND_UP[100];
00059 uint16_t size_sound_up;
00060 uint16_t SOUND_ON[100];
00061 uint16_t size_sound_on;
00062 uint16_t SOUND_OFF[100];
00063 uint16_t size_sound_off;
00064 uint16_t SOUND_CHARGE_IN[100];
00065 uint16_t size_sound_charge_in;
00066 uint16_t SOUND_CHARGE_OUT[100];
00067 uint16_t size_sound_charge_out;
00068 uint16_t SOUND_DEATH_BATTERY[100];
00069 uint16_t size_sound_death_battery;
00070 uint16_t SOUND_FULL_CHARGE[100];
00071 uint16_t size_sound_full_charge;
00072 uint16_t SOUND_LOW_BATTERY[100];
00073 uint16_t size_sound_low_battery;
00074 uint16_t SOUND_ERROR[100];
00075 uint16_t size_sound_error;
00076
00077 const uint16_t C_MODE_HACK_ALTERNATIVE = 0x10;
00078 const uint16_t C_MODE_DEFAULT = 0x11;
00079
00080 void InitBuzzer(uint16_t mode = C_MODE_DEFAULT)
00081 {
00082     pinMode(C_PIN_BUZZER, OUTPUT);
00083 }
```

```

00085     switch (mode)
00086     {
00087     case C_MODE_DEFAULT:
00088         memcpy(SOUND_START, C_NOTES_START_1, sizeof(C_NOTES_START_1));
00089         size_sound_start = (sizeof(C_NOTES_START_1) / sizeof(C_NOTES_START_1[0])) / 3;
00090         memcpy(SOUND_END, C_NOTES_END_1, sizeof(C_NOTES_END_1));
00091         size_sound_end = (sizeof(C_NOTES_END_1) / sizeof(C_NOTES_END_1[0])) / 3;
00092         memcpy(SOUND_UP, C_NOTES_UP_1, sizeof(C_NOTES_UP_1));
00093         size_sound_up = (sizeof(C_NOTES_UP_1) / sizeof(C_NOTES_UP_1[0])) / 3;
00094         memcpy(SOUND_DOWN, C_NOTES_DOWN_1, sizeof(C_NOTES_DOWN_1));
00095         size_sound_down = (sizeof(C_NOTES_DOWN_1) / sizeof(C_NOTES_DOWN_1[0])) / 3;
00096         memcpy(SOUND_ON, C_NOTES_ON_1, sizeof(C_NOTES_ON_1));
00097         size_sound_on = (sizeof(C_NOTES_ON_1) / sizeof(C_NOTES_ON_1[0])) / 3;
00098         memcpy(SOUND_OFF, C_NOTES_OFF_1, sizeof(C_NOTES_OFF_1));
00099         size_sound_off = (sizeof(C_NOTES_OFF_1) / sizeof(C_NOTES_OFF_1[0])) / 3;
00100         memcpy(SOUND_CHARGE_IN, C_NOTES_CHARGE_IN, sizeof(C_NOTES_CHARGE_IN));
00101         size_sound_charge_in = (sizeof(C_NOTES_CHARGE_IN) / sizeof(C_NOTES_CHARGE_IN[0])) / 3;
00102         memcpy(SOUND_CHARGE_OUT, C_NOTES_CHARGE_OUT, sizeof(C_NOTES_CHARGE_OUT));
00103         size_sound_charge_out = (sizeof(C_NOTES_CHARGE_OUT) / sizeof(C_NOTES_CHARGE_OUT[0])) / 3;
00104         memcpy(SOUND_DEATH_BATTERY, C_NOTES_DEATHBATTERY, sizeof(C_NOTES_DEATHBATTERY));
00105         size_sound_death_battery = (sizeof(C_NOTES_DEATHBATTERY) / sizeof(C_NOTES_DEATHBATTERY[0])) /
00106     3;
00106         memcpy(SOUND_FULL_CHARGE, C_NOTES_FULL_CHARGE, sizeof(C_NOTES_FULL_CHARGE));
00107         size_sound_full_charge = (sizeof(C_NOTES_FULL_CHARGE) / sizeof(C_NOTES_FULL_CHARGE[0])) / 3;
00108         memcpy(SOUND_LOW_BATTERY, C_NOTES_LOW_BATTERY, sizeof(C_NOTES_LOW_BATTERY));
00109         size_sound_low_battery = (sizeof(C_NOTES_LOW_BATTERY) / sizeof(C_NOTES_LOW_BATTERY[0])) / 3;
00110         memcpy(SOUND_ERROR, C_NOTES_ERROR, sizeof(C_NOTES_ERROR));
00111         size_sound_error = (sizeof(C_NOTES_ERROR) / sizeof(C_NOTES_ERROR[0])) / 3;
00112         break;
00113     }
00114     default:
00115         memcpy(SOUND_START, C_NOTES_START_1, sizeof(C_NOTES_START_1));
00116         size_sound_start = (sizeof(C_NOTES_START_1) / sizeof(C_NOTES_START_1[0])) / 3;
00117         memcpy(SOUND_END, C_NOTES_END_1, sizeof(C_NOTES_END_1));
00118         size_sound_end = (sizeof(C_NOTES_END_1) / sizeof(C_NOTES_END_1[0])) / 3;
00119         memcpy(SOUND_UP, C_NOTES_UP_1, sizeof(C_NOTES_UP_1));
00120         size_sound_up = (sizeof(C_NOTES_UP_1) / sizeof(C_NOTES_UP_1[0])) / 3;
00121         memcpy(SOUND_DOWN, C_NOTES_DOWN_1, sizeof(C_NOTES_DOWN_1));
00122         size_sound_down = (sizeof(C_NOTES_DOWN_1) / sizeof(C_NOTES_DOWN_1[0])) / 3;
00123         memcpy(SOUND_ON, C_NOTES_ON_1, sizeof(C_NOTES_ON_1));
00124         size_sound_on = (sizeof(C_NOTES_ON_1) / sizeof(C_NOTES_ON_1[0])) / 3;
00125         memcpy(SOUND_OFF, C_NOTES_OFF_1, sizeof(C_NOTES_OFF_1));
00126         size_sound_off = (sizeof(C_NOTES_OFF_1) / sizeof(C_NOTES_OFF_1[0])) / 3;
00127         break;
00128     }
00129 }
00130 uint16_t getSound(uint16_t *local, uint16_t sound_id)
00131 {
00132     if (sound_id == C_SOUND_START)
00133     {
00134         memcpy(local, SOUND_START, sizeof(SOUND_START));
00135         return (size_sound_start);
00136     }
00137     else if (sound_id == C_SOUND_END)
00138     {
00139         memcpy(local, SOUND_END, sizeof(SOUND_END));
00140         return (size_sound_end);
00141     }
00142     else if (sound_id == C_SOUND_UP)
00143     {
00144         memcpy(local, SOUND_UP, sizeof(SOUND_UP));
00145         return (size_sound_up);
00146     }
00147     else if (sound_id == C_SOUND_DOWN)
00148     {
00149         memcpy(local, SOUND_DOWN, sizeof(SOUND_DOWN));
00150         return (size_sound_down);
00151     }
00152     else if (sound_id == C_SOUND_ON)
00153     {
00154         memcpy(local, SOUND_ON, sizeof(SOUND_ON));
00155         return (size_sound_on);
00156     }
00157     else if (sound_id == C_SOUND_OFF)
00158     {
00159         memcpy(local, SOUND_OFF, sizeof(SOUND_OFF));
00160         return (size_sound_off);
00161     }
00162     else if (sound_id == C_SOUND_CHARGE_IN)
00163     {
00164         memcpy(local, SOUND_CHARGE_IN, sizeof(SOUND_CHARGE_IN));
00165         return (size_sound_charge_in);
00166     }
00167     else if (sound_id == C_SOUND_CHARGE_OUT)
00168     {
00169         memcpy(local, SOUND_CHARGE_OUT, sizeof(SOUND_CHARGE_OUT));
00170     }
00171 }

```

```

00178     return (size_sound_charge_out);
00179 }
00180
00181 else if (sound_id == C_SOUND_DEATH_BATTERY)
00182 {
00183     memcpy(local, SOUND_DEATH_BATTERY, sizeof(SOUND_DEATH_BATTERY));
00184     return (size_sound_death_battery);
00185 }
00186 else if (sound_id == C_SOUND_FULL_CHARGE)
00187 {
00188     memcpy(local, SOUND_FULL_CHARGE, sizeof(SOUND_FULL_CHARGE));
00189     return (size_sound_full_charge);
00190 }
00191
00192 else if (sound_id == C_SOUND_LOW_BATTERY)
00193 {
00194     memcpy(local, SOUND_LOW_BATTERY, sizeof(SOUND_LOW_BATTERY));
00195     return (size_sound_low_battery);
00196 }
00197 else if (sound_id == C_SOUND_ERROR)
00198 {
00199     memcpy(local, SOUND_ERROR, sizeof(SOUND_ERROR));
00200     return (size_sound_error);
00201 }
00202 return 0;
00203 }
00211 bool playSound(int16_t num)
00212 {
00213     static uint16_t index = 0;
00214     static bool buzzer_state = C_BUZZER_NOT_BUSSY;
00215     static uint16_t local_sound[100];
00216     static uint16_t size_sound = 0;
00217     if (buzzer_state == C_BUZZER_NOT_BUSSY)
00218     {
00219         size_sound = getSound(local_sound, num);
00220         index = 0;
00221         tone(C_PIN_BUZZER, local_sound[index], local_sound[index + 1]);
00222         timer_buzzer.set(local_sound[index + 1] + local_sound[index + 2]);
00223         buzzer_state = C_BUZZER_BUSSY;
00224     }
00225     else if (buzzer_state == C_BUZZER_BUSSY)
00226     {
00227         if (timer_buzzer.poll() != C_TIMER_NOT_EXPIRED)
00228         {
00229             index += 3;
00230             if ((index / 3) >= size_sound)
00231             {
00232                 buzzer_state = C_BUZZER_NOT_BUSSY;
00233                 //noTone(C_PIN_BUZZER);
00234             }
00235             else
00236             {
00237                 tone(C_PIN_BUZZER, local_sound[index], local_sound[index + 1]);
00238                 timer_buzzer.set(local_sound[index + 1] + local_sound[index + 2]);
00239                 buzzer_state = C_BUZZER_BUSSY;
00240             }
00241         }
00242     }
00243     return buzzer_state;
00244 }

```

## 15.61 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0003-BUZZER/notes.h File Reference

### Macros

- #define NOTE\_B0 31
- #define NOTE\_C1 33
- #define NOTE\_CS1 35
- #define NOTE\_D1 37
- #define NOTE\_DS1 39
- #define NOTE\_E1 41
- #define NOTE\_F1 44
- #define NOTE\_FS1 46
- #define NOTE\_G1 49

- #define NOTE\_GS1 52
- #define NOTE\_A1 55
- #define NOTE\_AS1 58
- #define NOTE\_B1 62
- #define NOTE\_C2 65
- #define NOTE\_CS2 69
- #define NOTE\_D2 73
- #define NOTE\_DS2 78
- #define NOTE\_E2 82
- #define NOTE\_F2 87
- #define NOTE\_FS2 93
- #define NOTE\_G2 98
- #define NOTE\_GS2 104
- #define NOTE\_A2 110
- #define NOTE\_AS2 117
- #define NOTE\_B2 123
- #define NOTE\_C3 131
- #define NOTE\_CS3 139
- #define NOTE\_D3 147
- #define NOTE\_DS3 156
- #define NOTE\_E3 165
- #define NOTE\_F3 175
- #define NOTE\_FS3 185
- #define NOTE\_G3 196
- #define NOTE\_GS3 208
- #define NOTE\_A3 220
- #define NOTE\_AS3 233
- #define NOTE\_B3 247
- #define NOTE\_C4 262
- #define NOTE\_CS4 277
- #define NOTE\_D4 294
- #define NOTE\_DS4 311
- #define NOTE\_E4 330
- #define NOTE\_F4 349
- #define NOTE\_FS4 370
- #define NOTE\_G4 392
- #define NOTE\_GS4 415
- #define NOTE\_A4 440
- #define NOTE\_AS4 466
- #define NOTE\_B4 494
- #define NOTE\_C5 523
- #define NOTE\_CS5 554
- #define NOTE\_D5 587
- #define NOTE\_DS5 622
- #define NOTE\_E5 659
- #define NOTE\_F5 698
- #define NOTE\_FS5 740
- #define NOTE\_G5 784
- #define NOTE\_GS5 831
- #define NOTE\_A5 880
- #define NOTE\_AS5 932
- #define NOTE\_B5 988
- #define NOTE\_C6 1047
- #define NOTE\_CS6 1109
- #define NOTE\_D6 1175

- #define NOTE\_DS6 1245
- #define NOTE\_E6 1319
- #define NOTE\_F6 1397
- #define NOTE\_FS6 1480
- #define NOTE\_G6 1568
- #define NOTE\_GS6 1661
- #define NOTE\_A6 1760
- #define NOTE\_AS6 1865
- #define NOTE\_B6 1976
- #define NOTE\_C7 2093
- #define NOTE\_CS7 2217
- #define NOTE\_D7 2349
- #define NOTE\_DS7 2489
- #define NOTE\_E7 2637
- #define NOTE\_F7 2794
- #define NOTE\_FS7 2960
- #define NOTE\_G7 3136
- #define NOTE\_GS7 3322
- #define NOTE\_A7 3520
- #define NOTE\_AS7 3729
- #define NOTE\_B7 3951
- #define NOTE\_C8 4186
- #define NOTE\_CS8 4435
- #define NOTE\_D8 4699
- #define NOTE\_DS8 4978
- #define NOTE\_E8 5274
- #define NOTE\_F8 5588
- #define NOTE\_FS8 5920
- #define NOTE\_G8 6272
- #define NOTE\_GS8 6644
- #define NOTE\_A8 7040
- #define NOTE\_AS8 7458
- #define NOTE\_B8 7902
- #define NOTE\_C9 8372
- #define NOTE\_CS9 8870
- #define NOTE\_D9 9398
- #define NOTE\_DS9 9956
- #define NOTE\_E9 10548
- #define NOTE\_F9 11176
- #define NOTE\_FS9 11840
- #define NOTE\_G9 12544
- #define NOTE\_GS9 13288
- #define NOTE\_A9 14080
- #define NOTE\_AS9 14916

### 15.61.1 Detailed Description

Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

Version

0.1

Date

2021-02-09

Copyright

Copyright (c) 2021

Definition in file [notes.h](#).

## 15.61.2 Macro Definition Documentation

### 15.61.2.1 NOTE\_A1

```
#define NOTE_A1 55
```

Definition at line [22](#) of file [notes.h](#).

### 15.61.2.2 NOTE\_A2

```
#define NOTE_A2 110
```

Definition at line [34](#) of file [notes.h](#).

### 15.61.2.3 NOTE\_A3

```
#define NOTE_A3 220
```

Definition at line [46](#) of file [notes.h](#).

### 15.61.2.4 NOTE\_A4

```
#define NOTE_A4 440
```

Definition at line [58](#) of file [notes.h](#).

### 15.61.2.5 NOTE\_A5

```
#define NOTE_A5 880
```

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

### 15.61.2.6 NOTE\_A6

```
#define NOTE_A6 1760
```

Definition at line [82](#) of file [notes.h](#).

### 15.61.2.7 NOTE\_A7

```
#define NOTE_A7 3520
```

Definition at line [94](#) of file [notes.h](#).

### 15.61.2.8 NOTE\_A8

```
#define NOTE_A8 7040
```

Definition at line [106](#) of file [notes.h](#).

### 15.61.2.9 NOTE\_A9

```
#define NOTE_A9 14080
Definition at line 118 of file notes.h.
```

### 15.61.2.10 NOTE\_AS1

```
#define NOTE_AS1 58
Definition at line 23 of file notes.h.
```

### 15.61.2.11 NOTE\_AS2

```
#define NOTE_AS2 117
Definition at line 35 of file notes.h.
```

### 15.61.2.12 NOTE\_AS3

```
#define NOTE_AS3 233
Definition at line 47 of file notes.h.
```

### 15.61.2.13 NOTE\_AS4

```
#define NOTE_AS4 466
Definition at line 59 of file notes.h.
```

### 15.61.2.14 NOTE\_AS5

```
#define NOTE_AS5 932
Definition at line 71 of file notes.h.
```

### 15.61.2.15 NOTE\_AS6

```
#define NOTE_AS6 1865
Definition at line 83 of file notes.h.
```

### 15.61.2.16 NOTE\_AS7

```
#define NOTE_AS7 3729
Definition at line 95 of file notes.h.
```

### 15.61.2.17 NOTE\_AS8

```
#define NOTE_AS8 7458
Definition at line 107 of file notes.h.
```

### 15.61.2.18 NOTE\_AS9

```
#define NOTE_AS9 14916
Definition at line 119 of file notes.h.
```

### 15.61.2.19 NOTE\_B0

```
#define NOTE_B0 31
```

Definition at line [12](#) of file [notes.h](#).

### 15.61.2.20 NOTE\_B1

```
#define NOTE_B1 62
```

Definition at line [24](#) of file [notes.h](#).

### 15.61.2.21 NOTE\_B2

```
#define NOTE_B2 123
```

Definition at line [36](#) of file [notes.h](#).

### 15.61.2.22 NOTE\_B3

```
#define NOTE_B3 247
```

Definition at line [48](#) of file [notes.h](#).

### 15.61.2.23 NOTE\_B4

```
#define NOTE_B4 494
```

Definition at line [60](#) of file [notes.h](#).

### 15.61.2.24 NOTE\_B5

```
#define NOTE_B5 988
```

Definition at line [72](#) of file [notes.h](#).

### 15.61.2.25 NOTE\_B6

```
#define NOTE_B6 1976
```

Definition at line [84](#) of file [notes.h](#).

### 15.61.2.26 NOTE\_B7

```
#define NOTE_B7 3951
```

Definition at line [96](#) of file [notes.h](#).

### 15.61.2.27 NOTE\_B8

```
#define NOTE_B8 7902
```

Definition at line [108](#) of file [notes.h](#).

### 15.61.2.28 NOTE\_C1

```
#define NOTE_C1 33
```

Definition at line [13](#) of file [notes.h](#).

### 15.61.2.29 NOTE\_C2

```
#define NOTE_C2 65
```

Definition at line [25](#) of file [notes.h](#).

### 15.61.2.30 NOTE\_C3

```
#define NOTE_C3 131
```

Definition at line [37](#) of file [notes.h](#).

### 15.61.2.31 NOTE\_C4

```
#define NOTE_C4 262
```

Definition at line [49](#) of file [notes.h](#).

### 15.61.2.32 NOTE\_C5

```
#define NOTE_C5 523
```

Definition at line [61](#) of file [notes.h](#).

### 15.61.2.33 NOTE\_C6

```
#define NOTE_C6 1047
```

Definition at line [73](#) of file [notes.h](#).

### 15.61.2.34 NOTE\_C7

```
#define NOTE_C7 2093
```

Definition at line [85](#) of file [notes.h](#).

### 15.61.2.35 NOTE\_C8

```
#define NOTE_C8 4186
```

Definition at line [97](#) of file [notes.h](#).

### 15.61.2.36 NOTE\_C9

```
#define NOTE_C9 8372
```

Definition at line [109](#) of file [notes.h](#).

### 15.61.2.37 NOTE\_CS1

```
#define NOTE_CS1 35
```

Definition at line [14](#) of file [notes.h](#).

### 15.61.2.38 NOTE\_CS2

```
#define NOTE_CS2 69
```

Definition at line [26](#) of file [notes.h](#).

#### 15.61.2.39 NOTE\_CS3

```
#define NOTE_CS3 139
```

Definition at line [38](#) of file [notes.h](#).

#### 15.61.2.40 NOTE\_CS4

```
#define NOTE_CS4 277
```

Definition at line [50](#) of file [notes.h](#).

#### 15.61.2.41 NOTE\_CS5

```
#define NOTE_CS5 554
```

Definition at line [62](#) of file [notes.h](#).

#### 15.61.2.42 NOTE\_CS6

```
#define NOTE_CS6 1109
```

Definition at line [74](#) of file [notes.h](#).

#### 15.61.2.43 NOTE\_CS7

```
#define NOTE_CS7 2217
```

Definition at line [86](#) of file [notes.h](#).

#### 15.61.2.44 NOTE\_CS8

```
#define NOTE_CS8 4435
```

Definition at line [98](#) of file [notes.h](#).

#### 15.61.2.45 NOTE\_CS9

```
#define NOTE_CS9 8870
```

Definition at line [110](#) of file [notes.h](#).

#### 15.61.2.46 NOTE\_D1

```
#define NOTE_D1 37
```

Definition at line [15](#) of file [notes.h](#).

#### 15.61.2.47 NOTE\_D2

```
#define NOTE_D2 73
```

Definition at line [27](#) of file [notes.h](#).

#### 15.61.2.48 NOTE\_D3

```
#define NOTE_D3 147
```

Definition at line [39](#) of file [notes.h](#).

**15.61.2.49 NOTE\_D4**

```
#define NOTE_D4 294
```

Definition at line [51](#) of file [notes.h](#).

**15.61.2.50 NOTE\_D5**

```
#define NOTE_D5 587
```

Definition at line [63](#) of file [notes.h](#).

**15.61.2.51 NOTE\_D6**

```
#define NOTE_D6 1175
```

Definition at line [75](#) of file [notes.h](#).

**15.61.2.52 NOTE\_D7**

```
#define NOTE_D7 2349
```

Definition at line [87](#) of file [notes.h](#).

**15.61.2.53 NOTE\_D8**

```
#define NOTE_D8 4699
```

Definition at line [99](#) of file [notes.h](#).

**15.61.2.54 NOTE\_D9**

```
#define NOTE_D9 9398
```

Definition at line [111](#) of file [notes.h](#).

**15.61.2.55 NOTE\_DS1**

```
#define NOTE_DS1 39
```

Definition at line [16](#) of file [notes.h](#).

**15.61.2.56 NOTE\_DS2**

```
#define NOTE_DS2 78
```

Definition at line [28](#) of file [notes.h](#).

**15.61.2.57 NOTE\_DS3**

```
#define NOTE_DS3 156
```

Definition at line [40](#) of file [notes.h](#).

**15.61.2.58 NOTE\_DS4**

```
#define NOTE_DS4 311
```

Definition at line [52](#) of file [notes.h](#).

#### 15.61.2.59 NOTE\_DS5

```
#define NOTE_DS5 622
```

Definition at line [64](#) of file [notes.h](#).

#### 15.61.2.60 NOTE\_DS6

```
#define NOTE_DS6 1245
```

Definition at line [76](#) of file [notes.h](#).

#### 15.61.2.61 NOTE\_DS7

```
#define NOTE_DS7 2489
```

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

#### 15.61.2.62 NOTE\_DS8

```
#define NOTE_DS8 4978
```

Definition at line [100](#) of file [notes.h](#).

#### 15.61.2.63 NOTE\_DS9

```
#define NOTE_DS9 9956
```

Definition at line [112](#) of file [notes.h](#).

#### 15.61.2.64 NOTE\_E1

```
#define NOTE_E1 41
```

Definition at line [17](#) of file [notes.h](#).

#### 15.61.2.65 NOTE\_E2

```
#define NOTE_E2 82
```

Definition at line [29](#) of file [notes.h](#).

#### 15.61.2.66 NOTE\_E3

```
#define NOTE_E3 165
```

Definition at line [41](#) of file [notes.h](#).

#### 15.61.2.67 NOTE\_E4

```
#define NOTE_E4 330
```

Definition at line [53](#) of file [notes.h](#).

#### 15.61.2.68 NOTE\_E5

```
#define NOTE_E5 659
```

Definition at line [65](#) of file [notes.h](#).

**15.61.2.69 NOTE\_E6**

```
#define NOTE_E6 1319
```

Definition at line [77](#) of file [notes.h](#).

**15.61.2.70 NOTE\_E7**

```
#define NOTE_E7 2637
```

Definition at line [89](#) of file [notes.h](#).

**15.61.2.71 NOTE\_E8**

```
#define NOTE_E8 5274
```

Definition at line [101](#) of file [notes.h](#).

**15.61.2.72 NOTE\_E9**

```
#define NOTE_E9 10548
```

Definition at line [113](#) of file [notes.h](#).

**15.61.2.73 NOTE\_F1**

```
#define NOTE_F1 44
```

Definition at line [18](#) of file [notes.h](#).

**15.61.2.74 NOTE\_F2**

```
#define NOTE_F2 87
```

Definition at line [30](#) of file [notes.h](#).

**15.61.2.75 NOTE\_F3**

```
#define NOTE_F3 175
```

Definition at line [42](#) of file [notes.h](#).

**15.61.2.76 NOTE\_F4**

```
#define NOTE_F4 349
```

Definition at line [54](#) of file [notes.h](#).

**15.61.2.77 NOTE\_F5**

```
#define NOTE_F5 698
```

Definition at line [66](#) of file [notes.h](#).

**15.61.2.78 NOTE\_F6**

```
#define NOTE_F6 1397
```

Definition at line [78](#) of file [notes.h](#).

#### 15.61.2.79 NOTE\_F7

```
#define NOTE_F7 2794
```

Definition at line [90](#) of file [notes.h](#).

#### 15.61.2.80 NOTE\_F8

```
#define NOTE_F8 5588
```

Definition at line [102](#) of file [notes.h](#).

#### 15.61.2.81 NOTE\_F9

```
#define NOTE_F9 11176
```

Definition at line [114](#) of file [notes.h](#).

#### 15.61.2.82 NOTE\_FS1

```
#define NOTE_FS1 46
```

Definition at line [19](#) of file [notes.h](#).

#### 15.61.2.83 NOTE\_FS2

```
#define NOTE_FS2 93
```

Definition at line [31](#) of file [notes.h](#).

#### 15.61.2.84 NOTE\_FS3

```
#define NOTE_FS3 185
```

Definition at line [43](#) of file [notes.h](#).

#### 15.61.2.85 NOTE\_FS4

```
#define NOTE_FS4 370
```

Definition at line [55](#) of file [notes.h](#).

#### 15.61.2.86 NOTE\_FS5

```
#define NOTE_FS5 740
```

Definition at line [67](#) of file [notes.h](#).

#### 15.61.2.87 NOTE\_FS6

```
#define NOTE_FS6 1480
```

Definition at line [79](#) of file [notes.h](#).

#### 15.61.2.88 NOTE\_FS7

```
#define NOTE_FS7 2960
```

Definition at line [91](#) of file [notes.h](#).

**15.61.2.89 NOTE\_FS8**

```
#define NOTE_FS8 5920
Definition at line 103 of file notes.h.
```

**15.61.2.90 NOTE\_FS9**

```
#define NOTE_FS9 11840
Definition at line 115 of file notes.h.
```

**15.61.2.91 NOTE\_G1**

```
#define NOTE_G1 49
Definition at line 20 of file notes.h.
```

**15.61.2.92 NOTE\_G2**

```
#define NOTE_G2 98
Definition at line 32 of file notes.h.
```

**15.61.2.93 NOTE\_G3**

```
#define NOTE_G3 196
Definition at line 44 of file notes.h.
```

**15.61.2.94 NOTE\_G4**

```
#define NOTE_G4 392
Definition at line 56 of file notes.h.
```

**15.61.2.95 NOTE\_G5**

```
#define NOTE_G5 784
Definition at line 68 of file notes.h.
```

**15.61.2.96 NOTE\_G6**

```
#define NOTE_G6 1568
Definition at line 80 of file notes.h.
```

**15.61.2.97 NOTE\_G7**

```
#define NOTE_G7 3136
Definition at line 92 of file notes.h.
```

**15.61.2.98 NOTE\_G8**

```
#define NOTE_G8 6272
Definition at line 104 of file notes.h.
```

### 15.61.2.99 NOTE\_G9

```
#define NOTE_G9 12544
```

Definition at line 116 of file [notes.h](#).

### 15.61.2.100 NOTE\_GS1

```
#define NOTE_GS1 52
```

Definition at line 21 of file [notes.h](#).

### 15.61.2.101 NOTE\_GS2

```
#define NOTE_GS2 104
```

Definition at line 33 of file [notes.h](#).

### 15.61.2.102 NOTE\_GS3

```
#define NOTE_GS3 208
```

Definition at line 45 of file [notes.h](#).

### 15.61.2.103 NOTE\_GS4

```
#define NOTE_GS4 415
```

Definition at line 57 of file [notes.h](#).

### 15.61.2.104 NOTE\_GS5

```
#define NOTE_GS5 831
```

Definition at line 69 of file [notes.h](#).

### 15.61.2.105 NOTE\_GS6

```
#define NOTE_GS6 1661
```

Definition at line 81 of file [notes.h](#).

### 15.61.2.106 NOTE\_GS7

```
#define NOTE_GS7 3322
```

Definition at line 93 of file [notes.h](#).

### 15.61.2.107 NOTE\_GS8

```
#define NOTE_GS8 6644
```

Definition at line 105 of file [notes.h](#).

### 15.61.2.108 NOTE\_GS9

```
#define NOTE_GS9 13288
```

Definition at line 117 of file [notes.h](#).

## 15.62 notes.h

[Go to the documentation of this file.](#)

```
00001  
00012 #define NOTE_B0 31  
00013 #define NOTE_C1 33  
00014 #define NOTE_CS1 35  
00015 #define NOTE_D1 37  
00016 #define NOTE_DS1 39  
00017 #define NOTE_E1 41  
00018 #define NOTE_F1 44  
00019 #define NOTE_FS1 46  
00020 #define NOTE_G1 49  
00021 #define NOTE_GS1 52  
00022 #define NOTE_A1 55  
00023 #define NOTE_AS1 58  
00024 #define NOTE_B1 62  
00025 #define NOTE_C2 65  
00026 #define NOTE_CS2 69  
00027 #define NOTE_D2 73  
00028 #define NOTE_DS2 78  
00029 #define NOTE_E2 82  
00030 #define NOTE_F2 87  
00031 #define NOTE_FS2 93  
00032 #define NOTE_G2 98  
00033 #define NOTE_GS2 104  
00034 #define NOTE_A2 110  
00035 #define NOTE_AS2 117  
00036 #define NOTE_B2 123  
00037 #define NOTE_C3 131  
00038 #define NOTE_CS3 139  
00039 #define NOTE_D3 147  
00040 #define NOTE_DS3 156  
00041 #define NOTE_E3 165  
00042 #define NOTE_F3 175  
00043 #define NOTE_FS3 185  
00044 #define NOTE_G3 196  
00045 #define NOTE_GS3 208  
00046 #define NOTE_A3 220  
00047 #define NOTE_AS3 233  
00048 #define NOTE_B3 247  
00049 #define NOTE_C4 262  
00050 #define NOTE_CS4 277  
00051 #define NOTE_D4 294  
00052 #define NOTE_DS4 311  
00053 #define NOTE_E4 330  
00054 #define NOTE_F4 349  
00055 #define NOTE_FS4 370  
00056 #define NOTE_G4 392  
00057 #define NOTE_GS4 415  
00058 #define NOTE_A4 440  
00059 #define NOTE_AS4 466  
00060 #define NOTE_B4 494  
00061 #define NOTE_C5 523  
00062 #define NOTE_CS5 554  
00063 #define NOTE_D5 587  
00064 #define NOTE_DS5 622  
00065 #define NOTE_E5 659  
00066 #define NOTE_F5 698  
00067 #define NOTE_FS5 740  
00068 #define NOTE_G5 784  
00069 #define NOTE_GS5 831  
00070 #define NOTE_A5 880  
00071 #define NOTE_AS5 932  
00072 #define NOTE_B5 988  
00073 #define NOTE_C6 1047  
00074 #define NOTE_CS6 1109  
00075 #define NOTE_D6 1175  
00076 #define NOTE_DS6 1245  
00077 #define NOTE_E6 1319  
00078 #define NOTE_F6 1397  
00079 #define NOTE_FS6 1480  
00080 #define NOTE_G6 1568  
00081 #define NOTE_GS6 1661  
00082 #define NOTE_A6 1760  
00083 #define NOTE_AS6 1865  
00084 #define NOTE_B6 1976  
00085 #define NOTE_C7 2093  
00086 #define NOTE_CS7 2217  
00087 #define NOTE_D7 2349  
00088 #define NOTE_DS7 2489  
00089 #define NOTE_E7 2637  
00090 #define NOTE_F7 2794  
00091 #define NOTE_FS7 2960  
00092 #define NOTE_G7 3136  
00093 #define NOTE_GS7 3322
```

```
00094 #define NOTE_A7 3520
00095 #define NOTE_A8 3729
00096 #define NOTE_B7 3951
00097 #define NOTE_C8 4186
00098 #define NOTE_CS8 4435
00099 #define NOTE_D8 4699
00100 #define NOTE_DS8 4978
00101 #define NOTE_E8 5274
00102 #define NOTE_F8 5588
00103 #define NOTE_FS8 5920
00104 #define NOTE_G8 6272
00105 #define NOTE_GS8 6644
00106 #define NOTE_A8 7040
00107 #define NOTE_AS8 7458
00108 #define NOTE_B8 7902
00109 #define NOTE_C9 8372
00110 #define NOTE_CS9 8870
00111 #define NOTE_D9 9398
00112 #define NOTE_DS9 9956
00113 #define NOTE_E9 10548
00114 #define NOTE_F9 11176
00115 #define NOTE_FS9 11840
00116 #define NOTE_G9 12544
00117 #define NOTE_GS9 13288
00118 #define NOTE_A9 14080
00119 #define NOTE_AS9 14916
00120 #define NOTE_B9 15804
```

## 15.63 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0004-DCDC/DCDC.h File Reference

Libreria que describe la clase del DCDC.

```
#include <Wire.h>
```

### Classes

- class [dcdc\\_controller](#)

### Variables

- const uint16\_t [LenDCDCvalues](#) = 121
- const byte [ADDR\\_I2C\\_DCDC](#) = 0
- const bool [C\\_BOOST\\_MODE](#) = true
- const bool [C\\_NON\\_BOOST\\_MODE](#) = false
- [PROGMEM](#) const byte [BOOST\\_ARRAY](#) [[LenDCDCvalues](#)] = {36, 164, 100, 228, 20, 148, 84, 52, 180, 116, 244, 12, 140, 76, 204, 44, 172, 108, 236, 28, 92, 220, 60, 188, 124, 252, 2, 130, 194, 34, 162, 98, 226, 18, 146, 82, 50, 178, 114, 242, 10, 138, 74, 202, 42, 170, 106, 234, 26, 90, 218, 58, 186, 122, 250, 6, 70, 198, 38, 166, 102, 230, 22, 150, 214, 54, 182, 118, 246, 14, 142, 78, 206, 46, 174, 110, 238, 30, 94, 222, 62, 190, 126, 254, 193, 33, 161, 97, 17, 145, 81, 209, 49, 177, 113, 241, 9, 137, 73, 201, 41, 105, 233, 25, 153, 89, 217, 57, 121, 249, 5, 133, 69, 197, 37, 101, 229, 21, 149, 85, 213}
- [PROGMEM](#) const byte [NON\\_BOOST\\_ARRAY](#) [[LenDCDCvalues](#)] = {248, 4, 132, 68, 36, 164, 100, 228, 20, 148, 84, 52, 180, 116, 244, 12, 140, 76, 204, 44, 172, 108, 236, 28, 92, 220, 60, 188, 124, 252, 2, 130, 194, 34, 162, 98, 226, 18, 146, 82, 50, 178, 114, 242, 10, 138, 74, 202, 42, 170, 106, 234, 26, 90, 218, 58, 186, 122, 250, 6, 70, 198, 38, 166, 102, 230, 22, 150, 214, 54, 182, 118, 246, 14, 142, 78, 206, 46, 174, 110, 238, 30, 94, 222, 62, 190, 126, 254, 193, 33, 161, 97, 17, 145, 81, 209, 49, 177, 113, 241, 9, 137, 73, 201, 41, 105, 233, 25, 153, 89, 217, 57, 121, 249, 5, 133, 69, 197, 37, 101, 229}

### 15.63.1 Detailed Description

Libreria que describe la clase del DCDC.

**Author**

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

**Version**

2

**Date**

2020-12-18

**Copyright**

Copyright (c) 2020

Definition in file [DCDC.h](#).

## 15.63.2 Variable Documentation

### 15.63.2.1 ADDR\_I2C\_DCDC

```
const byte ADDR_I2C_DCDC = 0
```

Definition at line [14](#) of file [DCDC.h](#).

### 15.63.2.2 BOOST\_ARRAY

```
PROGMEM const byte BOOST_ARRAY[LenDCDCvalues] = {36, 164, 100, 228, 20, 148, 84, 52, 180, 116,
244, 12, 140, 76, 204, 44, 172, 108, 236, 28, 92, 220, 60, 188, 124, 252, 2, 130, 194, 34,
162, 98, 226, 18, 146, 82, 50, 178, 114, 242, 10, 138, 74, 202, 42, 170, 106, 234, 26, 90,
218, 58, 186, 122, 250, 6, 70, 198, 38, 166, 102, 230, 22, 150, 214, 54, 182, 118, 246, 14,
142, 78, 206, 46, 174, 110, 238, 30, 94, 222, 62, 190, 126, 254, 193, 33, 161, 97, 17, 145,
81, 209, 49, 177, 113, 241, 9, 137, 73, 201, 41, 105, 233, 25, 153, 89, 217, 57, 121, 249, 5,
133, 69, 197, 37, 101, 229, 21, 149, 85, 213}
```

Definition at line [17](#) of file [DCDC.h](#).

### 15.63.2.3 C\_BOOST\_MODE

```
const bool C_BOOST_MODE = true
```

Definition at line [15](#) of file [DCDC.h](#).

### 15.63.2.4 C\_NON\_BOOST\_MODE

```
const bool C_NON_BOOST_MODE = false
```

Definition at line [16](#) of file [DCDC.h](#).

### 15.63.2.5 LenDCDCvalues

```
const uint16_t LenDCDCvalues = 121
```

Definition at line [13](#) of file [DCDC.h](#).

### 15.63.2.6 NON\_BOOST\_ARRAY

```
PROGMEM const byte NON_BOOST_ARRAY[LenDCDCvalues] = {248, 4, 132, 68, 36, 164, 100, 228, 20, 148, 84, 52, 180, 116, 244, 12, 140, 76, 204, 44, 172, 108, 236, 28, 92, 220, 60, 188, 124, 252, 2, 130, 194, 34, 162, 98, 226, 18, 146, 82, 50, 178, 114, 242, 10, 138, 74, 202, 42, 170, 106, 234, 26, 90, 218, 58, 186, 122, 250, 6, 70, 198, 38, 166, 102, 230, 22, 150, 214, 54, 182, 118, 246, 14, 142, 78, 206, 46, 174, 110, 238, 30, 94, 222, 62, 190, 126, 254, 193, 33, 161, 97, 17, 145, 81, 209, 49, 177, 113, 241, 9, 137, 73, 201, 41, 105, 233, 25, 153, 89, 217, 57, 121, 249, 5, 133, 69, 197, 37, 101, 229}
```

Definition at line 19 of file DCDC.h.

## 15.64 DCDC.h

[Go to the documentation of this file.](#)

```
00001
00011 #include <Wire.h>
00012
00013 const uint16_t LenDCDCvalues = 121;
00014 const byte ADDR_I2C_DCDC = 0;
00015 const bool C_BOOST_MODE = true;
00016 const bool C_NON_BOOST_MODE = false;
00017 PROGMEM const byte BOOST_ARRAY[LenDCDCvalues] = {36, 164, 100, 228, 20, 148, 84, 52, 180, 116, 244, 12, 140, 76, 204, 44, 172, 108, 236, 28, 92, 220, 60, 188, 124, 252, 2, 130, 194, 34, 162, 98, 226, 18, 146, 82, 50, 178, 114, 242, 10, 138, 74, 202, 42, 170, 106, 234, 26, 90, 218, 58, 186, 122, 250, 6, 70, 198, 38, 166, 102, 230, 22, 150, 214, 54, 182, 118, 246, 14, 142, 78, 206, 46, 174, 110, 238, 30, 94, 222, 62, 190, 126, 254, 193, 33, 161, 97, 17, 145, 81, 209, 49, 177, 113, 241, 9, 137, 73, 201, 41, 105, 233, 25, 153, 89, 217, 57, 121, 249, 5, 133, 69, 197, 37, 101, 229, 21, 149, 85, 213};
00018
00019 PROGMEM const byte NON_BOOST_ARRAY[LenDCDCvalues] = {248, 4, 132, 68, 36, 164, 100, 228, 20, 148, 84, 52, 180, 116, 244, 12, 140, 76, 204, 44, 172, 108, 236, 28, 92, 220, 60, 188, 124, 252, 2, 130, 194, 34, 162, 98, 226, 18, 146, 82, 50, 178, 114, 242, 10, 138, 74, 202, 42, 170, 106, 234, 26, 90, 218, 58, 186, 122, 250, 6, 70, 198, 38, 166, 102, 230, 22, 150, 214, 54, 182, 118, 246, 14, 142, 78, 206, 46, 174, 110, 238, 30, 94, 222, 62, 190, 126, 254, 193, 33, 161, 97, 17, 145, 81, 209, 49, 177, 113, 241, 9, 137, 73, 201, 41, 105, 233, 25, 153, 89, 217, 57, 121, 249, 5, 133, 69, 197, 37, 101, 229};
00020
00021 class dcdc_controller
00022 {
00023     private:
00030         void Write_TPIC2810(byte address, byte data)
00031     {
00032         Wire.beginTransmission(byte(96)); // transmit command to device TPIC2810
00033         Wire.write(byte(68));           // Command to transfer next value to output register
00034         Wire.write(byte(data));
00035         Wire.endTransmission(); // stop transmitting
00036     }
00037
00038     public:
00039         uint16_t encoderPos;
00040         byte TPICvalue;
00041         int16_t pin_enable;
00042
00043         dcdc_controller(int16_t pin)
00044     {
00050
00051             pin_enable = pin;
00052             pinMode(pin_enable, OUTPUT);
00053             digitalWrite(pin_enable, LOW);
00054     }
00055
00062         void SetVoltage(int volt, bool mode)
00063     {
00064             volt = constrain(volt, 40, 160);
00065             encoderPos = volt - 40;
00066             if (mode == C_BOOST_MODE)
00067             {
00068                 TPICvalue = pgm_read_byte_near(BOOST_ARRAY + encoderPos);
00069             }
00070             else if (mode == C_NON_BOOST_MODE)
00071             {
00072                 TPICvalue = pgm_read_byte_near(NON_BOOST_ARRAY + encoderPos);
00073             }
00074
00075             Write_TPIC2810(ADDR_I2C_DCDC, TPICvalue);
00076     }
00082         void EnableDCDC(bool enable)
00083     {
00084             digitalWrite(pin_enable, enable);
00085     }
00086 },
```

## 15.65 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0005-SENSING/HealthMonitor.h File Reference

```
#include <batt_SAMD_AnalogCorrection.h>
```

### Classes

- class [HealthMonitor](#)

### Variables

- const int16\_t [C\\_TIME\\_HEALTH\\_MONITORING](#) = 50
- const int16\_t [C\\_SAMPLING\\_TIME](#) = 10
- const int16\_t [C\\_RISE\\_COUNT](#) = 20
- const int16\_t [C\\_COUNTER\\_LIMIT](#) = [C\\_RISE\\_COUNT](#) \* [C\\_TIME\\_HEALTH\\_MONITORING](#) / [C\\_SAMPLING\\_TIME](#)

#### 15.65.1 Detailed Description

##### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

##### Version

4

##### Date

2020-3-18

##### Copyright

Copyright (c) 2020

Definition in file [HealthMonitor.h](#).

#### 15.65.2 Variable Documentation

##### 15.65.2.1 C\_COUNTER\_LIMIT

```
const int16_t C_COUNTER_LIMIT = C\_RISE\_COUNT * C\_TIME\_HEALTH\_MONITORING / C\_SAMPLING\_TIME
Definition at line 18 of file HealthMonitor.h.
```

##### 15.65.2.2 C\_RISE\_COUNT

```
const int16_t C_RISE_COUNT = 20
Definition at line 16 of file HealthMonitor.h.
```

##### 15.65.2.3 C\_SAMPLING\_TIME

```
const int16_t C_SAMPLING_TIME = 10
Definition at line 14 of file HealthMonitor.h.
```

### 15.65.2.4 C\_TIME\_HEALTH\_MONITORING

const int16\_t C\_TIME\_HEALTH\_MONITORING = 50  
 Definition at line 13 of file [HealthMonitor.h](#).

## 15.66 HealthMonitor.h

[Go to the documentation of this file.](#)

```

00001
00011 #include <batt_SAMD_AnalogCorrection.h>
00012
00013 const int16_t C_TIME_HEALTH_MONITORING = 50; // ms
00014 const int16_t C_SAMPLING_TIME = 10;           // ms
00015
00016 const int16_t C_RISE_COUNT = 20;
00017
00018 const int16_t C_COUNTER_LIMIT = C_RISE_COUNT * C_TIME_HEALTH_MONITORING / C_SAMPLING_TIME;
00019
00020 class HealthMonitor
00021 {
00022   // Class Member Variables
00023   uint16_t counter; // private, only accesible by setCounter method
00024
00025 public:
00026   uint16_t threshold;
00027   uint8_t rampUPinc;
00028   uint8_t rampDOWNdec;
00029   uint16_t limit;
00030   bool alarm;
00031   int16_t sense_values[8]; // Filter samples
00032
00041 HealthMonitor(uint16_t thres, uint8_t inc, uint8_t dec, uint16_t lim)
00042 {
00043   threshold = thres;
00044   rampUPinc = inc;
00045   rampDOWNdec = dec;
00046   limit = lim;
00047   counter = 0;
00048 }
00054 void setCounter(uint16_t value)
00055 {
00056   counter = constrain(value, 0, UINT16_MAX - 1);
00057 }
00058
00064 uint16_t getCounter()
00065 {
00066   uint16_t percent;
00067
00068   percent = counter * 100 / C_COUNTER_LIMIT;
00069
00070   return percent;
00071 }
00072
00079 boolean check(uint16_t sample)
00080 {
00081
00082   if (sample > threshold)
00083   {
00084     counter = constrain(counter, 0, UINT16_MAX - rampUPinc); // prevent counter overflow before
00085     counter = counter + rampUPinc;
00086   }
00087   else
00088   {
00089     counter = constrain(counter, rampDOWNdec, UINT16_MAX - rampUPinc); // prevent counter overflow
00090     counter = counter - rampDOWNdec;
00091   }
00092
00093   if (counter >= limit)
00094   {
00095     counter = constrain(counter, rampDOWNdec, limit);
00096     alarm = true;
00097   }
00098   else
00099   {
00100     alarm = false;
00101   }
00102   return (alarm);
00103 }
00104
00111 uint16_t getSample(uint16_t ADCpin)
00112 {

```

```

0013     analogReadCorrection(12, 2055);
0014     analogReadResolution(12); // Set analog input resolution to max, 12-bits
0015     int16_t out_sample = 0;
0016     int16_t sample = 0;
0017     for (int i = (sizeof(sense_values) / sizeof(sense_values[0])) - 1; i >= 1; i--)
0018     {
0019         sense_values[i] = sense_values[i - 1];
0020         out_sample += sense_values[i];
0021     }
0022
0023     for (int i = 0; i < 8; i++)
0024     {
0025         sample += analogRead(ADCpin);
0026     }
0027
0028     sample = (sample / 8 + out_sample) / 8;
0029     sense_values[0] = sample;
0030     return sample;
0031 }
0032 };

```

## 15.67 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0006-POWERBAR/power\_bar.h File Reference

Esta libreria contiene las funciones y constantes necesarias para controlar la ultima fila de leds de matriz de leds de 15x7, con el funcion de reproducir el funcionamiento de una barra de potencia. Esta barra iluminara mas leds o menos en funcion de la potencia que se este suministrando por parte de la bateria a la maquina.

### Functions

- void `PowerBar` (int16\_t leds, Adafruit\_IS31FL3731\_Wing ledmatrix, bool mode\_bright=`C_MODE_HIGH_BRIGHT`)  
*Esta funcion se encarga de encender y apagar los leds necesario en la barra de potencia.*
- void `UpdatePowerBar` (int16\_t power\_sample, Adafruit\_IS31FL3731\_Wing ledmatrix, bool mode\_bright\_display)  
*Actualiza la cantidad de leds encendidos en la barra de potencia segun el parametro de entrada "power\_sample".*

### Variables

- const int16\_t `MAX_POWER_DISPLAYED` = 2500
- const int16\_t `LEDS_IN_POWERBAR` = 15
- const int16\_t `power_by_led` = `MAX_POWER_DISPLAYED` / `LEDS_IN_POWERBAR`
- MilliTimmer `refresh_timer`

#### 15.67.1 Detailed Description

Esta libreria contiene las funciones y constantes necesarias para controlar la ultima fila de leds de matriz de leds de 15x7, con el funcion de reproducir el funcionamiento de una barra de potencia. Esta barra iluminara mas leds o menos en funcion de la potencia que se este suministrando por parte de la bateria a la maquina.

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com) )

#### Version

1

#### Date

2021-02-24

#### Copyright

Copyright (c) 2021

Definition in file [power\\_bar.h](#).

## 15.67.2 Function Documentation

### 15.67.2.1 PowerBar()

```
void PowerBar (
    int16_t leds,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool mode_bright = C_MODE_HIGH_BRIGHT )
```

Esta funcion se encarga de encender y apagar los leds necesario en la barra de potencia.

#### Parameters

leds	
ledmatrix	
mode_bright	

Definition at line [22](#) of file [power\\_bar.h](#).

### 15.67.2.2 UpdatePowerBar()

```
void UpdatePowerBar (
    int16_t power_sample,
    Adafruit_IS31FL3731_Wing ledmatrix,
    bool mode_bright_display )
```

Actualiza la cantidad de leds encendidos en la barra de potencia segun el parametro de entrada "power\_sample".

#### Parameters

power_sample	
ledmatrix	
mode_bright_display	

Definition at line [64](#) of file [power\\_bar.h](#).

## 15.67.3 Variable Documentation

### 15.67.3.1 LEDS\_IN\_POWERBAR

```
const int16_t LEDS_IN_POWERBAR = 15
```

Definition at line [16](#) of file [power\\_bar.h](#).

### 15.67.3.2 MAX\_POWER\_DISPLAYED

```
const int16_t MAX_POWER_DISPLAYED = 2500
```

Definition at line [15](#) of file [power\\_bar.h](#).

### 15.67.3.3 power\_by\_led

```
const int16_t power_by_led = MAX_POWER_DISPLAYED / LEDS_IN_POWERBAR
```

Definition at line [18](#) of file [power\\_bar.h](#).

### 15.67.3.4 refresh\_timer

`MilliTimmer refresh_timer`  
 Definition at line 20 of file [power\\_bar.h](#).

## 15.68 power\_bar.h

Go to the documentation of this file.

```

00001
00012 //#include <Adafruit_IS31FL3731.h>
00013 //#include <display.h>
00014 //#include <MilliTimmer.h>
00015 const int16_t MAX_POWER_DISPLAYED = 2500;
00016 const int16_t LEDS_IN_POWERBAR = 15;
00017
00018 const int16_t power_by_led = MAX_POWER_DISPLAYED / LEDS_IN_POWERBAR;
00019
00020 MilliTimmer refresh_timer;
00021
00022 void PowerBar(int16_t leds, Adafruit_IS31FL3731_Wing ledmatrix, bool mode_bright = C_MODE_HIGH_BRIGHT)
00030 {
00031     leds = constrain(leds, 0, LEDS_IN_POWERBAR);
00032     if (mode_bright == C_MODE_HIGH_BRIGHT)
00033     {
00034         C_DISPLAY_ON = C_HIGH_BRIGHTNESS;
00035     }
00036     else if (mode_bright == C_MODE_LOW_BRIGHT)
00037     {
00038         C_DISPLAY_ON = C_LOW_BRIGHTNESS;
00039     }
00040     if (leds == 0)
00041     {
00042         ledmatrix.drawFastHLine(0, C_PWBART_Y_AXE, LEDS_IN_POWERBAR, C_DISPLAY_OFF);
00043     }
00044     else
00045     {
00046         for (int16_t i = 0; i <= leds; i++)
00047         {
00048             ledmatrix.drawPixel(i, C_PWBART_Y_AXE, C_DISPLAY_ON);
00049         }
00050         for (int16_t i = leds; i < LEDS_IN_POWERBAR; i++)
00051         {
00052             ledmatrix.drawPixel(i, C_PWBART_Y_AXE, C_DISPLAY_OFF);
00053         }
00054     }
00055 }
00064 void UpdatePowerBar(int16_t power_sample, Adafruit_IS31FL3731_Wing ledmatrix, bool
  mode_bright_display)
00065 {
00066     static int16_t high_sample = 0;
00067
00068     if (high_sample < power_sample)
00069     {
00070         high_sample = power_sample;
00071     }
00072
00073     if (refresh_timer.poll(50) != C_TIMER_NOT_EXPIRED)
00074     {
00075         uint16_t num_leds = (high_sample + (power_by_led / 2)) / power_by_led;
00076         PowerBar(num_leds, ledmatrix, mode_bright_display);
00077         high_sample = 0;
00078     }
00079 }
```

## 15.69 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0007-SlowSoftI2CMaster/batt\_SlowSoft←  
 I2CMaster.cpp File Reference

```
#include <batt_SlowSoftI2CMaster.h>
```

## 15.70 batt\_SlowSoftI2CMaster.cpp

[Go to the documentation of this file.](#)

```

00001 /* Arduino Slow Software I2C Master
00002 Copyright (c) 2017 Bernhard Nebel.
00003
00004 This library is free software; you can redistribute it and/or
00005 modify it under the terms of the GNU Lesser General Public License
00006 as published by the Free Software Foundation; either version 3 of
00007 the License, or (at your option) any later version.
00008
00009 This library is distributed in the hope that it will be useful, but
00010 WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
00012 Lesser General Public License for more details.
00013
00014 You should have received a copy of the GNU Lesser General Public
00015 License along with this library; if not, write to the Free Software
00016 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301
00017 USA
00018 */
00019
00020 #include <batt_SlowSoftI2CMaster.h>
00021
00022 SlowSoftI2CMaster::SlowSoftI2CMaster(uint8_t sda, uint8_t scl) {
00023     _sda = sda;
00024     _scl = scl;
00025     _pullup = false;
00026 }
00027
00028 SlowSoftI2CMaster::SlowSoftI2CMaster(uint8_t sda, uint8_t scl, bool pullup) {
00029     _sda = sda;
00030     _scl = scl;
00031     _pullup = pullup;
00032 }
00033 // Init function. Needs to be called once in the beginning.
00034 // Returns false if SDA or SCL are low, which probably means
00035 // a I2C bus lockup or that the lines are not pulled up.
00036 bool SlowSoftI2CMaster::i2c_init(void) {
00037     digitalWrite(_sda, LOW);
00038     digitalWrite(_scl, LOW);
00039     setHigh(_sda);
00040     setHigh(_scl);
00041     if (digitalRead(_sda) == LOW || digitalRead(_scl) == LOW) return false;
00042     return true;
00043 }
00044
00045 // Start transfer function: <addr> is the 8-bit I2C address (including the R/W
00046 // bit).
00047 // Return: true if the slave replies with an "acknowledge", false otherwise
00048 bool SlowSoftI2CMaster::i2c_start(uint8_t addr) {
00049     setLow(_sda);
00050     delayMicroseconds(DELAY);
00051     setLow(_scl);
00052     return i2c_write(addr);
00053 }
00054
00055 // Try to start transfer until an ACK is returned
00056 bool SlowSoftI2CMaster::i2c_start_wait(uint8_t addr) {
00057     long retry = I2C_MAXWAIT;
00058     while (!i2c_start(addr)) {
00059         i2c_stop();
00060         if (--retry == 0) return false;
00061     }
00062     return true;
00063 }
00064
00065 // Repeated start function: After having claimed the bus with a start condition,
00066 // you can address another or the same chip again without an intervening
00067 // stop condition.
00068 // Return: true if the slave replies with an "acknowledge", false otherwise
00069 bool SlowSoftI2CMaster::i2c_rep_start(uint8_t addr) {
00070     setHigh(_sda);
00071     setHigh(_scl);
00072     delayMicroseconds(DELAY);
00073     return i2c_start(addr);
00074 }
00075
00076 // Issue a stop condition, freeing the bus.
00077 void SlowSoftI2CMaster::i2c_stop(void) {
00078     setLow(_sda);
00079     delayMicroseconds(DELAY);
00080     setHigh(_scl);
00081     delayMicroseconds(DELAY);
00082     setHigh(_sda);
00083     delayMicroseconds(DELAY);

```

```

00084 }
00085
00086 // Write one byte to the slave chip that had been addressed
00087 // by the previous start call. <value> is the byte to be sent.
00088 // Return: true if the slave replies with an "acknowledge", false otherwise
00089 bool SlowSoftI2CMaster::i2c_write(uint8_t value) {
00090     for (uint8_t curr = 0x80; curr != 0; curr >= 1) {
00091         if (curr & value) setHigh(_sda); else setLow(_sda);
00092         setHigh(_scl);
00093         delayMicroseconds(DELAY);
00094         setLow(_scl);
00095     }
00096     // get Ack or Nak
00097     setHigh(_sda);
00098     setHigh(_scl);
00099     delayMicroseconds(DELAY/2);
00100    uint8_t ack = digitalRead(_sda);
00101    setLow(_scl);
00102    delayMicroseconds(DELAY/2);
00103    setLow(_sda);
00104    return ack == 0;
00105 }
00106
00107 // Read one byte. If <last> is true, we send a NAK after having received
00108 // the byte in order to terminate the read sequence.
00109 uint8_t SlowSoftI2CMaster::i2c_read(bool last) {
00110     uint8_t b = 0;
00111     setHigh(_sda);
00112     for (uint8_t i = 0; i < 8; i++) {
00113         b <<= 1;
00114         delayMicroseconds(DELAY);
00115         setHigh(_scl);
00116         if (digitalRead(_sda)) b |= 1;
00117         setLow(_scl);
00118     }
00119     if (last) setHigh(_sda); else setLow(_sda);
00120     setHigh(_scl);
00121     delayMicroseconds(DELAY/2);
00122     setLow(_scl);
00123     delayMicroseconds(DELAY/2);
00124     setLow(_sda);
00125     return b;
00126 }
00127
00128 void SlowSoftI2CMaster::setLow(uint8_t pin) {
00129     noInterrupts();
00130     if (_pullup)
00131         digitalWrite(pin, LOW);
00132     pinMode(pin, OUTPUT);
00133     interrupts();
00134 }
00135
00136
00137 void SlowSoftI2CMaster::setHigh(uint8_t pin) {
00138     noInterrupts();
00139     if (_pullup)
00140         pinMode(pin, INPUT_PULLUP);
00141     else
00142         pinMode(pin, INPUT);
00143     interrupts();
00144 }
00145

```

## 15.71 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0007-SlowSoftI2CMaster/batt\_SlowSoftI2CMaster.h File Reference

```
#include <Arduino.h>
#include <iinttypes.h>
```

### Classes

- class [SlowSoftI2CMaster](#)

## Macros

- #define I2C\_READ 1
- #define I2C\_WRITE 0
- #define DELAY 4
- #define BUFFER\_LENGTH 32
- #define I2C\_MAXWAIT 5000

### 15.71.1 Macro Definition Documentation

#### 15.71.1.1 BUFFER\_LENGTH

```
#define BUFFER_LENGTH 32
Definition at line 28 of file batt_SlowSoftI2CMaster.h.
```

#### 15.71.1.2 DELAY

```
#define DELAY 4
Definition at line 27 of file batt_SlowSoftI2CMaster.h.
```

#### 15.71.1.3 I2C\_MAXWAIT

```
#define I2C_MAXWAIT 5000
Definition at line 29 of file batt_SlowSoftI2CMaster.h.
```

#### 15.71.1.4 I2C\_READ

```
#define I2C_READ 1
Definition at line 25 of file batt_SlowSoftI2CMaster.h.
```

#### 15.71.1.5 I2C\_WRITE

```
#define I2C_WRITE 0
Definition at line 26 of file batt_SlowSoftI2CMaster.h.
```

## 15.72 batt\_SlowSoftI2CMaster.h

[Go to the documentation of this file.](#)

```
00001 /* Arduino Slow Software I2C Master
00002 Copyright (c) 2017 Bernhard Nebel.
00003
00004 This library is free software; you can redistribute it and/or
00005 modify it under the terms of the GNU Lesser General Public
00006 License as published by the Free Software Foundation; either
00007 version 3 of the License, or (at your option) any later version.
00008
00009 This library is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
00012 Lesser General Public License for more details.
00013
00014 You should have received a copy of the GNU Lesser General Public
00015 License along with this library; if not, write to the Free Software
00016 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00017 */
00018
00019 #ifndef SLOW_SOFT_I2C_MASTER_H
00020 #define SLOW_SOFT_I2C_MASTER_H
00021
00022 #include <Arduino.h>
```

```

00023 #include <inttypes.h>
00024
00025 #define I2C_READ 1
00026 #define I2C_WRITE 0
00027 #define DELAY 4 // usec delay
00028 #define BUFFER_LENGTH 32
00029 #define I2C_MAXWAIT 5000
00030
00031 class SlowSoftI2CMaster {
00032 public:
00033     SlowSoftI2CMaster(uint8_t sda, uint8_t scl);
00034     SlowSoftI2CMaster(uint8_t sda, uint8_t scl, bool internal_pullup);
00035     bool i2c_init(void);
00036     bool i2c_start(uint8_t addr);
00037     bool i2c_start_wait(uint8_t addr);
00038     bool i2c_rep_start(uint8_t addr);
00039     void i2c_stop(void);
00040     bool i2c_write(uint8_t value);
00041     uint8_t i2c_read(bool last);
00042     bool error;
00043
00044 private:
00045     void setHigh(uint8_t pin);
00046     void setLow(uint8_t pin);
00047     uint8_t _sda;
00048     uint8_t _scl;
00049     bool _pullup;
00050 };
00051
00052 #endif

```

## 15.73 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/batt\_ Adafruit\_GFX.cpp File Reference

```
#include "batt_Adafruit_GFX.h"
#include "batt_glcdfont.c"
```

### Macros

- #define pgm\_read\_byte(addr) (\*(const unsigned char \*)(addr))
- #define pgm\_read\_word(addr) (\*(const unsigned short \*)(addr))
- #define pgm\_read\_dword(addr) (\*(const unsigned long \*)(addr))
- #define pgm\_read\_pointer(addr) ((void \*)pgm\_read\_dword(addr))
- #define min(a, b) (((a) < (b)) ? (a) : (b))
- #define \_swap\_int16\_t(a, b)

### Functions

- GFXglyph \* pgm\_read\_glyph\_ptr (const GFXfont \*gfxFont, uint8\_t c)
- uint8\_t \* pgm\_read\_bitmap\_ptr (const GFXfont \*gfxFont)

#### 15.73.1 Macro Definition Documentation

##### 15.73.1.1 \_swap\_int16\_t

```
#define _swap_int16_t(
    a,
    b )
```

##### Value:

```
{
    int16_t t = a;
```

```
a = b;  
b = t;  
}
```

Definition at line 94 of file [batt\\_Adafuit\\_GFX.cpp](#).

### 15.73.1.2 min

```
#define min(  
    a,  
    b ) (((a) < (b)) ? (a) : (b))
```

Definition at line 90 of file [batt\\_Adafuit\\_GFX.cpp](#).

### 15.73.1.3 pgm\_read\_byte

```
#define pgm_read_byte(  
    addr ) (*(const unsigned char *) (addr))
```

Definition at line 47 of file [batt\\_Adafuit\\_GFX.cpp](#).

### 15.73.1.4 pgm\_read\_dword

```
#define pgm_read_dword(  
    addr ) (*(const unsigned long *) (addr))
```

Definition at line 53 of file [batt\\_Adafuit\\_GFX.cpp](#).

### 15.73.1.5 pgm\_read\_pointer

```
#define pgm_read_pointer(  
    addr ) ((void *)pgm_read_dword(addr))
```

Definition at line 60 of file [batt\\_Adafuit\\_GFX.cpp](#).

### 15.73.1.6 pgm\_read\_word

```
#define pgm_read_word(  
    addr ) (*(const unsigned short *) (addr))
```

Definition at line 50 of file [batt\\_Adafuit\\_GFX.cpp](#).

## 15.73.2 Function Documentation

### 15.73.2.1 pgm\_read\_bitmap\_ptr()

```
uint8_t * pgm_read_bitmap_ptr (  
    const GFXfont * gfxFont ) [inline]
```

Definition at line 77 of file [batt\\_Adafuit\\_GFX.cpp](#).

### 15.73.2.2 pgm\_read\_glyph\_ptr()

```
GFXglyph * pgm_read_glyph_ptr (  
    const GFXfont * gfxFont,  
    uint8_t c ) [inline]
```

Definition at line 65 of file [batt\\_Adafuit\\_GFX.cpp](#).

## 15.74 batt\_Adafruit\_GFX.cpp

[Go to the documentation of this file.](#)

```

00001 /*
00002 This is the core graphics library for all our displays, providing a common
00003 set of graphics primitives (points, lines, circles, etc.). It needs to be
00004 paired with a hardware-specific library for each display device we carry
00005 (to handle the lower-level functions).
00006
00007 Adafruit invests time and resources providing this open source code, please
00008 support Adafruit & open-source hardware by purchasing products from Adafruit!
00009
00010 Copyright (c) 2013 Adafruit Industries. All rights reserved.
00011
00012 Redistribution and use in source and binary forms, with or without
00013 modification, are permitted provided that the following conditions are met:
00014
00015 - Redistributions of source code must retain the above copyright notice,
00016 this list of conditions and the following disclaimer.
00017 - Redistributions in binary form must reproduce the above copyright notice,
00018 this list of conditions and the following disclaimer in the documentation
00019 and/or other materials provided with the distribution.
00020
00021 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00022 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00023 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00024 ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
00025 LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00026 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00027 SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00028 INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00029 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00030 ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00031 POSSIBILITY OF SUCH DAMAGE.
00032 */
00033
00034 #include "batt_Adafruit_GFX.h"
00035 #include "batt_glcdfont.c"
00036 #ifdef __AVR__
00037 #include <avr/pgmspace.h>
00038 #elif defined(ESP8266) || defined(ESP32)
00039 #include <pgmspace.h>
00040 #endif
00041
00042 // Many (but maybe not all) non-AVR board installs define macros
00043 // for compatibility with existing PROGMEM-reading AVR code.
00044 // Do our own checks and defines here for good measure...
00045
00046 #ifndef pgm_read_byte
00047 #define pgm_read_byte(addr) (*(const unsigned char *) (addr))
00048 #endif
00049 #ifndef pgm_read_word
00050 #define pgm_read_word(addr) (*(const unsigned short *) (addr))
00051 #endif
00052 #ifndef pgm_read_dword
00053 #define pgm_read_dword(addr) (*(const unsigned long *) (addr))
00054 #endif
00055
00056 // Pointers are a peculiar case...typically 16-bit on AVR boards,
00057 // 32 bits elsewhere. Try to accommodate both...
00058
00059 #if !defined(__INT_MAX__) || (__INT_MAX__ > 0xFFFF)
00060 #define pgm_read_pointer(addr) ((void *)pgm_read_dword(addr))
00061 #else
00062 #define pgm_read_pointer(addr) ((void *)pgm_read_word(addr))
00063 #endif
00064
00065 inline GFXglyph *pgm_read_glyph_ptr(const GFXfont *gfxFont, uint8_t c) {
00066 #ifdef __AVR__
00067     return &((GFXglyph *)pgm_read_pointer(&gfxFont->glyph))[c];
00068 #else
00069     // expression in __AVR__ section may generate "dereferencing type-punned
00070     // pointer will break strict-aliasing rules" warning In fact, on other
00071     // platforms (such as STM32) there is no need to do this pointer magic as
00072     // program memory may be read in a usual way So expression may be simplified
00073     return gfxFont->glyph + c;
00074 #endif //__AVR__
00075 }
00076
00077 inline uint8_t *pgm_read_bitmap_ptr(const GFXfont *gfxFont) {
00078 #ifdef __AVR__
00079     return (uint8_t *)pgm_read_pointer(&gfxFont->bitmap);
00080 #else
00081     // expression in __AVR__ section generates "dereferencing type-punned pointer
00082     // will break strict-aliasing rules" warning In fact, on other platforms (such
00083     // as STM32) there is no need to do this pointer magic as program memory may

```

```
00084 // be read in a usual way So expression may be simplified
00085     return gfxFont->bitmap;
00086 #endif //__AVR__
00087 }
00088
00089 #ifndef min
00090 #define min(a, b) (((a) < (b)) ? (a) : (b))
00091 #endif
00092
00093 #ifndef _swap_int16_t
00094 #define _swap_int16_t(a, b)
00095 {
00096     int16_t t = a;
00097     a = b;
00098     b = t;
00099 }
00100#endif
00101
00102 /*****
00103 ****/
00104 Adafruit_GFX::Adafruit_GFX(int16_t w, int16_t h) : WIDTH(w), HEIGHT(h) {
00111     _width = WIDTH;
00112     _height = HEIGHT;
00113     rotation = 0;
00114     cursor_y = cursor_x = 0;
00115     textSize_x = textSize_y = 1;
00116     textColor = textbgcolor = 0xFFFF;
00117     wrap = true;
00118     _cp437 = false;
00119     gfxFont = NULL;
00120 }
00121
00122 /*****
00123 ****/
00124 void Adafruit_GFX::writeLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
00125                               uint16_t color) {
00126 #if defined(ESP8266)
00127     yield();
00128 #endif
00129     int16_t steep = abs(y1 - y0) > abs(x1 - x0);
00130     if (steep) {
00131         _swap_int16_t(x0, y0);
00132         _swap_int16_t(x1, y1);
00133     }
00134     if (x0 > x1) {
00135         _swap_int16_t(x0, x1);
00136         _swap_int16_t(y0, y1);
00137     }
00138     int16_t dx, dy;
00139     dx = x1 - x0;
00140     dy = abs(y1 - y0);
00141
00142     int16_t err = dx / 2;
00143     int16_t ystep;
00144
00145     if (y0 < y1) {
00146         ystep = 1;
00147     } else {
00148         ystep = -1;
00149     }
00150
00151     for (; x0 <= x1; x0++) {
00152         if (steep) {
00153             writePixel(y0, x0, color);
00154         } else {
00155             writePixel(x0, y0, color);
00156         }
00157         err -= dy;
00158         if (err < 0) {
00159             y0 += ystep;
00160             err += dx;
00161         }
00162     }
00163 }
00164
00165 /*****
00166 ****/
00167 void Adafruit_GFX::startWrite() {}
00168
00169 /*****
00170 ****/
00171 void Adafruit_GFX::writePixel(int16_t x, int16_t y, uint16_t color) {
00172     drawPixel(x, y, color);
00173 }
00174
00175 /*****
00176 ****/
00177 void Adafruit_GFX::drawPixel(int16_t x, int16_t y, uint16_t color) {
00178 }
```

```

00194 /*****
00203 *****/
00204 void Adafruit_GFX::writeFastVLine(int16_t x, int16_t y, int16_t h,
00205     uint16_t color) {
00206     // Overwrite in subclasses if startWrite is defined!
00207     // Can be just writeLine(x, y, x, y+h-1, color);
00208     // or writeFillRect(x, y, 1, h, color);
00209     drawFastVLine(x, y, h, color);
00210 }
00211
00212 /*****
00221 *****/
00222 void Adafruit_GFX::writeFastHLine(int16_t x, int16_t y, int16_t w,
00223     uint16_t color) {
00224     // Overwrite in subclasses if startWrite is defined!
00225     // Example: writeLine(x, y, x+w-1, y, color);
00226     // or writeFillRect(x, y, w, 1, color);
00227     drawFastHLine(x, y, w, color);
00228 }
00229
00230 /*****
00240 *****/
00241 void Adafruit_GFX::writeFillRect(int16_t x, int16_t y, int16_t w, int16_t h,
00242     uint16_t color) {
00243     // Overwrite in subclasses if desired!
00244     fillRect(x, y, w, h, color);
00245 }
00246
00247 /*****
00252 *****/
00253 void Adafruit_GFX::endWrite() {}
00254
00255 /*****
00264 *****/
00265 void Adafruit_GFX::drawFastVLine(int16_t x, int16_t y, int16_t h,
00266     uint16_t color) {
00267     startWrite();
00268     writeLine(x, y, x, y + h - 1, color);
00269     endWrite();
00270 }
00271
00272 /*****
00281 *****/
00282 void Adafruit_GFX::drawFastHLine(int16_t x, int16_t y, int16_t w,
00283     uint16_t color) {
00284     startWrite();
00285     writeLine(x, y, x + w - 1, y, color);
00286     endWrite();
00287 }
00288
00289 /*****
00299 *****/
00300 void Adafruit_GFX::fillRect(int16_t x, int16_t y, int16_t w, int16_t h,
00301     uint16_t color) {
00302     startWrite();
00303     for (int16_t i = x; i < x + w; i++) {
00304         writeFastVLine(i, y, h, color);
00305     }
00306     endWrite();
00307 }
00308
00309 /*****
00315 *****/
00316 void Adafruit_GFX::fillScreen(uint16_t color) {
00317     fillRect(0, 0, _width, _height, color);
00318 }
00319
00320 /*****
00329 *****/
00330 void Adafruit_GFX::drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
00331     uint16_t color) {
00332     // Update in subclasses if desired!
00333     if (x0 == x1) {
00334         if (y0 > y1)
00335             _swap_int16_t(y0, y1);
00336         drawFastVLine(x0, y0, y1 - y0 + 1, color);
00337     } else if (y0 == y1) {
00338         if (x0 > x1)
00339             _swap_int16_t(x0, x1);
00340         drawFastHLine(x0, y0, x1 - x0 + 1, color);
00341     } else {
00342         startWrite();
00343         writeLine(x0, y0, x1, y1, color);
00344         endWrite();
00345     }
00346 }
00347

```

```
00348 /*****  
00356 *****/  
00357 void Adafruit_GFX::drawCircle(int16_t x0, int16_t y0, int16_t r,  
00358                                     uint16_t color) {  
00359 #if defined(ESP8266)  
00360     yield();  
00361 #endif  
00362     int16_t f = 1 - r;  
00363     int16_t ddF_x = 1;  
00364     int16_t ddF_y = -2 * r;  
00365     int16_t x = 0;  
00366     int16_t y = r;  
00367  
00368     startWrite();  
00369     writePixel(x0, y0 + r, color);  
00370     writePixel(x0, y0 - r, color);  
00371     writePixel(x0 + r, y0, color);  
00372     writePixel(x0 - r, y0, color);  
00373  
00374     while (x < y) {  
00375         if (f >= 0) {  
00376             y--;  
00377             ddF_y += 2;  
00378             f += ddF_y;  
00379         }  
00380         x++;  
00381         ddF_x += 2;  
00382         f += ddF_x;  
00383  
00384         writePixel(x0 + x, y0 + y, color);  
00385         writePixel(x0 - x, y0 + y, color);  
00386         writePixel(x0 + x, y0 - y, color);  
00387         writePixel(x0 - x, y0 - y, color);  
00388         writePixel(x0 + y, y0 + x, color);  
00389         writePixel(x0 - y, y0 + x, color);  
00390         writePixel(x0 + y, y0 - x, color);  
00391         writePixel(x0 - y, y0 - x, color);  
00392     }  
00393     endWrite();  
00394 }  
00395  
00396 /*****  
00406 *****/  
00407 void Adafruit_GFX::drawCircleHelper(int16_t x0, int16_t y0, int16_t r,  
00408                                         uint8_t cornername, uint16_t color) {  
00409     int16_t f = 1 - r;  
00410     int16_t ddF_x = 1;  
00411     int16_t ddF_y = -2 * r;  
00412     int16_t x = 0;  
00413     int16_t y = r;  
00414  
00415     while (x < y) {  
00416         if (f >= 0) {  
00417             y--;  
00418             ddF_y += 2;  
00419             f += ddF_y;  
00420         }  
00421         x++;  
00422         ddF_x += 2;  
00423         f += ddF_x;  
00424         if (cornername & 0x4) {  
00425             writePixel(x0 + x, y0 + y, color);  
00426             writePixel(x0 + y, y0 + x, color);  
00427         }  
00428         if (cornername & 0x2) {  
00429             writePixel(x0 + x, y0 - y, color);  
00430             writePixel(x0 + y, y0 - x, color);  
00431         }  
00432         if (cornername & 0x8) {  
00433             writePixel(x0 - y, y0 + x, color);  
00434             writePixel(x0 - x, y0 + y, color);  
00435         }  
00436         if (cornername & 0x1) {  
00437             writePixel(x0 - y, y0 - x, color);  
00438             writePixel(x0 - x, y0 - y, color);  
00439         }  
00440     }  
00441 }  
00442  
00443 /*****  
00451 *****/  
00452 void Adafruit_GFX::fillCircle(int16_t x0, int16_t y0, int16_t r,  
00453                                         uint16_t color) {  
00454     startWrite();  
00455     writeFastVLine(x0, y0 - r, 2 * r + 1, color);  
00456     fillCircleHelper(x0, y0, r, 3, 0, color);  
00457     endWrite();
```

```

00458 }
00459
00460 /*****
00470 *****/
00471 void Adafruit_GFX::fillCircleHelper(int16_t x0, int16_t y0, int16_t r,
00472                                     uint8_t corners, int16_t delta,
00473                                     uint16_t color) {
00474
00475     int16_t f = 1 - r;
00476     int16_t ddF_x = 1;
00477     int16_t ddF_y = -2 * r;
00478     int16_t x = 0;
00479     int16_t y = r;
00480     int16_t px = x;
00481     int16_t py = y;
00482
00483     delta++; // Avoid some +1's in the loop
00484
00485     while (x < y) {
00486         if (f >= 0) {
00487             y--;
00488             ddF_y += 2;
00489             f += ddF_y;
00490         }
00491         x++;
00492         ddF_x += 2;
00493         f += ddF_x;
00494         // These checks avoid double-drawing certain lines, important
00495         // for the SSD1306 library which has an INVERT drawing mode.
00496         if (x < (y + 1)) {
00497             if (corners & 1)
00498                 writeFastVLine(x0 + x, y0 - y, 2 * y + delta, color);
00499             if (corners & 2)
00500                 writeFastVLine(x0 - x, y0 - y, 2 * y + delta, color);
00501         }
00502         if (y != py) {
00503             if (corners & 1)
00504                 writeFastVLine(x0 + py, y0 - px, 2 * px + delta, color);
00505             if (corners & 2)
00506                 writeFastVLine(x0 - py, y0 - px, 2 * px + delta, color);
00507             py = y;
00508         }
00509         px = x;
00510     }
00511 }
00512
00513 /*****
00522 *****/
00523 void Adafruit_GFX::drawRect(int16_t x, int16_t y, int16_t w, int16_t h,
00524                               uint16_t color) {
00525     startWrite();
00526     writeFastHLine(x, y, w, color);
00527     writeFastHLine(x, y + h - 1, w, color);
00528     writeFastVLine(x, y, h, color);
00529     writeFastVLine(x + w - 1, y, h, color);
00530     endWrite();
00531 }
00532
00533 /*****
00543 *****/
00544 void Adafruit_GFX::drawRoundRect(int16_t x, int16_t y, int16_t w, int16_t h,
00545                                   int16_t r, uint16_t color) {
00546     int16_t max_radius = ((w < h) ? w : h) / 2; // 1/2 minor axis
00547     if (r > max_radius)
00548         r = max_radius;
00549     // smarter version
00550     startWrite();
00551     writeFastHLine(x + r, y, w - 2 * r, color); // Top
00552     writeFastHLine(x + r, y + h - 1, w - 2 * r, color); // Bottom
00553     writeFastVLine(x, y + r, h - 2 * r, color); // Left
00554     writeFastVLine(x + w - 1, y + r, h - 2 * r, color); // Right
00555     // draw four corners
00556     drawCircleHelper(x + r, y + r, r, 1, color);
00557     drawCircleHelper(x + w - r - 1, y + r, r, 2, color);
00558     drawCircleHelper(x + w - r - 1, y + h - r - 1, r, 4, color);
00559     drawCircleHelper(x + r, y + h - r - 1, r, 8, color);
00560     endWrite();
00561 }
00562
00563 /*****
00573 *****/
00574 void Adafruit_GFX::fillRoundRect(int16_t x, int16_t y, int16_t w, int16_t h,
00575                                   int16_t r, uint16_t color) {
00576     int16_t max_radius = ((w < h) ? w : h) / 2; // 1/2 minor axis
00577     if (r > max_radius)
00578         r = max_radius;
00579     // smarter version

```

```

00580     startWrite();
00581     writeFillRect(x + r, y, w - 2 * r, h, color);
00582     // draw four corners
00583     fillCircleHelper(x + w - r - 1, y + r, r, 1, h - 2 * r - 1, color);
00584     fillCircleHelper(x + r, y + r, r, 2, h - 2 * r - 1, color);
00585     endWrite();
00586 }
00587
00588 /***** *****
00589 /***** *****
00590 void Adafruit_GFX::drawTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
00591                                 int16_t x2, int16_t y2, uint16_t color) {
00592     drawLine(x0, y0, x1, y1, color);
00593     drawLine(x1, y1, x2, y2, color);
00594     drawLine(x2, y2, x0, y0, color);
00595 }
00596
00597 /***** *****
00598 /***** *****
00599 void Adafruit_GFX::fillTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
00600                                 int16_t x2, int16_t y2, uint16_t color) {
00601
00602     int16_t a, b, y, last;
00603
00604     // Sort coordinates by Y order (y2 >= y1 >= y0)
00605     if (y0 > y1) {
00606         _swap_int16_t(y0, y1);
00607         _swap_int16_t(x0, x1);
00608     }
00609     if (y1 > y2) {
00610         _swap_int16_t(y2, y1);
00611         _swap_int16_t(x2, x1);
00612     }
00613     if (y0 > y1) {
00614         _swap_int16_t(y0, y1);
00615         _swap_int16_t(x0, x1);
00616     }
00617
00618     startWrite();
00619     if (y0 == y2) { // Handle awkward all-on-same-line case as its own thing
00620         a = b = x0;
00621         if (x1 < a)
00622             a = x1;
00623         else if (x1 > b)
00624             b = x1;
00625         if (x2 < a)
00626             a = x2;
00627         else if (x2 > b)
00628             b = x2;
00629         writeFastHLine(a, y0, b - a + 1, color);
00630         endWrite();
00631         return;
00632     }
00633
00634     int16_t dx01 = x1 - x0, dy01 = y1 - y0, dx02 = x2 - x0, dy02 = y2 - y0,
00635             dx12 = x2 - x1, dy12 = y2 - y1;
00636     int32_t sa = 0, sb = 0;
00637
00638     // For upper part of triangle, find scanline crossings for segments
00639     // 0-1 and 0-2. If y1=y2 (flat-bottomed triangle), the scanline y1
00640     // is included here (and second loop will be skipped, avoiding a /0
00641     // error there), otherwise scanline y1 is skipped here and handled
00642     // in the second loop...which also avoids a /0 error here if y0=y1
00643     // (flat-topped triangle).
00644     if (y1 == y2)
00645         last = y1; // Include y1 scanline
00646     else
00647         last = y1 - 1; // Skip it
00648
00649     for (y = y0; y <= last; y++) {
00650         a = x0 + sa / dy01;
00651         b = x0 + sb / dy02;
00652         sa += dx01;
00653         sb += dx02;
00654         /* longhand:
00655             a = x0 + (x1 - x0) * (y - y0) / (y1 - y0);
00656             b = x0 + (x2 - x0) * (y - y0) / (y2 - y0);
00657         */
00658         if (a > b)
00659             _swap_int16_t(a, b);
00660         writeFastHLine(a, y, b - a + 1, color);
00661     }
00662
00663     // For lower part of triangle, find scanline crossings for segments
00664     // 0-2 and 1-2. This loop is skipped if y1=y2.
00665     sa = (int32_t)dx12 * (y - y1);
00666     sb = (int32_t)dx02 * (y - y0);

```

```

00687     for (; y <= y2; y++) {
00688         a = x1 + sa / dy12;
00689         b = x0 + sb / dy02;
00690         sa += dx12;
00691         sb += dx02;
00692         /* longhand:
00693             a = x1 + (x2 - x1) * (y - y1) / (y2 - y1);
00694             b = x0 + (x2 - x0) * (y - y0) / (y2 - y0);
00695         */
00696         if (a > b)
00697             _swap_int16_t(a, b);
00698         writeFastHLine(a, y, b - a + 1, color);
00699     }
00700     endWrite();
00701 }
00702
00703 // BITMAP / XBITMAP / GRayscale / RGB BITMAP FUNCTIONS -----
00704
00705 /***** *****
00716 /***** *****
00717 void Adafruit_GFX::drawBitmap(int16_t x, int16_t y, const uint8_t bitmap[],
00718                                 int16_t w, int16_t h, uint16_t color) {
00719
00720     int16_t byteWidth = (w + 7) / 8; // Bitmap scanline pad = whole byte
00721     uint8_t byte = 0;
00722
00723     startWrite();
00724     for (int16_t j = 0; j < h; j++, y++) {
00725         for (int16_t i = 0; i < w; i++) {
00726             if (i & 7)
00727                 byte <= 1;
00728             else
00729                 byte = pgm_read_byte(&bitmap[j * byteWidth + i / 8]);
00730             if (byte & 0x80)
00731                 writePixel(x + i, y, color);
00732         }
00733     }
00734     endWrite();
00735 }
00736
00737 /***** *****
00750 /***** *****
00751 void Adafruit_GFX::drawBitmap(int16_t x, int16_t y, const uint8_t bitmap[],
00752                                 int16_t w, int16_t h, uint16_t color,
00753                                 uint16_t bg) {
00754
00755     int16_t byteWidth = (w + 7) / 8; // Bitmap scanline pad = whole byte
00756     uint8_t byte = 0;
00757
00758     startWrite();
00759     for (int16_t j = 0; j < h; j++, y++) {
00760         for (int16_t i = 0; i < w; i++) {
00761             if (i & 7)
00762                 byte <= 1;
00763             else
00764                 byte = pgm_read_byte(&bitmap[j * byteWidth + i / 8]);
00765             writePixel(x + i, y, (byte & 0x80) ? color : bg);
00766         }
00767     }
00768     endWrite();
00769 }
00770
00771 /***** *****
00782 /***** *****
00783 void Adafruit_GFX::drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w,
00784                                 int16_t h, uint16_t color) {
00785
00786     int16_t byteWidth = (w + 7) / 8; // Bitmap scanline pad = whole byte
00787     uint8_t byte = 0;
00788
00789     startWrite();
00790     for (int16_t j = 0; j < h; j++, y++) {
00791         for (int16_t i = 0; i < w; i++) {
00792             if (i & 7)
00793                 byte <= 1;
00794             else
00795                 byte = bitmap[j * byteWidth + i / 8];
00796             if (byte & 0x80)
00797                 writePixel(x + i, y, color);
00798         }
00799     }
00800     endWrite();
00801 }
00802
00803 /***** *****
00816 /***** *****
00817 void Adafruit_GFX::drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w,

```

```
00818     int16_t h, uint16_t color, uint16_t bg) {
00819
00820     int16_t byteWidth = (w + 7) / 8; // Bitmap scanline pad = whole byte
00821     uint8_t byte = 0;
00822
00823     startWrite();
00824     for (int16_t j = 0; j < h; j++, y++) {
00825         for (int16_t i = 0; i < w; i++) {
00826             if (i & 7)
00827                 byte <<= 1;
00828             else
00829                 byte = bitmap[j * byteWidth + i / 8];
00830             writePixel(x + i, y, (byte & 0x80) ? color : bg);
00831         }
00832     }
00833     endWrite();
00834 }
00835
00836 /*****
00837 ****/
00838 /*****
00839 ****/
00840 void Adafruit_GFX::drawXBitmap(int16_t x, int16_t y, const uint8_t bitmap[],
00841                                 int16_t w, int16_t h, uint16_t color) {
00842
00843     int16_t byteWidth = (w + 7) / 8; // Bitmap scanline pad = whole byte
00844     uint8_t byte = 0;
00845
00846     startWrite();
00847     for (int16_t j = 0; j < h; j++, y++) {
00848         for (int16_t i = 0; i < w; i++) {
00849             if (i & 7)
00850                 byte >>= 1;
00851             else
00852                 byte = pgm_read_byte(&bitmap[j * byteWidth + i / 8]);
00853             // Nearly identical to drawBitmap(), only the bit order
00854             // is reversed here (left-to-right = LSB to MSB):
00855             if (byte & 0x01)
00856                 writePixel(x + i, y, color);
00857         }
00858     }
00859     endWrite();
00860 }
00861
00862 /*****
00863 ****/
00864 /*****
00865 ****/
00866 void Adafruit_GFX::drawGrayscaleBitmap(int16_t x, int16_t y,
00867                                         const uint8_t bitmap[], int16_t w,
00868                                         int16_t h) {
00869     startWrite();
00870     for (int16_t j = 0; j < h; j++, y++) {
00871         for (int16_t i = 0; i < w; i++) {
00872             writePixel(x + i, y, (uint8_t)pgm_read_byte(&bitmap[j * w + i]));
00873         }
00874     }
00875     endWrite();
00876 }
00877
00878 /*****
00879 ****/
00880 /*****
00881 ****/
00882 void Adafruit_GFX::drawGrayscaleBitmap(int16_t x, int16_t y, uint8_t *bitmap,
00883                                         int16_t w, int16_t h) {
00884     startWrite();
00885     for (int16_t j = 0; j < h; j++, y++) {
00886         for (int16_t i = 0; i < w; i++) {
00887             writePixel(x + i, y, bitmap[j * w + i]);
00888         }
00889     }
00890     endWrite();
00891 }
00892
00893 /*****
00894 ****/
00895 /*****
00896 ****/
00897 void Adafruit_GFX::drawGrayscaleBitmap(int16_t x, int16_t y,
00898                                         const uint8_t bitmap[], int16_t w,
00899                                         const uint8_t mask[], int16_t h) {
00900     startWrite();
00901     for (int16_t j = 0; j < h; j++, y++) {
00902         for (int16_t i = 0; i < w; i++) {
00903             writePixel(x + i, y, bitmap[j * w + i] & mask[j * w + i]);
00904         }
00905     }
00906     endWrite();
00907 }
00908
00909 /*****
00910 ****/
00911 /*****
00912 ****/
00913 void Adafruit_GFX::drawGrayscaleBitmap(int16_t x, int16_t y,
00914                                         const uint8_t bitmap[], int16_t w,
00915                                         const uint8_t mask[], int16_t h) {
00916     startWrite();
00917     for (int16_t j = 0; j < h; j++, y++) {
00918         for (int16_t i = 0; i < w; i++) {
00919             if (i & 7)
00920                 byte <<= 1;
00921             else
00922                 byte = pgm_read_byte(&mask[j * bw + i / 8]);
00923             if (byte & 0x80)
00924                 writePixel(x + i, y, (uint8_t)pgm_read_byte(&bitmap[j * w + i]));
00925         }
00926     }
00927     endWrite();
00928 }
```

```

00951     }
00952   }
00953   endWrite();
00954 }
00955
00956 /*****
00970 *****/
00971 void Adafruit_GFX::drawGrayscaleBitmap(int16_t x, int16_t y, uint8_t *bitmap,
00972                                         uint8_t *mask, int16_t w, int16_t h) {
00973   int16_t bw = (w + 7) / 8; // Bitmask scanline pad = whole byte
00974   uint8_t byte = 0;
00975   startWrite();
00976   for (int16_t j = 0; j < h; j++, y++) {
00977     for (int16_t i = 0; i < w; i++) {
00978       if (i & 7)
00979         byte <= 1;
00980       else
00981         byte = mask[j * bw + i / 8];
00982       if (byte & 0x80) {
00983         writePixel(x + i, y, bitmap[j * w + i]);
00984       }
00985     }
00986   }
00987   endWrite();
00988 }
00989
00990 *****/
01000 *****/
01001 void Adafruit_GFX::drawRGBBitmap(int16_t x, int16_t y, const uint16_t bitmap[],
01002                                     int16_t w, int16_t h) {
01003   startWrite();
01004   for (int16_t j = 0; j < h; j++, y++) {
01005     for (int16_t i = 0; i < w; i++) {
01006       writePixel(x + i, y, pgm_read_word(&bitmap[j * w + i]));
01007     }
01008   }
01009   endWrite();
01010 }
01011
01012 *****/
01022 *****/
01023 void Adafruit_GFX::drawRGBBitmap(int16_t x, int16_t y, uint16_t *bitmap,
01024                                     int16_t w, int16_t h) {
01025   startWrite();
01026   for (int16_t j = 0; j < h; j++, y++) {
01027     for (int16_t i = 0; i < w; i++) {
01028       writePixel(x + i, y, bitmap[j * w + i]);
01029     }
01030   }
01031   endWrite();
01032 }
01033
01034 *****/
01047 *****/
01048 void Adafruit_GFX::drawRGBBitmap(int16_t x, int16_t y, const uint16_t bitmap[],
01049                                     const uint8_t mask[], int16_t w, int16_t h) {
01050   int16_t bw = (w + 7) / 8; // Bitmask scanline pad = whole byte
01051   uint8_t byte = 0;
01052   startWrite();
01053   for (int16_t j = 0; j < h; j++, y++) {
01054     for (int16_t i = 0; i < w; i++) {
01055       if (i & 7)
01056         byte <= 1;
01057       else
01058         byte = pgm_read_byte(&mask[j * bw + i / 8]);
01059       if (byte & 0x80) {
01060         writePixel(x + i, y, pgm_read_word(&bitmap[j * w + i]));
01061       }
01062     }
01063   }
01064   endWrite();
01065 }
01066
01067 *****/
01080 *****/
01081 void Adafruit_GFX::drawRGBBitmap(int16_t x, int16_t y, uint16_t *bitmap,
01082                                     uint8_t *mask, int16_t w, int16_t h) {
01083   int16_t bw = (w + 7) / 8; // Bitmask scanline pad = whole byte
01084   uint8_t byte = 0;
01085   startWrite();
01086   for (int16_t j = 0; j < h; j++, y++) {
01087     for (int16_t i = 0; i < w; i++) {
01088       if (i & 7)
01089         byte <= 1;
01090       else
01091         byte = mask[j * bw + i / 8];
01092       if (byte & 0x80) {

```

```

01093         writePixel(x + i, y, bitmap[j * w + i]);
01094     }
01095   }
01096 }
01097 endWrite();
01098 }
01099
01100 // TEXT- AND CHARACTER-HANDLING FUNCTIONS -----
01101
01102 // Draw a character
01103 //*****
01104 //*****
01105 void Adafruit_GFX::drawChar(int16_t x, int16_t y, unsigned char c,
01106                               uint16_t color, uint16_t bg, uint8_t size) {
01107   drawChar(x, y, c, color, bg, size, size);
01108 }
01109
01110 // Draw a character
01111 //*****
01112 //*****
01113 void Adafruit_GFX::drawChar(int16_t x, int16_t y, unsigned char c,
01114                               uint16_t color, uint16_t bg, uint8_t size_x,
01115                               uint8_t size_y) {
01116
01117   if (!gfxFont) { // 'Classic' built-in font
01118
01119     if ((x >= _width) || // Clip right
01120         (y >= _height) || // Clip bottom
01121         ((x + 6 * size_x - 1) < 0) || // Clip left
01122         ((y + 8 * size_y - 1) < 0)) // Clip top
01123       return;
01124
01125     if (!_cp437 && (c >= 176))
01126       c++; // Handle 'classic' charset behavior
01127
01128     startWrite();
01129     for (int8_t i = 0; i < 5; i++) { // Char bitmap = 5 columns
01130       uint8_t line = pgm_read_byte(&font[c * 5 + i]);
01131       for (int8_t j = 0; j < 8; j++, line >>= 1) {
01132         if (line & 1) {
01133           if (size_x == 1 && size_y == 1)
01134             writePixel(x + i, y + j, color);
01135           else
01136             writeFillRect(x + i * size_x, y + j * size_y, size_x, size_y,
01137                           color);
01138         } else if (bg != color) {
01139           if (size_x == 1 && size_y == 1)
01140             writePixel(x + i, y + j, bg);
01141           else
01142             writeFillRect(x + i * size_x, y + j * size_y, size_x, size_y, bg);
01143         }
01144       }
01145       if (bg != color) { // If opaque, draw vertical line for last column
01146         if (size_x == 1 && size_y == 1)
01147           writeFastVLine(x + 5, y, 8, bg);
01148         else
01149           writeFillRect(x + 5 * size_x, y, size_x, 8 * size_y, bg);
01150     }
01151   }
01152   endWrite();
01153
01154 } else { // Custom font
01155
01156   // Character is assumed previously filtered by write() to eliminate
01157   // newlines, returns, non-printable characters, etc. Calling
01158   // drawChar() directly with 'bad' characters of font may cause mayhem!
01159
01160   c -= (uint8_t)pgm_read_byte(&gfxFont->first);
01161   GFXglyph *glyph = pgm_read_glyph_ptr(gfxFont, c);
01162   uint8_t *bitmap = pgm_read_bitmap_ptr(gfxFont);
01163
01164   uint16_t bo = pgm_read_word(&glyph->bitmapOffset);
01165   uint8_t w = pgm_read_byte(&glyph->width), h = pgm_read_byte(&glyph->height);
01166   int8_t xo = pgm_read_byte(&glyph->xOffset),
01167         yo = pgm_read_byte(&glyph->yOffset);
01168   uint8_t xx, yy, bits = 0, bit = 0;
01169   int16_t xo16 = 0, yo16 = 0;
01170
01171   if (size_x > 1 || size_y > 1) {
01172     xo16 = xo;
01173     yo16 = yo;
01174   }
01175
01176   // Todo: Add character clipping here
01177
01178   // NOTE: THERE IS NO 'BACKGROUND' COLOR OPTION ON CUSTOM FONTS.
01179   // THIS IS ON PURPOSE AND BY DESIGN. The background color feature
01180

```

```

01201 // has typically been used with the 'classic' font to overwrite old
01202 // screen contents with new data. This ONLY works because the
01203 // characters are a uniform size; it's not a sensible thing to do with
01204 // proportionally-spaced fonts with glyphs of varying sizes (and that
01205 // may overlap). To replace previously-drawn text when using a custom
01206 // font, use the getTextBounds() function to determine the smallest
01207 // rectangle encompassing a string, erase the area with fillRect(),
01208 // then draw new text. This WILL unfortunately 'blink' the text, but
01209 // is unavoidable. Drawing 'background' pixels will NOT fix this,
01210 // only creates a new set of problems. Have an idea to work around
01211 // this (a canvas object type for MCUs that can afford the RAM and
01212 // displays supporting setAddrWindow() and pushColors()), but haven't
01213 // implemented this yet.
01214
01215     startWrite();
01216     for (yy = 0; yy < h; yy++) {
01217         for (xx = 0; xx < w; xx++) {
01218             if (!(bit++ & 7)) {
01219                 bits = pgm_read_byte(&bitmap[bo++]);
01220             }
01221             if (bits & 0x80) {
01222                 if (size_x == 1 && size_y == 1) {
01223                     writePixel(x + xo + xx, y + yo + yy, color);
01224                 } else {
01225                     writeFillRect(x + (xo16 + xx) * size_x, y + (yo16 + yy) * size_y,
01226                                   size_x, size_y, color);
01227                 }
01228             }
01229             bits <= 1;
01230         }
01231     }
01232     endWrite();
01233
01234 } // End classic vs custom font
01235 }
01236 /*********************************************************************
01237 ****
01238 size_t Adafruit_GFX::write(uint8_t c) {
01239     if (!gfxFont) { // 'Classic' built-in font
01240
01241         if (c == '\n') { // Newline?
01242             cursor_x = 0; // Reset x to zero,
01243             cursor_y += textsize_y * 8; // advance y one line
01244         } else if (c != '\r') { // Ignore carriage returns
01245             if (wrap && ((cursor_x + textsize_x * 6) > _width)) { // Off right?
01246                 cursor_x = 0; // Reset x to zero,
01247                 cursor_y += textsize_y * 8; // advance y one line
01248             }
01249             drawChar(cursor_x, cursor_y, c, textcolor, textbgcolor, textsize_x,
01250                     textsize_y);
01251             cursor_x += textsize_x * 6; // Advance x one char
01252         }
01253
01254     } else { // Custom font
01255
01256         if (c == '\n') {
01257             cursor_x = 0;
01258             cursor_y +=
01259                 (int16_t)textsize_y * (uint8_t)pgm_read_byte(&gfxFont->yAdvance);
01260         } else if (c != '\r') {
01261             uint8_t first = pgm_read_byte(&gfxFont->first);
01262             if ((c >= first) && (c <= (uint8_t)pgm_read_byte(&gfxFont->last))) {
01263                 GFXglyph *glyph = pgm_read_glyph_ptr(gfxFont, c - first);
01264                 uint8_t w = pgm_read_byte(&glyph->width),
01265                         h = pgm_read_byte(&glyph->height);
01266                 if ((w > 0) && (h > 0)) { // Is there an associated bitmap?
01267                     int16_t xo = (int8_t)pgm_read_byte(&glyph->xOffset); // sic
01268                     if (wrap && ((cursor_x + textsize_x * (xo + w)) > _width)) {
01269                         cursor_x = 0;
01270                         cursor_y += (int16_t)textsize_y *
01271                             (uint8_t)pgm_read_byte(&gfxFont->yAdvance);
01272                     }
01273                     drawChar(cursor_x, cursor_y, c, textcolor, textbgcolor, textsize_x,
01274                             textsize_y);
01275                 }
01276                 cursor_x +=
01277                     (uint8_t)pgm_read_byte(&glyph->xAdvance) * (int16_t)textsize_x;
01278             }
01279         }
01280     }
01281     return 1;
01282 }
01283 }
01284 }
01285
01286 }
01287
01288 /*********************************************************************
01289 ****
01290 void Adafruit_GFX::setTextSize(uint8_t s) { setTextSize(s, s); }
01291

```

```

01297 /***** ****
01304 /***** ****
01305 void Adafruit_GFX::setTextSize(uint8_t s_x, uint8_t s_y) {
01306     textsize_x = (s_x > 0) ? s_x : 1;
01307     textsize_y = (s_y > 0) ? s_y : 1;
01308 }
01309
01310 /***** ****
01315 /***** ****
01316 void Adafruit_GFX::setRotation(uint8_t x) {
01317     rotation = (x & 3);
01318     switch (rotation) {
01319         case 0:
01320             _width = WIDTH;
01321             _height = HEIGHT;
01322             break;
01324         case 1:
01325         case 3:
01326             _width = HEIGHT;
01327             _height = WIDTH;
01328             break;
01329     }
01330 }
01331
01332 /***** ****
01337 /***** ****
01338 void Adafruit_GFX::setFont(const GFXfont *f) {
01339     if (f) {           // Font struct pointer passed in?
01340         if (!gfxFont) { // And no current font struct?
01341             // Switching from classic to new font behavior.
01342             // Move cursor pos down 6 pixels so it's on baseline.
01343             cursor_y += 6;
01344         }
01345     } else if (gfxFont) { // NULL passed. Current font struct defined?
01346         // Switching from new to classic font behavior.
01347         // Move cursor pos up 6 pixels so it's at top-left of char.
01348         cursor_y -= 6;
01349     }
01350     gfxFont = (GFXfont *)f;
01351 }
01352
01353 /***** ****
01370 /***** ****
01371 void Adafruit_GFX::charBounds(unsigned char c, int16_t *x, int16_t *y,
01372                                 int16_t *minx, int16_t *miny, int16_t *maxx,
01373                                 int16_t *maxy) {
01374
01375     if (gfxFont) {
01376
01377         if (c == '\n') { // Newline?
01378             *x = 0;          // Reset x to zero, advance y by one line
01379             *y += textsize_y * (uint8_t)pgm_read_byte(&gfxFont->yAdvance);
01380         } else if (c != '\r') { // Not a carriage return; is normal char
01381             uint8_t first = pgm_read_byte(&gfxFont->first),
01382                     last = pgm_read_byte(&gfxFont->last);
01383             if ((c >= first) && (c <= last)) { // Char present in this font?
01384                 GFXglyph *glyph = pgm_read_glyph_ptr(gfxFont, c - first);
01385                 uint8_t gw = pgm_read_byte(&glyph->width),
01386                         gh = pgm_read_byte(&glyph->height),
01387                         xa = pgm_read_byte(&glyph->xAdvance);
01388                 int8_t xo = pgm_read_byte(&glyph->xOffset),
01389                         yo = pgm_read_byte(&glyph->yOffset);
01390                 if (wrap && (((*x + ((int16_t)xo + gw) * textsize_x)) > _width)) {
01391                     *x = 0;          // Reset x to zero, advance y by one line
01392                     *y += textsize_y * (uint8_t)pgm_read_byte(&gfxFont->yAdvance);
01393                 }
01394                 int16_t tsx = (int16_t)textsize_x, tsy = (int16_t)textsize_y,
01395                         x1 = *x + xo * tsx, y1 = *y + yo * tsy, x2 = x1 + gw * tsx - 1,
01396                         y2 = y1 + gh * tsy - 1;
01397                 if (x1 < *minx)
01398                     *minx = x1;
01399                 if (y1 < *miny)
01400                     *miny = y1;
01401                 if (x2 > *maxx)
01402                     *maxx = x2;
01403                 if (y2 > *maxy)
01404                     *maxy = y2;
01405                 *x += xa * tsx;
01406             }
01407         }
01408     } else { // Default font
01409         if (c == '\n') {           // Newline?
01410             *x = 0;          // Reset x to zero,
01411             *y += textsize_y * 8; // advance y one line

```

```

01414     // min/max x/y unchanged -- that waits for next 'normal' character
01415 } else if (c != '\r') { // Normal char; ignore carriage returns
01416     if (wrap && ((*x + textsize_x * 6) > _width)) { // Off right?
01417         *x = 0;                                         // Reset x to zero,
01418         *y += textsize_y * 8;                           // advance y one line
01419     }
01420     int x2 = *x + textsize_x * 6 - 1, // Lower-right pixel of char
01421         y2 = *y + textsize_y * 8 - 1;
01422     if (x2 > *maxx)
01423         *maxx = x2; // Track max x, y
01424     if (y2 > *maxy)
01425         *maxy = y2;
01426     if (*x < *minx)
01427         *minx = *x; // Track min x, y
01428     if (*y < *miny)
01429         *miny = *y;
01430     *x += textsize_x * 6; // Advance x one char
01431 }
01432 }
01433 }
01434
01435 /*****
01436 *****/
01437 /*****
01438 void Adafruit_GFX::getTextBounds(const char *str, int16_t x, int16_t y,
01439                                     int16_t *x1, int16_t *y1, uint16_t *w,
01440                                     uint16_t *h) {
01441
01442     uint8_t c; // Current character
01443     int16_t minx = 0xFFFF, miny = 0x7FFF, maxx = -1, maxy = -1; // Bound rect
01444     // Bound rect is intentionally initialized inverted, so 1st char sets it
01445
01446     *x1 = x; // Initial position is value passed in
01447     *y1 = y;
01448     *w = *h = 0; // Initial size is zero
01449
01450     while ((c = *str++)) {
01451         // charBounds() modifies x/y to advance for each character,
01452         // and min/max x/y are updated to incrementally build bounding rect.
01453         charBounds(c, &x, &y, &minx, &miny, &maxx, &maxy);
01454     }
01455
01456     if (maxx >= minx) { // If legit string bounds were found...
01457         *x1 = minx; // Update x1 to least X coord,
01458         *w = maxx - minx + 1; // And w to bound rect width
01459     }
01460     if (maxy >= miny) { // Same for height
01461         *y1 = miny;
01462         *h = maxy - miny + 1;
01463     }
01464 }
01465
01466 /*****
01467 *****/
01468 /*****
01469 void Adafruit_GFX::getTextBounds(const String &str, int16_t x, int16_t y,
01470                                     int16_t *x1, int16_t *y1, uint16_t *w,
01471                                     uint16_t *h) {
01472     if (str.length() != 0) {
01473         getTextBounds(const_cast<char*>(str.c_str()), x, y, x1, y1, w, h);
01474     }
01475 }
01476
01477 /*****
01478 *****/
01479 /*****
01480 void Adafruit_GFX::getTextBounds(const __FlashStringHelper *str, int16_t x,
01481                                     int16_t y, int16_t *x1, int16_t *y1,
01482                                     uint16_t *w, uint16_t *h) {
01483     uint8_t *s = (uint8_t *)str, c;
01484
01485     *x1 = x;
01486     *y1 = y;
01487     *w = *h = 0;
01488
01489     int16_t minx = _width, miny = _height, maxx = -1, maxy = -1;
01490
01491     while ((c = pgm_read_byte(s++)))
01492         charBounds(c, &x, &y, &minx, &miny, &maxx, &maxy);
01493
01494     if (maxx >= minx) {
01495         *x1 = minx;
01496         *w = maxx - minx + 1;
01497     }
01498     if (maxy >= miny) {
01499         *y1 = miny;
01500         *h = maxy - miny + 1;
01501     }
01502 }
01503 */
01504

```

```
01534 /*****  
01539 *****/  
01540 void Adafruit_GFX::invertDisplay(bool i) {  
01541     // Do nothing, must be subclassed if supported by hardware  
01542     (void)i; // disable -Wunused-parameter warning  
01543 }  
01544  
01545 /*****  
01546 *****/  
01547 /*****  
01551 *****/  
01552 Adafruit_GFX_Button::Adafruit_GFX_Button(void) { _gfx = 0; }  
01553  
01554 /*****  
01558 *****/  
01559 // Classic initButton() function: pass center & size  
01560 void Adafruit_GFX_Button::initButton(Adafruit_GFX *gfx, int16_t x, int16_t y,  
01561         uint16_t w, uint16_t h, uint16_t outline,  
01562         uint16_t fill, uint16_t textcolor,  
01563         char *label, uint8_t textsize) {  
01564     // Tweak arguments and pass to the newer initButtonUL() function...  
01565     initButtonUL(gfx, x - (w / 2), y - (h / 2), w, h, outline, fill, textcolor,  
01566             label, textsize);  
01577 }  
01578  
01579 /*****  
01594 *****/  
01595 // Classic initButton() function: pass center & size  
01596 void Adafruit_GFX_Button::initButton(Adafruit_GFX *gfx, int16_t x, int16_t y,  
01597         uint16_t w, uint16_t h, uint16_t outline,  
01598         uint16_t fill, uint16_t textcolor,  
01599         char *label, uint8_t textsize_x,  
01600         uint8_t textsize_y) {  
01601     // Tweak arguments and pass to the newer initButtonUL() function...  
01602     initButtonUL(gfx, x - (w / 2), y - (h / 2), w, h, outline, fill, textcolor,  
01603             label, textsize_x, textsize_y);  
01604 }  
01605  
01606 /*****  
01621 *****/  
01622 void Adafruit_GFX_Button::initButtonUL(Adafruit_GFX *gfx, int16_t x1,  
01623         int16_t y1, uint16_t w, uint16_t h,  
01624         uint16_t outline, uint16_t fill,  
01625         uint16_t textcolor, char *label,  
01626         uint8_t textsize) {  
01627     initButtonUL(gfx, x1, y1, w, h, outline, fill, textcolor, label, textsize,  
01628             textsize);  
01629 }  
01630  
01631 /*****  
01647 *****/  
01648 void Adafruit_GFX_Button::initButtonUL(Adafruit_GFX *gfx, int16_t x1,  
01649         int16_t y1, uint16_t w, uint16_t h,  
01650         uint16_t outline, uint16_t fill,  
01651         uint16_t textcolor, char *label,  
01652         uint8_t textsize_x, uint8_t textsize_y) {  
01653     _x1 = x1;  
01654     _y1 = y1;  
01655     _w = w;  
01656     _h = h;  
01657     _outlinecolor = outline;  
01658     _fillcolor = fill;  
01659     _textcolor = textcolor;  
01660     _textsize_x = textsize_x;  
01661     _textsize_y = textsize_y;  
01662     _gfx = gfx;  
01663     strncpy(_label, label, 9);  
01664 }  
01665  
01666 /*****  
01672 *****/  
01673 void Adafruit_GFX_Button::drawButton(bool inverted) {  
01674     uint16_t fill, outline, text;  
01675  
01676     if (!inverted) {  
01677         fill = _fillcolor;  
01678         outline = _outlinecolor;  
01679         text = _textcolor;  
01680     } else {  
01681         fill = _textcolor;  
01682         outline = _outlinecolor;  
01683         text = _fillcolor;  
01684     }  
01685  
01686     uint8_t r = min(_w, _h) / 4; // Corner radius  
01687     _gfx->fillRoundRect(_x1, _y1, _w, _h, r, fill);  
01688     _gfx->drawRoundRect(_x1, _y1, _w, _h, r, outline);
```

```

01689     _gfx->setCursor(_x1 + (_w / 2) - (strlen(_label) * 3 * _textsize_x),
01690                         _y1 + (_h / 2) - (4 * _textsize_y));
01691     _gfx->setTextColor(text);
01692     _gfx->setTextSize(_textsize_x, _textsize_y);
01693     _gfx->print(_label);
01694 }
01695 */
01696
01697 //*****
01698 //*****
01699 //*****
01700 bool Adafruit_GFX_Button::contains(int16_t x, int16_t y) {
01701     return ((x >= _x1) && (x < (int16_t)(_x1 + _w)) && (y >= _y1) &&
01702             (y < (int16_t)(_y1 + _h)));
01703 }
01704
01705 //*****
01706 //*****
01707 //*****
01708 bool Adafruit_GFX_Button::justPressed() { return (currstate && !laststate); }
01709
01710 //*****
01711 //*****
01712 //*****
01713 bool Adafruit_GFX_Button::justReleased() { return (!currstate && laststate); }
01714
01715 // -----
01716
01717 // GFXcanvas1, GFXcanvas8 and GFXcanvas16 (currently a WIP, don't get too
01718 // comf'y with the implementation) provide 1-, 8- and 16-bit offscreen
01719 // canvases, the address of which can be passed to drawBitmap() or
01720 // pushColors() (the latter appears only in a couple of GFX-subclassed TFT
01721 // libraries at this time). This is here mostly to help with the recently-
01722 // added proportionally-spaced fonts; adds a way to refresh a section of the
01723 // screen without a massive flickering clear-and-redraw...but maybe you'll
01724 // find other uses too. VERY RAM-intensive, since the buffer is in MCU
01725 // memory and not the display driver...GXFcanvas1 might be minimally useful
01726 // on an Uno-class board, but this and the others are much more likely to
01727 // require at least a Mega or various recent ARM-type boards (recommended,
01728 // as the text+bitmap draw can be pokey). GFXcanvas1 requires 1 bit per
01729 // pixel (rounded up to nearest byte per scanline), GFXcanvas8 is 1 byte
01730 // per pixel (no scanline pad), and GFXcanvas16 uses 2 bytes per pixel (no
01731 // scanline pad).
01732
01733 // NOT EXTENSIVELY TESTED YET. MAY CONTAIN WORST BUGS KNOWN TO HUMANKIND.
01734
01735 #ifdef __AVR__
01736 // Bitmask tables of 0x80»X and ~(0x80»X), because X»Y is slow on AVR
01737 const uint8_t PROGMEM GFXcanvas1::GFXxsetBit[] = {0x80, 0x40, 0x20, 0x10,
01738                                         0x08, 0x04, 0x02, 0x01};
01739 const uint8_t PROGMEM GFXcanvas1::GFXclrBit[] = {0x7F, 0xBF, 0xDF, 0xEF,
01740                                         0xF7, 0xFB, 0xFD, 0xFE};
01741
01742 #endif
01743
01744 //*****
01745 //*****
01746 //*****
01747 // GFXcanvas1:GFXcanvas1(uint16_t w, uint16_t h) : Adafruit_GFX(w, h) {
01748     uint16_t bytes = ((w + 7) / 8) * h;
01749     if ((buffer = (uint8_t *)malloc(bytes))) {
01750         memset(buffer, 0, bytes);
01751     }
01752 }
01753
01754 //*****
01755 //*****
01756 //*****
01757 // GFXcanvas1:~GFXcanvas1(void) {
01758     if (buffer)
01759         free(buffer);
01760 }
01761
01762 //*****
01763 //*****
01764 void GFXcanvas1::drawPixel(int16_t x, int16_t y, uint16_t color) {
01765     if (buffer) {
01766         if ((x < 0) || (y < 0) || (x >= _width) || (y >= _height))
01767             return;
01768         int16_t t;
01769         switch (rotation) {
01770         case 1:
01771             t = x;
01772             x = WIDTH - 1 - y;
01773             y = t;
01774             break;
01775         case 2:
01776             x = WIDTH - 1 - x;
01777             y = HEIGHT - 1 - y;
01778             break;
01779         case 3:
01780             t = x;
01781             x = y;
01782             y = t;
01783         }
01784     }
01785 }
01786
01787 //*****
01788 //*****
01789 //*****
01790 //*****
01791 //*****
01792 //*****
01793 //*****
01794 //*****
01795 //*****
01796 //*****
01797 //*****
01798 //*****
01799 //*****
01800 //*****
01801 //*****
01802 //*****
01803 //*****
01804 //*****

```

```

01805     y = HEIGHT - 1 - t;
01806     break;
01807 }
01808
01809     uint8_t *ptr = &buffer[(x / 8) + y * ((WIDTH + 7) / 8)];
01810 #ifdef __AVR__
01811     if (color)
01812         *ptr |= pgm_read_byte(&GFXsetBit[x & 7]);
01813     else
01814         *ptr &= pgm_read_byte(&GFXclrBit[x & 7]);
01815 #else
01816     if (color)
01817         *ptr |= 0x80 >> (x & 7);
01818     else
01819         *ptr &= ~ (0x80 >> (x & 7));
01820 #endif
01821 }
01822 }
01823
01824 /*****
01825 ****/
01826 /*****
01827 ****/
01828 bool GFXcanvas1::getPixel(int16_t x, int16_t y) const {
01829     int16_t t;
01830     switch (rotation) {
01831     case 1:
01832         t = x;
01833         x = WIDTH - 1 - y;
01834         y = t;
01835         break;
01836     case 2:
01837         x = WIDTH - 1 - x;
01838         y = HEIGHT - 1 - y;
01839         break;
01840     case 3:
01841         t = x;
01842         x = y;
01843         y = HEIGHT - 1 - t;
01844         break;
01845     }
01846     return getRawPixel(x, y);
01847 }
01848
01849 /*****
01850 ****/
01851 /*****
01852 ****/
01853
01854 /*****
01855 ****/
01856 /*****
01857 ****/
01858 bool GFXcanvas1::getRawPixel(int16_t x, int16_t y) const {
01859     if ((x < 0) || (y < 0) || (x >= WIDTH) || (y >= HEIGHT))
01860         return 0;
01861     if (this->getBuffer()) {
01862         uint8_t *buffer = this->getBuffer();
01863         uint8_t *ptr = &buffer[(x / 8) + y * ((WIDTH + 7) / 8)];
01864
01865 #ifdef __AVR__
01866         return ((*ptr) & pgm_read_byte(&GFXsetBit[x & 7])) != 0;
01867 #else
01868         return ((*ptr) & (0x80 >> (x & 7))) != 0;
01869 #endif
01870     }
01871     return 0;
01872 }
01873
01874 /*****
01875 ****/
01876 /*****
01877 ****/
01878 void GFXcanvas1::fillScreen(uint16_t color) {
01879     if (buffer) {
01880         uint16_t bytes = ((WIDTH + 7) / 8) * HEIGHT;
01881         memset(buffer, color ? 0xFF : 0x00, bytes);
01882     }
01883
01884 /*****
01885 ****/
01886 /*****
01887 ****/
01888 void GFXcanvas1::drawFastVLine(int16_t x, int16_t y, int16_t h,
01889                                 uint16_t color) {
01890
01891     if (h < 0) { // Convert negative heights to positive equivalent
01892         h *= -1;
01893         y -= h - 1;
01894         if (y < 0) {
01895             h += y;
01896             y = 0;
01897         }
01898     }
01899
01900     // Edge rejection (no-draw if totally off canvas)
01901     if ((x < 0) || (x >= width()) || (y >= height()) || ((y + h - 1) < 0)) {
01902         return;
01903     }
01904 }
```

```

01919
01920     if (y < 0) { // Clip top
01921         h += y;
01922         y = 0;
01923     }
01924     if (y + h > height()) { // Clip bottom
01925         h = height() - y;
01926     }
01927
01928     if (getRotation() == 0) {
01929         drawFastRawVLine(x, y, h, color);
01930     } else if (getRotation() == 1) {
01931         int16_t t = x;
01932         x = WIDTH - 1 - y;
01933         y = t;
01934         x -= h - 1;
01935         drawFastRawHLine(x, y, h, color);
01936     } else if (getRotation() == 2) {
01937         x = WIDTH - 1 - x;
01938         y = HEIGHT - 1 - y;
01939
01940         y -= h - 1;
01941         drawFastRawVLine(x, y, h, color);
01942     } else if (getRotation() == 3) {
01943         int16_t t = x;
01944         x = y;
01945         y = HEIGHT - 1 - t;
01946         drawFastRawHLine(x, y, h, color);
01947     }
01948 }
01949
01950 /*****
01951 ****/
01952 /*****
01953 void GFXcanvas1::drawFastHLine(int16_t x, int16_t y, int16_t w,
01954                                 uint16_t color) {
01955     if (w < 0) { // Convert negative widths to positive equivalent
01956         w *= -1;
01957         x -= w - 1;
01958         if (x < 0) {
01959             w += x;
01960             x = 0;
01961         }
01962     }
01963
01964     // Edge rejection (no-draw if totally off canvas)
01965     if ((y < 0) || (y >= height()) || (x >= width()) || ((x + w - 1) < 0)) {
01966         return;
01967     }
01968
01969     if (x < 0) { // Clip left
01970         w += x;
01971         x = 0;
01972     }
01973     if (x + w >= width()) { // Clip right
01974         w = width() - x;
01975     }
01976
01977     if (getRotation() == 0) {
01978         drawFastRawHLine(x, y, w, color);
01979     } else if (getRotation() == 1) {
01980         int16_t t = x;
01981         x = WIDTH - 1 - y;
01982         y = t;
01983         drawFastRawVLine(x, y, w, color);
01984     } else if (getRotation() == 2) {
01985         x = WIDTH - 1 - x;
01986         y = HEIGHT - 1 - y;
01987
01988         x -= w - 1;
01989         drawFastRawHLine(x, y, w, color);
01990     } else if (getRotation() == 3) {
01991         int16_t t = x;
01992         x = y;
01993         y = HEIGHT - 1 - t;
01994         y -= w - 1;
01995         drawFastRawVLine(x, y, w, color);
01996     }
01997 }
01998
01999 /*****
02000 ****/
02001 void GFXcanvas1::drawFastRawVLine(int16_t x, int16_t y, int16_t h,
02002                                     uint16_t color) {
02003     // x & y already in raw (rotation 0) coordinates, no need to transform.
02004     int16_t row_bytes = ((WIDTH + 7) / 8);
02005     uint8_t *buffer = this->getBuffer();
02006     uint8_t *ptr = &buffer[(x / 8) + y * row_bytes];

```

```

02020     if (color > 0) {
02021 #ifdef __AVR__
02022     uint8_t bit_mask = pgm_read_byte(&GFXsetBit[x & 7]);
02023 #else
02024     uint8_t bit_mask = (0x80 >> (x & 7));
02025 #endif
02026     for (int16_t i = 0; i < h; i++) {
02027         *ptr |= bit_mask;
02028         ptr += row_bytes;
02029     }
02030 } else {
02031 #ifdef __AVR__
02032     uint8_t bit_mask = pgm_read_byte(&GFXclrBit[x & 7]);
02033 #else
02034     uint8_t bit_mask = ~(0x80 >> (x & 7));
02035 #endif
02036     for (int16_t i = 0; i < h; i++) {
02037         *ptr &= bit_mask;
02038         ptr += row_bytes;
02039     }
02040 }
02041 }
02042 }
02043
02044 /*****
02045 *****/
02046 /*****
02047 *****/
02048 void GFXcanvas1::drawFastRawHLine(int16_t x, int16_t y, int16_t w,
02049                                     uint16_t color) {
02050 // x & y already in raw (rotation 0) coordinates, no need to transform.
02051 int16_t rowBytes = ((WIDTH + 7) / 8);
02052 uint8_t *buffer = this->getBuffer();
02053 uint8_t *ptr = &buffer[(x / 8) + y * rowBytes];
02054 size_t remainingWidthBits = w;
02055
02056 // check to see if first byte needs to be partially filled
02057 if ((x & 7) > 0) {
02058     // create bit mask for first byte
02059     uint8_t startByteBitMask = 0x00;
02060     for (int8_t i = (x & 7); (i < 8) && (remainingWidthBits > 0); i++) {
02061 #ifdef __AVR__
02062         startByteBitMask |= pgm_read_byte(&GFXsetBit[i]);
02063 #else
02064         startByteBitMask |= (0x80 >> i);
02065 #endif
02066     remainingWidthBits--;
02067 }
02068     if (color > 0) {
02069         *ptr |= startByteBitMask;
02070     } else {
02071         *ptr &= ~startByteBitMask;
02072     }
02073     ptr++;
02074 }
02075 // do the next remainingWidthBits bits
02076 if (remainingWidthBits > 0) {
02077     size_t remainingWholeBytes = remainingWidthBits / 8;
02078     size_t lastByteBits = remainingWidthBits % 8;
02079     uint8_t wholeByteColor = color > 0 ? 0xFF : 0x00;
02080
02081     memset(ptr, wholeByteColor, remainingWholeBytes);
02082
02083     if (lastByteBits > 0) {
02084         uint8_t lastByteBitMask = 0x00;
02085         for (size_t i = 0; i < lastByteBits; i++) {
02086 #ifdef __AVR__
02087             lastByteBitMask |= pgm_read_byte(&GFXsetBit[i]);
02088 #else
02089             lastByteBitMask |= (0x80 >> i);
02090 #endif
02091         }
02092         ptr += remainingWholeBytes;
02093     }
02094     if (color > 0) {
02095         *ptr |= lastByteBitMask;
02096     } else {
02097         *ptr &= ~lastByteBitMask;
02098     }
02099 }
02100
02101 /*****
02102 *****/
02103 /*****
02104 *****/
02105
02106
02107
02108 }
02109
02110 /*****
02111 *****/
02112 /*****
02113 *****/
02114 GFXcanvas8::GFXcanvas8(uint16_t w, uint16_t h) : Adafruit_GFX(w, h) {
02115     uint32_t bytes = w * h;

```

```

02119  if ((buffer = (uint8_t *)malloc(bytes))) {
02120      memset(buffer, 0, bytes);
02121  }
02122 }
02123
02124 /***** *****
02125 ****
02126 ****
02127 ****
02128 ****
02129 GFXcanvas8::~GFXcanvas8(void) {
02130     if (buffer)
02131         free(buffer);
02132 }
02133
02134 /***** *****
02135 ****
02136 ****
02137 ****
02138 ****
02139 void GFXcanvas8::drawPixel(int16_t x, int16_t y, uint16_t color) {
02140     if (buffer) {
02141         if ((x < 0) || (y < 0) || (x >= _width) || (y >= _height))
02142             return;
02143
02144         int16_t t;
02145         switch (rotation) {
02146             case 1:
02147                 t = x;
02148                 x = WIDTH - 1 - y;
02149                 y = t;
02150                 break;
02151             case 2:
02152                 x = WIDTH - 1 - x;
02153                 y = HEIGHT - 1 - y;
02154                 break;
02155             case 3:
02156                 t = x;
02157                 x = y;
02158                 y = HEIGHT - 1 - t;
02159                 break;
02160             default:
02161                 break;
02162         }
02163
02164         buffer[x + y * WIDTH] = color;
02165     }
02166 }
02167 }
02168
02169 /***** *****
02170 ****
02171 ****
02172 ****
02173 ****
02174 uint8_t GFXcanvas8::getPixel(int16_t x, int16_t y) const {
02175     int16_t t;
02176     switch (rotation) {
02177         case 1:
02178             t = x;
02179             x = WIDTH - 1 - y;
02180             y = t;
02181             break;
02182         case 2:
02183             x = WIDTH - 1 - x;
02184             y = HEIGHT - 1 - y;
02185             break;
02186         case 3:
02187             t = x;
02188             x = y;
02189             y = HEIGHT - 1 - t;
02190             break;
02191         default:
02192             break;
02193     }
02194
02195     return getRawPixel(x, y);
02196 }
02197
02198 /***** *****
02199 ****
02200 ****
02201 ****
02202 ****
02203 uint8_t GFXcanvas8::getRawPixel(int16_t x, int16_t y) const {
02204     if ((x < 0) || (y < 0) || (x >= WIDTH) || (y >= HEIGHT))
02205         return 0;
02206     if (buffer) {
02207         return buffer[x + y * WIDTH];
02208     }
02209     return 0;
02210 }
02211
02212 /***** *****
02213 ****
02214 ****
02215 ****
02216 ****
02217 /***** *****
02218 ****
02219 ****
02220 ****
02221 ****
02222 ****
02223 void GFXcanvas8::fillScreen(uint16_t color) {
02224     if (buffer) {
02225         memset(buffer, color, WIDTH * HEIGHT);
02226     }
02227 }
02228
02229 /***** *****
02230 ****
02231 ****
02232 ****
02233 ****
02234 void GFXcanvas8::drawFastVLine(int16_t x, int16_t y, int16_t h,
02235                                 uint16_t color) {
02236 }
```

```

02241 if (h < 0) { // Convert negative heights to positive equivalent
02242     h *= -1;
02243     y -= h - 1;
02244     if (y < 0) {
02245         h += y;
02246         y = 0;
02247     }
02248 }
02249
02250 // Edge rejection (no-draw if totally off canvas)
02251 if ((x < 0) || (x >= width()) || (y >= height()) || ((y + h - 1) < 0)) {
02252     return;
02253 }
02254
02255 if (y < 0) { // Clip top
02256     h += y;
02257     y = 0;
02258 }
02259 if (y + h > height()) { // Clip bottom
02260     h = height() - y;
02261 }
02262
02263 if (getRotation() == 0) {
02264     drawFastRawVLine(x, y, h, color);
02265 } else if (getRotation() == 1) {
02266     int16_t t = x;
02267     x = WIDTH - 1 - y;
02268     y = t;
02269     x -= h - 1;
02270     drawFastRawHLine(x, y, h, color);
02271 } else if (getRotation() == 2) {
02272     x = WIDTH - 1 - x;
02273     y = HEIGHT - 1 - y;
02274
02275     y -= h - 1;
02276     drawFastRawVLine(x, y, h, color);
02277 } else if (getRotation() == 3) {
02278     int16_t t = x;
02279     x = y;
02280     y = HEIGHT - 1 - t;
02281     drawFastRawHLine(x, y, h, color);
02282 }
02283 }
02284
02285 /***** *****
02286 *****
02287
02288 void GFXcanvas8::drawFastHLine(int16_t x, int16_t y, int16_t w,
02289                                 uint16_t color) {
02290
02291     if (w < 0) { // Convert negative widths to positive equivalent
02292         w *= -1;
02293         x -= w - 1;
02294         if (x < 0) {
02295             w += x;
02296             x = 0;
02297         }
02298
02299     // Edge rejection (no-draw if totally off canvas)
02300     if ((y < 0) || (y >= height()) || (x >= width()) || ((x + w - 1) < 0)) {
02301         return;
02302     }
02303
02304     if (x < 0) { // Clip left
02305         w += x;
02306         x = 0;
02307     }
02308     if (x + w >= width()) { // Clip right
02309         w = width() - x;
02310     }
02311
02312     if (getRotation() == 0) {
02313         drawFastRawHLine(x, y, w, color);
02314     } else if (getRotation() == 1) {
02315         int16_t t = x;
02316         x = WIDTH - 1 - y;
02317         y = t;
02318         drawFastRawVLine(x, y, w, color);
02319     } else if (getRotation() == 2) {
02320         x = WIDTH - 1 - x;
02321         y = HEIGHT - 1 - y;
02322         w -= w - 1;
02323         drawFastRawHLine(x, y, w, color);
02324     } else if (getRotation() == 3) {
02325         int16_t t = x;
02326         x = y;
02327     }
02328 }
```

```

02336     y = HEIGHT - 1 - t;
02337     y -= w - 1;
02338     drawFastRawVLine(x, y, w, color);
02339 }
02340 }
02341 ****
02342 ****
02343 void GFXcanvas8::drawFastRawVLine(int16_t x, int16_t y, int16_t h,
02344                                     uint16_t color) {
02345     // x & y already in raw (rotation 0) coordinates, no need to transform.
02346     uint8_t *buffer_ptr = buffer + y * WIDTH + x;
02347     for (int16_t i = 0; i < h; i++) {
02348         (*buffer_ptr) = color;
02349         buffer_ptr += WIDTH;
02350     }
02351 }
02352 ****
02353 ****
02354 void GFXcanvas8::drawFastRawHLine(int16_t x, int16_t y, int16_t w,
02355                                     uint16_t color) {
02356     // x & y already in raw (rotation 0) coordinates, no need to transform.
02357     memset(buffer + y * WIDTH + x, color, w);
02358 }
02359 ****
02360 ****
02361 ****
02362 ****
02363 ****
02364 ****
02365 GFXcanvas16::GFXcanvas16(uint16_t w, uint16_t h) : Adafruit_GFX(w, h) {
02366     uint32_t bytes = w * h * 2;
02367     if ((buffer = (uint16_t *)malloc(bytes))) {
02368         memset(buffer, 0, bytes);
02369     }
02370 }
02371 ****
02372 ****
02373 ****
02374 ****
02375 ****
02376 ****
02377 ****
02378 ****
02379 ****
02380 ****
02381 ****
02382 ****
02383 ****
02384 ****
02385 ****
02386 ****
02387 ****
02388 ****
02389 ****
02390 ****
02391 ****
02392 ****
02393 ****
02394 ****
02395 ****
02396 ****
02397 GFXcanvas16::~GFXcanvas16(void) {
02398     if (buffer)
02399         free(buffer);
02400 }
02401 ****
02402 ****
02403 ****
02404 ****
02405 ****
02406 ****
02407 ****
02408 ****
02409 ****
02410 void GFXcanvas16::drawPixel(int16_t x, int16_t y, uint16_t color) {
02411     if (buffer) {
02412         if ((x < 0) || (y < 0) || (x >= _width) || (y >= _height))
02413             return;
02414
02415         int16_t t;
02416         switch (rotation) {
02417             case 1:
02418                 t = x;
02419                 x = WIDTH - 1 - y;
02420                 y = t;
02421                 break;
02422             case 2:
02423                 x = WIDTH - 1 - x;
02424                 y = HEIGHT - 1 - y;
02425                 break;
02426             case 3:
02427                 t = x;
02428                 x = y;
02429                 y = HEIGHT - 1 - t;
02430                 break;
02431         }
02432
02433         buffer[x + y * WIDTH] = color;
02434     }
02435 }
02436 ****
02437 ****
02438 ****
02439 ****
02440 ****
02441 ****
02442 ****
02443 ****
02444 ****
02445 uint16_t GFXcanvas16::getPixel(int16_t x, int16_t y) const {
02446     int16_t t;
02447     switch (rotation) {
02448         case 1:
02449             t = x;
02450             x = WIDTH - 1 - y;
02451             y = t;
02452             break;
02453         case 2:
02454             x = WIDTH - 1 - x;
02455             y = HEIGHT - 1 - y;
02456             break;
02457         case 3:
02458             t = x;
02459     }

```

```

02459     x = y;
02460     y = HEIGHT - 1 - t;
02461     break;
02462 }
02463 return getRawPixel(x, y);
02464 }
02465
02466 /*****
02467 ****/
02468 uint16_t GFXcanvas16::getRawPixel(int16_t x, int16_t y) const {
02469     if ((x < 0) || (y < 0) || (x >= WIDTH) || (y >= HEIGHT))
02470         return 0;
02471     if (buffer) {
02472         return buffer[x + y * WIDTH];
02473     }
02474     return 0;
02475 }
02476
02477 /*****
02478 ****/
02479 void GFXcanvas16::fillScreen(uint16_t color) {
02480     if (buffer) {
02481         uint8_t hi = color >> 8, lo = color & 0xFF;
02482         if (hi == lo) {
02483             memset(buffer, lo, WIDTH * HEIGHT * 2);
02484         } else {
02485             uint32_t i, pixels = WIDTH * HEIGHT;
02486             for (i = 0; i < pixels; i++)
02487                 buffer[i] = color;
02488         }
02489     }
02490 }
02491
02492 /*****
02493 ****/
02494 void GFXcanvas16::byteSwap(void) {
02495     if (buffer) {
02496         uint32_t i, pixels = WIDTH * HEIGHT;
02497         for (i = 0; i < pixels; i++)
02498             buffer[i] = __builtin_bswap16(buffer[i]);
02499     }
02500 }
02501
02502 }
02503
02504 /*****
02505 ****/
02506 void GFXcanvas16::drawFastVLine(int16_t x, int16_t y, int16_t h,
02507                                 uint16_t color) {
02508     if (h < 0) { // Convert negative heights to positive equivalent
02509         h *= -1;
02510         y -= h - 1;
02511         if (y < 0) {
02512             h += y;
02513             y = 0;
02514         }
02515     }
02516
02517     // Edge rejection (no-draw if totally off canvas)
02518     if ((x < 0) || (x >= width()) || (y >= height()) || ((y + h - 1) < 0)) {
02519         return;
02520     }
02521
02522     if (y < 0) { // Clip top
02523         h += y;
02524         y = 0;
02525     }
02526     if (y + h > height()) { // Clip bottom
02527         h = height() - y;
02528     }
02529
02530     if (getRotation() == 0) {
02531         drawFastRawVLine(x, y, h, color);
02532     } else if (getRotation() == 1) {
02533         int16_t t = x;
02534         x = WIDTH - 1 - y;
02535         y = t;
02536         x -= h - 1;
02537         drawFastRawHLine(x, y, h, color);
02538     } else if (getRotation() == 2) {
02539         x = WIDTH - 1 - x;
02540         y = HEIGHT - 1 - y;
02541
02542         y -= h - 1;
02543         drawFastRawVLine(x, y, h, color);
02544     } else if (getRotation() == 3) {
02545         int16_t t = x;
02546         x = y;
02547         y = HEIGHT - 1 - t;
02548     }
02549
02550 }
02551
02552
02553
02554
02555
02556
02557
02558
02559
02560
02561
02562
02563
02564
02565
02566
02567
02568
02569
02570
02571
02572
02573
02574
02575

```

```

02576     drawFastRawHLine(x, y, h, color);
02577 }
02578 }
02579
02580 /***** *****
02581 *****
02582 *****
02583 void GFXcanvas16::drawFastHLine(int16_t x, int16_t y, int16_t w,
02584                                 uint16_t color) {
02585     if (w < 0) { // Convert negative widths to positive equivalent
02586         w *= -1;
02587         x -= w - 1;
02588         if (x < 0) {
02589             w += x;
02590             x = 0;
02591         }
02592     }
02593
02594     // Edge rejection (no-draw if totally off canvas)
02595     if ((y < 0) || (y >= height()) || (x >= width()) || ((x + w - 1) < 0)) {
02596         return;
02597     }
02598
02599
02600     if (x < 0) { // Clip left
02601         w += x;
02602         x = 0;
02603     }
02604     if (x + w >= width()) { // Clip right
02605         w = width() - x;
02606     }
02607
02608     if (getRotation() == 0) {
02609         drawFastRawHLine(x, y, w, color);
02610     } else if (getRotation() == 1) {
02611         int16_t t = x;
02612         x = WIDTH - 1 - y;
02613         y = t;
02614         drawFastRawVLine(x, y, w, color);
02615     } else if (getRotation() == 2) {
02616         x = WIDTH - 1 - x;
02617         y = HEIGHT - 1 - y;
02618
02619         x -= w - 1;
02620         drawFastRawHLine(x, y, w, color);
02621     } else if (getRotation() == 3) {
02622         int16_t t = x;
02623         x = y;
02624         y = HEIGHT - 1 - t;
02625         y -= w - 1;
02626         drawFastRawVLine(x, y, w, color);
02627     }
02628 }
02629
02630
02631
02632
02633 }
02634
02635 /***** *****
02636 *****
02637 *****
02638 *****
02639 void GFXcanvas16::drawFastRawVLine(int16_t x, int16_t y, int16_t h,
02640                                   uint16_t color) {
02641     // x & y already in raw (rotation 0) coordinates, no need to transform.
02642     uint16_t *buffer_ptr = buffer + y * WIDTH + x;
02643     for (int16_t i = 0; i < h; i++) {
02644         (*buffer_ptr) = color;
02645         buffer_ptr += WIDTH;
02646     }
02647 }
02648
02649
02650
02651
02652 }
02653
02654 /***** *****
02655 *****
02656 *****
02657 *****
02658 *****
02659 *****
02660 *****
02661 *****
02662 *****
02663 *****
02664 *****
02665 *****
02666 *****
02667 *****
02668 *****
02669 *****
02670 }

```

## 15.75 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/batt\_Adafruit\_GFX.h File Reference

```
#include <Arduino.h>
#include "batt_gfxfont.h"
```

### Classes

- class [Adafruit\\_GFX](#)
- class [Adafruit\\_GFX\\_Button](#)  
*A simple drawn button UI element.*
- class [GFXcanvas1](#)  
*A GFX 1-bit canvas context for graphics.*
- class [GFXcanvas8](#)  
*A GFX 8-bit canvas context for graphics.*
- class [GFXcanvas16](#)  
*A GFX 16-bit canvas context for graphics.*

## 15.76 batt\_Adafruit\_GFX.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _ADAFRUIT_GFX_H
00002 #define _ADAFRUIT_GFX_H
00003
00004 #if ARDUINO >= 100
00005 #include "Arduino.h"
00006 #include "Print.h"
00007 #else
00008 #include <Arduino.h>
00009 #endif
00010 #include "batt_gfxfont.h"
00011
00015 class Adafruit_GFX : public Print {
00016
00017 public:
00018     Adafruit_GFX(int16_t w, int16_t h); // Constructor
00019
00020     /*****
00021     *****/
00022     /*****
00023     *****/
00029     virtual void drawPixel(int16_t x, int16_t y, uint16_t color) = 0;
00030
00031     // TRANSACTION API / CORE DRAW API
00032     // These MAY be overridden by the subclass to provide device-specific
00033     // optimized code. Otherwise 'generic' versions are used.
00034     virtual void startWrite(void);
00035     virtual void writePixel(int16_t x, int16_t y, uint16_t color);
00036     virtual void writeFillRect(int16_t x, int16_t y, int16_t w, int16_t h,
00037                               uint16_t color);
00038     virtual void writeFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color);
00039     virtual void writeFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color);
00040     virtual void writeLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
00041                           uint16_t color);
00042     virtual void endWrite(void);
00043
00044     // CONTROL API
00045     // These MAY be overridden by the subclass to provide device-specific
00046     // optimized code. Otherwise 'generic' versions are used.
00047     virtual void setRotation(uint8_t r);
00048     virtual void invertDisplay(bool i);
00049
00050     // BASIC DRAW API
00051     // These MAY be overridden by the subclass to provide device-specific
00052     // optimized code. Otherwise 'generic' versions are used.
00053
00054     // It's good to implement those, even if using transaction API
00055     virtual void drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color);
00056     virtual void drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color);
00057     virtual void fillRect(int16_t x, int16_t y, int16_t w, int16_t h,
```

```

00058         uint16_t color);
00059     virtual void fillScreen(uint16_t color);
00060     // Optional and probably not necessary to change
00061     virtual void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1,
00062                           uint16_t color);
00063     virtual void drawRect(int16_t x, int16_t y, int16_t w, int16_t h,
00064                           uint16_t color);
00065
00066     // These exist only with Adafruit_GFX (no subclass overrides)
00067     void drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color);
00068     void drawCircleHelper(int16_t x0, int16_t y0, int16_t r, uint8_t cornername,
00069                           uint16_t color);
00070     void fillCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color);
00071     void fillCircleHelper(int16_t x0, int16_t y0, int16_t r, uint8_t cornername,
00072                           int16_t delta, uint16_t color);
00073     void drawTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2,
00074                       int16_t y2, uint16_t color);
00075     void fillTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2,
00076                       int16_t y2, uint16_t color);
00077     void drawRoundRect(int16_t x0, int16_t y0, int16_t w, int16_t h,
00078                         int16_t radius, uint16_t color);
00079     void fillRoundRect(int16_t x0, int16_t y0, int16_t w, int16_t h,
00080                         int16_t radius, uint16_t color);
00081     void drawBitmap(int16_t x, int16_t y, const uint8_t bitmap[], int16_t w,
00082                      int16_t h, uint16_t color);
00083     void drawBitmap(int16_t x, int16_t y, const uint8_t bitmap[], int16_t w,
00084                      int16_t h, uint16_t color, uint16_t bg);
00085     void drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h,
00086                      uint16_t color);
00087     void drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h,
00088                      uint16_t color, uint16_t bg);
00089     void drawXBitmap(int16_t x, int16_t y, const uint8_t bitmap[], int16_t w,
00090                      int16_t h, uint16_t color);
00091     void drawGrayscaleBitmap(int16_t x, int16_t y, const uint8_t bitmap[],
00092                               int16_t w, int16_t h);
00093     void drawGrayscaleBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w,
00094                               int16_t h);
00095     void drawGrayscaleBitmap(int16_t x, int16_t y, const uint8_t bitmap[],
00096                               const uint8_t mask[], int16_t w, int16_t h);
00097     void drawGrayscaleBitmap(int16_t x, int16_t y, uint8_t *bitmap, uint8_t *mask,
00098                               int16_t w, int16_t h);
00099     void drawRGBitmap(int16_t x, int16_t y, const uint16_t bitmap[], int16_t w,
00100                      int16_t h);
00101     void drawRGBitmap(int16_t x, int16_t y, uint16_t *bitmap, int16_t w,
00102                      int16_t h);
00103     void drawRGBitmap(int16_t x, int16_t y, const uint16_t bitmap[],
00104                      const uint8_t mask[], int16_t w, int16_t h);
00105     void drawRGBitmap(int16_t x, int16_t y, uint16_t *bitmap, uint8_t *mask,
00106                      int16_t w, int16_t h);
00107     void drawChar(int16_t x, int16_t y, unsigned char c, uint16_t color,
00108                   uint16_t bg, uint8_t size);
00109     void drawChar(int16_t x, int16_t y, unsigned char c, uint16_t color,
00110                   uint16_t bg, uint8_t size_x, uint8_t size_y);
00111     void getTextBounds(const char *string, int16_t x, int16_t y, int16_t *x1,
00112                         int16_t *y1, uint16_t *w, uint16_t *h);
00113     void getTextBounds(const __FlashStringHelper *s, int16_t x, int16_t y,
00114                         int16_t *x1, int16_t *y1, uint16_t *w, uint16_t *h);
00115     void getTextBounds(const String &str, int16_t x, int16_t y, int16_t *x1,
00116                         int16_t *y1, uint16_t *w, uint16_t *h);
00117     void setTextSize(uint8_t s);
00118     void setTextSize(uint8_t sx, uint8_t sy);
00119     void setFont(const GFXfont *f = NULL);
00120
00121     /*****
00122     *****/
00123     void setCursor(int16_t x, int16_t y) {
00124         cursor_x = x;
00125         cursor_y = y;
00126     }
00127
00128     /*****
00129     *****/
00130     void setTextColor(uint16_t c) { textcolor = textbgcolor = c; }
00131
00132     /*****
00133     *****/
00134     void setTextColor(uint16_t c, uint16_t bg) {
00135         textcolor = c;
00136         textbgcolor = bg;
00137     }
00138
00139     /*****
00140     *****/
00141     void setTextColor(uint16_t c) { textcolor = textbgcolor = c; }
00142
00143     /*****
00144     *****/
00145     void setTextColor(uint16_t c, uint16_t bg) {
00146         textcolor = c;
00147         textbgcolor = bg;
00148     }
00149
00150     /*****
00151     *****/
00152     void setTextColor(uint16_t c, uint16_t bg) {
00153         textcolor = c;
00154         textbgcolor = bg;
00155     }
00156
00157     /*****
00158     *****/
00159     void setTextWrap(bool w) { wrap = w; }
00160
00161     /*****
00162     *****/
00163     void setTextWrap(bool w) { wrap = w; }
00164
00165     /*****
00166     *****/
00167 
```

```

00178     void cp437(bool x = true) { _cp437 = x; }
00179
00180     using Print::write;
00181 #if ARDUINO >= 100
00182     virtual size_t write(uint8_t);
00183 #else
00184     virtual size_t write(uint8_t);
00185 #endif
00186
00187 //*****
00188 //*****
00189 int16_t width(void) const { return _width; }
00190
00191 //*****
00192 //*****
00193 int16_t height(void) const { return _height; }
00194
00195 //*****
00196 //*****
00197 int16_t getRotation(void) const { return rotation; }
00198
00199 // get current cursor position (get rotation safe maximum values,
00200 // using: width() for x, height() for y)
00201 //*****
00202 //*****
00203 int16_t getCursorX(void) const { return cursor_x; }
00204
00205 //*****
00206 //*****
00207 int16_t getCursorY(void) const { return cursor_y; }
00208
00209 protected:
00210     void charBounds(unsigned char c, int16_t *x, int16_t *y, int16_t *minx,
00211                     int16_t *miny, int16_t *maxx, int16_t *maxy);
00212     int16_t WIDTH;
00213     int16_t HEIGHT;
00214     int16_t _width;
00215     int16_t _height;
00216     int16_t cursor_x;
00217     int16_t cursor_y;
00218     uint16_t textcolor;
00219     uint16_t textbgcolor;
00220     uint8_t textsize_x;
00221     uint8_t textsize_y;
00222     uint8_t rotation;
00223     bool wrap;
00224     bool _cp437;
00225     GFXfont *gfxFont;
00226 };
00227
00228 class Adafruit_GFX_Button {
00229 public:
00230     Adafruit_GFX_Button(void);
00231     // "Classic" initButton() uses center & size
00232     void initButton(Adafruit_GFX *gfx, int16_t x, int16_t y, uint16_t w,
00233                     uint16_t h, uint16_t outline, uint16_t fill,
00234                     uint16_t textcolor, char *label, uint8_t textsize);
00235     void initButton(Adafruit_GFX *gfx, int16_t x, int16_t y, uint16_t w,
00236                     uint16_t h, uint16_t outline, uint16_t fill,
00237                     uint16_t textcolor, char *label, uint8_t textsize_x,
00238                     uint8_t textsize_y);
00239     // New/alt initButton() uses upper-left corner & size
00240     void initButtonUL(Adafruit_GFX *gfx, int16_t x1, int16_t y1, uint16_t w,
00241                     uint16_t h, uint16_t outline, uint16_t fill,
00242                     uint16_t textcolor, char *label, uint8_t textsize);
00243     void initButtonUL(Adafruit_GFX *gfx, int16_t x1, int16_t y1, uint16_t w,
00244                     uint16_t h, uint16_t outline, uint16_t fill,
00245                     uint16_t textcolor, char *label, uint8_t textsize_x,
00246                     uint8_t textsize_y);
00247     void drawButton(bool inverted = false);
00248     bool contains(int16_t x, int16_t y);
00249
00250     //*****
00251     void press(bool p) {
00252         laststate = currstate;
00253         currstate = p;
00254     }
00255
00256     bool justPressed();
00257     bool justReleased();
00258
00259     //*****
00260     bool isPressed(void) { return currstate; }
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293

```

```

00294 private:
00295     Adafruit_GFX *_gfx;
00296     int16_t _x1, _y1; // Coordinates of top-left corner
00297     uint16_t _w, _h;
00298     uint8_t _textsize_x;
00299     uint8_t _textsize_y;
00300     uint16_t _outlinecolor, _fillcolor, _textcolor;
00301     char _label[10];
00302
00303     bool currstate, laststate;
00304 };
00305
00306 class GFXcanvas1 : public Adafruit_GFX {
00307     public:
00308         GFXcanvas1(uint16_t w, uint16_t h);
00309         ~GFXcanvas1(void);
00310         void drawPixel(int16_t x, int16_t y, uint16_t color);
00311         void fillScreen(uint16_t color);
00312         void drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color);
00313         void drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color);
00314         bool getPixel(int16_t x, int16_t y) const;
00315         protected:
00316         /*****
00317         *****/
00318         uint8_t *getBuffer(void) const { return buffer; }
00319
00320     private:
00321         uint8_t *buffer;
00322
00323     #ifdef __AVR__
00324         // Bitmask tables of 0x80»X and ~(0x80»X), because X»Y is slow on AVR
00325         static const uint8_t PROGMEM GFXsetBit[], GFXclrBit[];
00326     #endif
00327 };
00328
00329 class GFXcanvas8 : public Adafruit_GFX {
00330     public:
00331         GFXcanvas8(uint16_t w, uint16_t h);
00332         ~GFXcanvas8(void);
00333         void drawPixel(int16_t x, int16_t y, uint16_t color);
00334         void fillScreen(uint16_t color);
00335         void drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color);
00336         void drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color);
00337         uint8_t getPixel(int16_t x, int16_t y) const;
00338         protected:
00339         uint8_t *getBuffer(void) const { return buffer; }
00340
00341     private:
00342         uint8_t *buffer;
00343
00344 class GFXcanvas16 : public Adafruit_GFX {
00345     public:
00346         GFXcanvas16(uint16_t w, uint16_t h);
00347         ~GFXcanvas16(void);
00348         void drawPixel(int16_t x, int16_t y, uint16_t color);
00349         void fillScreen(uint16_t color);
00350         void byteSwap(void);
00351         void drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color);
00352         void drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color);
00353         uint16_t getPixel(int16_t x, int16_t y) const;
00354         protected:
00355         uint16_t *getBuffer(void) const { return buffer; }
00356
00357     private:
00358         uint16_t *buffer;
00359
00360 #endif // _ADAFRUIT_GFX_H

```

## 15.77 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/batt\_↔ gfxfont.h File Reference

### Classes

- struct **GFXglyph**  
*Font data stored PER GLYPH.*
- struct **GFXfont**  
*Data stored for FONT AS A WHOLE.*

## 15.78 batt\_gfxfont.h

[Go to the documentation of this file.](#)

```
00001 // Font structures for newer Adafruit_GFX (1.1 and later).  
00002 // Example fonts are included in 'Fonts' directory.  
00003 // To use a font in your Arduino sketch, #include the corresponding .h  
00004 // file and pass address of GFXfont struct to setFont(). Pass NULL to  
00005 // revert to 'classic' fixed-space bitmap font.  
00006  
00007 #ifndef _GFXFONT_H_  
00008 #define _GFXFONT_H_  
00009  
00011 typedef struct {  
00012     uint16_t bitmapOffset;  
00013     uint8_t width;  
00014     uint8_t height;  
00015     uint8_t xAdvance;  
00016     int8_t xOffset;  
00017     int8_t yOffset;  
00018 } GFXglyph;  
00019  
00021 typedef struct {  
00022     uint8_t *bitmap;  
00023     GFXglyph *glyph;  
00024     uint16_t first;  
00025     uint16_t last;  
00026     uint8_t yAdvance;  
00027 } GFXfont;  
00028  
00029 #endif // _GFXFONT_H_
```

## 15.79 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0008-Adafruit\_GFX\_Library/batt\_↔ glcdfont.c File Reference

### Macros

- **#define FONT5X7\_H**
- **#define PROGMEM**

#### 15.79.1 Macro Definition Documentation

##### 15.79.1.1 FONT5X7\_H

```
#define FONT5X7_H  
Definition at line 5 of file batt_glcdfont.c.
```

### 15.79.1.2 PROGMEM

```
#define PROGMEM
```

Definition at line 17 of file [batt\\_glcdfont.c](#).

## 15.80 batt\_glcdfont.c

[Go to the documentation of this file.](#)

```
00001 // This is the 'classic' fixed-space bitmap font for Adafruit_GFX since 1.0.
00002 // See gfxfont.h for newer custom bitmap font info.
00003
00004 #ifndef FONT5X7_H
00005 #define FONT5X7_H
00006
00007 #ifdef __AVR__
00008 #include <avr/io.h>
00009 #include <avr/pgmspace.h>
00010 #elif defined(ESP8266)
00011 #include <pgmspace.h>
00012 #elif defined(__IMXRT1052__) || defined(__IMXRT1062__)
00013 // PROGMEM is defefind for T4 to place data in specific memory section
00014 #undef PROGMEM
00015 #define PROGMEM
00016 #else
00017 #define PROGMEM
00018 #endif
00019
00020 // Standard ASCII 5x7 font
00021
00022 static const unsigned char font[] PROGMEM = {
00023     0x00, 0x00, 0x00, 0x00, 0x3E, 0x5B, 0x4F, 0x5B, 0x3E, 0x3E, 0x6B,
00024     0x4F, 0x6B, 0x3E, 0x1C, 0x3E, 0x7C, 0x3E, 0x1C, 0x18, 0x3C, 0x7E, 0x3C,
00025     0x18, 0x1C, 0x57, 0x7D, 0x57, 0x1C, 0x1C, 0x5E, 0x7F, 0x5E, 0x1C, 0x00,
00026     0x18, 0x3C, 0x18, 0x00, 0xFF, 0xE7, 0xC3, 0xE7, 0xFF, 0x00, 0x18, 0x24,
00027     0x18, 0x00, 0xFF, 0xE7, 0xDB, 0xE7, 0xFF, 0x30, 0x48, 0x3A, 0x06, 0x0E,
00028     0x26, 0x29, 0x79, 0x29, 0x26, 0x40, 0x7F, 0x05, 0x05, 0x07, 0x40, 0x7F,
00029     0x05, 0x25, 0x3F, 0x5A, 0x3C, 0xE7, 0x3C, 0x5A, 0x7F, 0x3E, 0x1C, 0x1C,
00030     0x08, 0x08, 0x1C, 0x1C, 0x3E, 0x7F, 0x14, 0x22, 0x7F, 0x22, 0x14, 0x5F,
00031     0x5F, 0x00, 0x5F, 0x5F, 0x06, 0x09, 0x7F, 0x01, 0x7F, 0x00, 0x66, 0x89,
00032     0x95, 0x6A, 0x60, 0x60, 0x60, 0x60, 0x94, 0xA2, 0xFF, 0xA2, 0x94,
00033     0x08, 0x04, 0x7E, 0x04, 0x08, 0x10, 0x20, 0x7E, 0x20, 0x10, 0x08, 0x08,
00034     0x2A, 0x1C, 0x08, 0x08, 0x1C, 0x2A, 0x08, 0x08, 0x1E, 0x10, 0x10, 0x10,
00035     0x10, 0x0C, 0x1E, 0x0C, 0x1E, 0x0C, 0x30, 0x38, 0x3E, 0x38, 0x30, 0x06,
00036     0x0E, 0x3E, 0x0E, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x5F,
00037     0x00, 0x00, 0x00, 0x07, 0x00, 0x07, 0x00, 0x14, 0x7F, 0x14, 0x7F, 0x14,
00038     0x24, 0x2A, 0x7F, 0x2A, 0x12, 0x23, 0x13, 0x08, 0x64, 0x62, 0x36, 0x49,
00039     0x56, 0x20, 0x50, 0x00, 0x08, 0x07, 0x03, 0x00, 0x00, 0x1C, 0x22, 0x41,
00040     0x00, 0x00, 0x41, 0x22, 0x1C, 0x00, 0x2A, 0x1C, 0x7F, 0x1C, 0x2A, 0x08,
00041     0x08, 0x3E, 0x08, 0x08, 0x00, 0x80, 0x70, 0x30, 0x00, 0x08, 0x08, 0x08,
00042     0x08, 0x08, 0x00, 0x00, 0x60, 0x60, 0x00, 0x20, 0x10, 0x08, 0x04, 0x02,
00043     0x3E, 0x51, 0x49, 0x45, 0x3E, 0x00, 0x42, 0x7F, 0x40, 0x00, 0x72, 0x49,
00044     0x49, 0x49, 0x46, 0x21, 0x41, 0x49, 0x4D, 0x33, 0x18, 0x14, 0x12, 0x7F,
00045     0x10, 0x27, 0x45, 0x45, 0x45, 0x39, 0x3C, 0x4A, 0x49, 0x49, 0x31, 0x41,
00046     0x21, 0x11, 0x09, 0x07, 0x36, 0x49, 0x49, 0x49, 0x36, 0x46, 0x49, 0x49,
00047     0x29, 0x1E, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00, 0x40, 0x34, 0x00, 0x00,
00048     0x00, 0x08, 0x14, 0x22, 0x41, 0x14, 0x14, 0x14, 0x14, 0x00, 0x41,
00049     0x22, 0x14, 0x08, 0x02, 0x01, 0x59, 0x09, 0x06, 0x3E, 0x41, 0x5D, 0x59,
00050     0x4E, 0x7C, 0x12, 0x11, 0x12, 0x7C, 0x7F, 0x49, 0x49, 0x49, 0x36, 0x3E,
00051     0x41, 0x41, 0x41, 0x22, 0x7F, 0x41, 0x41, 0x41, 0x3E, 0x7F, 0x49, 0x49,
00052     0x49, 0x41, 0x7F, 0x09, 0x09, 0x01, 0x3E, 0x41, 0x41, 0x51, 0x73,
00053     0x7F, 0x08, 0x08, 0x7F, 0x00, 0x41, 0x7F, 0x41, 0x00, 0x20, 0x40,
00054     0x41, 0x3F, 0x01, 0x7F, 0x08, 0x14, 0x22, 0x41, 0x7F, 0x40, 0x40, 0x40,
00055     0x40, 0x7F, 0x02, 0x1C, 0x02, 0x7F, 0x7F, 0x04, 0x08, 0x10, 0x7F, 0x3E,
00056     0x41, 0x41, 0x41, 0x3E, 0x7F, 0x09, 0x09, 0x01, 0x06, 0x3E, 0x41, 0x51,
00057     0x21, 0x5E, 0x7F, 0x09, 0x19, 0x29, 0x46, 0x26, 0x49, 0x49, 0x49, 0x32,
00058     0x03, 0x01, 0x7F, 0x01, 0x03, 0x3F, 0x40, 0x40, 0x40, 0x3F, 0x1F, 0x20,
00059     0x40, 0x20, 0x1F, 0x3F, 0x40, 0x38, 0x40, 0x3F, 0x63, 0x14, 0x08, 0x14,
00060     0x63, 0x03, 0x04, 0x78, 0x04, 0x03, 0x61, 0x59, 0x49, 0x4D, 0x43, 0x00,
00061     0x7F, 0x41, 0x41, 0x41, 0x02, 0x04, 0x08, 0x10, 0x20, 0x00, 0x41, 0x41,
00062     0x41, 0x7F, 0x04, 0x02, 0x01, 0x02, 0x04, 0x40, 0x40, 0x40, 0x40, 0x40,
00063     0x00, 0x03, 0x07, 0x08, 0x00, 0x20, 0x54, 0x54, 0x78, 0x40, 0x7F, 0x28,
00064     0x44, 0x44, 0x38, 0x38, 0x44, 0x44, 0x44, 0x28, 0x38, 0x44, 0x44, 0x28,
00065     0x7F, 0x38, 0x54, 0x54, 0x54, 0x54, 0x18, 0x00, 0x08, 0x7E, 0x09, 0x02, 0x18,
00066     0xA4, 0xA4, 0x9C, 0x78, 0x7F, 0x08, 0x04, 0x04, 0x78, 0x00, 0x44, 0x7D,
00067     0x40, 0x00, 0x20, 0x40, 0x40, 0x3D, 0x00, 0x7F, 0x10, 0x28, 0x44, 0x00,
00068     0x00, 0x41, 0x7F, 0x40, 0x00, 0x7C, 0x04, 0x78, 0x04, 0x78, 0x7C, 0x08,
00069     0x04, 0x04, 0x78, 0x38, 0x44, 0x44, 0x44, 0x38, 0xFC, 0x18, 0x24, 0x24,
00070     0x18, 0x18, 0x24, 0x24, 0x18, 0xFC, 0x7C, 0x08, 0x04, 0x04, 0x08, 0x48,
00071     0x54, 0x54, 0x54, 0x24, 0x04, 0x04, 0x3F, 0x44, 0x24, 0x3C, 0x40, 0x40,
00072     0x20, 0x7C, 0x1C, 0x20, 0x40, 0x20, 0x1C, 0x3C, 0x40, 0x30, 0x40, 0x3C,
00073     0x44, 0x28, 0x10, 0x28, 0x44, 0x4C, 0x90, 0x90, 0x90, 0x7C, 0x44, 0x64,
00074     0x54, 0x4C, 0x44, 0x00, 0x08, 0x36, 0x41, 0x00, 0x00, 0x00, 0x77, 0x00,
00075     0x00, 0x00, 0x41, 0x36, 0x08, 0x00, 0x02, 0x01, 0x04, 0x02, 0x3C,
```

```

00076      0x26, 0x23, 0x26, 0x3C, 0x1E, 0xA1, 0xA1, 0x61, 0x12, 0x3A, 0x40, 0x40,
00077      0x20, 0x7A, 0x38, 0x54, 0x54, 0x54, 0x55, 0x59, 0x21, 0x55, 0x55, 0x79, 0x41,
00078      0x22, 0x54, 0x54, 0x78, 0x42, // a-umlaut
00079      0x21, 0x55, 0x54, 0x78, 0x40, 0x20, 0x54, 0x55, 0x79, 0x40, 0x0C, 0x1E,
00080      0x52, 0x72, 0x12, 0x39, 0x55, 0x55, 0x55, 0x59, 0x39, 0x54, 0x54, 0x54,
00081      0x59, 0x39, 0x55, 0x54, 0x54, 0x58, 0x00, 0x00, 0x45, 0x7C, 0x41, 0x00,
00082      0x02, 0x45, 0x7D, 0x42, 0x00, 0x01, 0x45, 0x7C, 0x40, 0x7D, 0x12, 0x11,
00083      0x12, 0x7D, // A-umlaut
00084      0xF0, 0x28, 0x25, 0x28, 0xF0, 0x7C, 0x54, 0x55, 0x45, 0x00, 0x20, 0x54,
00085      0x54, 0x7C, 0x54, 0x7C, 0x0A, 0x09, 0x7F, 0x49, 0x32, 0x49, 0x49, 0x49,
00086      0x32, 0x3A, 0x44, 0x44, 0x44, 0x3A, // o-umlaut
00087      0x32, 0x4A, 0x48, 0x48, 0x30, 0x3A, 0x41, 0x41, 0x21, 0x7A, 0x3A, 0x42,
00088      0x40, 0x20, 0x78, 0x00, 0x9D, 0xA0, 0xA0, 0x7D, 0x3D, 0x42, 0x42, 0x42,
00089      0x3D, // O-umlaut
00090      0x3D, 0x40, 0x40, 0x40, 0x3D, 0x3C, 0x24, 0xFF, 0x24, 0x24, 0x48, 0x7E,
00091      0x49, 0x43, 0x66, 0x2B, 0x2F, 0xFC, 0x2F, 0x2B, 0xFF, 0x09, 0x29, 0xF6,
00092      0x20, 0xC0, 0x88, 0x7E, 0x09, 0x03, 0x20, 0x54, 0x54, 0x79, 0x41, 0x00,
00093      0x00, 0x44, 0x7D, 0x41, 0x30, 0x48, 0x48, 0x4A, 0x32, 0x38, 0x40, 0x40,
00094      0x22, 0x7A, 0x00, 0x7A, 0x0A, 0x0A, 0x72, 0x7D, 0x0D, 0x19, 0x31, 0x7D,
00095      0x26, 0x29, 0x29, 0x2F, 0x28, 0x26, 0x29, 0x29, 0x29, 0x26, 0x30, 0x48,
00096      0x4D, 0x40, 0x20, 0x38, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
00097      0x38, 0x2F, 0x10, 0xC8, 0xAC, 0xBA, 0x2F, 0x10, 0x28, 0x34, 0xFA, 0x00,
00098      0x00, 0x7B, 0x00, 0x00, 0x08, 0x2A, 0x14, 0x22, 0x14, 0x2A, 0x14,
00099      0x14, 0x08, 0x55, 0x00, 0x55, 0x00, 0x55, // #176 (25% block) missing in old
00100      // code
00101      0xAA, 0x55, 0xAA, 0x55, 0xAA, // 50% block
00102      0xFF, 0x55, 0xFF, 0x55, 0xFF, // 75% block
00103      0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x10, 0x10, 0x10, 0x10, 0xFF, 0x00, 0x14, 0x14,
00104      0x14, 0xFF, 0x00, 0x10, 0x10, 0xFF, 0x00, 0xFF, 0x10, 0x10, 0xF0, 0x10,
00105      0xF0, 0x14, 0x14, 0x14, 0xFC, 0x00, 0x14, 0x14, 0xF7, 0x00, 0xFF, 0x00,
00106      0x00, 0xFF, 0x00, 0x0F, 0x14, 0x14, 0xF4, 0x04, 0xFC, 0x14, 0x14, 0x17,
00107      0x10, 0x1F, 0x10, 0x10, 0x1F, 0x10, 0x1F, 0x14, 0x14, 0x14, 0x1F, 0x00,
00108      0x10, 0x10, 0x10, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x10, 0x10, 0x10,
00109      0x10, 0x1F, 0x10, 0x10, 0x10, 0xF0, 0x10, 0x00, 0x00, 0x00, 0xFF,
00110      0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0xFF, 0x10, 0x00,
00111      0x00, 0x00, 0xFF, 0x14, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x1F,
00112      0x10, 0x17, 0x00, 0x00, 0xFC, 0x04, 0xF4, 0x14, 0x14, 0x17, 0x10, 0x17,
00113      0x14, 0x14, 0xF4, 0x04, 0xF4, 0x00, 0x00, 0xFF, 0x00, 0xF7, 0x14, 0x14,
00114      0x14, 0x14, 0x14, 0x14, 0xF7, 0x00, 0xF7, 0x14, 0x14, 0x14, 0x17,
00115      0x14, 0x10, 0x10, 0x1F, 0x10, 0x1F, 0x14, 0x14, 0x14, 0x14, 0x10,
00116      0x10, 0xF0, 0x10, 0xF0, 0x00, 0x00, 0x1F, 0x10, 0x1F, 0x00, 0x00, 0x00,
00117      0x1F, 0x14, 0x00, 0x00, 0x00, 0xFC, 0x14, 0x00, 0x00, 0xF0, 0x10, 0xF0,
00118      0x10, 0x10, 0xFF, 0x10, 0xFF, 0x14, 0x14, 0x14, 0xFF, 0x14, 0x10,
00119      0x10, 0x1F, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x10, 0xFF, 0xFF, 0x10, 0x00,
00120      0xFF, 0xF0, 0xF0, 0xF0, 0xF0, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
00121      0x00, 0x00, 0xFF, 0xFF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x38, 0x44, 0x44,
00122      0x38, 0x44, 0xFC, 0x4A, 0x4A, 0x4A, 0x34, // sharp-s or beta
00123      0x7E, 0x02, 0x02, 0x06, 0x06, 0x02, 0x7E, 0x02, 0x7E, 0x02, 0x63, 0x55,
00124      0x49, 0x41, 0x63, 0x38, 0x44, 0x44, 0x3C, 0x04, 0x40, 0x7E, 0x20, 0x1E,
00125      0x20, 0x06, 0x02, 0x7E, 0x02, 0x02, 0x99, 0xA5, 0xE7, 0xA5, 0x99, 0x1C,
00126      0x2A, 0x49, 0x2A, 0x1C, 0x4C, 0x72, 0x01, 0x72, 0x4C, 0x30, 0x4A, 0x4D,
00127      0x4D, 0x30, 0x30, 0x48, 0x78, 0x48, 0x30, 0xBC, 0x62, 0x5A, 0x46, 0x3D,
00128      0x3E, 0x49, 0x49, 0x49, 0x00, 0x7E, 0x01, 0x01, 0x01, 0x7E, 0x2A, 0x2A,
00129      0x2A, 0x2A, 0x2A, 0x44, 0x44, 0x5F, 0x44, 0x44, 0x40, 0x51, 0x4A, 0x44,
00130      0x40, 0x40, 0x44, 0x4A, 0x51, 0x40, 0x00, 0x00, 0xFF, 0x01, 0x03, 0xE0,
00131      0x80, 0xFF, 0x00, 0x00, 0x08, 0x08, 0x6B, 0x6B, 0x08, 0x36, 0x12, 0x36,
00132      0x24, 0x36, 0x06, 0x0F, 0x09, 0x0F, 0x06, 0x00, 0x00, 0x18, 0x18, 0x00,
00133      0x00, 0x00, 0x10, 0x10, 0x00, 0x30, 0x40, 0xFF, 0x01, 0x01, 0x00, 0x1F,
00134      0x01, 0x01, 0x1E, 0x00, 0x19, 0x1D, 0x17, 0x12, 0x00, 0x3C, 0x3C, 0x3C,
00135      0x3C, 0x00, 0x00, 0x00, 0x00, 0x00 // #255 NBSP
00136 };
00137
00138 // allow clean compilation with [-Wunused-const-variable=] and [-Wall]
00139 static inline void avoid_unused_const_variable_compiler_warning(void) {
00140     (void)font;
00141 }
00142
00143 #endif // FONT5X7_H

```

## 15.81 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/batt\_ArduinoLowPower.cpp File Reference

### 15.82 batt\_ArduinoLowPower.cpp

[Go to the documentation of this file.](#)

```
00001 #if defined(ARDUINO_ARCH_SAMD)
00002
```

```

00003 #include "batt_ArduinoLowPower.h"
00004
00005 static void configGCLK6()
00006 {
00007     // enable EIC clock
00008     GCLK->CLKCTRL.bit.CLKEN = 0; //disable GCLK module
00009     while (GCLK->STATUS.bit.SYNCBUSY)
00010     ;
00011
00012     GCLK->CLKCTRL.reg = (uint16_t)(GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK6 |
00013         GCLK_CLKCTRL_ID(GCM_EIC)); //EIC clock switched on GCLK6
00014     while (GCLK->STATUS.bit.SYNCBUSY)
00015     ;
00016
00017     GCLK->GENCTRL.reg = (GCLK_GENCTRL_GENEN | GCLK_GENCTRL_SRC_OSCULP32K | GCLK_GENCTRL_ID(6));
00018     //source for GCLK6 is OSCULP32K
00019     while (GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY)
00020     ;
00021
00022     GCLK->GENCTRL.bit.RUNSTDBY = 1; //GCLK6 run standby
00023     while (GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY)
00024     ;
00025
00026     /* Errata: Make sure that the Flash does not power all the way down
00027      * when in sleep mode. */
00028 }
00029 void ArduinoLowPowerClass::idle()
00030 {
00031     SCB->SCR &= ~SCB_SCR_SLEEPDEEP_Msk;
00032     PM->SLEEP.reg = 2;
00033     __DSB();
00034     __WFI();
00035 }
00036
00037 void ArduinoLowPowerClass::idle(uint32_t millis)
00038 {
00039     setAlarmIn(millis);
00040     idle();
00041 }
00042
00043 void ArduinoLowPowerClass::sleep()
00044 {
00045     bool restoreUSBDevice = false;
00046     if (SERIAL_PORT_USBVIRTUAL)
00047     {
00048         //USBDevice.standby();
00049     }
00050     else
00051     {
00052         USBDevice.detach();
00053         restoreUSBDevice = true;
00054     }
00055     // Disable systick interrupt: See
00056     //www.avrfreaks.net/forum/samd21-samd21e16b-sporadically-locks-and-does-not-wake-standby-sleep-mode
00057     SysTick->CTRL &= ~SysTick_CTRL_TICKINT_Msk;
00058     SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
00059     __DSB();
00060     __WFI();
00061     // Enable systick interrupt
00062     SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;
00063     if (restoreUSBDevice)
00064     {
00065         USBDevice.attach();
00066     }
00067 void ArduinoLowPowerClass::deatchRTCInterrupt()
00068 {
00069     rtc.detachInterrupt();
00070 }
00071 void ArduinoLowPowerClass::sleep(uint32_t millis)
00072 {
00073     setAlarmIn(millis);
00074     sleep();
00075 }
00076
00077 void ArduinoLowPowerClass::deepSleep()
00078 {
00079     sleep();
00080 }
00081
00082 void ArduinoLowPowerClass::deepSleep(uint32_t millis)
00083 {
00084     sleep(millis);
00085 }
00086

```

```
00087 void ArduinoLowPowerClass::setAlarmIn(uint32_t millis)
00088 {
00089
00090     if (!rtc.isConfigured())
00091     {
00092         attachInterruptWakeup(RTC_ALARM_WAKEUP, NULL, (irq_mode)0);
00093     }
00094
00095     uint32_t now = rtc.getEpoch();
00096     rtc.setAlarmEpoch(now + millis / 1000);
00097     rtc.enableAlarm(rtc.MATCH_YYMMDDHHMMSS);
00098 }
00099
00100 void ArduinoLowPowerClass::attachInterruptWakeup(uint32_t pin, voidFuncPtr callback, irq_mode mode)
00101 {
00102
00103     if (pin > PINS_COUNT)
00104     {
00105         // check for external wakeup sources
00106         // RTC library should call this API to enable the alarm subsystem
00107         switch (pin)
00108         {
00109             case RTC_ALARM_WAKEUP:
00110                 rtc.begin(false);
00111                 rtc.attachInterrupt(callback);
00112                 /*case UART_WAKEUP:*/
00113             }
00114         return;
00115     }
00116
00117     EExt_Interrupts in = g_APinDescription[pin].ulExtInt;
00118     if (in == NOT_AN_INTERRUPT || in == EXTERNAL_INT_NMI)
00119         return;
00120
00121     //pinMode(pin, INPUT_PULLUP);
00122     attachInterrupt(pin, callback, mode);
00123
00124     configGCLK6();
00125
00126     // Enable wakeup capability on pin in case being used during sleep
00127     EIC->WAKEUP.reg |= (1 << in);
00128 }
00129
00130 void ArduinoLowPowerClass::attachAdcInterrupt(uint32_t pin, voidFuncPtr callback, adc_interrupt mode,
00131     uint16_t lo, uint16_t hi)
00132 {
00133     uint8_t winmode = 0;
00134
00135     switch (mode)
00136     {
00137         case ADC_INT_BETWEEN:
00138             winmode = ADC_WINCTRL_WINMODE_MODE3;
00139             break;
00140         case ADC_INT_OUTSIDE:
00141             winmode = ADC_WINCTRL_WINMODE_MODE4;
00142             break;
00143         case ADC_INT_ABOVE_MIN:
00144             winmode = ADC_WINCTRL_WINMODE_MODE1;
00145             break;
00146         case ADC_INT_BELOW_MAX:
00147             winmode = ADC_WINCTRL_WINMODE_MODE2;
00148             break;
00149         default:
00150             return;
00151     }
00152
00153     adc_cb = callback;
00154
00155     configGCLK6();
00156
00157     // Configure ADC to use GCLK6 (OSCULP32K)
00158     while (GCLK->STATUS.bit.SYNCBUSY)
00159     {
00160     }
00161     GCLK->CLKCTRL.reg = GCLK_CLKCTRL_ID_ADC | GCLK_CLKCTRL_GEN_GCLK6 | GCLK_CLKCTRL_CLKEN;
00162     while (GCLK->STATUS.bit.SYNCBUSY)
00163     {
00164     }
00165
00166     // Set ADC prescaler as low as possible
00167     ADC->CTRLB.bit.PRESCALER = ADC_CTRLB_PRESCALER_DIV4;
00168     while (ADC->STATUS.bit.SYNCBUSY)
00169     {
00170     }
00171
00172     // Configure window mode
00173     ADC->WINLT.reg = lo;
```

```
00173     ADC->WINUT.reg = hi;
00174     ADC->WINCTRL.reg = winmode;
00175     while (ADC->STATUS.bit.SYNCBUSY)
00176     {
00177     }
00178
00179     // Enable window interrupt
00180     ADC->INTENSET.bit.WINMON = 1;
00181     while (ADC->STATUS.bit.SYNCBUSY)
00182     {
00183     }
00184
00185     // Enable ADC in standby mode
00186     ADC->CTRLA.bit.RUNSTDBY = 1;
00187     while (ADC->STATUS.bit.SYNCBUSY)
00188     {
00189     }
00190
00191     // Enable continuous conversions
00192     ADC->CTRLB.bit.FREERUN = 1;
00193     while (ADC->STATUS.bit.SYNCBUSY)
00194     {
00195     }
00196
00197     // Configure input mux
00198     ADC->INPUTCTRL.bit.MUXPOS = g_APinDescription[pin].ulADCChannelNumber;
00199     while (ADC->STATUS.bit.SYNCBUSY)
00200     {
00201     }
00202
00203     // Enable the ADC
00204     ADC->CTRLA.bit.ENABLE = 1;
00205     while (ADC->STATUS.bit.SYNCBUSY)
00206     {
00207     }
00208
00209     // Start continuous conversions
00210     ADC->SWTRIG.bit.START = 1;
00211     while (ADC->STATUS.bit.SYNCBUSY)
00212     {
00213     }
00214
00215     // Enable the ADC interrupt
00216     NVIC_EnableIRQ(ADC_IRQn);
00217 }
00218
00219 void ArduinoLowPowerClass::detachAdcInterrupt()
00220 {
00221     // Disable the ADC interrupt
00222     NVIC_DisableIRQ(ADC_IRQn);
00223
00224     // Disable the ADC
00225     ADC->CTRLA.bit.ENABLE = 0;
00226     while (ADC->STATUS.bit.SYNCBUSY)
00227     {
00228     }
00229
00230     // Disable continuous conversions
00231     ADC->CTRLB.bit.FREERUN = 0;
00232     while (ADC->STATUS.bit.SYNCBUSY)
00233     {
00234     }
00235
00236     // Disable ADC in standby mode
00237     ADC->CTRLA.bit.RUNSTDBY = 1;
00238     while (ADC->STATUS.bit.SYNCBUSY)
00239     {
00240     }
00241
00242     // Disable window interrupt
00243     ADC->INTENCLR.bit.WINMON = 1;
00244     while (ADC->STATUS.bit.SYNCBUSY)
00245     {
00246     }
00247
00248     // Disable window mode
00249     ADC->WINCTRL.reg = ADC_WINCTRL_WINMODE_DISABLE;
00250     while (ADC->STATUS.bit.SYNCBUSY)
00251     {
00252     }
00253
00254     // Restore ADC prescaler
00255     ADC->CTRLB.bit.PRESCALER = ADC_CTRLB_PRESCALER_DIV512_Val;
00256     while (ADC->STATUS.bit.SYNCBUSY)
00257     {
00258     }
00259 }
```

```
00260 // Restore ADC clock
00261 while (GCLK->STATUS.bit.SYNCBUSY)
00262 {
00263 }
00264 GCLK->CLKCTRL.reg = GCLK_CLKCTRL_ID_ADC | GCLK_CLKCTRL_GEN_GCLK0 | GCLK_CLKCTRL_CLKEN;
00265 while (GCLK->STATUS.bit.SYNCBUSY)
00266 {
00267 }
00268
00269 adc_cb = nullptr;
00270 }
00271
00272 void ADC_Handler()
00273 {
00274 // Clear the interrupt flag
00275 ADC->INTFLAG.bit.WINMON = 1;
00276 LowPower.adc_cb();
00277 }
00278
00279 ArduinoLowPowerClass LowPower;
00280
00281 #endif // ARDUINO_ARCH_SAMD
```

## 15.83 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/batt\_ArduinoLowPower.h File Reference

```
#include <Arduino.h>
```

### Classes

- class [ArduinoLowPowerClass](#)

### Macros

- `#define RTC_ALARM_WAKEUP 0xFF`

### Typedefs

- using `irq_mode` = `uint32_t`
- `typedef void(* onOffFuncPtr) (bool)`

### Enumerations

- enum `wakeup_reason` { `OTHER_WAKEUP` = 0 , `GPIO_WAKEUP` = 1 , `NFC_WAKEUP` = 2 , `ANALOG_COMPARATOR_WAKEUP` = 3 }

### Variables

- `ArduinoLowPowerClass LowPower`

#### 15.83.1 Macro Definition Documentation

##### 15.83.1.1 RTC\_ALARM\_WAKEUP

```
#define RTC_ALARM_WAKEUP 0xFF
Definition at line 19 of file batt\_ArduinoLowPower.h.
```

## 15.83.2 Typedef Documentation

### 15.83.2.1 irq\_mode

```
using irq_mode = uint32_t
Definition at line 24 of file batt_ArduinoLowPower.h.
```

### 15.83.2.2 onOffFuncPtr

```
typedef void(* onOffFuncPtr) (bool)
Definition at line 28 of file batt_ArduinoLowPower.h.
```

## 15.83.3 Enumeration Type Documentation

### 15.83.3.1 wakeup\_reason

```
enum wakeup_reason
```

Enumerator

OTHER_WAKEUP	
GPIO_WAKEUP	
NFC_WAKEUP	
ANALOG_COMPARATOR_WAKEUP	

Definition at line 30 of file batt\_ArduinoLowPower.h.

## 15.83.4 Variable Documentation

### 15.83.4.1 LowPower

```
ArduinoLowPowerClass LowPower [extern]
```

## 15.84 batt\_ArduinoLowPower.h

Go to the documentation of this file.

```
00001 #ifndef _ARDUINO_LOW_POWER_H_
00002 #define _ARDUINO_LOW_POWER_H_
00003
00004 #include <Arduino.h>
00005
00006 #ifdef ARDUINO_ARCH_AVR
00007 #error The library is not compatible with AVR boards
00008 #endif
00009
00010 #ifdef ARDUINO_ARCH_SAMD
00011 #include "batt_RTCZero.h"
00012 #endif
00013
00014 #if defined(ARDUINO_SAMD_TIAN) || defined(ARDUINO_NRF52_PRIMO)
00015 // add here any board with companion chip which can be woken up
00016 #define BOARD_HAS_COMPANION_CHIP
00017 #endif
00018
00019 #define RTC_ALARM_WAKEUP 0xFF
00020
00021 #ifdef ARDUINO_API_VERSION
00022 using irq_mode = PinStatus;
```

```
00023 #else
00024     using irq_mode = uint32_t;
00025 #endif
00026
00027 //typedef void (*voidFuncPtr)( void ) ;
00028 typedef void (*onOffFuncPtr)(bool);
00029
00030 typedef enum
00031 {
00032     OTHER_WAKEUP = 0,
00033     GPIO_WAKEUP = 1,
00034     NFC_WAKEUP = 2,
00035     ANALOG_COMPARATOR_WAKEUP = 3
00036 } wakeup_reason;
00037
00038 #ifdef ARDUINO_ARCH_SAMD
00039 enum adc_interrupt
00040 {
00041     ADC_INT_BETWEEN,
00042     ADC_INT_OUTSIDE,
00043     ADC_INT_ABOVE_MIN,
00044     ADC_INT_BELOW_MAX,
00045 };
00046 #endif
00047
00048 class ArduinoLowPowerClass
00049 {
00050 public:
00051     void idle(void);
00052     void idle(uint32_t millis);
00053     void idle(int millis)
00054     {
00055         idle((uint32_t)millis);
00056     }
00057
00058     void sleep(void);
00059     void sleep(uint32_t millis);
00060     void sleep(int millis)
00061     {
00062         sleep((uint32_t)millis);
00063     }
00064
00065     void deepSleep(void);
00066     void deepSleep(uint32_t millis);
00067     void deepSleep(int millis)
00068     {
00069         deepSleep((uint32_t)millis);
00070     }
00071
00072     void attachInterruptWakeup(uint32_t pin, voidFuncPtr callback, irq_mode mode);
00073     void deatchRTCInterrupt(void);
00074 #ifdef BOARD_HAS_COMPANION_CHIP
00075     void companionLowPowerCallback(onOffFuncPtr callback)
00076     {
00077         companionSleepCB = callback;
00078     }
00079     void companionSleep()
00080     {
00081         companionSleepCB(true);
00082     }
00083     void companionWakeUp()
00084     {
00085         companionSleepCB(false);
00086     }
00087 #endif
00088
00089 #ifdef ARDUINO_ARCH_NRF52
00090     void enableWakeupFrom(wakeup_reason peripheral, uint32_t pin = 0xFF, uint32_t event = 0xFF,
00091     uint32_t option = 0xFF);
00092     wakeup_reason wakeupReason();
00093 #endif
00094 #ifdef ARDUINO_ARCH_SAMD
00095     void attachAdcInterrupt(uint32_t pin, voidFuncPtr callback, adc_interrupt mode, uint16_t lo,
00096     uint16_t hi);
00097     void detachAdcInterrupt();
00098 #endif
00099 private:
00100     void setAlarmIn(uint32_t millis);
00101 #ifdef ARDUINO_ARCH_SAMD
00102     RTCZero rtc;
00103     voidFuncPtr adc_cb;
00104     friend void ADC_Handler();
00105 #endif
00106 #ifdef BOARD_HAS_COMPANION_CHIP
00107     void (*companionSleepCB)(bool);
```

```
00108 #endif
00109 };
00110
00111 extern ArduinoLowPowerClass LowPower;
00112
00113 #endif
```

## 15.85 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/examples/Adc← Wakeup/AdcWakeup.ino File Reference

```
#include "ArduinoLowPower.h"
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)
- void [repetitionsIncrease\(\)](#)

### Variables

- volatile int [repetitions](#) = 1
- const int [pin](#) = A0
- const int [margin](#) = 10

#### 15.85.1 Function Documentation

##### 15.85.1.1 [loop\(\)](#)

```
void loop ()  
Definition at line 29 of file AdcWakeup.ino.
```

##### 15.85.1.2 [repetitionsIncrease\(\)](#)

```
void repetitionsIncrease ()  
Definition at line 56 of file AdcWakeup.ino.
```

##### 15.85.1.3 [setup\(\)](#)

```
void setup ()  
Definition at line 24 of file AdcWakeup.ino.
```

#### 15.85.2 Variable Documentation

##### 15.85.2.1 [margin](#)

```
const int margin = 10  
Definition at line 22 of file AdcWakeup.ino.
```

### 15.85.2.2 pin

```
const int pin = A0
```

Definition at line 20 of file [AdcWakeup.ino](#).

### 15.85.2.3 repetitions

```
volatile int repetitions = 1
```

Definition at line 17 of file [AdcWakeup.ino](#).

## 15.86 AdcWakeup.ino

[Go to the documentation of this file.](#)

```
00001 /*
00002   AdcWakeup
00003
00004   This sketch demonstrates the usage of the ADC to wakeup a chip in sleep mode.
00005   Sleep modes allow a significant drop in the power usage of a board while it does nothing waiting for
00006   an event to happen. Battery powered application can take advantage of these modes to enhance battery
00007   life significantly.
00008
00009   In this sketch, changing the voltage on pin A0 will wake up the board. You can test this by
00010   connecting a potentiometer between VCC, A0, and GND.
00011   Please note that, if the processor is sleeping, a new sketch can't be uploaded. To overcome this,
00012   manually reset the board (usually with a single or double tap to the RESET button)
00013
00014   This example code is in the public domain.
00015 */
00016
00017 #include "ArduinoLowPower.h"
00018
00019 // Blink sequence number
00020 // Declare it volatile since it's incremented inside an interrupt
00021 volatile int repetitions = 1;
00022
00023 const int pin = A0;
00024 // How sensitive to be to changes in voltage
00025 const int margin = 10;
00026
00027
00028
00029 void setup() {
00030   pinMode(LED_BUILTIN, OUTPUT);
00031   pinMode(pin, INPUT);
00032 }
00033
00034 void loop() {
00035   for (int i = 0; i < repetitions; i++) {
00036     digitalWrite(LED_BUILTIN, HIGH);
00037     delay(500);
00038     digitalWrite(LED_BUILTIN, LOW);
00039     delay(500);
00040   }
00041
00042   // Read the voltage at the ADC pin
00043   int value = analogRead(pin);
00044
00045   // Define a window around that value
00046   uint16_t lo = max(value - margin, 0);
00047   uint16_t hi = min(value + margin, UINT16_MAX);
00048
00049   // Attach an ADC interrupt on pin A0, calling repetitionsIncrease when the voltage is outside the
00050   // given range.
00051   // This should be called immediately before LowPower.sleep() because it reconfigures the ADC
00052   // internally.
00053   LowPower.attachAdcInterrupt(pin, repetitionsIncrease, ADC_INT_OUTSIDE, lo, hi);
00054
00055   // Triggers an infinite sleep (the device will be woken up only by the registered wakeup sources)
00056   // The power consumption of the chip will drop consistently
00057   LowPower.sleep();
00058
00059   // Detach the ADC interrupt. This should be called immediately after LowPower.sleep() because it
00060   // restores the ADC configuration after waking up.
00061   LowPower.detachAdcInterrupt();
00062 }
```

```

00059 // Remember to avoid calling delay() and long running functions since this functions executes in
00060 interrupt context
00061 repetitions++;

```

## 15.87 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/examples/External← Wakeup/ExternalWakeups.ino File Reference

```
#include "ArduinoLowPower.h"
```

### Functions

- void `setup()`
- void `loop()`
- void `repetitionsIncrease()`

### Variables

- volatile int `repetitions` = 1
- const int `pin` = 8

#### 15.87.1 Function Documentation

##### 15.87.1.1 `loop()`

```
void loop ()
```

Definition at line 37 of file [ExternalWakeups.ino](#).

##### 15.87.1.2 `repetitionsIncrease()`

```
void repetitionsIncrease ()
```

Definition at line 51 of file [ExternalWakeups.ino](#).

##### 15.87.1.3 `setup()`

```
void setup ()
```

Definition at line 22 of file [ExternalWakeups.ino](#).

#### 15.87.2 Variable Documentation

##### 15.87.2.1 `pin`

```
const int pin = 8
```

Definition at line 20 of file [ExternalWakeups.ino](#).

### 15.87.2.2 repetitions

```
volatile int repetitions = 1  
Definition at line 17 of file ExternalWakeup.ino.
```

## 15.88 ExternalWakeup.ino

Go to the documentation of this file.

```
00001 /*  
00002  ExternalWakeup  
00003  
00004  This sketch demonstrates the usage of External Interrupts (on pins) to wakeup a chip in sleep mode.  
00005  Sleep modes allow a significant drop in the power usage of a board while it does nothing waiting for  
00006  an event to happen. Battery powered application can take advantage of these modes to enhance battery  
00007  life significantly.  
00008  In this sketch, shorting pin 8 to a GND will wake up the board.  
00009  Please note that, if the processor is sleeping, a new sketch can't be uploaded. To overcome this,  
00010  manually reset the board (usually with a single or double tap to the RESET button)  
00011 */  
00012  
00013 #include "ArduinoLowPower.h"  
00014  
00015 // Blink sequence number  
00016 // Declare it volatile since it's incremented inside an interrupt  
00017 volatile int repetitions = 1;  
00018  
00019 // Pin used to trigger a wakeup  
00020 const int pin = 8;  
00021  
00022 void setup()  
00023 {  
00024     Serial.begin(9600);  
00025     // while (!Serial)  
00026     // {  
00027     //     /* code */  
00028     // }  
00029  
00030     pinMode(LED_BUILTIN, OUTPUT);  
00031     // Set pin 8 as INPUT_PULLUP to avoid spurious wakeup  
00032     pinMode(pin, INPUT_PULLUP);  
00033     // Attach a wakeup interrupt on pin 8, calling repetitionsIncrease when the device is woken up  
00034     LowPower.attachInterruptWakeup(pin, repetitionsIncrease, RISING);  
00035 }  
00036  
00037 void loop()  
00038 {  
00039     for (int i = 0; i < repetitions; i++)  
00040     {  
00041         digitalWrite(LED_BUILTIN, HIGH);  
00042         delay(500);  
00043         digitalWrite(LED_BUILTIN, LOW);  
00044         delay(500);  
00045     }  
00046     // Triggers an infinite sleep (the device will be woken up only by the registered wakeup sources)  
00047     // The power consumption of the chip will drop consistently  
00048     LowPower.sleep();  
00049 }  
00050  
00051 void repetitionsIncrease()  
00052 {  
00053     // This function will be called once on device wakeup  
00054     // You can do some little operations here (like changing variables which will be used in the loop)  
00055     // Remember to avoid calling delay() and long running functions since this functions executes in  
00056     // interrupt context  
00056     repetitions++;  
00057 }
```

## 15.89 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/examples/PrimoDeep← Sleep/PrimoDeepSleep.ino File Reference

```
#include "ArduinoLowPower.h"
```

### Functions

- void `StmEspPM` (bool sleep)
- void `setup` ()
- void `loop` ()
- void `doMyStuff` ()
- void `doMyStuffWithNFC` ()
- void `doOtherStuff` ()

### Variables

- const int `digitalPin` = 10
- const int `analogPin` = A0

#### 15.89.1 Function Documentation

##### 15.89.1.1 `doMyStuff()`

```
void doMyStuff ()
```

Definition at line [82](#) of file [PrimoDeepSleep.ino](#).

##### 15.89.1.2 `doMyStuffWithNFC()`

```
void doMyStuffWithNFC ()
```

Definition at line [86](#) of file [PrimoDeepSleep.ino](#).

##### 15.89.1.3 `doOtherStuff()`

```
void doOtherStuff ()
```

Definition at line [90](#) of file [PrimoDeepSleep.ino](#).

##### 15.89.1.4 `loop()`

```
void loop ()
```

Definition at line [80](#) of file [PrimoDeepSleep.ino](#).

##### 15.89.1.5 `setup()`

```
void setup ()
```

Definition at line [43](#) of file [PrimoDeepSleep.ino](#).

### 15.89.1.6 StmEspPM()

```
void StmEspPM (
    bool sleep )
```

Definition at line 34 of file [PrimoDeepSleep.ino](#).

## 15.89.2 Variable Documentation

### 15.89.2.1 analogPin

```
const int analogPin = A0
```

Definition at line 31 of file [PrimoDeepSleep.ino](#).

### 15.89.2.2 digitalPin

```
const int digitalPin = 10
```

Definition at line 28 of file [PrimoDeepSleep.ino](#).

## 15.90 PrimoDeepSleep.ino

[Go to the documentation of this file.](#)

```
00001 /*
00002   PrimoDeepSleep.ino
00003
00004   Written by Chiara Ruggeri (chiara@arduino.org)
00005
00006   This example for the Arduino Primo board shows how to use
00007   low power library to enter in power off mode and save power.
00008   This mode ensure the deepest power saving mode. If you need
00009   a faster response from the board use standby function instead.
00010
00011   Please note that once exited from the deepest sleep mode the
00012   board will reset (so setup will be run again).
00013
00014   The functions enableWakeupFrom set the peripheral that will wake up
00015   the board. By calling it more than once you can choose more than
00016   a wakeup source.
00017   The board will be reset when it wakes up from power off.
00018   You can use wakeUpCause() function to find out what signals woke up
00019   the board if you use more than one wakeUpBy.. function.
00020
00021   This example code is in the public domain.
00022 */
00023
00024 #include "ArduinoLowPower.h"
00025
00026
00027 // Pin used to wakeup the board
00028 const int digitalPin = 10;
00029
00030 // Pin used in Compatarot module to wake up the board
00031 const int analogPin = A0;
00032
00033
00034 void StmEspPM(bool sleep){
00035   // enable USER1_BUTTON to turn STM32 off and on when pressed.
00036   // note that when STM32 is off you cannot load any new sketch.
00037   pinMode(USER1_BUTTON, STM32_IT);
00038
00039   // turn ESP8266 off or on
00040   digitalWrite(GPIO_ESP_FW, sleep ? LOW : HIGH);
00041 }
00042
00043 void setup() {
00044   Serial.begin(9600);
00045   pinMode(LED_BUILTIN, OUTPUT);
00046   digitalWrite(LED_BUILTIN, HIGH);
00047   delay(500);
00048   digitalWrite(LED_BUILTIN, LOW);
00049   delay(500);
00050
00051   //look for what peripheral woke up the board
00052   //reason is 0 at the first execution
```

```

00053     wakeup_reason reason=LowPower.wakeupReason();
00054     if(reason==GPIO_WAKEUP) //GPIO caused the wake up
00055         doMyStuff();
00056     else
00057         if(reason==NFC_WAKEUP) //NFC caused the wake up
00058             doMyStuffWithNFC();
00059     else
00060         if(reason==ANALOG_COMPARATOR_WAKEUP) //Comparator caused the wake up
00061             doOtherStuff();
00062
00063     Serial.println("Hi all, I return to sleep");
00064
00065     LowPower.companionLowPowerCallback(StmEspPM);
00066     // Send sleep command to ESP and enable USER1_BUTTON to turn STM off
00067     LowPower.companionSleep();
00068
00069     //set digital pin 10 to wake up the board when LOW level is detected
00070     LowPower.enableWakeupFrom(GPIO_WAKEUP, digitalPin, LOW);
00071     //let the board be woken up by any NFC field
00072     LowPower.enableWakeupFrom(NFC_WAKEUP);
00073     //wake up the board when the voltage on pin A0 goes below the voltage on pin AREF
00074     LowPower.enableWakeupFrom(ANALOG_COMPARATOR_WAKEUP, analogPin, AREF, UP);
00075     //go in low power mode. Note that the board will reset once it is woken up
00076     LowPower.deepSleep();
00077 }
00078
00079
00080 void loop() {}
00081
00082 void doMyStuff(){
00083     //insert your code here
00084 }
00085
00086 void doMyStuffWithNFC(){
00087     //insert your code here
00088 }
00089
00090 void doOtherStuff(){
00091     //insert your code here
00092 }

```

## 15.91 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/examples/TianStandby/TianStandby.ino File Reference

```
#include "ArduinoLowPower.h"
```

### Macros

- #define MIPS\_PIN 32

### Functions

- void MipsPM (bool sleep)
- void setup ()
- void loop ()
- void onWakeup ()

#### 15.91.1 Macro Definition Documentation

##### 15.91.1.1 MIPS\_PIN

```
#define MIPS_PIN 32
```

Definition at line 18 of file [TianStandby.ino](#).

## 15.91.2 Function Documentation

### 15.91.2.1 loop()

```
void loop ()  
Definition at line 32 of file TianStandby.ino.
```

### 15.91.2.2 MipsPM()

```
void MipsPM (bool sleep)  
Definition at line 20 of file TianStandby.ino.
```

### 15.91.2.3 onWakeup()

```
void onWakeup ()  
Definition at line 44 of file TianStandby.ino.
```

### 15.91.2.4 setup()

```
void setup ()  
Definition at line 25 of file TianStandby.ino.
```

## 15.92 TianStandby.ino

[Go to the documentation of this file.](#)

```
00001 /*  
00002   TianStandby  
00003  
00004   This sketch demonstrates the usage of SAMD chip to furtherly reduce the power usage of Tian  
00005   board. This method can be applied to any board with companion chips which expose a method  
00006   (via direct pin interrupt or via a command) to enter and exit standby.  
00007   Sleep modes allow a significant drop in the power usage of a board while it does nothing waiting for  
an event to happen. Battery powered application can take advantage of these modes to enhance battery  
life significantly.  
00008  
00009   In this sketch, the internal RTC of SAMD chip will wake up the processor every 20 seconds.  
00010   Before going to sleep, the SAMD chip tells the MIPS CPU to standby too.  
00011   Please note that, if the processor is sleeping, a new sketch can't be uploaded. To overcome this,  
manually reset the board (usually with a single or double tap to the RESET button)  
00012  
00013   This example code is in the public domain.  
00014 */  
00015  
00016 #include "ArduinoLowPower.h"  
00017  
00018 #define MIPS_PIN 32  
00019  
00020 void MipsPM(bool sleep) {  
00021   pinMode(MIPS_PIN, OUTPUT);  
00022   digitalWrite(MIPS_PIN, sleep ? LOW: HIGH);  
00023 }  
00024  
00025 void setup() {  
00026   pinMode(LED_BUILTIN, OUTPUT);  
00027   LowPower.companionLowPowerCallback(MipsPM);  
00028   // Uncomment this function if you wish to attach function dummy when RTC wakes up the chip  
00029   LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, onWakeup, CHANGE);  
00030 }  
00031  
00032 void loop() {  
00033   digitalWrite(LED_BUILTIN, HIGH);  
00034   delay(500);  
00035   digitalWrite(LED_BUILTIN, LOW);  
00036   delay(500);  
00037   // Triggers a 2000 ms sleep (the device will be woken up only by the registered wakeup sources and  
// by internal RTC)  
00038   // The power consumption of the chip will drop consistently
```

```

00039 // Send sleep command to MIPS CPU and then go to sleep
00040 LowPower.companionSleep();
00041 LowPower.sleep(20000);
00042 }
00043
00044 void onWakeup() {
00045 // This function will be called once on device wakeup
00046 // You can do some little operations here (like changing variables which will be used in the loop)
00047 // Remember to avoid calling delay() and long running functions since this functions executes in
00048 // interrupt context
00049 // Wakeup the companion chip, too
00050 LowPower.companionWakeup();
00051 }

```

## 15.93 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0009-LOWPOWER/examples/TimedWakeups/TimedWakeups.ino File Reference

```
#include "ArduinoLowPower.h"
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)
- void [dummy\(\)](#)

#### 15.93.1 Function Documentation

##### 15.93.1.1 [dummy\(\)](#)

```
void dummy()
```

Definition at line 31 of file [TimedWakeups.ino](#).

##### 15.93.1.2 [loop\(\)](#)

```
void loop()
```

Definition at line 21 of file [TimedWakeups.ino](#).

##### 15.93.1.3 [setup\(\)](#)

```
void setup()
```

Definition at line 15 of file [TimedWakeups.ino](#).

## 15.94 TimedWakeups.ino

[Go to the documentation of this file.](#)

```

00001 /*
00002  * TimedWakeups
00003
00004  * This sketch demonstrates the usage of Internal Interrupts to wakeup a chip in sleep mode.
00005  * Sleep modes allow a significant drop in the power usage of a board while it does nothing waiting for
00006  * an event to happen. Battery powered application can take advantage of these modes to enhance battery
00007  * life significantly.
00008  * In this sketch, the internal RTC will wake up the processor every 2 seconds.
00009  * Please note that, if the processor is sleeping, a new sketch can't be uploaded. To overcome this,
00010  * manually reset the board (usually with a single or double tap to the RESET button)

```

```

00009
00010  This example code is in the public domain.
00011 */
00012
00013 #include "ArduinoLowPower.h"
00014
00015 void setup() {
00016     pinMode(LED_BUILTIN, OUTPUT);
00017     // Uncomment this function if you wish to attach function dummy when RTC wakes up the chip
00018     // LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, dummy, CHANGE);
00019 }
00020
00021 void loop() {
00022     digitalWrite(LED_BUILTIN, HIGH);
00023     delay(500);
00024     digitalWrite(LED_BUILTIN, LOW);
00025     delay(500);
00026     // Triggers a 2000 ms sleep (the device will be woken up only by the registered wakeup sources and
00027     // by internal RTC)
00028     // The power consumption of the chip will drop consistently
00029     LowPower.sleep(2000);
00030 }
00031 void dummy() {
00032     // This function will be called once on device wakeup
00033     // You can do some little operations here (like changing variables which will be used in the loop)
00034     // Remember to avoid calling delay() and long running functions since this functions executes in
00035     interrupt context
}

```

## 15.95 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0010-LOGGING/diagnostic.h File Reference

```
#include <batt_FlashStorage.h>
#include <batt_SlowSoftI2CMaster.h>
#include "EepromBitBang.h"
#include "Eeprom_LUT.h"
```

### Classes

- struct [EEPROM\\_EMULATION](#)
- struct [Element](#)

### Macros

- [#define EEPROM\\_EMULATION\\_SIZE 127](#)

### Functions

- [FlashStorage](#) (eeprom\_flash, [EEPROM\\_EMULATION](#))  
*Puesta a 0 de las primeras 4095 posiciones de la EEPROM de la Bateria.*
- void [ResetBateriaEeprom](#) ()  
*Inicializacion del valor de los elementos de diagnostico. Cada elemento posee el valor, el nombre y la categoria:*
- void [LogDiagnosticData](#) (int16\_t data, int16\_t address)  
*Actualizar el valor en Ram de una variable de adiagnostico.*
- uint16\_t [ReadDiagnosticData](#) (int16\_t address)  
*Lectura del valor de un elemento de diagnostico.*
- void [IncrementDiagnosticData](#) (int16\_t inc, int16\_t address)  
*Incrementar el valor de una variable de diagnostico en un valor.*
- void [PrintDiagnosticData](#) ()

*Muestreo por el puerto serie de las variable de diagnostico con categoria de Serial Port 'S'.*

- void [SaveEepromChasis \(\)](#)

*Guardado en la EEPROM del chasis de las variables de diagnostico con categoria de Memoria 'C'.*

- bool [isValid \(\)](#)

*Devuelve true si la memoria RAM contiene datos y false si la memoria RAM esta vacia.*

- void [Stats \(uint16\\_t volts, uint16\\_t wats\)](#)

*Incremento de las estadisticas del valor de voltahe y potencia usado.*

- void [UpdateEepromBatory \(\)](#)

*Actualizacion de la memoria EEPROM de la Bateria.*

- void [PrintStats \(\)](#)

*Muestreo por el puerto serie de los valores acumulados de las estadisticas de potencia y voltage.*

- void [PrintStaticData \(\)](#)

*Impresion por el puerto serie de los datos estaticos.*

## Variables

- const uint16\_t [MIN\\_VOLTAGE](#) = 5000
- const uint16\_t [MAX\\_VOLTAGE](#) = 160000
- const int16\_t [MIN\\_WATS](#) = 500
- const int16\_t [MAX\\_WATS](#) = 5000
- Element elements [[C\\_NUM\\_ELEMENTS](#)]
- EEPROM\_EMULATION \_eeprom\_RAM
- bool [\\_dirty](#) = false
- int [eeprom\\_index](#) = 0
- uint16\_t [voltage\\_values](#) [111]
- uint16\_t [power\\_values](#) [46]
- uint8\_t [init\\_flag](#) = 0

### 15.95.1 Detailed Description

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

#### Version

1

#### Date

2021-01-14

#### Copyright

Copyright (c) 2021

Definition in file [diagnostic.h](#).

### 15.95.2 Macro Definition Documentation

#### 15.95.2.1 EEPROM\_EMULATION\_SIZE

```
#define EEPROM_EMULATION_SIZE 127
```

Definition at line 13 of file [diagnostic.h](#).

### 15.95.3 Function Documentation

#### 15.95.3.1 FlashStorage()

```
FlashStorage (
    eeprom_flash ,
    EEPROM_EMULATION )
```

#### 15.95.3.2 IncrementDiagnosticData()

```
void IncrementDiagnosticData (
    int16_t inc,
    int16_t address )
```

Incrementar el valor de una variable de diagnostico en un valor.

##### Parameters

<i>inc</i>	Valor de incremento
<i>address</i>	Direccion en la memoria RAM de la variable.

Definition at line [232](#) of file [diagnostic.h](#).

#### 15.95.3.3 Init\_diagnostic\_elements()

```
void Init_diagnostic_elements ( )
```

Inicializacion del valor de los elementos de diagnostico. Cada elemento posee el valor, el nombre y la categoria:

- Serial Port (X o S). Con X no se muestra con S si.
- Guardado en EEPROM (X,B,C). X no se guarda, B se guarda en la Bateria y C se guarda en el Chasis.

Definition at line [83](#) of file [diagnostic.h](#).

#### 15.95.3.4 isValid()

```
bool isValid ( )
```

Devuelve true si la memoria RAM contiene datos y false si la memoria RAM esta vacia.

##### Returns

true

false

Definition at line [288](#) of file [diagnostic.h](#).

#### 15.95.3.5 LogDiagnosticData()

```
void LogDiagnosticData (
    int16_t data,
    int16_t address )
```

Actualizar el valor en Ram de una variable de adiagnostico.

##### Parameters

<i>data</i>	Valor del dato
<i>address</i>	Direccion en la memoria RAM de la variable

Definition at line 207 of file [diagnostic.h](#).

#### 15.95.3.6 PrintDiagnosticData()

```
void PrintDiagnosticData ( )
```

Muestreo por el puerto serie de las variable de diagnostico con categoria de Serial Port 'S'.

Definition at line 243 of file [diagnostic.h](#).

#### 15.95.3.7 PrintStaticData()

```
void PrintStaticData ( )
```

Impresion por el puerto serie de los datos estaticos.

Definition at line 404 of file [diagnostic.h](#).

#### 15.95.3.8 PrintStats()

```
void PrintStats ( )
```

Muestreo por el puerto serie de los valores acumulados de las estadisticas de potencia y voltage.

Definition at line 386 of file [diagnostic.h](#).

#### 15.95.3.9 ReadDiagnosticData()

```
uint16_t ReadDiagnosticData ( int16_t address )
```

Lectura del valor de un elemento de diagnostico.

##### Parameters

address	<input type="text"/>
---------	----------------------

Definition at line 222 of file [diagnostic.h](#).

#### 15.95.3.10 ResetBateriaEeprom()

```
void ResetBateriaEeprom ( )
```

Puesta a 0 de las primeras 4095 posiciones de la EEPROM de la Bateria.

##### FUNCTIONS

Definition at line 69 of file [diagnostic.h](#).

#### 15.95.3.11 SaveEepromChasis()

```
void SaveEepromChasis ( )
```

Guardado en la EEPROM del chasis de las variables de diagnostico con categoria de Memoria 'C'.

Definition at line 262 of file [diagnostic.h](#).

#### 15.95.3.12 Stats()

```
void Stats ( uint16_t volts, uint16_t wats )
```

Incremento de las estadisticas del valor de voltahe y potencia usado.

Parameters

<i>volts</i>	
<i>wats</i>	

Definition at line 299 of file [diagnostic.h](#).

### 15.95.3.13 UpdateEepromBatory()

`void UpdateEepromBatory ( )`

Actualizacion de la memoria EEPROM de la Bateria.

Definition at line 315 of file [diagnostic.h](#).

## 15.95.4 Variable Documentation

### 15.95.4.1 \_dirty

`bool _dirty = false`

Definition at line 52 of file [diagnostic.h](#).

### 15.95.4.2 \_eeprom\_RAM

`EEPROM_EMULATION _eeprom_RAM`

Definition at line 50 of file [diagnostic.h](#).

### 15.95.4.3 eeprom\_index

`int eeprom_index = 0`

Definition at line 53 of file [diagnostic.h](#).

### 15.95.4.4 elements

`Element elements[C_NUM_ELEMENTS]`

Definition at line 49 of file [diagnostic.h](#).

### 15.95.4.5 init\_flag

`uint8_t init_flag = 0`

Definition at line 58 of file [diagnostic.h](#).

### 15.95.4.6 MAX\_VOLTAGE

`const uint16_t MAX_VOLTAGE = 160000`

Definition at line 29 of file [diagnostic.h](#).

### 15.95.4.7 MAX\_WATS

`const int16_t MAX_WATS = 5000`

Definition at line 31 of file [diagnostic.h](#).

#### 15.95.4.8 MIN\_VOLTAGE

```
const uint16_t MIN_VOLTAGE = 5000
CONSTANTS
```

Definition at line 28 of file [diagnostic.h](#).

#### 15.95.4.9 MIN\_WATS

```
const int16_t MIN_WATS = 500
Definition at line 30 of file diagnostic.h.
```

#### 15.95.4.10 power\_values

```
uint16_t power_values[46]
Definition at line 56 of file diagnostic.h.
```

#### 15.95.4.11 voltage\_values

```
uint16_t voltage_values[111]
Definition at line 55 of file diagnostic.h.
```

### 15.96 diagnostic.h

[Go to the documentation of this file.](#)

```
00001
00012 #ifndef EEPROM_EMULATION_SIZE
00013 #define EEPROM_EMULATION_SIZE 127
00014 #endif
00015
00016 #include <batt_FlashStorage.h>
00017 #include <batt_SlowSoftI2CMaster.h>
00018 #include "EepromBitBang.h"
00019 #include "Eeprom_LUT.h"
00020 /*extern "C"
00021 {
00022 #include <SEGGER_RTT.h>
00023 }
00024 */
00028 const uint16_t MIN_VOLTAGE = 5000;
00029 const uint16_t MAX_VOLTAGE = 160000;
00030 const int16_t MIN_WATS = 500;
00031 const int16_t MAX_WATS = 5000;
00032
00036 typedef struct
00037 {
00038     int16_t data[EEPROM_EMULATION_SIZE];
00039     boolean valid;
00040 } EEPROM_EMULATION;
00041
00042 typedef struct
00043 {
00044     int16_t value = 0;
00045     char category[2] = {'X', 'X'}; // category[0] = 'S' (Serial Port); category[1] = 'B' Baterie
        Eeprom, 'C' Chasis Eeprom
00046     String name = "NoName";
00047 } Element;
00048
00049 Element elements[C_NUM_ELEMENTS];
00050 EEPROM_EMULATION _eeprom_RAM;
00051
00052 bool _dirty = false;
00053 int eeprom_index = 0;
00054
00055 uint16_t voltage_values[111]; // 5000V - 16000 step of 100mV = 111 positions
00056 uint16_t power_values[46]; // 500mW - 5000mW step of 100mW = 46 postions.
00057
00058 uint8_t init_flag = 0;
00059
00060 FlashStorage(eeprom_flash, EEPROM_EMULATION);
00061
00069 void ResetBateriaEeprom()
```

```

00070 {
00071     for (uint32_t i = 0; i < 0x3FF; i++)
00072     {
00073         writeEEPROM(i, 0);
00074     }
00075 }
00083 void Init_diagnostic_elements()
00084 {
00085     // Read EEPROM memory.
00086     _eprom_RAM = eeprom_flash.read();
00087     readEEPROM(POS_EEPROM_INIT, &init_flag);
00088     if (init_flag != C_INIT_STATE)
00089     {
00090         ResetBateriaEeprom();
00091         writeEEPROM(POS_EEPROM_INIT, C_INIT_STATE);
00092         WriteWordEEPROM(POS_EEPROM_MODEL, 1);
00093     }
00094
00095     elements[C_HACK_START_SOUND].category[0] = {'X'};
00096     elements[C_HACK_START_SOUND].category[1] = {'C'};
00097     elements[C_HACK_START_SOUND].name = "HACK_START_SOUND";
00098
00099     elements[C_HACK_END_SOUND].category[0] = {'S'};
00100    elements[C_HACK_END_SOUND].category[1] = {'C'};
00101    elements[C_HACK_END_SOUND].name = "HACK_END_SOUND";
00102
00103    elements[C_HACK_CHARGE_SOUND].category[0] = {'S'};
00104    elements[C_HACK_CHARGE_SOUND].category[1] = {'C'};
00105    elements[C_HACK_CHARGE_SOUND].name = "HACK_CHARGE_SOUND";
00106
00107    elements[C_HACK_DEATH_BATTERY_SOUND].category[0] = {'X'};
00108    elements[C_HACK_DEATH_BATTERY_SOUND].category[1] = {'C'};
00109    elements[C_HACK_DEATH_BATTERY_SOUND].name = "HACK_DEATH_BATTERY_SOUND";
00110
00111    elements[C_HACK_FULL_CHARGE_SOUND].category[0] = {'X'};
00112    elements[C_HACK_FULL_CHARGE_SOUND].category[1] = {'C'};
00113    elements[C_HACK_FULL_CHARGE_SOUND].name = "HACK_FULL_CHARGE_SOUND";
00114
00115    elements[C_HACK_START_DISPLAY].category[0] = {'X'};
00116    elements[C_HACK_START_DISPLAY].category[1] = {'C'};
00117    elements[C_HACK_START_DISPLAY].name = "HACK_START_DISPLAY";
00118
00119    elements[C_HACK_END_DISPLAY].category[0] = {'X'};
00120    elements[C_HACK_END_DISPLAY].category[1] = {'C'};
00121    elements[C_HACK_END_DISPLAY].name = "HACK_END_DISPLAY";
00122
00123    elements[C_SERIAL_NUMBER].category[0] = {'X'};
00124    elements[C_SERIAL_NUMBER].category[1] = {'B'};
00125    elements[C_SERIAL_NUMBER].name = "SERIAL NUMBER";
00126    elements[C_SERIAL_NUMBER].value = ReadWordEEPROM(POS_EEPROM_SERIAL_NUMBER);
00127
00128    elements[C_NUMBER_CYCLES].category[0] = {'X'};
00129    elements[C_NUMBER_CYCLES].category[1] = {'B'};
00130    elements[C_NUMBER_CYCLES].name = "NUMBER CYCLES";
00131    elements[C_NUMBER_CYCLES].value = ReadWordEEPROM(POS_EEPROM_NUMBER_CYCLES);
00132
00133    elements[C_WORK_TIME].category[0] = {'X'};
00134    elements[C_WORK_TIME].category[1] = {'B'};
00135    elements[C_WORK_TIME].name = "Work Time";
00136    elements[C_WORK_TIME].value = ReadWordEEPROM(POS_EEPROM_WORK_TIME);
00137
00138    elements[C_POWER_ERROR].category[0] = {'X'};
00139    elements[C_POWER_ERROR].category[1] = {'B'};
00140    elements[C_POWER_ERROR].name = "Power Error";
00141    elements[C_POWER_ERROR].value = ReadWordEEPROM(POS_EEPROM_POWER_ERROR);
00142
00143    elements[C_CONSUMPTION_ERROR].category[0] = {'X'};
00144    elements[C_CONSUMPTION_ERROR].category[1] = {'B'};
00145    elements[C_CONSUMPTION_ERROR].name = "Consumption Error";
00146    elements[C_CONSUMPTION_ERROR].value = ReadWordEEPROM(POS_EEPROM_CONSUMPTION_ERROR);
00147
00148    elements[C_VOLTAGE_ERROR].category[0] = {'X'};
00149    elements[C_VOLTAGE_ERROR].category[1] = {'B'};
00150    elements[C_VOLTAGE_ERROR].name = "Voltage Error";
00151    elements[C_VOLTAGE_ERROR].value = ReadWordEEPROM(POS_EEPROM_VOLTAGE_ERROR);
00152
00153    elements[C_INSTANT_VOLTAGE].category[0] = {'S'};
00154    elements[C_INSTANT_VOLTAGE].category[1] = {'X'};
00155    elements[C_INSTANT_VOLTAGE].name = "Instant Voltage";
00156    elements[C_INSTANT_VOLTAGE].value = 0;
00157
00158    elements[C_INSTANT_POWER].category[0] = {'S'};
00159    elements[C_INSTANT_POWER].category[1] = {'X'};
00160    elements[C_INSTANT_POWER].name = "Instant Power";
00161    elements[C_INSTANT_POWER].value = 0;
00162
00163    elements[C_INSTANT_CURRENT].category[0] = {'S'};

```

```

00164     elements[C_INSTANT_CURRENT].category[1] = {'X'};
00165     elements[C_INSTANT_CURRENT].name = "Instant Power";
00166     elements[C_INSTANT_CURRENT].value = 0;
00167
00168     elements[C_THEORY_VOLTAGE].category[0] = {'X'};
00169     elements[C_THEORY_VOLTAGE].category[1] = {'C'};
00170     elements[C_THEORY_VOLTAGE].name = "Voltage programed";
00171
00172     elements[C_MODEL].category[0] = {'X'};
00173     elements[C_MODEL].category[1] = {'C'};
00174     elements[C_MODEL].name = "Model";
00175     elements[C_MODEL].value = ReadWordEEPROM(POS_EEPROM_MODEL);
00176
00177     elements[C_VOLTAGE_INPUT_ERROR].category[0] = {'X'};
00178     elements[C_VOLTAGE_INPUT_ERROR].category[1] = {'C'};
00179     elements[C_VOLTAGE_INPUT_ERROR].name = "Model";
00180     elements[C_VOLTAGE_INPUT_ERROR].value = ReadWordEEPROM(POS_EEPROM_VOLTAGE_INPUT_ERROR);
00181
00182     elements[C_CAPACITY].category[0] = {'S'};
00183     elements[C_CAPACITY].category[1] = {'X'};
00184     elements[C_CAPACITY].name = "Capacity";
00185
00186     ReadArrayEEPROM(POS_EEPROM_VOLTS_ARRAY, &voltage_values[0], (sizeof(voltage_values) /
00187     sizeof(voltage_values[0])));
00188     ReadArrayEEPROM(POS_EEPROM_POWER_ARRAY, &power_values[0], (sizeof(power_values) /
00189     sizeof(power_values[0])));
00190
00191     eeprom_index = 0;
00192     for (int i = 0; i < C_NUM_ELEMENTS; i++)
00193     {
00194         if ((elements[i].category[1] == 'C'))
00195         {
00196             elements[i].value = _eprom_RAM.data[eeprom_index];
00197             eeprom_index++;
00198         }
00199     }
00200
00201 void LogDiagnosticData(int16_t data, int16_t address)
00202 {
00203     address = constrain(address, 0, C_NUM_ELEMENTS);
00204     if (elements[address].value != data)
00205     {
00206         _dirty = true;
00207         elements[address].value = data;
00208     }
00209 }
00210
00211 uint16_t ReadDiagnosticData(int16_t address)
00212 {
00213     return elements[address].value;
00214 }
00215
00216 void IncrementDiagnosticData(int16_t inc, int16_t address)
00217 {
00218     elements[address].value = elements[address].value + inc;
00219     constrain(elements[address].value, 0, 65535);
00220     _dirty = true;
00221 }
00222
00223 void PrintDiagnosticData()
00224 {
00225     for (int16_t i = 0; i < C_NUM_ELEMENTS; i++)
00226     {
00227         if (elements[i].category[0] == 'S')
00228         {
00229             //SEGGER_RTT_printf(0, "%s%d", RTT_CTRL_TEXT_WHITE, elements[i].value);
00230             Serial.print(elements[i].value);
00231             Serial.print(";");
00232         }
00233     }
00234     //SEGGER_RTT_printf(0, "\n");
00235     Serial.println();
00236 }
00237
00238 void SaveEepromChasis()
00239 {
00240     if (_dirty == true)
00241     {
00242         eeprom_index = 0;
00243         for (int16_t i = 0; i < C_NUM_ELEMENTS; i++)
00244         {
00245             if ((elements[i].category[1] == 'C'))
00246             {
00247                 _eprom_RAM.data[eeprom_index] = elements[i].value;
00248                 eeprom_index++;
00249                 eeprom_index = constrain(eeprom_index, 0, EEPROM_EMULATION_SIZE);
00250             }
00251         }
00252     }
00253 }
```

```

00275         }
00276         _eeprom_RAM.valid = true;
00277         eeprom_flash.write(_eeprom_RAM);
00278     _dirty = false;
00279 }
00280 }
00281
00288 bool isValid()
00289 {
00290     return _eeprom_RAM.valid;
00291 }
00292
00299 void Stats(uint16_t volts, uint16_t wats)
00300 {
00301     int16_t voltage_pos = 0, wats_pos = 0;
00302     voltage_pos = (volts - MIN_VOLTAGE) / 100;
00303     voltage_values[voltage_pos]++;
00304
00305     wats_pos = (wats - MIN_WATS) / 100;
00306     power_values[wats_pos]++;
00307 }
00308
00310
00315 void UpdateEepromBatory()
00316 {
00317
00318     uint16_t scrap = 0;
00319     uint16_t scrap1_array[111];
00320     uint16_t scrap2_array[46];
00321
00322     scrap = ReadWordEEPROM(POS_EEPROM_NUMBER_CYCLES);
00323     if (elements[C_NUMBER_CYCLES].value != scrap)
00324     {
00325         WriteWordEEPROM(POS_EEPROM_NUMBER_CYCLES, elements[C_NUMBER_CYCLES].value);
00326     }
00327
00328     scrap = ReadWordEEPROM(POS_EEPROM_SERIAL_NUMBER);
00329     if (elements[C_SERIAL_NUMBER].value != scrap)
00330     {
00331         WriteWordEEPROM(POS_EEPROM_SERIAL_NUMBER, elements[C_SERIAL_NUMBER].value);
00332     }
00333
00334     scrap = ReadWordEEPROM(POS_EEPROM_WORK_TIME);
00335     if (elements[C_WORK_TIME].value != scrap)
00336     {
00337         WriteWordEEPROM(POS_EEPROM_WORK_TIME, elements[C_WORK_TIME].value);
00338     }
00339
00340     scrap = ReadWordEEPROM(POS_EEPROM_CONSUMPTION_ERROR);
00341     if (elements[C_CONSUMPTION_ERROR].value != scrap)
00342     {
00343         WriteWordEEPROM(POS_EEPROM_CONSUMPTION_ERROR, elements[C_CONSUMPTION_ERROR].value);
00344     }
00345
00346     scrap = ReadWordEEPROM(POS_EEPROM_POWER_ERROR);
00347     if (elements[C_POWER_ERROR].value != scrap)
00348     {
00349         WriteWordEEPROM(POS_EEPROM_POWER_ERROR, elements[C_POWER_ERROR].value);
00350     }
00351
00352     scrap = ReadWordEEPROM(POS_EEPROM_VOLTAGE_ERROR);
00353     if (elements[C_VOLTAGE_ERROR].value != scrap)
00354     {
00355         WriteWordEEPROM(POS_EEPROM_VOLTAGE_ERROR, elements[C_VOLTAGE_ERROR].value);
00356     }
00357
00358     scrap = ReadWordEEPROM(POS_EEPROM_VOLTAGE_INPUT_ERROR);
00359     if (elements[C_VOLTAGE_INPUT_ERROR].value != scrap)
00360     {
00361         WriteWordEEPROM(POS_EEPROM_VOLTAGE_INPUT_ERROR, elements[C_VOLTAGE_INPUT_ERROR].value);
00362     }
00363
00364     ReadArrayEEPROM(POS_EEPROM_VOLTS_ARRAY, &scrap1_array[0], (sizeof(scrap1_array) /
00365     sizeof(scrap1_array[0])));
00366     for (int i = 0; i < (sizeof(voltage_values) / sizeof(voltage_values[0])); i++)
00367     {
00368         if (scrap1_array[i] != voltage_values[i])
00369         {
00370             WriteWordEEPROM(POS_EEPROM_VOLTS_ARRAY + i * 2, voltage_values[i]);
00371         }
00372     }
00373
00374     ReadArrayEEPROM(POS_EEPROM_POWER_ARRAY, &scrap2_array[0], (sizeof(scrap2_array) /
00375     sizeof(scrap2_array[0])));
00376     for (int i = 0; i < (sizeof(power_values) / sizeof(power_values[0])); i++)
00377     {

```

```

00376     if (scrap2_array[i] != power_values[i])
00377     {
00378         WriteWordEEPROM(POS_EEPROM_POWER_ARRAY + i * 2, power_values[i]);
00379     }
00380 }
00381 }
00382 void PrintStats()
00383 {
00384     for (int i = 0; i < sizeof(voltage_values) / sizeof(voltage_values[0]); i++)
00385     {
00386         Serial.print(voltage_values[i]);
00387         Serial.print(";");
00388     }
00389     for (int i = 0; i < sizeof(power_values) / sizeof(power_values[0]); i++)
00390     {
00391         Serial.print(power_values[i]);
00392         Serial.print(";");
00393     }
00394 }
00395 }
00396 }
00397 }
00398 }
00399
00400 void PrintStaticData()
00401 {
00402     Serial.print("\t");
00403     Serial.print(elements[C_MODEL].value);
00404     Serial.print(";");
00405     Serial.print(elements[C_SERIAL_NUMBER].value);
00406     Serial.print(";");
00407     Serial.print(elements[C_CONSUMPTION_ERROR].value);
00408     Serial.print(";");
00409     Serial.print(elements[C_POWER_ERROR].value);
00410     Serial.print(";");
00411     Serial.print(elements[C_VOLTAGE_ERROR].value);
00412     Serial.print(";");
00413     Serial.print(elements[C_NUMBER_CYCLES].value);
00414     Serial.print(";");
00415     Serial.print(elements[C_WORK_TIME].value);
00416     Serial.print(";");
00417     Serial.print(elements[C_VOLTAGE_INPUT_ERROR].value);
00418     Serial.print(";");
00419     PrintStats();
00420     Serial.print("\n");
00421
00422 //SEGGER_RTT_printf(0, "\n");
00423 }
```

## 15.97 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0010-LOGGING/Eeprom\_LUT.h File Reference

Look Up Table de direcciones de memoria de la eeprom de la bateria.

### Variables

- const uint8\_t **C\_INIT\_STATE** = 0xA5
- const int16\_t **C\_HACK\_START\_SOUND** = 0  
*RAM LUT EEPROM.*
- const int16\_t **C\_HACK\_END\_SOUND** = 1
- const int16\_t **C\_HACK\_CHARGE\_SOUND** = 2
- const int16\_t **C\_HACK\_DEATH\_BATTERY\_SOUND** = 3
- const int16\_t **C\_HACK\_FULL\_CHARGE\_SOUND** = 4
- const int16\_t **C\_HACK\_START\_DISPLAY** = 5
- const int16\_t **C\_SERIAL\_NUMBER** = 6
- const int16\_t **C\_NUMBER\_CYCLES** = 7
- const int16\_t **C\_WORK\_TIME** = 8
- const int16\_t **C\_POWER\_ERROR** = 9
- const int16\_t **C\_CONSUMPTION\_ERROR** = 10
- const int16\_t **C\_VOLTAGE\_ERROR** = 11
- const int16\_t **C\_INSTANT\_VOLTAGE** = 12

- const int16\_t [C\\_INSTANT\\_CURRENT](#) = 13
- const int16\_t [C\\_INSTANT\\_POWER](#) = 14
- const int16\_t [C THEORY\\_VOLTAGE](#) = 15
- const int16\_t [C\\_MODEL](#) = 16
- const int16\_t [C\\_CAPACITY](#) = 17
- const int16\_t [C\\_VOLTAGE\\_INPUT\\_ERROR](#) = 18
- const int16\_t [C\\_HACK\\_END\\_DISPLAY](#) = 19
- const int16\_t [C\\_NUM\\_ELEMENTS](#) = 20
- const int16\_t [POS\\_EEPROM\\_INIT](#) = 0x0000
  - Baterie LUT EEPROM.*
- const int16\_t [POS\\_EEPROM\\_MODEL](#) = 0x0001
- const int16\_t [POS\\_EEPROM\\_SERIAL\\_NUMBER](#) = 0x0003
- const int16\_t [POS\\_EEPROM\\_NUMBER\\_CYCLES](#) = 0x0005
- const int16\_t [POS\\_EEPROM\\_VOLTS\\_ARRAY](#) = 0x0007
- const int16\_t [POS\\_EEPROM\\_POWER\\_ARRAY](#) = 0x00E5
- const int16\_t [POS\\_EEPROM\\_WORK\\_TIME](#) = 0x0141
- const int16\_t [POS\\_EEPROM\\_POWER\\_ERROR](#) = 0x0143
- const int16\_t [POS\\_EEPROM\\_CONSUMPTION\\_ERROR](#) = 0x0145
- const int16\_t [POS\\_EEPROM\\_VOLTAGE\\_ERROR](#) = 0x0147
- const int16\_t [POS\\_EEPROM\\_VOLTAGE\\_INPUT\\_ERROR](#) = 0x0149

### 15.97.1 Detailed Description

Look Up Table de direcciones de memoria de la eeprom de la bateria.

Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

Version

1

Date

2021-03-22

Copyright

Copyright (c) 2021

Definition in file [Eeprom\\_LUT.h](#).

### 15.97.2 Variable Documentation

#### 15.97.2.1 C\_CAPACITY

const int16\_t C\_CAPACITY = 17

Definition at line 34 of file [Eeprom\\_LUT.h](#).

#### 15.97.2.2 C\_CONSUMPTION\_ERROR

const int16\_t C\_CONSUMPTION\_ERROR = 10

Definition at line 27 of file [Eeprom\\_LUT.h](#).

### 15.97.2.3 C\_HACK\_CHARGE\_SOUND

```
const int16_t C_HACK_CHARGE_SOUND = 2
Definition at line 19 of file Eeprom\_LUT.h.
```

### 15.97.2.4 C\_HACK\_DEATH\_BATTERY\_SOUND

```
const int16_t C_HACK_DEATH_BATTERY_SOUND = 3
Definition at line 20 of file Eeprom\_LUT.h.
```

### 15.97.2.5 C\_HACK\_END\_DISPLAY

```
const int16_t C_HACK_END_DISPLAY = 19
Definition at line 36 of file Eeprom\_LUT.h.
```

### 15.97.2.6 C\_HACK\_END\_SOUND

```
const int16_t C_HACK_END_SOUND = 1
Definition at line 18 of file Eeprom\_LUT.h.
```

### 15.97.2.7 C\_HACK\_FULL\_CHARGE\_SOUND

```
const int16_t C_HACK_FULL_CHARGE_SOUND = 4
Definition at line 21 of file Eeprom\_LUT.h.
```

### 15.97.2.8 C\_HACK\_START\_DISPLAY

```
const int16_t C_HACK_START_DISPLAY = 5
Definition at line 22 of file Eeprom\_LUT.h.
```

### 15.97.2.9 C\_HACK\_START\_SOUND

```
const int16_t C_HACK_START_SOUND = 0
RAM LUT EEPROM.
Definition at line 17 of file Eeprom\_LUT.h.
```

### 15.97.2.10 C\_INIT\_STATE

```
const uint8_t C_INIT_STATE = 0xA5
Definition at line 11 of file Eeprom\_LUT.h.
```

### 15.97.2.11 C\_INSTANT\_CURRENT

```
const int16_t C_INSTANT_CURRENT = 13
Definition at line 30 of file Eeprom\_LUT.h.
```

### 15.97.2.12 C\_INSTANT\_POWER

```
const int16_t C_INSTANT_POWER = 14
Definition at line 31 of file Eeprom\_LUT.h.
```

### 15.97.2.13 C\_INSTANT\_VOLTAGE

```
const int16_t C_INSTANT_VOLTAGE = 12
Definition at line 29 of file Eeprom_LUT.h.
```

### 15.97.2.14 C\_MODEL

```
const int16_t C_MODEL = 16
Definition at line 33 of file Eeprom_LUT.h.
```

### 15.97.2.15 C\_NUM\_ELEMENTS

```
const int16_t C_NUM_ELEMENTS = 20
Definition at line 37 of file Eeprom_LUT.h.
```

### 15.97.2.16 C\_NUMBER\_CYCLES

```
const int16_t C_NUMBER_CYCLES = 7
Definition at line 24 of file Eeprom_LUT.h.
```

### 15.97.2.17 C\_POWER\_ERROR

```
const int16_t C_POWER_ERROR = 9
Definition at line 26 of file Eeprom_LUT.h.
```

### 15.97.2.18 C\_SERIAL\_NUMBER

```
const int16_t C_SERIAL_NUMBER = 6
Definition at line 23 of file Eeprom_LUT.h.
```

### 15.97.2.19 C\_THEORY\_VOLTAGE

```
const int16_t C_THEORY_VOLTAGE = 15
Definition at line 32 of file Eeprom_LUT.h.
```

### 15.97.2.20 C\_VOLTAGE\_ERROR

```
const int16_t C_VOLTAGE_ERROR = 11
Definition at line 28 of file Eeprom_LUT.h.
```

### 15.97.2.21 C\_VOLTAGE\_INPUT\_ERROR

```
const int16_t C_VOLTAGE_INPUT_ERROR = 18
Definition at line 35 of file Eeprom_LUT.h.
```

### 15.97.2.22 C\_WORK\_TIME

```
const int16_t C_WORK_TIME = 8
Definition at line 25 of file Eeprom_LUT.h.
```

**15.97.2.23 POS\_EEPROM\_CONSUMPTION\_ERROR**

```
const int16_t POS EEPROM CONSUMPTION ERROR = 0x0145
Definition at line 51 of file Eeprom_LUT.h.
```

**15.97.2.24 POS\_EEPROM\_INIT**

```
const int16_t POS EEPROM INIT = 0x0000
Baterie LUT EEPROM.
Definition at line 43 of file Eeprom_LUT.h.
```

**15.97.2.25 POS\_EEPROM\_MODEL**

```
const int16_t POS EEPROM MODEL = 0x0001
Definition at line 44 of file Eeprom_LUT.h.
```

**15.97.2.26 POS\_EEPROM\_NUMBER\_CYCLES**

```
const int16_t POS EEPROM NUMBER CYCLES = 0x0005
Definition at line 46 of file Eeprom_LUT.h.
```

**15.97.2.27 POS\_EEPROM\_POWER\_ARRAY**

```
const int16_t POS EEPROM POWER ARRAY = 0x00E5
Definition at line 48 of file Eeprom_LUT.h.
```

**15.97.2.28 POS\_EEPROM\_POWER\_ERROR**

```
const int16_t POS EEPROM POWER ERROR = 0x0143
Definition at line 50 of file Eeprom_LUT.h.
```

**15.97.2.29 POS\_EEPROM\_SERIAL\_NUMBER**

```
const int16_t POS EEPROM SERIAL NUMBER = 0x0003
Definition at line 45 of file Eeprom_LUT.h.
```

**15.97.2.30 POS\_EEPROM\_VOLTAGE\_ERROR**

```
const int16_t POS EEPROM VOLTAGE ERROR = 0x0147
Definition at line 52 of file Eeprom_LUT.h.
```

**15.97.2.31 POS\_EEPROM\_VOLTAGE\_INPUT\_ERROR**

```
const int16_t POS EEPROM VOLTAGE INPUT ERROR = 0x0149
Definition at line 53 of file Eeprom_LUT.h.
```

**15.97.2.32 POS\_EEPROM\_VOLTS\_ARRAY**

```
const int16_t POS EEPROM VOLTS ARRAY = 0x0007
Definition at line 47 of file Eeprom_LUT.h.
```

### 15.97.2.33 POS\_EEPROM\_WORK\_TIME

```
const int16_t POS EEPROM WORK TIME = 0x0141
Definition at line 49 of file Eeprom_LUT.h.
```

## 15.98 Eeprom\_LUT.h

Go to the documentation of this file.

```
00001
00011 const uint8_t C_INIT_STATE = 0xA5;
00012
00017 const int16_t C HACK START SOUND = 0;
00018 const int16_t C HACK END SOUND = 1;
00019 const int16_t C HACK CHARGE SOUND = 2;
00020 const int16_t C HACK DEATH BATTERY SOUND = 3;
00021 const int16_t C HACK FULL CHARGE SOUND = 4;
00022 const int16_t C HACK START DISPLAY = 5;
00023 const int16_t C SERIAL NUMBER = 6;
00024 const int16_t C NUMBER CYCLES = 7;
00025 const int16_t C WORK TIME = 8;
00026 const int16_t C POWER ERROR = 9;
00027 const int16_t C CONSUMPTION ERROR = 10;
00028 const int16_t C VOLTAGE ERROR = 11;
00029 const int16_t C INSTANT VOLTAGE = 12;
00030 const int16_t C INSTANT CURRENT = 13;
00031 const int16_t C INSTANT POWER = 14;
00032 const int16_t C THEORY VOLTAGE = 15;
00033 const int16_t C MODEL = 16;
00034 const int16_t C CAPACITY = 17;
00035 const int16_t C VOLTAGE INPUT ERROR = 18;
00036 const int16_t C HACK END DISPLAY = 19;
00037 const int16_t C NUM ELEMENTS = 20; // NOTE: MODIFICAR CADA VEZ QUE SE AÑADA UN ELEMENTO!!!!
00038
00043 const int16_t POS EEPROM INIT = 0x0000; // byte -> 1 pos
00044 const int16_t POS EEPROM MODEL = 0x0001; // int16_t -> 2 pos
00045 const int16_t POS EEPROM SERIAL NUMBER = 0x0003; // int16_t -> 2 pos
00046 const int16_t POS EEPROM NUMBER CYCLES = 0x0005; // int16_t -> 2 pos
00047 const int16_t POS EEPROM VOLTS ARRAY = 0x0007; // int16_t[11] -> 222 pos
00048 const int16_t POS EEPROM POWER ARRAY = 0x00E5; // int16_t[46] -> 92 pos
00049 const int16_t POS EEPROM WORK TIME = 0x0141; // int16_t -> 2 pos
00050 const int16_t POS EEPROM POWER ERROR = 0x0143; // int16_t -> 2 pos
00051 const int16_t POS EEPROM CONSUMPTION ERROR = 0x0145; // int16_t -> 2 pos
00052 const int16_t POS EEPROM VOLTAGE ERROR = 0x0147; // int16_t -> 2 pos
00053 const int16_t POS EEPROM VOLTAGE INPUT ERROR = 0x0149; // int16_t -> 2 pos
```

## 15.99 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0010-LOGGING/EepromBitBang.h File Reference

### Macros

- #define SDA\_PIN 13
- #define SCL\_PIN 12
- #define EEPROMADDR 0xA0

### Functions

- boolean `readEEPROM` (unsigned long address, uint8\_t \*byte)
- boolean `writeEEPROM` (long unsigned address, uint8\_t byte)
- uint16\_t `ReadWordEEPROM` (int16\_t address)
- void `WriteWordEEPROM` (int16\_t address, int16\_t value)
- void `ReadArrayEEPROM` (uint16\_t address, uint16\_t \*array, uint16\_t length\_array)
- void `WriteArrayEEPROM` (uint16\_t address, uint16\_t \*array, uint16\_t length\_array)

### Variables

- SlowSoftI2CMaster si = SlowSoftI2CMaster(SDA\_PIN, SCL\_PIN, true)

## 15.99.1 Macro Definition Documentation

### 15.99.1.1 EEPROMADDR

```
#define EEPROMADDR 0xA0
```

Definition at line 9 of file [EepromBitBang.h](#).

### 15.99.1.2 SCL\_PIN

```
#define SCL_PIN 12
```

Definition at line 5 of file [EepromBitBang.h](#).

### 15.99.1.3 SDA\_PIN

```
#define SDA_PIN 13
```

Definition at line 4 of file [EepromBitBang.h](#).

## 15.99.2 Function Documentation

### 15.99.2.1 ReadArrayEEPROM()

```
void ReadArrayEEPROM (
    uint16_t address,
    uint16_t * array,
    uint16_t length_array )
```

Definition at line 81 of file [EepromBitBang.h](#).

### 15.99.2.2 readEEPROM()

```
boolean readEEPROM (
    unsigned long address,
    uint8_t * byte )
```

Definition at line 15 of file [EepromBitBang.h](#).

### 15.99.2.3 ReadWordEEPROM()

```
uint16_t ReadWordEEPROM (
    int16_t address )
```

Definition at line 64 of file [EepromBitBang.h](#).

### 15.99.2.4 WriteArrayEEPROM()

```
void WriteArrayEEPROM (
    uint16_t address,
    uint16_t * array,
    uint16_t length_array )
```

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

### 15.99.2.5 writeEEPROM()

```
boolean writeEEPROM (
    long unsigned address,
    uint8_t byte )
```

Definition at line 40 of file [EepromBitBang.h](#).

### 15.99.2.6 WriteWordEEPROM()

```
void WriteWordEEPROM (
    int16_t address,
    int16_t value )
```

Definition at line 74 of file [EepromBitBang.h](#).

## 15.99.3 Variable Documentation

### 15.99.3.1 si

```
SlowSoftI2CMaster si = SlowSoftI2CMaster(SDA_PIN, SCL_PIN, true)
```

Definition at line 7 of file [EepromBitBang.h](#).

## 15.100 EepromBitBang.h

[Go to the documentation of this file.](#)

```
00001 // Sketch to explore 24AA1024 using SlowSoftI2CMaster
00002
00003 //#include <batt_SlowSoftI2CMaster.h>
00004 #define SDA_PIN 13
00005 #define SCL_PIN 12
00006
00007 SlowSoftI2CMaster si = SlowSoftI2CMaster(SDA_PIN, SCL_PIN, true);
00008
00009 #define EEPROMADDR 0xA0 // set by jumper (A0 and A1 = High)
00010
00011 //-----
00012 /*
00013 * read one byte from address
00014 */
00015 boolean readEEPROM(unsigned long address, uint8_t *byte)
00016 {
00017     // issue a start condition, send device address and write direction bit
00018     if (!si.i2c_start(EEPROMADDR | I2C_WRITE | (address & 0x10000 ? 8 : 0)))
00019         return false;
00020
00021     // send the address
00022     if (!si.i2c_write((address >> 8) & 0xFF))
00023         return false;
00024     if (!si.i2c_write(address & 0xFF))
00025         return false;
00026
00027     // issue a repeated start condition, send device address and read direction bit
00028     if (!si.i2c_rep_start(EEPROMADDR | I2C_READ | (address & 0x10000 ? 8 : 0)))
00029         return false;
00030
00031     *byte = si.i2c_read(true);
00032
00033     si.i2c_stop();
00034     return true;
00035 }
00036
00037 /*
00038 * write 1 byte to 'address' in eeprom
00039 */
00040 boolean writeEEPROM(long unsigned address, uint8_t byte)
00041 {
00042     // issue a start condition, send device address and write direction bit
00043     if (!si.i2c_start(EEPROMADDR | I2C_WRITE | (address & 0x10000 ? 8 : 0)))
00044         return false;
00045
00046     // send the address
00047     if (!si.i2c_write((address >> 8) & 0xFF))
```

```

00048     return false;
00049     if (!si.i2c_write(address & 0xFF))
00050         return false;
00051
00052     // send data to EEPROM
00053     if (!si.i2c_write(byte))
00054         return false;
00055
00056     // issue a stop condition
00057     si.i2c_stop();
00058
00059     delay(6);
00060
00061     return true;
00062 }
00063
00064 uint16_t ReadWordEEPROM(int16_t address)
00065 {
00066     uint8_t low_byte = 0;
00067     uint8_t high_byte = 0;
00068     uint16_t value = 0;
00069     readEEPROM(address, &low_byte);
00070     readEEPROM(address + 1, &high_byte);
00071     value = (high_byte << 8) | low_byte;
00072     return value;
00073 }
00074 void WriteWordEEPROM(int16_t address, int16_t value)
00075 {
00076     uint8_t low_byte = (byte)value;
00077     uint8_t high_byte = (byte)(value >> 8);
00078     writeEEPROM(address, low_byte);
00079     writeEEPROM(address + 1, high_byte);
00080 }
00081 void ReadArrayEEPROM(uint16_t address, uint16_t *array, uint16_t length_array)
00082 {
00083     for (int i = 0; i < length_array; i++)
00084     {
00085         array[i] = ReadWordEEPROM(address + i * 2);
00086     }
00087 }
00088 void WriteArrayEEPROM(uint16_t address, uint16_t *array, uint16_t length_array)
00089 {
00090     for (int i = 0; i < length_array; i++)
00091     {
00092         WriteWordEEPROM(address + i * 2, array[i]);
00093     }
00094 }

```

## 15.101 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_↔ Library/batt\_Adafuit\_IS31FL3731.cpp File Reference

```
#include <batt_Adafuit_GFX.h>
#include <batt_Adafuit_IS31FL3731.h>
#include <Arduino.h>
#include <Wire.h>
```

### Macros

- #define \_swap\_int16\_t(a, b)

#### 15.101.1 Macro Definition Documentation

##### 15.101.1.1 \_swap\_int16\_t

```
#define _swap_int16_t(
    a,
    b )
```

**Value:**

```
{
    int16_t t = a;
    a = b;
    b = t;
}
```

```
\ \
\ \
\
```

Definition at line 7 of file [batt\\_Adafruit\\_IS31FL3731.cpp](#).

## 15.102 batt\_Adafruit\_IS31FL3731.cpp

[Go to the documentation of this file.](#)

```
00001 #include <batt_Adafruit_GFX.h>
00002 #include <batt_Adafruit_IS31FL3731.h>
00003 #include <Arduino.h>
00004 #include <Wire.h>
00005
00006 #ifndef _swap_int16_t
00007 #define _swap_int16_t(a, b)
00008 {
00009     int16_t t = a;
00010     a = b;
00011     b = t;
00012 }
00013 #endif
00014
00015 /***** Adafruit_IS31FL3731::Adafruit_IS31FL3731(uint8_t width, uint8_t height) *****/
00021 /***** Adafruit_IS31FL3731_Wing::Adafruit_IS31FL3731_Wing(void) *****/
00022
00023 Adafruit_IS31FL3731::Adafruit_IS31FL3731(uint8_t width, uint8_t height)
00024     : Adafruit_GFX(width, height) {}
00025
00026 /***** Adafruit_IS31FL3731::begin(uint8_t addr) *****/
00030 /***** Adafruit_IS31FL3731_Wing::Adafruit_IS31FL3731_Wing(void) *****/
00031 Adafruit_IS31FL3731_Wing::Adafruit_IS31FL3731_Wing(void)
00032     : Adafruit_IS31FL3731(15, 7) {}
00033
00034 /***** Adafruit_IS31FL3731::begin(uint8_t addr) *****/
00040 /***** Adafruit_IS31FL3731::begin(uint8_t addr) *****/
00041 bool Adafruit_IS31FL3731::begin(uint8_t addr) {
00042     Wire.begin();
00043     Wire.setClock(400000);
00044
00045     _i2caddr = addr;
00046     _frame = 0;
00047
00048     // A basic scanner, see if it ACK's
00049     Wire.beginTransmission(_i2caddr);
00050     if (Wire.endTransmission() != 0) {
00051         return false;
00052     }
00053
00054     // shutdown
00055     writeRegister8(ISSI_BANK_FUNCTIONREG, ISSI_REG_SHUTDOWN, 0x00);
00056
00057     delay(10);
00058
00059     // out of shutdown
00060     writeRegister8(ISSI_BANK_FUNCTIONREG, ISSI_REG_SHUTDOWN, 0x01);
00061
00062     // picture mode
00063     writeRegister8(ISSI_BANK_FUNCTIONREG, ISSI_REG_CONFIG,
00064                     ISSI_REG_CONFIG_PICTUREMODE);
00065
00066     displayFrame(_frame);
00067
00068     // all LEDs on & 0 PWM
00069     clear(); // set each led to 0 PWM
00070
00071     for (uint8_t f = 0; f < 8; f++) {
00072         for (uint8_t i = 0; i <= 0x11; i++)
00073             writeRegister8(f, i, 0xff); // each 8 LEDs on
00074     }
00075
00076     audioSync(false);
00077
00078     return true;
00079 }
00080
00081 /***** Adafruit_IS31FL3731::clear(void) *****/
00085 /***** Adafruit_IS31FL3731::clear(void) *****/
00086 void Adafruit_IS31FL3731::clear(void) {
00087     selectBank(_frame);
00088
00089     for (uint8_t i = 0; i < 6; i++) {
```

```

00090     Wire.beginTransmission(_i2caddr);
00091     Wire.write((byte)0x24 + i * 24);
00092     // write 24 bytes at once
00093     for (uint8_t j = 0; j < 24; j++) {
00094         Wire.write((byte)0);
00095     }
00096     Wire.endTransmission();
00097 }
00098 }
00099
00100 /*****
00101 ****/
00102 /*****
00103 ****/
00104 void Adafruit_IS31FL3731::setLEDPWM(uint8_t lednum, uint8_t pwm, uint8_t bank) {
00105     if (lednum >= 144)
00106         return;
00107     writeRegister8(bank, 0x24 + lednum, pwm);
00108 }
00109
00110 /*****
00111 ****/
00112 /*****
00113 ****/
00114 void Adafruit_IS31FL3731_Wing::drawPixel(int16_t x, int16_t y, uint16_t color) {
00115     // check rotation, move pixel around if necessary
00116     switch (getRotation()) {
00117         case 1:
00118             _swap_int16_t(x, y);
00119             x = 15 - x - 1;
00120             break;
00121         case 2:
00122             x = 15 - x - 1;
00123             y = 7 - y - 1;
00124             break;
00125         case 3:
00126             _swap_int16_t(x, y);
00127             y = 9 - y - 1;
00128             break;
00129     }
00130
00131     // charlie wing is smaller:
00132     if ((x < 0) || (x >= 16) || (y < 0) || (y >= 7))
00133         return;
00134
00135     if (x > 7) {
00136         x = 15 - x;
00137         y += 8;
00138     } else {
00139         y = 7 - y;
00140     }
00141
00142     _swap_int16_t(x, y);
00143
00144     if (color > 255)
00145         color = 255; // PWM 8bit max
00146
00147     setLEDPWM(x + y * 16, color, _frame);
00148     return;
00149 }
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159 }
00160
00161 /*****
00162 ****/
00163 /*****
00164 ****/
00165 void Adafruit_IS31FL3731::drawPixel(int16_t x, int16_t y, uint16_t color) {
00166     // check rotation, move pixel around if necessary
00167     switch (getRotation()) {
00168         case 1:
00169             _swap_int16_t(x, y);
00170             x = 16 - x - 1;
00171             break;
00172         case 2:
00173             x = 16 - x - 1;
00174             y = 9 - y - 1;
00175             break;
00176         case 3:
00177             _swap_int16_t(x, y);
00178             y = 9 - y - 1;
00179             break;
00180     }
00181
00182     if ((x < 0) || (x >= 16))
00183         return;
00184     if ((y < 0) || (y >= 9))
00185         return;
00186     if (color > 255)
00187         color = 255; // PWM 8bit max
00188
00189     setLEDPWM(x + y * 16, color, _frame);
00190     return;
00191 }
00192
00193
00194
00195
00196 }
00197

```

## **15.103 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_Library/batt\_Adafruit\_IS31FL3731.h File Reference**

```
00198 /*****
00203 ****
00204 void Adafruit_IS31FL3731::setFrame(uint8_t frame) { _frame = frame; }
00205 ****
00206 ****
00211 ****
00212 void Adafruit_IS31FL3731::displayFrame(uint8_t frame) {
00213     if (frame > 7)
00214         frame = 0;
00215     writeRegister8(ISSI_BANK_FUNCTIONREG, ISSI_REG_PICTUREFRAME, frame);
00216 }
00217 ****
00218 ****
00223 ****
00224 void Adafruit_IS31FL3731::selectBank(uint8_t bank) {
00225     Wire.beginTransmission(_i2caddr);
00226     Wire.write((byte)ISSI_COMMANDREGISTER);
00227     Wire.write(bank);
00228     Wire.endTransmission();
00229 }
00230 ****
00231 ****
00236 ****
00237 void Adafruit_IS31FL3731::audioSync(bool sync) {
00238     if (sync) {
00239         writeRegister8(ISSI_BANK_FUNCTIONREG, ISSI_REG_AUDIOSYNC, 0x1);
00240     } else {
00241         writeRegister8(ISSI_BANK_FUNCTIONREG, ISSI_REG_AUDIOSYNC, 0x0);
00242     }
00243 }
00244 ****
00245 ****
00252 ****
00253 void Adafruit_IS31FL3731::writeRegister8(uint8_t bank, uint8_t reg,
00254                                     uint8_t data) {
00255     selectBank(bank);
00256     Wire.beginTransmission(_i2caddr);
00257     Wire.write((byte)reg);
00258     Wire.write((byte)data);
00260     Wire.endTransmission();
00261     // Serial.print("$"); Serial.print(reg, HEX);
00262     // Serial.print(" = 0x"); Serial.println(data, HEX);
00263 }
00264 ****
00272 ****
00273 uint8_t Adafruit_IS31FL3731::readRegister8(uint8_t bank, uint8_t reg) {
00274     uint8_t x;
00275     selectBank(bank);
00277     Wire.beginTransmission(_i2caddr);
00278     Wire.write((byte)reg);
00280     Wire.endTransmission();
00281     Wire.requestFrom(_i2caddr, (size_t)1);
00283     x = Wire.read();
00284     // Serial.print("$"); Serial.print(reg, HEX);
00286     // Serial.print(": 0x"); Serial.println(x, HEX);
00287     return x;
00289 }
```

## **15.103 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_Library/batt\_Adafruit\_IS31FL3731.h File Reference**

```
#include <batt_Adafruit_GFX.h>
#include <Arduino.h>
#include <Wire.h>
```

## Classes

- class [Adafruit\\_IS31FL3731](#)  
*Constructor for generic IS31FL3731 breakout version.*
- class [Adafruit\\_IS31FL3731\\_Wing](#)  
*Constructor for FeatherWing IS31FL3731 version.*

## Macros

- #define ISSI\_ADDR\_DEFAULT 0x74
- #define ISSI\_REG\_CONFIG 0x00
- #define ISSI\_REG\_CONFIG\_PICTUREMODE 0x00
- #define ISSI\_REG\_CONFIG\_AUTOPLAYMODE 0x08
- #define ISSI\_REG\_CONFIG\_AUDIOPLAYMODE 0x18
- #define ISSI\_CONF\_PICTUREMODE 0x00
- #define ISSI\_CONF\_AUTOFRAMEMODE 0x04
- #define ISSI\_CONF\_AUDIOMODE 0x08
- #define ISSI\_REG\_PICTUREFRAME 0x01
- #define ISSI\_REG\_SHUTDOWN 0x0A
- #define ISSI\_REG\_AUDIOSYNC 0x06
- #define ISSI\_COMMANDREGISTER 0xFD
- #define ISSI\_BANK\_FUNCTIONREG 0x0B

### 15.103.1 Macro Definition Documentation

#### 15.103.1.1 ISSI\_ADDR\_DEFAULT

```
#define ISSI_ADDR_DEFAULT 0x74
```

Definition at line [8](#) of file [batt\\_Adafruit\\_IS31FL3731.h](#).

#### 15.103.1.2 ISSI\_BANK\_FUNCTIONREG

```
#define ISSI_BANK_FUNCTIONREG 0x0B
```

Definition at line [25](#) of file [batt\\_Adafruit\\_IS31FL3731.h](#).

#### 15.103.1.3 ISSI\_COMMANDREGISTER

```
#define ISSI_COMMANDREGISTER 0xFD
```

Definition at line [24](#) of file [batt\\_Adafruit\\_IS31FL3731.h](#).

#### 15.103.1.4 ISSI\_CONF\_AUDIOMODE

```
#define ISSI_CONF_AUDIOMODE 0x08
```

Definition at line [17](#) of file [batt\\_Adafruit\\_IS31FL3731.h](#).

#### 15.103.1.5 ISSI\_CONF\_AUTOFRAMEMODE

```
#define ISSI_CONF_AUTOFRAMEMODE 0x04
```

Definition at line [16](#) of file [batt\\_Adafruit\\_IS31FL3731.h](#).

### 15.103.1.6 ISSI\_CONF\_PICTUREMODE

```
#define ISSI_CONF_PICTUREMODE 0x00
Definition at line 15 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.7 ISSI\_REG\_AUDIOSYNC

```
#define ISSI_REG_AUDIOSYNC 0x06
Definition at line 22 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.8 ISSI\_REG\_CONFIG

```
#define ISSI_REG_CONFIG 0x00
Definition at line 10 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.9 ISSI\_REG\_CONFIG\_AUDIOPLAYMODE

```
#define ISSI_REG_CONFIG_AUDIOPLAYMODE 0x18
Definition at line 13 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.10 ISSI\_REG\_CONFIG\_AUTOPLAYMODE

```
#define ISSI_REG_CONFIG_AUTOPLAYMODE 0x08
Definition at line 12 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.11 ISSI\_REG\_CONFIG\_PICTUREMODE

```
#define ISSI_REG_CONFIG_PICTUREMODE 0x00
Definition at line 11 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.12 ISSI\_REG\_PICTUREFRAME

```
#define ISSI_REG_PICTUREFRAME 0x01
Definition at line 19 of file batt_Adafruit_IS31FL3731.h.
```

### 15.103.1.13 ISSI\_REG\_SHUTDOWN

```
#define ISSI_REG_SHUTDOWN 0x0A
Definition at line 21 of file batt_Adafruit_IS31FL3731.h.
```

## 15.104 batt\_Adafruit\_IS31FL3731.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _ADAFRUIT_IS31FL3731_H_
00002 #define _ADAFRUIT_IS31FL3731_H_
00003
00004 #include <batt_Adafruit_GFX.h>
00005 #include <Arduino.h>
00006 #include <Wire.h>
00007
00008 #define ISSI_ADDR_DEFAULT 0x74
00009
00010 #define ISSI_REG_CONFIG 0x00
00011 #define ISSI_REG_CONFIG_PICTUREMODE 0x00
00012 #define ISSI_REG_CONFIG_AUTOPLAYMODE 0x08
00013 #define ISSI_REG_CONFIG_AUDIOPLAYMODE 0x18
00014
```

```

00015 #define ISSI_CONF_PICTUREMODE 0x00
00016 #define ISSI_CONF_AUTOFRAMEMODE 0x04
00017 #define ISSI_CONF_AUDIOMODE 0x08
00018
00019 #define ISSI_REG_PICTUREFRAME 0x01
00020
00021 #define ISSI_REG_SHUTDOWN 0x0A
00022 #define ISSI_REG_AUDIOSYNC 0x06
00023
00024 #define ISSI_COMMANDREGISTER 0xFD
00025 #define ISSI_BANK_FUNCTIONREG 0x0B // helpfully called 'page nine'
00026
00027 /*****
00031 *****/
00032 class Adafruit_IS31FL3731 : public Adafruit_GFX {
00033 public:
00034   Adafruit_IS31FL3731(uint8_t x = 16, uint8_t y = 9);
00035   bool begin(uint8_t addr = ISSI_ADDR_DEFAULT);
00036   void drawPixel(int16_t x, int16_t y, uint16_t color);
00037   void clear(void);
00038
00039   void setLEDPWM(uint8_t lednum, uint8_t pwm, uint8_t bank = 0);
00040   void audioSync(bool sync);
00041   void setFrame(uint8_t b);
00042   void displayFrame(uint8_t frame);
00043
00044 protected:
00045   void selectBank(uint8_t bank);
00046   void writeRegister8(uint8_t bank, uint8_t reg, uint8_t data);
00047   uint8_t readRegister8(uint8_t bank, uint8_t reg);
00048   uint8_t _i2caddr,
00049   _frame;
00050 };
00051
00052 *****/
00056 *****/
00057 class Adafruit_IS31FL3731_Wing : public Adafruit_IS31FL3731 {
00058 public:
00059   Adafruit_IS31FL3731_Wing(void);
00060   void drawPixel(int16_t x, int16_t y, uint16_t color);
00061 };
00062
00063 #endif

```

## 15.105 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_← Library/examples/gfxdemo/gfxdemo.ino File Reference

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_IS31FL3731.h>
```

### Functions

- void **setup ()**
- void **loop ()**

### Variables

- Adafruit\_IS31FL3731 matrix = **Adafruit\_IS31FL3731()**

#### 15.105.1 Function Documentation

##### 15.105.1.1 **loop()**

```
void loop ( )
```

Definition at line 53 of file [gfxdemo.ino](#).

### 15.105.1.2 setup()

```
void setup ()
```

Definition at line 40 of file [gfxdemo.ino](#).

## 15.105.2 Variable Documentation

### 15.105.2.1 matrix

```
Adafruit_IS31FL3731 matrix = Adafruit_IS31FL3731()
```

Definition at line 6 of file [gfxdemo.ino](#).

## 15.106 gfxdemo.ino

[Go to the documentation of this file.](#)

```
00001 #include <Wire.h>
00002 #include <Adafruit_GFX.h>
00003 #include <Adafruit_IS31FL3731.h>
00004
00005 // If you're using the full breakout...
00006 Adafruit_IS31FL3731 matrix = Adafruit_IS31FL3731();
00007 // If you're using the FeatherWing version
00008 //Adafruit_IS31FL3731_Wing matrix = Adafruit_IS31FL3731_Wing();
00009
00010 static const uint8_t PROGMEM
00011 smile_bmp[] =
00012 { B00111100,
00013   B01000010,
00014   B10100101,
00015   B10000001,
00016   B10100101,
00017   B10011001,
00018   B01000010,
00019   B00111100 },
00020 neutral_bmp[] =
00021 { B00111100,
00022   B01000010,
00023   B10100101,
00024   B10000001,
00025   B10111101,
00026   B10000001,
00027   B01000010,
00028   B00111100 },
00029 frown_bmp[] =
00030 { B00111100,
00031   B01000010,
00032   B10100101,
00033   B10000001,
00034   B10011001,
00035   B10100101,
00036   B01000010,
00037   B00111100 };
00038
00039
00040 void setup() {
00041
00042   Serial.begin(9600);
00043   Serial.println("IS31 manual animation test");
00044   if (! matrix.begin()) {
00045     Serial.println("IS31 not found");
00046     while (1);
00047   }
00048   Serial.println("IS31 Found!");
00049
00050 }
00051
00052
00053 void loop() {
00054   matrix.setRotation(0);
00055
00056   matrix.clear();
00057   matrix.drawBitmap(3, 0, smile_bmp, 8, 8, 255);
```

```

00058     delay(500);
00059
00060     matrix.clear();
00061     matrix.drawBitmap(3, 0, neutral_bmp, 8, 8, 64);
00062     delay(500);
00063
00064     matrix.clear();
00065     matrix.drawBitmap(3, 0, frown_bmp, 8, 8, 32);
00066     delay(500);
00067
00068     matrix.clear();
00069     matrix.drawPixel(0, 0, 255);
00070     delay(500);
00071
00072     matrix.clear();
00073     matrix.drawLine(0,0, matrix.width()-1, matrix.height()-1, 127);
00074     delay(500);
00075
00076     matrix.clear();
00077     matrix.drawRect(0,0, matrix.width(), matrix.height(), 255);
00078     matrix.fillRect(2,2, matrix.width()-4, matrix.height()-4, 20);
00079     delay(500);
00080
00081     matrix.clear();
00082     matrix.drawCircle(8,4, 4, 64);
00083     matrix.drawCircle(8,4, 2, 32);
00084     delay(500);
00085
00086
00087     matrix.setTextSize(1);
00088     matrix.setTextWrap(false); // we dont want text to wrap so it scrolls nicely
00089     matrix.setTextColor(100);
00090     for (int8_t x=0; x>=-32; x--) {
00091         matrix.clear();
00092         matrix.setCursor(x,0);
00093         matrix.print("Hello");
00094         delay(100);
00095     }
00096
00097     matrix.setTextSize(2);
00098     matrix.setTextWrap(false); // we dont want text to wrap so it scrolls nicely
00099     matrix.setTextColor(32);
00100     matrix.setRotation(1);
00101     for (int8_t x=7; x>=-64; x--) {
00102         matrix.clear();
00103         matrix.setCursor(x,0);
00104         matrix.print("World");
00105         delay(100);
00106     }
00107 }

```

## 15.107 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_↔ Library/examples/manualanim/manualanim.ino File Reference

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_IS31FL3731.h>
```

### Functions

- void **setup** ()
- void **loop** ()

### Variables

- Adafruit\_IS31FL3731 ledmatrix = Adafruit\_IS31FL3731()
- int **x** = 0

## 15.107.1 Function Documentation

### 15.107.1.1 loop()

```
void loop ()  
Definition at line 37 of file manualanim.ino.
```

### 15.107.1.2 setup()

```
void setup ()  
Definition at line 10 of file manualanim.ino.
```

## 15.107.2 Variable Documentation

### 15.107.2.1 ledmatrix

```
Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731()  
Definition at line 6 of file manualanim.ino.
```

### 15.107.2.2 x

```
int x = 0  
Definition at line 35 of file manualanim.ino.
```

## 15.108 manualanim.ino

[Go to the documentation of this file.](#)

```
00001 #include <Wire.h>  
00002 #include <Adafruit_GFX.h>  
00003 #include <Adafruit_IS31FL3731.h>  
00004  
00005 // If you're using the full breakout...  
00006 Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731();  
00007 // If you're using the FeatherWing version  
00008 //Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();  
00009  
00010 void setup() {  
00011  
00012   Serial.begin(9600);  
00013   Serial.println("IS31 manual animation test");  
00014   if (! ledmatrix.begin()) {  
00015     Serial.println("IS31 not found");  
00016     while (1);  
00017   }  
00018   Serial.println("IS31 Found!");  
00019  
00020   ledmatrix.setTextWrap(false);  
00021   ledmatrix.setTextColor(64); // quarter brightness  
00022   ledmatrix.setTextSize(1);  
00023  
00024   // fill all 8 frames with some text  
00025   for (uint8_t frame = 0; frame < 8; frame++) {  
00026     ledmatrix.setFrame(frame);  
00027     ledmatrix.clear();  
00028     ledmatrix.setCursor(0,0);  
00029     ledmatrix.write('a'+frame*3);  
00030     ledmatrix.write('b'+frame*3);  
00031     ledmatrix.write('c'+frame*3);  
00032   }  
00033 }  
00034  
00035 int x = 0;  
00036  
00037 void loop() {  
00038   // display each frame for 250 milliseconds  
00039   for (uint8_t frame = 0; frame < 8; frame++) {
```

```

00040     ledmatrix.displayFrame(frame);
00041     delay(250);
00042 }
00043 }
```

## 15.109 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0011-Adafruit\_IS31FL3731\_↔ Library/examples/swirldemo/swirldemo.ino File Reference

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_IS31FL3731.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

### Variables

- Adafruit\_IS31FL3731 ledmatrix = Adafruit\_IS31FL3731()
- uint8\_t sweep[] = {1, 2, 3, 4, 6, 8, 10, 15, 20, 30, 40, 60, 60, 40, 30, 20, 15, 10, 8, 6, 4, 3, 2, 1}

#### 15.109.1 Function Documentation

##### 15.109.1.1 [loop\(\)](#)

```
void loop ( )
```

Definition at line 25 of file [swirldemo.ino](#).

##### 15.109.1.2 [setup\(\)](#)

```
void setup ( )
```

Definition at line 14 of file [swirldemo.ino](#).

#### 15.109.2 Variable Documentation

##### 15.109.2.1 [ledmatrix](#)

```
Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731()
```

Definition at line 6 of file [swirldemo.ino](#).

##### 15.109.2.2 [sweep](#)

```
uint8_t sweep[] = {1, 2, 3, 4, 6, 8, 10, 15, 20, 30, 40, 60, 60, 40, 30, 20, 15, 10, 8, 6, 4, 3, 2, 1}
```

Definition at line 12 of file [swirldemo.ino](#).

## 15.110 swirldemo.ino

[Go to the documentation of this file.](#)

```

00001 #include <Wire.h>
00002 #include <Adafruit_GFX.h>
00003 #include <Adafruit_IS31FL3731.h>
00004
00005 // If you're using the full breakout...
00006 Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731();
00007 // If you're using the FeatherWing version
00008 //Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
00009
00010
00011 // The lookup table to make the brightness changes be more visible
00012 uint8_t sweep[] = {1, 2, 3, 4, 6, 8, 10, 15, 20, 30, 40, 60, 60, 40, 30, 20, 15, 10, 8, 6, 4, 3, 2,
1};
00013
00014 void setup() {
00015   Serial.begin(9600);
00016   Serial.println("ISSI swirl test");
00017
00018   if (! ledmatrix.begin()) {
00019     Serial.println("IS31 not found");
00020     while (1);
00021   }
00022   Serial.println("IS31 found!");
00023 }
00024
00025 void loop() {
00026   // animate over all the pixels, and set the brightness from the sweep table
00027   for (uint8_t incr = 0; incr < 24; incr++)
00028     for (uint8_t x = 0; x < 16; x++)
00029       for (uint8_t y = 0; y < 9; y++)
00030         ledmatrix.drawPixel(x, y, sweep[(x+y+incr)%24]);
00031   delay(20);
00032 }
```

## 15.111 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/examples/ EmulateEEPROM/EmulateEEPROM.ino File Reference

```
#include <FlashAsEEPROM.h>
```

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

#### 15.111.1 Function Documentation

##### 15.111.1.1 [loop\(\)](#)

```
void loop ()
```

Definition at line 55 of file [EmulateEEPROM.ino](#).

##### 15.111.1.2 [setup\(\)](#)

```
void setup ()
```

Definition at line 13 of file [EmulateEEPROM.ino](#).

## 15.112 EmulateEEPROM.ino

[Go to the documentation of this file.](#)

```

00001 /*
00002 Demonstrate how to use FlashStorage with an API that is similar to the EEPROM library.
00003 This example code is in the public domain.
00005
00006 Written by A. Christian
00007 Edited 14 Oct 2016 by Cristian Maglie
00008 */
00009
00010 // Include EEPROM-like API for FlashStorage
00011 #include <FlashAsEEPROM.h>
00012
00013 void setup() {
00014   Serial.begin(9600);
00015
00016 // If the EEPROM is empty then isValid() is false
00017 if (!EEPROM.isValid()) {
00018
00019   Serial.println("EEPROM is empty, writing some example data:");
00020   Serial.print(">");
00021   for (int i=0; i<20; i++) {
00022     EEPROM.write(i, 100+i);
00023     Serial.print(" ");
00024     Serial.print(100+i);
00025   }
00026   Serial.println();
00027
00028 // commit() saves all the changes to EEPROM, it must be called
00029 // every time the content of virtual EEPROM is changed to make
00030 // the change permanent.
00031 // This operation burns Flash write cycles and should not be
00032 // done too often. See readme for details:
00033 // https://github.com/cmaglie/FlashStorage#limited-number-of-writes
00034 EEPROM.commit();
00035   Serial.println("Done!");
00036
00037   Serial.print("After commit, calling isValid() returns ");
00038   Serial.println(EEPROM.isValid());
00039
00040 } else {
00041
00042   Serial.println("EEPROM has been written.");
00043   Serial.println("Here is the content of the first 20 bytes:");
00044
00045   Serial.print(">");
00046   for (int i=0; i<20; i++) {
00047     Serial.print(" ");
00048     Serial.print(EEPROM.read(i));
00049   }
00050   Serial.println();
00051
00052 }
00053 }
00054
00055 void loop() {
00056 }
```

## 15.113 C:/Users/Javi/Team Dropbox/Javi

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
 Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/examples/←  
 FlashStoreAndRetrieve/FlashStoreAndRetrieve.ino File  
 Reference

```
#include <FlashStorage.h>
```

### Functions

- `FlashStorage` (`my_flash_store, int`)
- `void setup ()`
- `void loop ()`

### 15.113.1 Function Documentation

#### 15.113.1.1 FlashStorage()

```
FlashStorage (
    my_flash_store ,
    int )
```

#### 15.113.1.2 loop()

```
void loop ()  
Definition at line 35 of file FlashStoreAndRetrieve.ino.
```

#### 15.113.1.3 setup()

```
void setup ()  
Definition at line 19 of file FlashStoreAndRetrieve.ino.
```

## 15.114 FlashStoreAndRetrieve.ino

[Go to the documentation of this file.](#)

```
00001 /*
00002   Store and retrieve an integer value in Flash memory.
00003   The value is increased each time the board is restarted.
00004
00005   This example code is in the public domain.
00006
00007   Written 30 Apr 2015 by Cristian Maglie
00008 */
00009
00010 #include <FlashStorage.h>
00011
00012 // Reserve a portion of flash memory to store an "int" variable
00013 // and call it "my_flash_store".
00014 FlashStorage(my_flash_store, int);
00015
00016 // Note: the area of flash memory reserved for the variable is
00017 // lost every time the sketch is uploaded on the board.
00018
00019 void setup() {
00020   SERIAL_PORT_MONITOR.begin(9600);
00021
00022   int number;
00023
00024   // Read the content of "my_flash_store" and assign it to "number"
00025   number = my_flash_store.read();
00026
00027   // Print the current number on the serial monitor
00028   SERIAL_PORT_MONITOR.println(number);
00029
00030   // Save into "my_flash_store" the number increased by 1 for the
00031   // next run of the sketch
00032   my_flash_store.write(number + 1);
00033 }
00034
00035 void loop() {
00036   // Do nothing...
00037 }
00038
```

## 15.115 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/examples/← StoreNameAndSurname/StoreNameAndSurname.ino File Reference

```
#include <FlashStorage.h>
```

### Classes

- struct [Person](#)

### Functions

- [FlashStorage](#) ([my\\_flash\\_store](#), [Person](#))
- void [setup](#) ()
- void [loop](#) ()

#### 15.115.1 Function Documentation

##### 15.115.1.1 [FlashStorage\(\)](#)

```
FlashStorage (
    my_flash_store ,
    Person )
```

##### 15.115.1.2 [loop\(\)](#)

```
void loop ()
```

Definition at line [78](#) of file [StoreNameAndSurname.ino](#).

##### 15.115.1.3 [setup\(\)](#)

```
void setup ()
```

Definition at line [27](#) of file [StoreNameAndSurname.ino](#).

## 15.116 StoreNameAndSurname.ino

[Go to the documentation of this file.](#)

```
00001 /*
00002   Store and retrieve structured data in Flash memory.
00003
00004   This example code is in the public domain.
00005
00006   Written 30 Apr 2015 by Cristian Maglie
00007 */
00008
00009 #include <FlashStorage.h>
00010
00011 // Create a structure that is big enough to contain a name
00012 // and a surname. The "valid" variable is set to "true" once
00013 // the structure is filled with actual data for the first time.
00014 typedef struct {
00015   boolean valid;
00016   char name[100];
00017   char surname[100];
00018 } Person;
```

```

00019
00020 // Reserve a portion of flash memory to store a "Person" and
00021 // call it "my_flash_store".
00022 FlashStorage(my_flash_store, Person);
00023
00024 // Note: the area of flash memory reserved lost every time
00025 // the sketch is uploaded on the board.
00026
00027 void setup() {
00028     SERIAL_PORT_MONITOR.begin(9600);
00029     while (!SERIAL_PORT_MONITOR) { }
00030
00031     // Create a "Person" variable and call it "owner"
00032     Person owner;
00033
00034     // Read the content of "my_flash_store" into the "owner" variable
00035     owner = my_flash_store.read();
00036
00037     // If this is the first run the "valid" value should be "false"...
00038     if (owner.valid == false) {
00039
00040         // ...in this case we ask for user data.
00041         SERIAL_PORT_MONITOR.setTimeout(30000);
00042         SERIAL_PORT_MONITOR.println("Insert your name:");
00043         String name = SERIAL_PORT_MONITOR.readStringUntil('\n');
00044         SERIAL_PORT_MONITOR.println("Insert your surname:");
00045         String surname = SERIAL_PORT_MONITOR.readStringUntil('\n');
00046
00047         // Fill the "owner" structure with the data entered by the user...
00048         name.toCharArray(owner.name, 100);
00049         surname.toCharArray(owner.surname, 100);
00050         // set "valid" to true, so the next time we know that we
00051         // have valid data inside
00052         owner.valid = true;
00053
00054         // ...and finally save everything into "my_flash_store"
00055         my_flash_store.write(owner);
00056
00057         // Print a confirmation of the data inserted.
00058         SERIAL_PORT_MONITOR.println();
00059         SERIAL_PORT_MONITOR.print("Your name: ");
00060         SERIAL_PORT_MONITOR.println(owner.name);
00061         SERIAL_PORT_MONITOR.print("And your surname: ");
00062         SERIAL_PORT_MONITOR.println(owner.surname);
00063         SERIAL_PORT_MONITOR.println("have been saved. Thank you!");
00064
00065     } else {
00066
00067         // Say hello to the returning user!
00068         SERIAL_PORT_MONITOR.println();
00069         SERIAL_PORT_MONITOR.print("Hi ");
00070         SERIAL_PORT_MONITOR.print(owner.name);
00071         SERIAL_PORT_MONITOR.print(" ");
00072         SERIAL_PORT_MONITOR.print(owner.surname);
00073         SERIAL_PORT_MONITOR.println(", nice to see you again :-)");
00074
00075     }
00076 }
00077
00078 void loop() {
00079     // Do nothing...
00080 }
00081

```

## 15.117 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/batt\_FlashAsEEPROM.cpp File Reference

```
#include "batt_FlashAsEEPROM.h"
```

### Functions

- **FlashStorage** (eeprom\_storage, EEPROM\_EMULATION)

## Variables

- EEPROMClass EEPROM

### 15.117.1 Function Documentation

#### 15.117.1.1 FlashStorage()

```
FlashStorage (
    eeprom_storage ,
    EEPROM_EMULATION )
```

### 15.117.2 Variable Documentation

#### 15.117.2.1 EEPROM

EEPROMClass EEPROM

Definition at line 74 of file [batt\\_FlashAsEEPROM.cpp](#).

## 15.118 batt\_FlashAsEEPROM.cpp

[Go to the documentation of this file.](#)

```
00001 /*
00002     EEPROM like API that uses Arduino Zero's flash memory.
00003     Written by A. Christian
00004
00005     Copyright (c) 2015-2016 Arduino LLC. All right reserved.
00006
00007     This library is free software; you can redistribute it and/or
00008     modify it under the terms of the GNU Lesser General Public
00009     License as published by the Free Software Foundation; either
00010     version 2.1 of the License, or (at your option) any later version.
00011
00012     This library is distributed in the hope that it will be useful,
00013     but WITHOUT ANY WARRANTY; without even the implied warranty of
00014     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00015     See the GNU Lesser General Public License for more details.
00016
00017     You should have received a copy of the GNU Lesser General Public
00018     License along with this library; if not, write to the Free Software
00019     Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00020 */
00021
00022 #include "batt_FlashAsEEPROM.h"
00023
00024 FlashStorage(eeprom_storage, EEPROM_EMULATION);
00025
00026 EEPROMClass::EEPROMClass(void) : _initialized(false), _dirty(false) {
00027     // Empty
00028 }
00029
00030 uint8_t EEPROMClass::read(int address)
00031 {
00032     if (!_initialized) init();
00033     return _eprom.data[address];
00034 }
00035
00036 void EEPROMClass::update(int address, uint8_t value)
00037 {
00038     if (!_initialized) init();
00039     if (_eprom.data[address] != value) {
00040         _dirty = true;
00041         _eprom.data[address] = value;
00042     }
00043 }
00044
00045 void EEPROMClass::write(int address, uint8_t value)
00046 {
00047     update(address, value);
00048 }
```

```
00050 void EEPROMClass::init()
00051 {
00052     _eeprom = eeprom_storage.read();
00053     if (!eeprom.valid) {
00054         memset(_eeprom.data, 0xFF, EEPROM_EMULATION_SIZE);
00055     }
00056     _initialized = true;
00057 }
00058
00059 bool EEPROMClass::isValid()
00060 {
00061     if (!_initialized) init();
00062     return _eeprom.valid;
00063 }
00064
00065 void EEPROMClass::commit()
00066 {
00067     if (!_initialized) init();
00068     if (_dirty) {
00069         _eeprom.valid = true;
00070         eeprom_storage.write(_eeprom);
00071     }
00072 }
00073
00074 EEPROMClass EEPROM;
```

## 15.119 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/batt\_FlashAsEEPROM.h File Reference

```
#include "batt_FlashStorage.h"
```

### Classes

- struct [EEPROM\\_EMULATION](#)
- class [EEPROMClass](#)

### Macros

- #define [EEPROM\\_EMULATION\\_SIZE](#) 1024

### Variables

- [EEPROMClass EEPROM](#)

#### 15.119.1 Macro Definition Documentation

##### 15.119.1.1 EEPROM\_EMULATION\_SIZE

```
#define EEPROM_EMULATION_SIZE 1024
Definition at line 28 of file batt\_FlashAsEEPROM.h.
```

#### 15.119.2 Variable Documentation

##### 15.119.2.1 EEPROM

```
EEPROMClass EEPROM [extern]
Definition at line 74 of file batt\_FlashAsEEPROM.cpp.
```

## 15.120 batt\_FlashAsEEPROM.h

[Go to the documentation of this file.](#)

```

00001 /*
00002   EEPROM like API that uses Arduino Zero's flash memory.
00003   Written by A. Christian
00004
00005   Copyright (c) 2015-2016 Arduino LLC. All right reserved.
00006
00007   This library is free software; you can redistribute it and/or
00008   modify it under the terms of the GNU Lesser General Public
00009   License as published by the Free Software Foundation; either
00010   version 2.1 of the License, or (at your option) any later version.
00011
00012   This library is distributed in the hope that it will be useful,
00013   but WITHOUT ANY WARRANTY; without even the implied warranty of
00014   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00015   See the GNU Lesser General Public License for more details.
00016
00017   You should have received a copy of the GNU Lesser General Public
00018   License along with this library; if not, write to the Free Software
00019   Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00020 */
00021
00022 #ifndef FLASH_AS_EEPROM_h
00023 #define FLASH_AS_EEPROM_h
00024
00025 #include "batt_FlashStorage.h"
00026
00027 #ifndef EEPROM_EMULATION_SIZE
00028 #define EEPROM_EMULATION_SIZE 1024
00029 #endif
00030
00031 typedef struct {
00032   byte data[EEPROM_EMULATION_SIZE];
00033   boolean valid;
00034 } EEPROM_EMULATION;
00035
00036
00037 class EEPROMClass {
00038
00039   public:
00040     EEPROMClass(void);
00041
00042     uint8_t read(int);
00043
00044     void write(int, uint8_t);
00045
00046     void update(int, uint8_t);
00047
00048     bool isValid();
00049
00050     void commit();
00051
00052     uint16_t length() { return EEPROM_EMULATION_SIZE; }
00053
00054   private:
00055     void init();
00056
00057     bool _initialized;
00058     EEPROM_EMULATION _eeprom;
00059     bool _dirty;
00060 };
00061
00062 extern EEPROMClass EEPROM;
00063
00064 #endif

```

## 15.121 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/batt\_FlashStorage.cpp File Reference

```
#include "batt_FlashStorage.h"
```

## 15.122 batt\_FlashStorage.cpp

Go to the documentation of this file.

```
00001 /*
00002 Copyright (c) 2015 Arduino LLC. All right reserved.
00003 Written by Cristian Maglie
00004
00005 This library is free software; you can redistribute it and/or
00006 modify it under the terms of the GNU Lesser General Public
00007 License as published by the Free Software Foundation; either
00008 version 2.1 of the License, or (at your option) any later version.
00009
00010 This library is distributed in the hope that it will be useful,
00011 but WITHOUT ANY WARRANTY; without even the implied warranty of
00012 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00013 See the GNU Lesser General Public License for more details.
00014
00015 You should have received a copy of the GNU Lesser General Public
00016 License along with this library; if not, write to the Free Software
00017 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00018 */
00019
00020 #include "batt_FlashStorage.h"
00021
00022 static const uint32_t pageSizes[] = { 8, 16, 32, 64, 128, 256, 512, 1024 };
00023
00024 FlashClass::FlashClass(const void *flash_addr, uint32_t size) :
00025 PAGE_SIZE(pageSizes[NVMCTRL->PARAM.bit.PSZ]),
00026 PAGES(NVMCTRL->PARAM.bit.NVMP),
00027 MAX_FLASH(PAGE_SIZE * PAGES),
00028 #if defined(__SAMD51__)
00029 ROW_SIZE(MAX_FLASH / 64),
00030 #else
00031 ROW_SIZE(PAGE_SIZE * 4),
00032 #endif
00033 flash_address((volatile void *)flash_addr),
00034 flash_size(size)
00035 {
00036 }
00037
00038 static inline uint32_t read_unaligned_uint32(const void *data)
00039 {
00040 union {
00041     uint32_t u32;
00042     uint8_t u8[4];
00043 } res;
00044 const uint8_t *d = (const uint8_t *)data;
00045 res.u8[0] = d[0];
00046 res.u8[1] = d[1];
00047 res.u8[2] = d[2];
00048 res.u8[3] = d[3];
00049 return res.u32;
00050 }
00051
00052 #if defined(__SAMD51__)
00053 // Invalidate all CMCC cache entries if CMCC cache is enabled.
00054 static void invalidate_CMCC_cache()
00055 {
00056     if (CMCC->SR.bit.CSTS) {
00057         CMCC->CTRL.bit.CEN = 0;
00058         while (CMCC->SR.bit.CSTS) {}
00059         CMCC->MAINT0.bit.INVALL = 1;
00060         CMCC->CTRL.bit.CEN = 1;
00061     }
00062 }
00063 #endif
00064
00065 void FlashClass::write(const volatile void *flash_ptr, const void *data, uint32_t size)
00066 {
00067     // Calculate data boundaries
00068     size = (size + 3) / 4;
00069     volatile uint32_t *dst_addr = (volatile uint32_t *)flash_ptr;
00070     const uint8_t *src_addr = (uint8_t *)data;
00071
00072     // Disable automatic page write
00073 #if defined(__SAMD51__)
00074     NVMCTRL->CTRLA.bit.WMODE = 0;
00075     while (NVMCTRL->STATUS.bit.READY == 0) {}
00076     // Disable NVMCTRL cache while writing, per SAMD51 errata.
00077     bool original_CACHEDISO = NVMCTRL->CTRLA.bit.CACHEDISO;
00078     bool original_CACHEDIS1 = NVMCTRL->CTRLA.bit.CACHEDIS1;
00079     NVMCTRL->CTRLA.bit.CACHEDISO = true;
00080     NVMCTRL->CTRLA.bit.CACHEDIS1 = true;
00081 #else
00082     NVMCTRL->CTRLB.bit.MANW = 1;
00083 #endif
```

```

00084     // Do writes in pages
00085     while (size) {
00086         // Execute "PBC" Page Buffer Clear
00087         #if defined(__SAMD51__)
00088             NVMCTRL->CTRLB.reg = NVMCTRL_CTRLB_CMDEX_KEY | NVMCTRL_CTRLB_CMD_PBC;
00089             while ((NVMCTRL->INTFLAG.bit.DONE == 0) { }
00090         #else
00091             NVMCTRL->CTRLA.reg = NVMCTRL_CTRLA_CMDEX_KEY | NVMCTRL_CTRLA_CMD_PBC;
00092             while ((NVMCTRL->INTFLAG.bit.READY == 0) { }
00093         #endif
00094     }
00095     // Fill page buffer
00096     uint32_t i;
00097     for (i=0; i<(PAGE_SIZE/4) && size; i++) {
00098         *dst_addr = read_unaligned_uint32(src_addr);
00099         src_addr += 4;
00100         dst_addr++;
00101         size--;
00102     }
00103 }
00104     // Execute "WP" Write Page
00105 #if defined(__SAMD51__)
00106     NVMCTRL->CTRLB.reg = NVMCTRL_CTRLB_CMDEX_KEY | NVMCTRL_CTRLB_CMD_WP;
00107     while ((NVMCTRL->INTFLAG.bit.DONE == 0) { }
00108     invalidate_CMCC_cache();
00109     // Restore original NVMCTRL cache settings.
00110     NVMCTRL->CTRLA.bit.CACHEDIS0 = original_CACHEDIS0;
00111     NVMCTRL->CTRLA.bit.CACHEDIS1 = original_CACHEDIS1;
00112 #else
00113     NVMCTRL->CTRLA.reg = NVMCTRL_CTRLA_CMDEX_KEY | NVMCTRL_CTRLA_CMD_WP;
00114     while ((NVMCTRL->INTFLAG.bit.READY == 0) { }
00115 #endif
00116 }
00117 }
00118 }
00119
00120 void FlashClass::erase(const volatile void *flash_ptr, uint32_t size)
00121 {
00122     const uint8_t *ptr = (const uint8_t *)flash_ptr;
00123     while (size > ROW_SIZE) {
00124         erase(ptr);
00125         ptr += ROW_SIZE;
00126         size -= ROW_SIZE;
00127     }
00128     erase(ptr);
00129 }
00130
00131 void FlashClass::erase(const volatile void *flash_ptr)
00132 {
00133 #if defined(__SAMD51__)
00134     NVMCTRL->ADDR.reg = ((uint32_t)flash_ptr);
00135     NVMCTRL->CTRLB.reg = NVMCTRL_CTRLB_CMDEX_KEY | NVMCTRL_CTRLB_CMD_EB;
00136     while (!NVMCTRL->INTFLAG.bit.DONE) { }
00137     invalidate_CMCC_cache();
00138 #else
00139     NVMCTRL->ADDR.reg = ((uint32_t)flash_ptr) / 2;
00140     NVMCTRL->CTRLA.reg = NVMCTRL_CTRLA_CMDEX_KEY | NVMCTRL_CTRLA_CMD_ER;
00141     while (!NVMCTRL->INTFLAG.bit.READY) { }
00142 #endif
00143 }
00144
00145 void FlashClass::read(const volatile void *flash_ptr, void *data, uint32_t size)
00146 {
00147     memcpy(data, (const void *)flash_ptr, size);
00148 }
00149

```

## 15.123 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0012-batt\_FlashStorage/src/batt\_FlashStorage.h File Reference

```
#include <Arduino.h>
```

### Classes

- class [FlashClass](#)

- class `FlashStorageClass< T >`

## Macros

- `#define PPCAT_NX(A, B) A ## B`
- `#define PPCAT(A, B) PPCAT_NX(A, B)`
- `#define Flash(name, size)`
- `#define FlashStorage(name, T)`

### 15.123.1 Macro Definition Documentation

#### 15.123.1.1 Flash

```
#define Flash(
    name,
    size )
```

**Value:**

```
__attribute__((__aligned__(256)) \
static const uint8_t PPCAT(_data, name)[(size+255)/256*256] = { }; \
FlashClass name(PPCAT(_data, name), size);
```

Definition at line 39 of file [batt\\_FlashStorage.h](#).

#### 15.123.1.2 FlashStorage

```
#define FlashStorage(
    name,
    T )
```

**Value:**

```
__attribute__((__aligned__(256)) \
static const uint8_t PPCAT(_data, name)[(sizeof(T)+255)/256*256] = { }; \
FlashStorageClass<T> name(PPCAT(_data, name));
```

Definition at line 44 of file [batt\\_FlashStorage.h](#).

#### 15.123.1.3 PPCAT

```
#define PPCAT(
    A,
    B ) PPCAT_NX(A, B)
```

Definition at line 26 of file [batt\\_FlashStorage.h](#).

#### 15.123.1.4 PPCAT\_NX

```
#define PPCAT_NX(
    A,
    B ) A ## B
```

Definition at line 25 of file [batt\\_FlashStorage.h](#).

## 15.124 batt\_FlashStorage.h

[Go to the documentation of this file.](#)

```
00001 /*
00002 Copyright (c) 2015 Arduino LLC. All right reserved.
00003 Written by Cristian Maglie
00004
00005 This library is free software; you can redistribute it and/or
00006 modify it under the terms of the GNU Lesser General Public
00007 License as published by the Free Software Foundation; either
```

```

00008 version 2.1 of the License, or (at your option) any later version.
00009
0010 This library is distributed in the hope that it will be useful,
0011 but WITHOUT ANY WARRANTY; without even the implied warranty of
0012 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
0013 See the GNU Lesser General Public License for more details.
0014
0015 You should have received a copy of the GNU Lesser General Public
0016 License along with this library; if not, write to the Free Software
0017 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
0018 */
0019
0020 #pragma once
0021
0022 #include <Arduino.h>
0023
0024 // Concatenate after macro expansion
0025 #define PPCAT_NX(A, B) A ## B
0026 #define PPCAT(A, B) PPCAT_NX(A, B)
0027
0028 #if defined(__SAMD51__)
0029     #define Flash(name, size) \
0030         __attribute__((__aligned__(8192))) \
0031         static const uint8_t PPCAT(_data, name)[(size+8191)/8192*8192] = { }; \
0032         FlashClass name(PPCAT(_data, name), size);
0033
0034 #define FlashStorage(name, T) \
0035     __attribute__((__aligned__(8192))) \
0036     static const uint8_t PPCAT(_data, name)[(sizeof(T)+8191)/8192*8192] = { }; \
0037     FlashStorageClass<T> name(PPCAT(_data, name));
0038 #else
0039     #define Flash(name, size) \
0040         __attribute__((__aligned__(256))) \
0041         static const uint8_t PPCAT(_data, name)[(size+255)/256*256] = { }; \
0042         FlashClass name(PPCAT(_data, name), size);
0043
0044 #define FlashStorage(name, T) \
0045     __attribute__((__aligned__(256))) \
0046     static const uint8_t PPCAT(_data, name)[(sizeof(T)+255)/256*256] = { }; \
0047     FlashStorageClass<T> name(PPCAT(_data, name));
0048 #endif
0049
0050 class FlashClass {
0051 public:
0052     FlashClass(const void *flash_addr = NULL, uint32_t size = 0);
0053
0054     void write(const void *data) { write(flash_address, data, flash_size); }
0055     void erase() { erase(flash_address, flash_size); }
0056     void read(void *data) { read(flash_address, data, flash_size); }
0057
0058     void write(const volatile void *flash_ptr, const void *data, uint32_t size);
0059     void erase(const volatile void *flash_ptr, uint32_t size);
0060     void read(const volatile void *flash_ptr, void *data, uint32_t size);
0061
0062 private:
0063     void erase(const volatile void *flash_ptr);
0064
0065     const uint32_t PAGE_SIZE, PAGES, MAX_FLASH, ROW_SIZE;
0066     const volatile void *flash_address;
0067     const uint32_t flash_size;
0068 };
0069
0070 template<class T>
0071 class FlashStorageClass {
0072 public:
0073     FlashStorageClass(const void *flash_addr) : flash(flash_addr, sizeof(T)) { }
0074
0075     // Write data into flash memory.
0076     // Compiler is able to optimize parameter copy.
0077     inline void write(T data) { flash.erase(); flash.write(&data); }
0078
0079     // Read data from flash into variable.
0080     inline void read(T *data) { flash.read(data); }
0081
0082     // Overloaded version of read.
0083     // Compiler is able to optimize copy-on-return.
0084     inline T read() { T data; read(&data); return data; }
0085
0086 private:
0087     FlashClass flash;
0088 };
0089

```

## 15.125 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/examples/CorrectADCResponse/CorrectADCResponse.ino File Reference

### Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/examples/CorrectADCResponse/CorrectADCResponse.ino File Reference

```
#include "batt_SAMD_AnalogCorrection.h"
```

#### Macros

- #define ADC\_GND\_PIN A1
- #define ADC\_3V3\_PIN A2
- #define ADC\_READS\_SHIFT 8
- #define ADC\_READS\_COUNT (1 << ADC\_READS\_SHIFT)
- #define ADC\_MIN\_GAIN 0x0400
- #define ADC\_UNITY\_GAIN 0x0800
- #define ADC\_MAX\_GAIN (0x1000 - 1)
- #define ADC\_RESOLUTION\_BITS 12
- #define ADC\_RANGE (1 << ADC\_RESOLUTION\_BITS)
- #define ADC\_TOP\_VALUE (ADC\_RANGE - 1)
- #define MAX\_TOP\_VALUE\_READS 10

#### Functions

- void setup ()
- void loop ()
- uint16\_t readGndLevel ()
- uint16\_t read3V3Level ()

#### 15.125.1 Macro Definition Documentation

##### 15.125.1.1 ADC\_3V3\_PIN

```
#define ADC_3V3_PIN A2
```

Definition at line 27 of file [CorrectADCResponse.ino](#).

##### 15.125.1.2 ADC\_GND\_PIN

```
#define ADC_GND_PIN A1
```

Definition at line 26 of file [CorrectADCResponse.ino](#).

##### 15.125.1.3 ADC\_MAX\_GAIN

```
#define ADC_MAX_GAIN (0x1000 - 1)
```

Definition at line 34 of file [CorrectADCResponse.ino](#).

##### 15.125.1.4 ADC\_MIN\_GAIN

```
#define ADC_MIN_GAIN 0x0400
```

Definition at line 32 of file [CorrectADCResponse.ino](#).

### 15.125.1.5 ADC\_RANGE

```
#define ADC_RANGE (1 << ADC_RESOLUTION_BITS)
```

Definition at line 36 of file [CorrectADCResponse.ino](#).

### 15.125.1.6 ADC\_READS\_COUNT

```
#define ADC_READS_COUNT (1 << ADC_READS_SHIFT)
```

Definition at line 30 of file [CorrectADCResponse.ino](#).

### 15.125.1.7 ADC\_READS\_SHIFT

```
#define ADC_READS_SHIFT 8
```

Definition at line 29 of file [CorrectADCResponse.ino](#).

### 15.125.1.8 ADC\_RESOLUTION\_BITS

```
#define ADC_RESOLUTION_BITS 12
```

Definition at line 35 of file [CorrectADCResponse.ino](#).

### 15.125.1.9 ADC\_TOP\_VALUE

```
#define ADC_TOP_VALUE (ADC_RANGE - 1)
```

Definition at line 37 of file [CorrectADCResponse.ino](#).

### 15.125.1.10 ADC\_UNITY\_GAIN

```
#define ADC_UNITY_GAIN 0x0800
```

Definition at line 33 of file [CorrectADCResponse.ino](#).

### 15.125.1.11 MAX\_TOP\_VALUE\_READS

```
#define MAX_TOP_VALUE_READS 10
```

Definition at line 39 of file [CorrectADCResponse.ino](#).

## 15.125.2 Function Documentation

### 15.125.2.1 loop()

```
void loop ()
```

Definition at line 174 of file [CorrectADCResponse.ino](#).

### 15.125.2.2 read3V3Level()

```
uint16_t read3V3Level ()
```

Definition at line 193 of file [CorrectADCResponse.ino](#).

**15.125.2.3 readGndLevel()**

```
uint16_t readGndLevel ( )
```

Definition at line 178 of file [CorrectADCResponse.ino](#).

**15.125.2.4 setup()**

```
void setup ( )
```

Definition at line 41 of file [CorrectADCResponse.ino](#).

**15.126 CorrectADCResponse.ino**

[Go to the documentation of this file.](#)

```
00001 /*
00002   This sketch easily and quickly finds the right ADC correction values for a particular Arduino ZERO
00003   board.
00004   The correction values that are found are only valid for the board where the sketch is executed.
00005   This example code is in the public domain.
00006
00007   Written 6 May 2015 by Claudio Indelicati
00008 */
00009
00010 /*
00011   How to use this sketch
00012
00013   1) Remove any connection cable, shield or jumper from your Arduino ZERO
00014   2) Connect pin A1 to the nearest GND pin using the shortest jumper possible
00015   3) Connect pin A2 to the 3.3V pin using the shortest jumper possible
00016   4) Connect the Arduino ZERO to your PC using a USB cable plugged in the USB programming port of the
00017   board
00018   5) Upload this sketch and leave the board powered on for at least one minute
00019   6) Open the Serial Monitor and press the reset button on the Arduino ZERO
00020   7) At the end of the procedure you can find logged
00021       - the offset and gain values for the board where the sketch has been just executed
00022       - the instruction line to copy/paste in the final sketch
00023
00024 #include "batt_SAMD_AnalogCorrection.h"
00025
00026 #define ADC_GND_PIN          A1
00027 #define ADC_3V3_PIN           A2
00028
00029 #define ADC_READS_SHIFT       8
00030 #define ADC_READS_COUNT        (1 << ADC_READS_SHIFT)
00031
00032 #define ADC_MIN_GAIN          0x0400
00033 #define ADC_UNITY_GAIN         0x0800
00034 #define ADC_MAX_GAIN          (0x1000 - 1)
00035 #define ADC_RESOLUTION_BITS    12
00036 #define ADC_RANGE              (1 << ADC_RESOLUTION_BITS)
00037 #define ADC_TOP_VALUE          (ADC_RANGE - 1)
00038
00039 #define MAX_TOP_VALUE_READS   10
00040
00041 void setup()
00042 {
00043   Serial.begin(9600);
00044
00045   Serial.println("\r\nCalibrating ADC with factory values");
00046
00047   analogReadResolution(ADC_RESOLUTION_BITS);
00048
00049   Serial.println("\r\nReading GND and 3.3V ADC levels");
00050   Serial.print("  ");
00051   readGndLevel();
00052   Serial.print("  ");
00053   read3V3Level();
00054
00055   int offsetCorrectionValue = 0;
00056   uint16_t gainCorrectionValue = ADC_UNITY_GAIN;
00057
00058   Serial.print("\r\nOffset correction (@gain = ");
00059   Serial.print(gainCorrectionValue);
00060   Serial.println(" (unity gain))");
00061
00062 // Set default correction values and enable correction
00063 analogReadCorrection(offsetCorrectionValue, gainCorrectionValue);
00064
```

```

00065  for (int offset = 0; offset < (int)(ADC_OFFSETCORR_MASK >> 1); ++offset)
00066  {
00067      analogReadCorrection(offset, gainCorrectionValue);
00068
00069      Serial.print("    Offset = ");
00070      Serial.print(offset);
00071      Serial.print(", ");
00072
00073      if (readGndLevel() == 0)
00074      {
00075          offsetCorrectionValue = offset;
00076          break;
00077      }
00078  }
00079
00080  Serial.println("\r\nGain correction");
00081
00082  uint8_t topValueReadsCount = 0U;
00083
00084  uint16_t minGain = 0U,
00085      maxGain = 0U;
00086
00087  analogReadCorrection(offsetCorrectionValue, gainCorrectionValue);
00088  Serial.print("    Gain = ");
00089  Serial.print(gainCorrectionValue);
00090  Serial.print(", ");
00091  uint16_t highLevelRead = read3V3Level();
00092
00093  if (highLevelRead < ADC_TOP_VALUE)
00094  {
00095      for (uint16_t gain = ADC_UNITY_GAIN + 1; gain <= ADC_MAX_GAIN; ++gain)
00096      {
00097          analogReadCorrection(offsetCorrectionValue, gain);
00098
00099          Serial.print("    Gain = ");
00100         Serial.print(gain);
00101         Serial.print(", ");
00102         highLevelRead = read3V3Level();
00103
00104         if (highLevelRead == ADC_TOP_VALUE)
00105         {
00106             if (minGain == 0U)
00107                 minGain = gain;
00108
00109             if (++topValueReadsCount >= MAX_TOP_VALUE_READS)
00110             {
00111                 maxGain = minGain;
00112                 break;
00113             }
00114
00115             maxGain = gain;
00116         }
00117
00118         if (highLevelRead > ADC_TOP_VALUE)
00119             break;
00120     }
00121
00122  else if (highLevelRead >= ADC_TOP_VALUE)
00123  {
00124      if (highLevelRead == ADC_TOP_VALUE)
00125          maxGain = ADC_UNITY_GAIN;
00126
00127      for (uint16_t gain = ADC_UNITY_GAIN - 1; gain >= ADC_MIN_GAIN; --gain)
00128      {
00129          analogReadCorrection(offsetCorrectionValue, gain);
00130
00131          Serial.print("    Gain = ");
00132          Serial.print(gain);
00133          Serial.print(", ");
00134          highLevelRead = read3V3Level();
00135
00136          if (highLevelRead == ADC_TOP_VALUE)
00137          {
00138              if (maxGain == 0U)
00139                  maxGain = gain;
00140
00141              minGain = gain;
00142          }
00143
00144          if (highLevelRead < ADC_TOP_VALUE)
00145              break;
00146      }
00147  }
00148
00149  gainCorrectionValue = (minGain + maxGain) >> 1;
00150  analogReadCorrection(offsetCorrectionValue, gainCorrectionValue);

```

```

00152
00153     Serial.println("\r\nReadings after corrections");
00154     Serial.print("    ");
00155     readGndLevel();
00156     Serial.print("    ");
00157     read3V3Level();
00158
00159     Serial.println("\r\n=====");
00160     Serial.println("\r\nCorrection values:");
00161     Serial.print("    Offset = ");
00162     Serial.println(offsetCorrectionValue);
00163     Serial.print("    Gain = ");
00164     Serial.println(gainCorrectionValue);
00165     Serial.println("\r\nAdd the next line to your sketch:");
00166     Serial.print("    analogReadCorrection()");
00167     Serial.print(offsetCorrectionValue);
00168     Serial.print(", ");
00169     Serial.print(gainCorrectionValue);
00170     Serial.println(");");
00171     Serial.println("\r\n=====");
00172 }
00173
00174 void loop()
00175 {
00176 }
00177
00178 uint16_t readGndLevel()
00179 {
00180     uint32_t readAccumulator = 0;
00181
00182     for (int i = 0; i < ADC_READS_COUNT; ++i)
00183         readAccumulator += analogRead(ADC_GND_PIN);
00184
00185     uint16_t readValue = readAccumulator >> ADC_READS_SHIFT;
00186
00187     Serial.print("ADC(GND) = ");
00188     Serial.println(readValue);
00189
00190     return readValue;
00191 }
00192
00193 uint16_t read3V3Level()
00194 {
00195     uint32_t readAccumulator = 0;
00196
00197     for (int i = 0; i < ADC_READS_COUNT; ++i)
00198         readAccumulator += analogRead(ADC_3V3_PIN);
00199
00200     uint16_t readValue = readAccumulator >> ADC_READS_SHIFT;
00201
00202     if (readValue < (ADC_RANGE >> 1))
00203         readValue += ADC_RANGE;
00204
00205     Serial.print("ADC(3.3V) = ");
00206     Serial.println(readValue);
00207
00208     return readValue;
00209 }
00210

```

## 15.127 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/src/batt\_SAMD\_AnalogCorrection.cpp File Reference

```
#include "batt_SAMD_AnalogCorrection.h"
```

### Functions

- void **analogReadCorrection** (int offset, uint16\_t gain)

#### 15.127.1 Function Documentation

### 15.127.1.1 analogReadCorrection()

```
void analogReadCorrection (
    int offset,
    uint16_t gain )
```

Definition at line 21 of file [batt\\_SAMD\\_AnalogCorrection.cpp](#).

## 15.128 batt\_SAMD\_AnalogCorrection.cpp

[Go to the documentation of this file.](#)

```
00001 /*
00002 Copyright (c) 2015 Arduino LLC. All right reserved.
00003
00004 This library is free software; you can redistribute it and/or
00005 modify it under the terms of the GNU Lesser General Public
00006 License as published by the Free Software Foundation; either
00007 version 2.1 of the License, or (at your option) any later version.
00008
00009 This library is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00012 See the GNU Lesser General Public License for more details.
00013
00014 You should have received a copy of the GNU Lesser General Public
00015 License along with this library; if not, write to the Free Software
00016 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00017 */
00018
00019 #include "batt_SAMD_AnalogCorrection.h"
00020
00021 void analogReadCorrection (int offset, uint16_t gain)
00022 {
00023     // Set correction values
00024     ADC->OFFSETCORR.reg = ADC_OFFSETCORR_OFFSETCORR(offset);
00025     ADC->GAINCORR.reg = ADC_GAINCORR_GAINCORR(gain);
00026
00027     // Enable digital correction logic
00028     ADC->CTRLB.bit.CORREN = 1;
00029     while (ADC->STATUS.bit.SYNCBUSY);
00030 }
00031
```

## 15.129 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0013-SAMD\_AnalogCorrection/src/batt\_SAMD\_AnalogCorrection.h File Reference

```
#include <Arduino.h>
```

### Functions

- void [analogReadCorrection](#) (int offset, uint16\_t gain)

### 15.129.1 Function Documentation

#### 15.129.1.1 analogReadCorrection()

```
void analogReadCorrection (
    int offset,
    uint16_t gain )
```

Definition at line 21 of file [batt\\_SAMD\\_AnalogCorrection.cpp](#).

## 15.130 batt\_SAMD\_AnalogCorrection.h

[Go to the documentation of this file.](#)

```

00001 /*
00002 Copyright (c) 2015 Arduino LLC. All right reserved.
00003
00004 This library is free software; you can redistribute it and/or
00005 modify it under the terms of the GNU Lesser General Public
00006 License as published by the Free Software Foundation; either
00007 version 2.1 of the License, or (at your option) any later version.
00008
00009 This library is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00012 See the GNU Lesser General Public License for more details.
00013
00014 You should have received a copy of the GNU Lesser General Public
00015 License along with this library; if not, write to the Free Software
00016 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00017 */
00018
00019 #pragma once
00020
00021 #include <Arduino.h>
00022
00023 void analogReadCorrection (int offset, uint16_t gain);
00024

```

## 15.131 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0014-MilliTimer/MilliTimer.cpp File Reference

```
#include "MilliTimer.h"
```

## 15.132 MilliTimer.cpp

[Go to the documentation of this file.](#)

```

00001 #include "MilliTimer.h"
00002
00003
00004 byte MilliTimer::poll(word ms)
00005 {
00006     byte ready = 0;
00007     if (armed)
00008     {
00009         word remain = next - millis();
00010         // since remain is unsigned, it will overflow to large values when
00011         // the timeout is reached, so this test works as long as poll() is
00012         // called no later than 5535 millisecs after the timer has expired
00013         if (remain <= 60000)
00014             return 0;
00015         // return a value between 1 and 255, being msecst1 past expiration
00016         // note: the actual return value is only reliable if poll() is
00017         // called no later than 255 millisecs after the timer has expired
00018         ready = -remain;
00019     }
00020     set(ms);
00021     return ready; // If check must be with != 0 (done) and == 0 not done
00022 }
00023
00024 word MilliTimer::remaining() const
00025 {
00026     word remain = armed ? next - millis() : 0;
00027     return remain <= 60000 ? remain : 0;
00028 }
00029
00030 void MilliTimer::set(word ms)
00031 {
00032     armed = ms != 0;
00033     if (armed)
00034         next = millis() + ms - 1;
00035 }

```

## 15.133 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0014-MilliTimer/MilliTimer.h File Reference

```
#include <Arduino.h>
```

### Classes

- class [MilliTimer](#)

### Macros

- [#define C\\_TIMER\\_NOT\\_EXPIRED 0](#)

#### 15.133.1 Macro Definition Documentation

##### 15.133.1.1 C\_TIMER\_NOT\_EXPIRED

```
#define C_TIMER_NOT_EXPIRED 0
Definition at line 5 of file MilliTimer.h.
```

## 15.134 MilliTimer.h

[Go to the documentation of this file.](#)

```
00001 #ifndef MilliTimer_h
00002 #define MilliTimer_h
00003 #endif
00004 #include <Arduino.h> // Arduino 1.0
00005 #define C_TIMER_NOT_EXPIRED 0
00006 class MilliTimer
00007 {
00008     word next;
00009     byte armed;
00010
00011 public:
00012     MilliTimer() : armed(0) {}
00013
00016     byte poll(word ms = 0);
00018     word remaining() const;
00020     byte idle() const { return !armed; }
00023     void set(word ms);
00024 };
```

## 15.135 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/examples/Epoch/ Epoch.ino File Reference

```
#include <RTCZero.h>
```

### Functions

- void [setup \(\)](#)
- void [loop \(\)](#)
- void [print2digits \(int number\)](#)

## Variables

- RTCZero rtc

## 15.135.1 Function Documentation

### 15.135.1.1 loop()

```
void loop ()  
Definition at line 27 of file Epoch.ino.
```

### 15.135.1.2 print2digits()

```
void print2digits (  
    int number )  
Definition at line 54 of file Epoch.ino.
```

### 15.135.1.3 setup()

```
void setup ()  
Definition at line 19 of file Epoch.ino.
```

## 15.135.2 Variable Documentation

### 15.135.2.1 rtc

```
RTCZero rtc  
Definition at line 17 of file Epoch.ino.
```

## 15.136 Epoch.ino

[Go to the documentation of this file.](#)

```
00001 /*  
00002     Epoch time example for Arduino Zero and MKR1000  
00003  
00004     Demonstrates how to set time using epoch for the Arduino Zero and MKR1000  
00005  
00006     This example code is in the public domain  
00007  
00008     created by Sandeep Mistry <s.mistry@arduino.cc>  
00009     31 Dec 2015  
00010     modified  
00011     18 Feb 2016  
00012 */  
00013  
00014 #include <RTCZero.h>  
00015  
00016 /* Create an rtc object */  
00017 RTCZero rtc;  
00018  
00019 void setup() {  
00020     Serial.begin(9600);  
00021  
00022     rtc.begin(); // initialize RTC  
00023  
00024     rtc.setEpoch(1451606400); // Jan 1, 2016  
00025 }  
00026  
00027 void loop() {  
00028     Serial.print("Unix time = ");  
00029     Serial.println(rtc.getEpoch());  
00030  
00031     Serial.print("Seconds since Jan 1 2000 = ");
```

```

00032     Serial.println(rtc.getY2kEpoch());
00033
00034     // Print date...
00035     Serial.print(rtc.getDay());
00036     Serial.print("/");
00037     Serial.print(rtc.getMonth());
00038     Serial.print("/");
00039     Serial.print(rtc.getYear());
00040     Serial.print("\t");
00041
00042     // ...and time
00043     print2digits(rtc.getHours());
00044     Serial.print(":");
00045     print2digits(rtc.getMinutes());
00046     Serial.print(":");
00047     print2digits(rtc.getSeconds());
00048
00049     Serial.println();
00050
00051     delay(1000);
00052 }
00053
00054 void print2digits(int number) {
00055     if (number < 10) {
00056         Serial.print("0");
00057     }
00058     Serial.print(number);
00059 }
00060

```

## 15.137 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/examples/SimpleRTC/← SimpleRTC.ino File Reference

```
#include <RTCZero.h>
```

### Functions

- void **setup** ()
- void **loop** ()
- void **print2digits** (int number)

### Variables

- RTCZero **rtc**
- const byte **seconds** = 0
- const byte **minutes** = 0
- const byte **hours** = 16
- const byte **day** = 15
- const byte **month** = 6
- const byte **year** = 15

#### 15.137.1 Function Documentation

##### 15.137.1.1 **loop()**

```
void loop ( )
```

Definition at line 54 of file [SimpleRTC.ino](#).

### 15.137.1.2 `print2digits()`

```
void print2digits (
    int number )
Definition at line 78 of file SimpleRTC.ino.
```

### 15.137.1.3 `setup()`

```
void setup ( )
Definition at line 33 of file SimpleRTC.ino.
```

## 15.137.2 Variable Documentation

### 15.137.2.1 `day`

```
const byte day = 15
Definition at line 29 of file SimpleRTC.ino.
```

### 15.137.2.2 `hours`

```
const byte hours = 16
Definition at line 26 of file SimpleRTC.ino.
```

### 15.137.2.3 `minutes`

```
const byte minutes = 0
Definition at line 25 of file SimpleRTC.ino.
```

### 15.137.2.4 `month`

```
const byte month = 6
Definition at line 30 of file SimpleRTC.ino.
```

### 15.137.2.5 `rtc`

```
RTCZero rtc
Definition at line 21 of file SimpleRTC.ino.
```

### 15.137.2.6 `seconds`

```
const byte seconds = 0
Definition at line 24 of file SimpleRTC.ino.
```

### 15.137.2.7 `year`

```
const byte year = 15
Definition at line 31 of file SimpleRTC.ino.
```

## 15.138 SimpleRTC.ino

[Go to the documentation of this file.](#)

```
00001 /*
00002  Simple RTC for Arduino Zero and MKR1000
00003
00004  Demonstrates the use of the RTC library for the Arduino Zero and MKR1000
00005
00006  This example code is in the public domain
00007
00008  http://arduino.cc/en/Tutorial/SimpleRTC
00009
00010  created by Arturo Guadalupi <a.guadalupi@arduino.cc>
00011  15 Jun 2015
00012  modified
00013  18 Feb 2016
00014  modified by Andrea Richetta <a.richetta@arduino.cc>
00015  24 Aug 2016
00016 */
00017
00018 #include <RTCZero.h>
00019
00020 /* Create an rtc object */
00021 RTCZero rtc;
00022
00023 /* Change these values to set the current initial time */
00024 const byte seconds = 0;
00025 const byte minutes = 0;
00026 const byte hours = 16;
00027
00028 /* Change these values to set the current initial date */
00029 const byte day = 15;
00030 const byte month = 6;
00031 const byte year = 15;
00032
00033 void setup()
00034 {
00035     Serial.begin(9600);
00036
00037     rtc.begin(); // initialize RTC
00038
00039     // Set the time
00040     rtc.setHours(hours);
00041     rtc.setMinutes(minutes);
00042     rtc.setSeconds(seconds);
00043
00044     // Set the date
00045     rtc.setDay(day);
00046     rtc.setMonth(month);
00047     rtc.setYear(year);
00048
00049     // you can use also
00050     //rtc.setTime(hours, minutes, seconds);
00051     //rtc.setDate(day, month, year);
00052 }
00053
00054 void loop()
00055 {
00056     // Print date...
00057     print2digits(rtc.getDay());
00058     Serial.print("/");
00059     print2digits(rtc.getMonth());
00060     Serial.print("/");
00061     print2digits(rtc.getYear());
00062     Serial.print(" ");
00063
00064     // ...and time
00065     print2digits(rtc.getHours());
00066     Serial.print(":");
00067     print2digits(rtc.getMinutes());
00068     Serial.print(":");
00069     print2digits(rtc.getSeconds());
00070
00071     Serial.println();
00072
00073     delay(1000);
00074 }
00075
00076
00077
00078 void print2digits(int number) {
00079     if (number < 10) {
00080         Serial.print("0"); // print a 0 before if the number is < than 10
00081     }
00082     Serial.print(number);
00083 }
```

## 15.139 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/examples/SimpleRTCAAlarm/SimpleRTCAAlarm.ino File Reference

```
#include <RTCZero.h>
```

### Functions

- void `setup()`
- void `loop()`
- void `alarmMatch()`

### Variables

- RTCZero `rtc`
- const byte `seconds` = 0
- const byte `minutes` = 0
- const byte `hours` = 16
- const byte `day` = 25
- const byte `month` = 9
- const byte `year` = 15

### 15.139.1 Function Documentation

#### 15.139.1.1 `alarmMatch()`

```
void alarmMatch()
```

Definition at line 52 of file [SimpleRTCAAlarm.ino](#).

#### 15.139.1.2 `loop()`

```
void loop()
```

Definition at line 47 of file [SimpleRTCAAlarm.ino](#).

#### 15.139.1.3 `setup()`

```
void setup()
```

Definition at line 32 of file [SimpleRTCAAlarm.ino](#).

### 15.139.2 Variable Documentation

#### 15.139.2.1 `day`

```
const byte day = 25
```

Definition at line 28 of file [SimpleRTCAAlarm.ino](#).

### 15.139.2.2 hours

```
const byte hours = 16
Definition at line 25 of file SimpleRTCAAlarm.ino.
```

### 15.139.2.3 minutes

```
const byte minutes = 0
Definition at line 24 of file SimpleRTCAAlarm.ino.
```

### 15.139.2.4 month

```
const byte month = 9
Definition at line 29 of file SimpleRTCAAlarm.ino.
```

### 15.139.2.5 rtc

```
RTCZero rtc
Definition at line 20 of file SimpleRTCAAlarm.ino.
```

### 15.139.2.6 seconds

```
const byte seconds = 0
Definition at line 23 of file SimpleRTCAAlarm.ino.
```

### 15.139.2.7 year

```
const byte year = 15
Definition at line 30 of file SimpleRTCAAlarm.ino.
```

## 15.140 SimpleRTCAAlarm.ino

[Go to the documentation of this file.](#)

```
00001 /*
00002   Simple RTC Alarm for Arduino Zero and MKR1000
00003
00004   Demonstrates how to set an RTC alarm for the Arduino Zero and MKR1000
00005
00006   This example code is in the public domain
00007
00008   http://arduino.cc/en/Tutorial/SimpleRTCAalarm
00009
00010  created by Arturo Guadalupi <a.guadalupi@arduino.cc>
00011  25 Sept 2015
00012
00013  modified
00014  21 Oct 2015
00015 */
00016
00017 #include <RTCZero.h>
00018
00019 /* Create an rtc object */
00020 RTCZero rtc;
00021
00022 /* Change these values to set the current initial time */
00023 const byte seconds = 0;
00024 const byte minutes = 0;
00025 const byte hours = 16;
00026
00027 /* Change these values to set the current initial date */
00028 const byte day = 25;
00029 const byte month = 9;
00030 const byte year = 15;
00031
```

```
00032 void setup()
00033 {
00034     Serial.begin(9600);
00035
00036     rtc.begin(); // initialize RTC 24H format
00037
00038     rtc.setTime(hours, minutes, seconds);
00039     rtc.setDate(day, month, year);
00040
00041     rtc.setAlarmTime(16, 0, 10);
00042     rtc.enableAlarm(rtc.MATCH_HHMMSS);
00043
00044     rtc.attachInterrupt(alarmMatch);
00045 }
00046
00047 void loop()
00048 {
00049
00050 }
00051
00052 void alarmMatch()
00053 {
00054     Serial.println("Alarm Match!");
00055 }
```

## 15.141 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/examples/Sleep<sub>↔</sub> RTCAuto/SleepRTCAuto.ino File Reference

```
#include <RTCZero.h>
```

### Functions

- void **setup** ()
- void **loop** ()
- void **alarmMatch** ()

### Variables

- RTCZero **rtc**
- const byte **seconds** = 0
- const byte **minutes** = 00
- const byte **hours** = 17
- const byte **day** = 17
- const byte **month** = 11
- const byte **year** = 15

#### 15.141.1 Function Documentation

##### 15.141.1.1 **alarmMatch()**

```
void alarmMatch ( )
```

Definition at line 59 of file [SleepRTCAuto.ino](#).

##### 15.141.1.2 **loop()**

```
void loop ( )
```

Definition at line 54 of file [SleepRTCAuto.ino](#).

**15.141.1.3 setup()**

```
void setup ( )  
Definition at line 36 of file SleepRTCAlarm.ino.
```

**15.141.2 Variable Documentation****15.141.2.1 day**

```
const byte day = 17  
Definition at line 32 of file SleepRTCAlarm.ino.
```

**15.141.2.2 hours**

```
const byte hours = 17  
Definition at line 29 of file SleepRTCAlarm.ino.
```

**15.141.2.3 minutes**

```
const byte minutes = 00  
Definition at line 28 of file SleepRTCAlarm.ino.
```

**15.141.2.4 month**

```
const byte month = 11  
Definition at line 33 of file SleepRTCAlarm.ino.
```

**15.141.2.5 rtc**

```
RTCZero rtc  
Definition at line 24 of file SleepRTCAlarm.ino.
```

**15.141.2.6 seconds**

```
const byte seconds = 0  
Definition at line 27 of file SleepRTCAlarm.ino.
```

**15.141.2.7 year**

```
const byte year = 15  
Definition at line 34 of file SleepRTCAlarm.ino.
```

**15.142 SleepRTCAlarm.ino**

[Go to the documentation of this file.](#)

```
00001 /*  
00002   Sleep RTC Alarm for Arduino Zero  
00003  
00004   Demonstrates the use an alarm to wake up an Arduino zero from Standby mode  
00005  
00006   This example code is in the public domain  
00007  
00008   http://arduino.cc/en/Tutorial/SleepRTCAlarm  
00009
```

```
00010  created by Arturo Guadalupe
00011  17 Nov 2015
00012  modified
00013  01 Mar 2016
00014
00015  NOTE:
00016  If you use this sketch with a MKR1000 you will see no output on the serial monitor.
00017  This happens because the USB clock is stopped so if the USB connection is stopped too.
00018  **To see again the USB port you have to double tap on the reset button!**
00019 */
00020
00021 #include <RTCZero.h>
00022
00023 /* Create an rtc object */
00024 RTCZero rtc;
00025
00026 /* Change these values to set the current initial time */
00027 const byte seconds = 0;
00028 const byte minutes = 00;
00029 const byte hours = 17;
00030
00031 /* Change these values to set the current initial date */
00032 const byte day = 17;
00033 const byte month = 11;
00034 const byte year = 15;
00035
00036 void setup()
00037 {
00038     pinMode(LED_BUILTIN, OUTPUT);
00039     digitalWrite(LED_BUILTIN, LOW);
00040
00041     rtc.begin();
00042
00043     rtc.setTime(hours, minutes, seconds);
00044     rtc.setDate(day, month, year);
00045
00046     rtc.setAlarmTime(17, 00, 10);
00047     rtc.enableAlarm(rtc.MATCH_HHMMSS);
00048
00049     rtc.attachInterrupt(alarmMatch);
00050
00051     rtc.standbyMode();
00052 }
00053
00054 void loop()
00055 {
00056     rtc.standbyMode(); // Sleep until next alarm match
00057 }
00058
00059 void alarmMatch()
00060 {
00061     digitalWrite(LED_BUILTIN, HIGH);
00062 }
```

## 15.143 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/src/batt\_RTCZero.cpp File Reference

```
#include <time.h>
#include "batt_RTCZero.h"
```

### Macros

- #define EPOCH\_TIME\_OFF 946684800
- #define EPOCH\_TIME\_YEAR\_OFF 100
- #define DEFAULT\_YEAR 2000
- #define DEFAULT\_MONTH 1
- #define DEFAULT\_DAY 1
- #define DEFAULT\_HOUR 0
- #define DEFAULT\_MINUTE 0
- #define DEFAULT\_SECOND 0

## Functions

- void [RTC\\_Handler](#) (void)

## Variables

- [voidFuncPtr RTC\\_callBack](#) = NULL

### 15.143.1 Macro Definition Documentation

#### 15.143.1.1 DEFAULT\_DAY

```
#define DEFAULT_DAY 1
Definition at line 30 of file batt\_RTCZero.cpp.
```

#### 15.143.1.2 DEFAULT\_HOUR

```
#define DEFAULT_HOUR 0
Definition at line 31 of file batt\_RTCZero.cpp.
```

#### 15.143.1.3 DEFAULT\_MINUTE

```
#define DEFAULT_MINUTE 0
Definition at line 32 of file batt\_RTCZero.cpp.
```

#### 15.143.1.4 DEFAULT\_MONTH

```
#define DEFAULT_MONTH 1
Definition at line 29 of file batt\_RTCZero.cpp.
```

#### 15.143.1.5 DEFAULT\_SECOND

```
#define DEFAULT_SECOND 0
Definition at line 33 of file batt\_RTCZero.cpp.
```

#### 15.143.1.6 DEFAULT\_YEAR

```
#define DEFAULT_YEAR 2000
Definition at line 28 of file batt\_RTCZero.cpp.
```

#### 15.143.1.7 EPOCH\_TIME\_OFF

```
#define EPOCH_TIME_OFF 946684800
Definition at line 24 of file batt\_RTCZero.cpp.
```

#### 15.143.1.8 EPOCH\_TIME\_YEAR\_OFF

```
#define EPOCH_TIME_YEAR_OFF 100
Definition at line 25 of file batt\_RTCZero.cpp.
```

## 15.143.2 Function Documentation

### 15.143.2.1 RTC\_Handler()

```
void RTC_Handler (
    void )
```

Definition at line 109 of file [batt\\_RTCZero.cpp](#).

## 15.143.3 Variable Documentation

### 15.143.3.1 RTC\_callBack

```
voidFuncPtr RTC_callBack = NULL
```

Definition at line 35 of file [batt\\_RTCZero.cpp](#).

## 15.144 batt\_RTCZero.cpp

[Go to the documentation of this file.](#)

```
00001 /*
00002     RTC library for Arduino Zero.
00003     Copyright (c) 2015 Arduino LLC. All right reserved.
00004
00005     This library is free software; you can redistribute it and/or
00006     modify it under the terms of the GNU Lesser General Public
00007     License as published by the Free Software Foundation; either
00008     version 2.1 of the License, or (at your option) any later version.
00009
00010     This library is distributed in the hope that it will be useful,
00011     but WITHOUT ANY WARRANTY; without even the implied warranty of
00012     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
00013     Lesser General Public License for more details.
00014
00015     You should have received a copy of the GNU Lesser General Public
00016     License along with this library; if not, write to the Free Software
00017     Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00018 */
00019
00020 #include <time.h>
00021
00022 #include "batt_RTCZero.h"
00023
00024 #define EPOCH_TIME_OFF      946684800 // This is 1st January 2000, 00:00:00 in epoch time
00025 #define EPOCH_TIME_YEAR_OFF 100       // years since 1900
00026
00027 // Default date & time after reset
00028 #define DEFAULT_YEAR      2000    // 2000..2063
00029 #define DEFAULT_MONTH     1        // 1..12
00030 #define DEFAULT_DAY       1        // 1..31
00031 #define DEFAULT_HOUR      0        // 1..23
00032 #define DEFAULT_MINUTE    0        // 0..59
00033 #define DEFAULT_SECOND    0        // 0..59
00034
00035 voidFuncPtr RTC_callBack = NULL;
00036
00037 RTCZero::RTCZero()
00038 {
00039     _configured = false;
00040 }
00041
00042 void RTCZero::begin(bool resetTime)
00043 {
00044     uint16_t tmp_reg = 0;
00045
00046     PM->APBAMASK.reg |= PM_APBAMASK_RTC; // turn on digital interface clock
00047     config32kOSC();
00048
00049     // If the RTC is in clock mode and the reset was
00050     // not due to POR or BOD, preserve the clock time
00051     // POR causes a reset anyway, BOD behaviour is?
00052     bool validTime = false;
00053     RTC_MODE2_CLOCK_Type oldTime;
00054
00055     if ((!resetTime) && (PM->RCAUSE.reg & (PM_RCAUSE_SYST | PM_RCAUSE_WDT | PM_RCAUSE_EXT))) {
```

```

00056     if (RTC->MODE2.CTRL.reg & RTC_MODE2_CTRL_MODE_CLOCK) {
00057         validTime = true;
00058         oldTime.reg = RTC->MODE2.CLOCK.reg;
00059     }
00060 }
00061
00062 // Setup clock GCLK2 with OSC32K divided by 32
00063 configureClock();
00064
00065 RTCdisable();
00066
00067 RTCreset();
00068
00069 tmp_reg |= RTC_MODE2_CTRL_MODE_CLOCK; // set clock operating mode
00070 tmp_reg |= RTC_MODE2_CTRL_PRESCALER_DIV1024; // set prescaler to 1024 for MODE2
00071 tmp_reg &= ~RTC_MODE2_CTRL_MATCHCLR; // disable clear on match
00072
00073 //According to the datasheet RTC_MODE2_CTRL_CLKREP = 0 for 24h
00074 tmp_reg &= ~RTC_MODE2_CTRL_CLKREP; // 24h time representation
00075
00076 RTC->MODE2.READREQ.reg &= ~RTC_READREQ_RCONT; // disable continuously mode
00077
00078 RTC->MODE2.CTRL.reg = tmp_reg;
00079 while (RTCiSyncing())
00080 ;
00081
00082 NVIC_EnableIRQ(RTC_IRQn); // enable RTC interrupt
00083 NVIC_SetPriority(RTC_IRQn, 0x00);
00084
00085 RTC->MODE2.INTENSET.reg |= RTC_MODE2_INTENSET_ALARM0; // enable alarm interrupt
00086 RTC->MODE2.Mode2Alarm[0].MASK.bit.SEL = MATCH_OFF; // default alarm match is off (disabled)
00087
00088 while (RTCiSyncing())
00089 ;
00090
00091 RTCEnable();
00092 RTCresetRemove();
00093
00094 // If desired and valid, restore the time value, else use first valid time value
00095 if (!resetTime) && (validTime) && (oldTime.reg != 0L) {
00096     RTC->MODE2.CLOCK.reg = oldTime.reg;
00097 }
00098 else {
00099     RTC->MODE2.CLOCK.reg = RTC_MODE2_CLOCK_YEAR(DEFAULT_YEAR - 2000) |
00100         RTC_MODE2_CLOCK_MONTH(DEFAULT_MONTH)
00101         | RTC_MODE2_CLOCK_DAY(DEFAULT_DAY) | RTC_MODE2_CLOCK_HOUR(DEFAULT_HOUR)
00102         | RTC_MODE2_CLOCK_MINUTE(DEFAULT_MINUTE) | RTC_MODE2_CLOCK_SECOND(DEFAULT_SECOND);
00103 }
00104 while (RTCiSyncing())
00105 ;
00106 _configured = true;
00107 }
00108
00109 void RTC_Handler(void)
00110 {
00111     if (RTC_callback != NULL) {
00112         RTC_callback();
00113     }
00114
00115     RTC->MODE2.INTFLAG.reg = RTC_MODE2_INTFLAG_ALARM0; // must clear flag at end
00116 }
00117
00118 void RTCZero::enableAlarm(Alarm_Match match)
00119 {
00120     if (_configured) {
00121         RTC->MODE2.Mode2Alarm[0].MASK.bit.SEL = match;
00122         while (RTCiSyncing())
00123     ;
00124 }
00125 }
00126
00127 void RTCZero::disableAlarm()
00128 {
00129     if (_configured) {
00130         RTC->MODE2.Mode2Alarm[0].MASK.bit.SEL = 0x00;
00131         while (RTCiSyncing())
00132     ;
00133 }
00134 }
00135
00136 void RTCZero::attachInterrupt(voidFuncPtr callback)
00137 {
00138     RTC_callback = callback;
00139 }
00140
00141 void RTCZero::detachInterrupt()

```

```
00142 {
00143     RTC_callBack = NULL;
00144 }
00145
00146 void RTCZero::standbyMode()
00147 {
00148     // Entering standby mode when connected
00149     // via the native USB port causes issues.
00150     SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
00151     __DSB();
00152     __WFI();
00153 }
00154
00155 /*
00156     * Get Functions
00157 */
00158
00159 uint8_t RTCZero::getSeconds()
00160 {
00161     RTCreadRequest();
00162     return RTC->MODE2.CLOCK.bit.SECOND;
00163 }
00164
00165 uint8_t RTCZero::getMinutes()
00166 {
00167     RTCreadRequest();
00168     return RTC->MODE2.CLOCK.bit.MINUTE;
00169 }
00170
00171 uint8_t RTCZero::getHours()
00172 {
00173     RTCreadRequest();
00174     return RTC->MODE2.CLOCK.bit.HOUR;
00175 }
00176
00177 uint8_t RTCZero::getDay()
00178 {
00179     RTCreadRequest();
00180     return RTC->MODE2.CLOCK.bit.DAY;
00181 }
00182
00183 uint8_t RTCZero::getMonth()
00184 {
00185     RTCreadRequest();
00186     return RTC->MODE2.CLOCK.bit.MONTH;
00187 }
00188
00189 uint8_t RTCZero::getYear()
00190 {
00191     RTCreadRequest();
00192     return RTC->MODE2.CLOCK.bit.YEAR;
00193 }
00194
00195 uint8_t RTCZero::getAlarmSeconds()
00196 {
00197     return RTC->MODE2.Mode2Alarm[0].ALARM.bit.SECOND;
00198 }
00199
00200 uint8_t RTCZero::getAlarmMinutes()
00201 {
00202     return RTC->MODE2.Mode2Alarm[0].ALARM.bit.MINUTE;
00203 }
00204
00205 uint8_t RTCZero::getAlarmHours()
00206 {
00207     return RTC->MODE2.Mode2Alarm[0].ALARM.bit.HOUR;
00208 }
00209
00210 uint8_t RTCZero::getAlarmDay()
00211 {
00212     return RTC->MODE2.Mode2Alarm[0].ALARM.bit.DAY;
00213 }
00214
00215 uint8_t RTCZero::getAlarmMonth()
00216 {
00217     return RTC->MODE2.Mode2Alarm[0].ALARM.bit.MONTH;
00218 }
00219
00220 uint8_t RTCZero::getAlarmYear()
00221 {
00222     return RTC->MODE2.Mode2Alarm[0].ALARM.bit.YEAR;
00223 }
00224
00225 /*
00226     * Set Functions
00227 */
00228
```

```
00229 void RTCZero::setSeconds(uint8_t seconds)
00230 {
00231     if (_configured) {
00232         RTC->MODE2.CLOCK.bit.SECOND = seconds;
00233         while (RTCiSyncing())
00234             ;
00235     }
00236 }
00237
00238 void RTCZero::setMinutes(uint8_t minutes)
00239 {
00240     if (_configured) {
00241         RTC->MODE2.CLOCK.bit.MINUTE = minutes;
00242         while (RTCiSyncing())
00243             ;
00244     }
00245 }
00246
00247 void RTCZero::setHours(uint8_t hours)
00248 {
00249     if (_configured) {
00250         RTC->MODE2.CLOCK.bit.HOUR = hours;
00251         while (RTCiSyncing())
00252             ;
00253     }
00254 }
00255
00256 void RTCZero::setTime(uint8_t hours, uint8_t minutes, uint8_t seconds)
00257 {
00258     if (_configured) {
00259         setSeconds(seconds);
00260         setMinutes(minutes);
00261         setHours(hours);
00262     }
00263 }
00264
00265 void RTCZero::setDay(uint8_t day)
00266 {
00267     if (_configured) {
00268         RTC->MODE2.CLOCK.bit.DAY = day;
00269         while (RTCiSyncing())
00270             ;
00271     }
00272 }
00273
00274 void RTCZero::setMonth(uint8_t month)
00275 {
00276     if (_configured) {
00277         RTC->MODE2.CLOCK.bit.MONTH = month;
00278         while (RTCiSyncing())
00279             ;
00280     }
00281 }
00282
00283 void RTCZero::setYear(uint8_t year)
00284 {
00285     if (_configured) {
00286         RTC->MODE2.CLOCK.bit.YEAR = year;
00287         while (RTCiSyncing())
00288             ;
00289     }
00290 }
00291
00292 void RTCZero:: setDate(uint8_t day, uint8_t month, uint8_t year)
00293 {
00294     if (_configured) {
00295         setDay(day);
00296         setMonth(month);
00297         setYear(year);
00298     }
00299 }
00300
00301 void RTCZero::setAlarmSeconds(uint8_t seconds)
00302 {
00303     if (_configured) {
00304         RTC->MODE2.Mode2Alarm[0].ALARM.bit.SECOND = seconds;
00305         while (RTCiSyncing())
00306             ;
00307     }
00308 }
00309
00310 void RTCZero::setAlarmMinutes(uint8_t minutes)
00311 {
00312     if (_configured) {
00313         RTC->MODE2.Mode2Alarm[0].ALARM.bit.MINUTE = minutes;
00314         while (RTCiSyncing())
00315             ;
```

```
00316     }
00317 }
00318
00319 void RTCZero::setAlarmHours(uint8_t hours)
00320 {
00321     if (_configured) {
00322         RTC->MODE2.Mode2Alarm[0].ALARM.bit.HOUR = hours;
00323         while (RTCiSyncing())
00324     ;
00325 }
00326 }
00327
00328 void RTCZero::setAlarmTime(uint8_t hours, uint8_t minutes, uint8_t seconds)
00329 {
00330     if (_configured) {
00331         setAlarmSeconds(seconds);
00332         setAlarmMinutes(minutes);
00333         setAlarmHours(hours);
00334     }
00335 }
00336
00337 void RTCZero::setAlarmDay(uint8_t day)
00338 {
00339     if (_configured) {
00340         RTC->MODE2.Mode2Alarm[0].ALARM.bit.DAY = day;
00341         while (RTCiSyncing())
00342     ;
00343 }
00344 }
00345
00346 void RTCZero::setAlarmMonth(uint8_t month)
00347 {
00348     if (_configured) {
00349         RTC->MODE2.Mode2Alarm[0].ALARM.bit.MONTH = month;
00350         while (RTCiSyncing())
00351     ;
00352 }
00353 }
00354
00355 void RTCZero::setAlarmYear(uint8_t year)
00356 {
00357     if (_configured) {
00358         RTC->MODE2.Mode2Alarm[0].ALARM.bit.YEAR = year;
00359         while (RTCiSyncing())
00360     ;
00361 }
00362 }
00363
00364 void RTCZero::setAlarmDate(uint8_t day, uint8_t month, uint8_t year)
00365 {
00366     if (_configured) {
00367         setAlarmDay(day);
00368         setAlarmMonth(month);
00369         setAlarmYear(year);
00370     }
00371 }
00372
00373 uint32_t RTCZero::getEpoch()
00374 {
00375     RTCreadRequest();
00376     RTC_MODE2_CLOCK_Type clockTime;
00377     clockTime.reg = RTC->MODE2.CLOCK.reg;
00378
00379     struct tm tm;
00380
00381     tm.tm_isdst = -1;
00382     tm.tm_yday = 0;
00383     tm.tm_wday = 0;
00384     tm.tm_year = clockTime.bit.YEAR + EPOCH_TIME_YEAR_OFF;
00385     tm.tm_mon = clockTime.bit.MONTH - 1;
00386     tm.tm_mday = clockTime.bit.DAY;
00387     tm.tm_hour = clockTime.bit.HOUR;
00388     tm.tm_min = clockTime.bit.MINUTE;
00389     tm.tm_sec = clockTime.bit.SECOND;
00390
00391     return mktime(&tm);
00392 }
00393
00394 uint32_t RTCZero::getY2kEpoch()
00395 {
00396     return (getEpoch() - EPOCH_TIME_OFF);
00397 }
00398
00399 void RTCZero::setAlarmEpoch(uint32_t ts)
00400 {
00401     if (_configured) {
00402         if (ts < EPOCH_TIME_OFF) {
```

```

00403     ts = EPOCH_TIME_OFF;
00404 }
00405
00406 time_t t = ts;
00407 struct tm* tmp = gmtime(&t);
00408
00409 setAlarmDate(tmp->tm_mday, tmp->tm_mon + 1, tmp->tm_year - EPOCH_TIME_YEAR_OFF);
00410 setAlarmTime(tmp->tm_hour, tmp->tm_min, tmp->tm_sec);
00411 }
00412 }
00413
00414 void RTCZero::setEpoch(uint32_t ts)
00415 {
00416     if (_configured) {
00417         if (ts < EPOCH_TIME_OFF) {
00418             ts = EPOCH_TIME_OFF;
00419         }
00420
00421         time_t t = ts;
00422         struct tm* tmp = gmtime(&t);
00423
00424         RTC_MODE2_CLOCK_Type clockTime;
00425
00426         clockTime.bit.YEAR = tmp->tm_year - EPOCH_TIME_YEAR_OFF;
00427         clockTime.bit.MONTH = tmp->tm_mon + 1;
00428         clockTime.bit.DAY = tmp->tm_mday;
00429         clockTime.bit.HOUR = tmp->tm_hour;
00430         clockTime.bit.MINUTE = tmp->tm_min;
00431         clockTime.bit.SECOND = tmp->tm_sec;
00432
00433         RTC->MODE2.CLOCK.reg = clockTime.reg;
00434
00435         while (RTCiSyncing())
00436         ;
00437     }
00438 }
00439
00440 void RTCZero::setY2kEpoch(uint32_t ts)
00441 {
00442     if (_configured) {
00443         setEpoch(ts + EPOCH_TIME_OFF);
00444     }
00445 }
00446
00447 /* Attach peripheral clock to 32k oscillator */
00448 void RTCZero::configureClock() {
00449     GCLK->GENDIV.reg = GCLK_GENDIV_ID(2)|GCLK_GENDIV_DIV(4);
00450     while (GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY)
00451     ;
00452 #ifdef CRYSTALLESS
00453     GCLK->GENCTRL.reg = (GCLK_GENCTRL_GENEN | GCLK_GENCTRL_SRC_OSCULP32K | GCLK_GENCTRL_ID(2) |
00454     GCLK_GENCTRL_DIVSEL );
00455 #else
00456     GCLK->GENCTRL.reg = (GCLK_GENCTRL_GENEN | GCLK_GENCTRL_SRC_XOSC32K | GCLK_GENCTRL_ID(2) |
00457     GCLK_GENCTRL_DIVSEL );
00458 #endif
00459     while (GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY)
00460     ;
00461     GCLK->CLKCTRL.reg = (uint32_t)((GCLK_CLKCTRL_CLKEN | GCLK_CLKCTRL_GEN_GCLK2 | (RTC_GCLK_ID <<
00462     GCLK_CLKCTRL_ID_Pos)));
00463     while (GCLK->STATUS.bit.SYNCBUSY)
00464     ;
00465
00466 /* Private Utility Functions
00467 */
00468
00469 void RTCZero::config32kOSC()
00470 {
00471 #ifndef CRYSTALLESS
00472     SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | 
00473                             SYSCTRL_XOSC32K_RUNSTDBY | 
00474                             SYSCTRL_XOSC32K_EN32K | 
00475                             SYSCTRL_XOSC32K_XTALEN | 
00476                             SYSCTRL_XOSC32K_STARTUP(6) | 
00477                             SYSCTRL_XOSC32K_ENABLE;
00478 #endif
00479 }
00480
00481 /* Synchronise the CLOCK register for reading*/
00482 inline void RTCZero::RTCreadRequest() {
00483     if (_configured) {
00484         RTC->MODE2.READREQ.reg = RTC_READREQ_RREQ;
00485         while (RTCiSyncing())
00486         ;
00487     }
00488 }

```

```
00487     }
00488 }
00489
00490 /* Wait for sync in write operations */
00491 inline bool RTCZero::RTCisSyncing()
00492 {
00493     return (RTC->MODE2.STATUS.bit.SYNCBUSY);
00494 }
00495
00496 void RTCZero::RTCdisable()
00497 {
00498     RTC->MODE2.CTRL.reg &= ~RTC_MODE2_CTRL_ENABLE; // disable RTC
00499     while (RTCisSyncing())
00500     ;
00501 }
00502
00503 void RTCZero::RTCeable()
00504 {
00505     RTC->MODE2.CTRL.reg |= RTC_MODE2_CTRL_ENABLE; // enable RTC
00506     while (RTCisSyncing())
00507     ;
00508 }
00509
00510 void RTCZero::RTCreset()
00511 {
00512     RTC->MODE2.CTRL.reg |= RTC_MODE2_CTRL_SWRST; // software reset
00513     while (RTCisSyncing())
00514     ;
00515 }
00516
00517 void RTCZero::RTCresetRemove()
00518 {
00519     RTC->MODE2.CTRL.reg &= ~RTC_MODE2_CTRL_SWRST; // software reset remove
00520     while (RTCisSyncing())
00521     ;
00522 }
```

## 15.145 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/0015-RTCZero/src/batt\_RTCZero.h File Reference

```
#include "Arduino.h"
```

### Classes

- class [RTCZero](#)

### TypeDefs

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

#### 15.145.1 TypeDef Documentation

##### 15.145.1.1 voidFuncPtr

```
typedef void(* voidFuncPtr) (void)
Definition at line 25 of file batt\_RTCZero.h.
```

## 15.146 batt\_RTCZero.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  RTC library for Arduino Zero.
00003  Copyright (c) 2015 Arduino LLC. All right reserved.
```

```

00004
00005 This library is free software; you can redistribute it and/or
00006 modify it under the terms of the GNU Lesser General Public
00007 License as published by the Free Software Foundation; either
00008 version 2.1 of the License, or (at your option) any later version.
00009
00010 This library is distributed in the hope that it will be useful,
00011 but WITHOUT ANY WARRANTY; without even the implied warranty of
00012 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
00013 Lesser General Public License for more details.
00014
00015 You should have received a copy of the GNU Lesser General Public
00016 License along with this library; if not, write to the Free Software
00017 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
00018 */
00019
00020 #ifndef RTC_ZERO_H
00021 #define RTC_ZERO_H
00022
00023 #include "Arduino.h"
00024
00025 typedef void(*voidFuncPtr)(void);
00026
00027 class RTCZero {
00028     public:
00029
00030     enum Alarm_Match: uint8_t // Should we have this enum or just use the identifiers from
00031     // /component/rtc.h ?
00032     {
00033         MATCH_OFF      = RTC_MODE2_MASK_SEL_OFF_Val,           // Never
00034         MATCH_SS       = RTC_MODE2_MASK_SEL_SS_Val,            // Every Minute
00035         MATCH_MMSS     = RTC_MODE2_MASK_SEL_MMSS_Val,          // Every Hour
00036         MATCH_HHMMSS   = RTC_MODE2_MASK_SEL_HHMMSS_Val,        // Every Day
00037         MATCH_DDHMMSS  = RTC_MODE2_MASK_SEL_DDHMMSS_Val,       // Every Month
00038         MATCH_MMDDHHMMSS = RTC_MODE2_MASK_SEL_MMDDHHMMSS_Val, // Every Year
00039         MATCH_YYMMDDHHMMSS = RTC_MODE2_MASK_SEL_YYMMDDHHMMSS_Val // Once, on a specific date and a
00040         // specific time
00041     };
00042
00043     RTCZero();
00044     void begin(bool resetTime = false);
00045
00046     void enableAlarm(Alarm_Match match);
00047     void disableAlarm();
00048
00049     void attachInterrupt(voidFuncPtr callback);
00050     void detachInterrupt();
00051
00052     /* Get Functions */
00053
00054     uint8_t getSeconds();
00055     uint8_t getMinutes();
00056     uint8_t getHours();
00057
00058     uint8_t getDay();
00059     uint8_t getMonth();
00060     uint8_t getYear();
00061
00062     uint8_t getAlarmSeconds();
00063     uint8_t getAlarmMinutes();
00064     uint8_t getAlarmHours();
00065
00066     uint8_t getAlarmDay();
00067     uint8_t getAlarmMonth();
00068     uint8_t getAlarmYear();
00069
00070     /* Set Functions */
00071
00072     void setSeconds(uint8_t seconds);
00073     void setMinutes(uint8_t minutes);
00074     void setHours(uint8_t hours);
00075     void setTime(uint8_t hours, uint8_t minutes, uint8_t seconds);
00076
00077     void setDay(uint8_t day);
00078     void setMonth(uint8_t month);
00079     void setYear(uint8_t year);
00080     void setDate(uint8_t day, uint8_t month, uint8_t year);
00081
00082     void setAlarmSeconds(uint8_t seconds);
00083     void setAlarmMinutes(uint8_t minutes);
00084     void setAlarmHours(uint8_t hours);
00085     void setAlarmTime(uint8_t hours, uint8_t minutes, uint8_t seconds);
00086
00087     void setAlarmDay(uint8_t day);
00088     void setAlarmMonth(uint8_t month);

```

```
00089 void setAlarmYear(uint8_t year);
00090 void setAlarmDate(uint8_t day, uint8_t month, uint8_t year);
00091
00092 /* Epoch Functions */
00093
00094 uint32_t getEpoch();
00095 uint32_t getY2kEpoch();
00096 void setEpoch(uint32_t ts);
00097 void setY2kEpoch(uint32_t ts);
00098 void setAlarmEpoch(uint32_t ts);
00099
00100 bool isConfigured() {
00101     return _configured;
00102 }
00103
00104 private:
00105     bool _configured;
00106
00107 void config32kOSC(void);
00108 void configureClock(void);
00109 void RTCreadRequest();
00110 bool RTCisSyncing(void);
00111 void RTCdisable();
00112 void RTCEnable();
00113 void RTCreset();
00114 void RTCresetRemove();
00115 };
00116
00117 #endif // RTC_ZERO_H
```

**15.147 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TEST/CPU/Validacion Diseño/ARDUINO/Test\_Shield\_Validation\_CPU/Test\_Shield\_Validation\_CPU.ino File Reference**

**15.148 Test\_Shield\_Validation\_CPU.ino**

[Go to the documentation of this file.](#)

**15.149 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TEST/CPU/Validacion Diseño/SAMD/PCB\_Validation\_CPU/PCB\_Validation\_CPU.ino File Reference**

**15.150 PCB\_Validation\_CPU.ino**

[Go to the documentation of this file.](#)

**15.151 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TEST/CPU/Validacion Produccion/ARDUINO/Test\_Production/Test\_Production.ino File Reference**

**Functions**

- void [setup \(\)](#)

- void `loop ()`

## Variables

- const uint16\_t `C_PIN_SW1` = 2
- const uint16\_t `C_PIN_SW2` = 3
- const uint16\_t `C_PIN_ENABLE_LDO` = 5
- const uint16\_t `C_PIN_SDA_2` = 7
- const uint16\_t `C_PIN_SCL_2` = 8
- const uint16\_t `C_PIN_ISENSE_RAW` = 10
- const uint16\_t `C_PIN_IOUT_SENSE` = A0
- const uint16\_t `C_PIN_VCC_2` = A1
- const uint16\_t `C_PIN_VCC` = A2
- const uint16\_t `C_PIN_LED_1` = 4
- const uint16\_t `C_PIN_LED_2` = 6
- const uint16\_t `C_PIN_LED_3` = 9
- char `data` = 0
- int16\_t `sample_vcc` = 0
- int16\_t `sample_vcc_2` = 0
- int16\_t `sample_iout_sense` = 0
- bool `sda_pin`
- bool `scl_pin`
- bool `test_start` = false

### 15.151.1 Detailed Description

#### Author

Javi ( [Javier@musotoku.com](mailto:Javier@musotoku.com))

#### Version

1

#### Date

2021-06-30

#### Copyright

Copyright (c) 2021

Definition in file [Test\\_Production.ino](#).

### 15.151.2 Function Documentation

#### 15.151.2.1 `loop()`

`void loop ()`

Definition at line [82](#) of file [Test\\_Production.ino](#).

#### 15.151.2.2 `setup()`

`void setup ()`

Definition at line [46](#) of file [Test\\_Production.ino](#).

## 15.151.3 Variable Documentation

---

### 15.151.3.1 C\_PIN\_ENABLE\_LDO

```
const uint16_t C_PIN_ENABLE_LDO = 5
```

Definition at line [20](#) of file [Test\\_Production.ino](#).

### 15.151.3.2 C\_PIN\_IOUT\_SENSE

```
const uint16_t C_PIN_IOUT_SENSE = A0
```

Definition at line [24](#) of file [Test\\_Production.ino](#).

### 15.151.3.3 C\_PIN\_ISENSE\_RAW

```
const uint16_t C_PIN_ISENSE_RAW = 10
```

Definition at line [23](#) of file [Test\\_Production.ino](#).

### 15.151.3.4 C\_PIN\_LED\_1

```
const uint16_t C_PIN_LED_1 = 4
```

Definition at line [27](#) of file [Test\\_Production.ino](#).

### 15.151.3.5 C\_PIN\_LED\_2

```
const uint16_t C_PIN_LED_2 = 6
```

Definition at line [28](#) of file [Test\\_Production.ino](#).

### 15.151.3.6 C\_PIN\_LED\_3

```
const uint16_t C_PIN_LED_3 = 9
```

Definition at line [29](#) of file [Test\\_Production.ino](#).

### 15.151.3.7 C\_PIN\_SCL\_2

```
const uint16_t C_PIN_SCL_2 = 8
```

Definition at line [22](#) of file [Test\\_Production.ino](#).

### 15.151.3.8 C\_PIN\_SDA\_2

```
const uint16_t C_PIN_SDA_2 = 7
```

Definition at line [21](#) of file [Test\\_Production.ino](#).

### 15.151.3.9 C\_PIN\_SW1

```
const uint16_t C_PIN_SW1 = 2
```

CONSTANTS

Definition at line [18](#) of file [Test\\_Production.ino](#).

**15.151.3.10 C\_PIN\_SW2**

```
const uint16_t C_PIN_SW2 = 3
Definition at line 19 of file Test\_Production.ino.
```

**15.151.3.11 C\_PIN\_VCC**

```
const uint16_t C_PIN_VCC = A2
Definition at line 26 of file Test\_Production.ino.
```

**15.151.3.12 C\_PIN\_VCC\_2**

```
const uint16_t C_PIN_VCC_2 = A1
Definition at line 25 of file Test\_Production.ino.
```

**15.151.3.13 data**

```
char data = 0
GLOBAL VARIABLES
Definition at line 34 of file Test\_Production.ino.
```

**15.151.3.14 sample\_iout\_sense**

```
int16_t sample_iout_sense = 0
Definition at line 37 of file Test\_Production.ino.
```

**15.151.3.15 sample\_vcc**

```
int16_t sample_vcc = 0
Definition at line 35 of file Test\_Production.ino.
```

**15.151.3.16 sample\_vcc\_2**

```
int16_t sample_vcc_2 = 0
Definition at line 36 of file Test\_Production.ino.
```

**15.151.3.17 scl\_pin**

```
bool scl_pin
Definition at line 40 of file Test\_Production.ino.
```

**15.151.3.18 sda\_pin**

```
bool sda_pin
Definition at line 39 of file Test\_Production.ino.
```

**15.151.3.19 test\_start**

```
bool test_start = false
FLAGS
Definition at line 44 of file Test\_Production.ino.
```

## 15.152 Test\_Production.ino

[Go to the documentation of this file.](#)

```

00001
00016 //      PINS
00017
00018 const uint16_t C_PIN_SW1 = 2;
00019 const uint16_t C_PIN_SW2 = 3;
00020 const uint16_t C_PIN_ENABLE_LDO = 5;
00021 const uint16_t C_PIN_SDA_2 = 7;
00022 const uint16_t C_PIN_SCL_2 = 8;
00023 const uint16_t C_PIN_ISENSE_RAW = 10;
00024 const uint16_t C_PIN_IOUT_SENSE = A0;
00025 const uint16_t C_PIN_VCC_2 = A1;
00026 const uint16_t C_PIN_VCC = A2;
00027 const uint16_t C_PIN_LED_1 = 4;
00028 const uint16_t C_PIN_LED_2 = 6;
00029 const uint16_t C_PIN_LED_3 = 9;
00034 char data = 0;
00035 int16_t sample_vcc = 0;
00036 int16_t sample_vcc_2 = 0;
00037 int16_t sample_iout_sense = 0;
00038
00039 bool sda_pin;
00040 bool scl_pin;
00044 bool test_start = false;
00045
00046 void setup()
00047 {
00048     Serial.begin(9600);
00049     pinMode(C_PIN_LED_1, OUTPUT);
00050     pinMode(C_PIN_LED_2, OUTPUT);
00051     pinMode(C_PIN_LED_3, OUTPUT);
00052     pinMode(C_PIN_ENABLE_LDO, OUTPUT);
00053     pinMode(C_PIN_ISENSE_RAW, OUTPUT);
00054     pinMode(C_PIN_SW1, OUTPUT);
00055     pinMode(C_PIN_SW2, OUTPUT);
00056     digitalWrite(C_PIN_LED_1, LOW);
00057     digitalWrite(C_PIN_LED_2, LOW);
00058     digitalWrite(C_PIN_LED_3, LOW);
00059     // while (test_start == false)
00060     // {
00061     //     if (Serial.available())
00062     //     {
00063     //         data = Serial.readString();
00064     //         data.trim();
00065     //         Serial.println(data);
00066     //         if (data == "Start test")
00067     //         {
00068     //             test_start = true;
00069     //         }
00070     //     }
00071     //     data = "";
00072     // }
00073     digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00074     digitalWrite(C_PIN_ISENSE_RAW, LOW);
00075     digitalWrite(C_PIN_SW1, HIGH);
00076     digitalWrite(C_PIN_SW2, LOW);
00077     digitalWrite(C_PIN_LED_1, LOW);
00078     digitalWrite(C_PIN_LED_2, LOW);
00079     digitalWrite(C_PIN_LED_3, LOW);
00080 }
00081
00082 void loop()
00083 {
00084     for (int i = 0; i < 8; i++)
00085     {
00086         sample_iout_sense += analogRead(C_PIN_IOUT_SENSE);
00087         sample_vcc += analogRead(C_PIN_VCC);
00088         sample_vcc_2 += analogRead(C_PIN_VCC_2);
00089     }
00090     sample_vcc_2 = sample_vcc_2 / 8;
00091     sample_vcc = sample_vcc / 8;
00092     sample_iout_sense = sample_iout_sense / 8;
00093     sda_pin = digitalRead(C_PIN_SDA_2);
00094     scl_pin = digitalRead(C_PIN_SCL_2);
00095
00096     if (Serial.available())
00097     {
00098         data = Serial.read();
00099         Serial.println(data);
00100
00102         switch (data)
00103         {
00104             // Estado base, Vbus = 4v. Iraw = OFF, Enable = ON.

```

```

00105     case '0':
00106         digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00107         digitalWrite(C_PIN_ISENSE_RAW, LOW);
00108         digitalWrite(C_PIN_SW1, HIGH);
00109         digitalWrite(C_PIN_SW2, LOW);
00110         digitalWrite(C_PIN_LED_1, LOW);
00111         digitalWrite(C_PIN_LED_2, LOW);
00112         digitalWrite(C_PIN_LED_3, LOW);
00113         break;
00114
00115     // BattLow, Vbus = 3v. Iraw = OFF, Enable = ON.
00116     case '1':
00117         digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00118         digitalWrite(C_PIN_ISENSE_RAW, LOW);
00119         digitalWrite(C_PIN_SW1, LOW);
00120         digitalWrite(C_PIN_SW2, LOW);
00121         digitalWrite(C_PIN_LED_1, HIGH);
00122         digitalWrite(C_PIN_LED_2, LOW);
00123         digitalWrite(C_PIN_LED_3, LOW);
00124         break;
00125
00126     // Charge, Vbus = 5v. Iraw = OFF, Enable = ON.
00127     case '2':
00128         digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00129         digitalWrite(C_PIN_ISENSE_RAW, LOW);
00130         digitalWrite(C_PIN_SW1, LOW);
00131         digitalWrite(C_PIN_SW2, HIGH);
00132         digitalWrite(C_PIN_LED_1, LOW);
00133         digitalWrite(C_PIN_LED_2, HIGH);
00134         digitalWrite(C_PIN_LED_3, LOW);
00135         break;
00136
00137     // Sensado Corriente, Vbus = 4v. Iraw = ON, Enable = ON.
00138     case '3':
00139         digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00140         digitalWrite(C_PIN_ISENSE_RAW, HIGH);
00141         digitalWrite(C_PIN_SW1, HIGH);
00142         digitalWrite(C_PIN_SW2, LOW);
00143         digitalWrite(C_PIN_LED_1, LOW);
00144         digitalWrite(C_PIN_LED_2, LOW);
00145         digitalWrite(C_PIN_LED_3, HIGH);
00146         break;
00147
00148     // Apagado, Vbus = 0v. Iraw = OFF, Enable = OFF.
00149     case '4':
00150         digitalWrite(C_PIN_ENABLE_LDO, LOW);
00151         digitalWrite(C_PIN_ISENSE_RAW, LOW);
00152         digitalWrite(C_PIN_SW1, HIGH);
00153         digitalWrite(C_PIN_SW2, LOW);
00154         digitalWrite(C_PIN_LED_1, HIGH);
00155         digitalWrite(C_PIN_LED_2, HIGH);
00156         digitalWrite(C_PIN_LED_3, HIGH);
00157         break;
00158
00159     default:
00160         break;
00161     }
00162 }
00163
00164 delay(1000);
00165 Serial.print("\t");
00166 Serial.print(sample_vcc);
00167 Serial.print(";");
00168 Serial.print(sample_vcc_2);
00169 Serial.print(";");
00170 Serial.print(sample_iout_sense);
00171 Serial.print(";");
00172 Serial.print(scl_pin);
00173 Serial.print(";");
00174 Serial.print(sda_pin);
00175 Serial.print(";");
00176 Serial.print(sample_iout_sense);
00177 Serial.print("\n");
00178 }
```

## 15.153 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery

Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery  
Pack/bat\_sw/TEST/CPU/Validacion Produccion/LABVIEW/CPU  
TEST/sketch\_jul14a/sketch\_jul14a.ino File Reference

### Functions

- void [setup\(\)](#)
- void [loop\(\)](#)

### Variables

- int16\_t [i](#)
- char [data](#)
- const uint16\_t [C\\_PIN\\_SW1](#) = 2
- const uint16\_t [C\\_PIN\\_SW2](#) = 3
- const uint16\_t [C\\_PIN\\_LED\\_3](#) = 4
- const uint16\_t [C\\_PIN\\_ENABLE\\_LDO](#) = 5
- const uint16\_t [C\\_PIN\\_LED\\_1](#) = 6
- const uint16\_t [C\\_PIN\\_SDA\\_2](#) = 7
- const uint16\_t [C\\_PIN\\_SCL\\_2](#) = 8
- const uint16\_t [C\\_PIN\\_LED\\_2](#) = 9
- const uint16\_t [C\\_PIN\\_ISENSE\\_RAW](#) = 10
- const uint16\_t [C\\_PIN\\_IOUT\\_SENSE](#) = A0
- const uint16\_t [C\\_PIN\\_VCC\\_2](#) = A1
- const uint16\_t [C\\_PIN\\_VCC](#) = A2
- int16\_t [iout\\_sense](#) = 0
- int16\_t [vcc](#) = 0
- int16\_t [vcc\\_2](#) = 0

### 15.153.1 Function Documentation

#### 15.153.1.1 [loop\(\)](#)

```
void loop ()
```

Definition at line [46](#) of file [sketch\\_jul14a.ino](#).

#### 15.153.1.2 [setup\(\)](#)

```
void setup ()
```

Definition at line [18](#) of file [sketch\\_jul14a.ino](#).

### 15.153.2 Variable Documentation

#### 15.153.2.1 [C\\_PIN\\_ENABLE\\_LDO](#)

```
const uint16_t C_PIN_ENABLE_LDO = 5
```

Definition at line [6](#) of file [sketch\\_jul14a.ino](#).

### 15.153.2.2 C\_PIN\_IOUT\_SENSE

```
const uint16_t C_PIN_IOUT_SENSE = A0
```

Definition at line 12 of file [sketch\\_jul14a.ino](#).

### 15.153.2.3 C\_PIN\_ISENSE\_RAW

```
const uint16_t C_PIN_ISENSE_RAW = 10
```

Definition at line 11 of file [sketch\\_jul14a.ino](#).

### 15.153.2.4 C\_PIN\_LED\_1

```
const uint16_t C_PIN_LED_1 = 6
```

Definition at line 7 of file [sketch\\_jul14a.ino](#).

### 15.153.2.5 C\_PIN\_LED\_2

```
const uint16_t C_PIN_LED_2 = 9
```

Definition at line 10 of file [sketch\\_jul14a.ino](#).

### 15.153.2.6 C\_PIN\_LED\_3

```
const uint16_t C_PIN_LED_3 = 4
```

Definition at line 5 of file [sketch\\_jul14a.ino](#).

### 15.153.2.7 C\_PIN\_SCL\_2

```
const uint16_t C_PIN_SCL_2 = 8
```

Definition at line 9 of file [sketch\\_jul14a.ino](#).

### 15.153.2.8 C\_PIN\_SDA\_2

```
const uint16_t C_PIN_SDA_2 = 7
```

Definition at line 8 of file [sketch\\_jul14a.ino](#).

### 15.153.2.9 C\_PIN\_SW1

```
const uint16_t C_PIN_SW1 = 2
```

Definition at line 3 of file [sketch\\_jul14a.ino](#).

### 15.153.2.10 C\_PIN\_SW2

```
const uint16_t C_PIN_SW2 = 3
```

Definition at line 4 of file [sketch\\_jul14a.ino](#).

### 15.153.2.11 C\_PIN\_VCC

```
const uint16_t C_PIN_VCC = A2
```

Definition at line 14 of file [sketch\\_jul14a.ino](#).

**15.153.2.12 C\_PIN\_VCC\_2**

```
const uint16_t C_PIN_VCC_2 = A1  
Definition at line 13 of file sketch\_jul14a.ino.
```

**15.153.2.13 data**

```
char data  
Definition at line 2 of file sketch\_jul14a.ino.
```

**15.153.2.14 i**

```
int16_t i  
Definition at line 1 of file sketch\_jul14a.ino.
```

**15.153.2.15 iout\_sense**

```
int16_t iout_sense = 0  
Definition at line 16 of file sketch\_jul14a.ino.
```

**15.153.2.16 vcc**

```
int16_t vcc = 0  
Definition at line 16 of file sketch\_jul14a.ino.
```

**15.153.2.17 vcc\_2**

```
int16_t vcc_2 = 0  
Definition at line 16 of file sketch\_jul14a.ino.
```

**15.154 sketch\_jul14a.ino**

[Go to the documentation of this file.](#)

```
00001 int16_t i;  
00002 char data;  
00003 const uint16_t C_PIN_SW1 = 2;  
00004 const uint16_t C_PIN_SW2 = 3;  
00005 const uint16_t C_PIN_LED_3 = 4;  
00006 const uint16_t C_PIN_ENABLE_LDO = 5;  
00007 const uint16_t C_PIN_LED_1 = 6;  
00008 const uint16_t C_PIN_SDA_2 = 7;  
00009 const uint16_t C_PIN_SCL_2 = 8;  
00010 const uint16_t C_PIN_LED_2 = 9;  
00011 const uint16_t C_PIN_ISENSE_RAW = 10;  
00012 const uint16_t C_PIN_IOUT_SENSE = A0;  
00013 const uint16_t C_PIN_VCC_2 = A1;  
00014 const uint16_t C_PIN_VCC = A2;  
00015  
00016 int16_t iout_sense = 0, vcc = 0, vcc_2 = 0;  
00017  
00018 void setup()  
00019 {  
00020     Serial.begin(9600);  
00021     analogReference(DEFAULT);  
00022     pinMode(C_PIN_LED_1, OUTPUT);  
00023     pinMode(C_PIN_LED_2, OUTPUT);  
00024     pinMode(C_PIN_LED_3, OUTPUT);  
00025     pinMode(C_PIN_ENABLE_LDO, OUTPUT);  
00026     pinMode(C_PIN_ISENSE_RAW, OUTPUT);  
00027     pinMode(C_PIN_SW1, OUTPUT);  
00028     pinMode(C_PIN_SW2, OUTPUT);  
00029     pinMode(C_PIN_IOUT_SENSE, INPUT);  
00030     pinMode(C_PIN_VCC_2, INPUT);  
00031     pinMode(C_PIN_VCC, INPUT);
```

```
00032 pinMode(LED_BUILTIN, OUTPUT);
00033 digitalWrite(C_PIN_LED_1, LOW);
00034 digitalWrite(C_PIN_LED_2, LOW);
00035 digitalWrite(C_PIN_LED_3, LOW);
00036 digitalWrite(C_PIN_ENABLE_LDO, LOW);
00037 digitalWrite(C_PIN_ISENSE_RAW, LOW);
00038 digitalWrite(C_PIN_SW1, LOW);
00039 digitalWrite(C_PIN_SW2, LOW);
00040
00041 digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00042 digitalWrite(C_PIN_ISENSE_RAW, HIGH);
00043 delay(100);
00044 }
00045
00046 void loop()
00047 {
00048
00049     digitalRead(C_PIN_SDA_2);
00050     digitalRead(C_PIN_SCL_2);
00051     iout_sense = 0;
00052     // for (int i = 0; i < 8; i++)
00053     // {
00054     //     iout_sense += analogRead(C_PIN_IOUT_SENSE);
00055     //     delay(100);
00056     // }
00057     // iout_sense = (iout_sense >> 3);
00058     // vcc = 0;
00059     // for (int i = 0; i < 8; i++)
00060     // {
00061     //     vcc += analogRead(C_PIN_VCC);
00062     //     delay(100);
00063     // }
00064     // vcc = (vcc >> 3);
00065     // vcc_2 = 0;
00066     // for (int i = 0; i < 8; i++)
00067     // {
00068     //     vcc_2 += analogRead(C_PIN_VCC_2);
00069     //     delay(100);
00070     // }
00071     // vcc_2 = (vcc_2 / 8);
00072
00073     if (Serial.available())
00074     {
00075         delay(500);
00076         digitalWrite(LED_BUILTIN, HIGH);
00077         data = Serial.read();
00078         if (data == '1')
00079         {
00080             digitalWrite(C_PIN_LED_1, HIGH);
00081             digitalWrite(C_PIN_SW1, LOW);
00082             digitalWrite(C_PIN_SW2, LOW);
00083             delay(200);
00084         }
00085         else if (data == '2')
00086         {
00087             digitalWrite(C_PIN_LED_2, HIGH);
00088             digitalWrite(C_PIN_SW1, HIGH);
00089             digitalWrite(C_PIN_SW2, LOW);
00090         }
00091         else if (data == '3')
00092         {
00093             digitalWrite(C_PIN_LED_3, HIGH);
00094             digitalWrite(C_PIN_SW1, LOW);
00095             digitalWrite(C_PIN_SW2, HIGH);
00096         }
00097         else if (data == '0')
00098         {
00099             digitalWrite(C_PIN_LED_1, LOW);
00100             digitalWrite(C_PIN_LED_2, LOW);
00101             digitalWrite(C_PIN_LED_3, LOW);
00102         }
00103
00104         Serial.print(data);
00105         Serial.print("\t");
00106         Serial.print(digitalRead(C_PIN_SCL_2));
00107         Serial.print(',');
00108         Serial.print(digitalRead(C_PIN_SDA_2));
00109         Serial.print(',');
00110         Serial.print(digitalRead(C_PIN_SCL_2));
00111         Serial.print(',');
00112         Serial.print(analogRead(C_PIN_IOUT_SENSE));
00113         Serial.print(',');
00114         Serial.print(analogRead(C_PIN_VCC));
00115         Serial.print(',');
00116         Serial.print(analogRead(C_PIN_VCC_2));
00117         Serial.print("\n");
00118         delay(500);
```

```

00119     digitalWrite(LED_BUILTIN, LOW);
00120 }
00121
00122 // -----
00123 // Test LDO
00124 // digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00125 // digitalWrite(C_PIN_ISENSE_RAW, HIGH);
00126
00127 // -----
00128 // Test Digital Ports
00129 // digitalWrite(C_PIN_LED_1, HIGH);
00130 // delay(500);
00131 // digitalWrite(C_PIN_LED_2, HIGH);
00132 // delay(500);
00133 // digitalWrite(C_PIN_LED_3, HIGH);
00134 // delay(500);
00135 // digitalWrite(C_PIN_ENABLE_LDO, HIGH);
00136 // delay(500);
00137 // digitalWrite(C_PIN_ISENSE_RAW, HIGH);
00138 // delay(500);
00139 // digitalWrite(C_PIN_SW1, HIGH);
00140 // delay(500);
00141 // digitalWrite(C_PIN_SW2, HIGH);
00142 // delay(500);
00143 // while (1)
00144 //
00145 // /* code */
00146 //
00147
00148 // digitalWrite(C_PIN_LED_1, LOW);
00149 // digitalWrite(C_PIN_LED_2, LOW);
00150 // digitalWrite(C_PIN_LED_3, LOW);
00151 // digitalWrite(C_PIN_ENABLE_LDO, LOW);
00152 // digitalWrite(C_PIN_ISENSE_RAW, LOW);
00153 // digitalWrite(C_PIN_SW1, LOW);
00154 // digitalWrite(C_PIN_SW2, LOW);
00155
00156 // -----
00157 // Test Serial Port
00158 //
00159 // if (Serial.available())
00160 //
00161 //   data = Serial.readString();
00162 //   data.trim();
00163 //   Serial.print(data);
00164 //
00165 // for (i = 0; i < 10; i++)
00166 //
00167 //   Serial.print("\t");
00168 //   Serial.print(i);
00169 //   Serial.print(";");
00170 //   Serial.print(i + 5);
00171 //   Serial.print(";");
00172 //   Serial.print(i * 10);
00173 //   Serial.print(";");
00174 //   Serial.print(i * 2);
00175 //   Serial.print("\n");
00176 //   delay(500);
00177 //
00178 }

```

## 15.155 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TEST/DCDC/Validacion Diseño/Arduino/Test\_Shield\_Validacion\_DCDC/Test\_Shield\_Validacion\_DCDC.ino File Reference

### 15.156 Test\_Shield\_Validacion\_DCDC.ino

[Go to the documentation of this file.](#)

**15.157 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TEST/DISPLAY/Validacion Diseño/Arduino/Test\_Shield\_Validacion\_Display/Test\_Shield\_Validacion\_Display.ino File Reference**

**15.158 Test\_Shield\_Validacion\_Display.ino**

[Go to the documentation of this file.](#)

**15.159 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TEST/INPUT/Validacion Diseño/Test\_Shield\_Validacion\_Input/Test\_Shield\_Validacion\_Input.ino File Reference**

**15.160 Test\_Shield\_Validacion\_Input.ino**

[Go to the documentation of this file.](#)

# Index

\_cp437  
    Adafruit\_GFX, 63  
\_dirty  
    diagnostic.h, 341  
\_eprom\_RAM  
    diagnostic.h, 341  
\_frame  
    Adafruit\_IS31FL3731, 73  
\_height  
    Adafruit\_GFX, 63  
\_i2caddr  
    Adafruit\_IS31FL3731, 74  
\_swap\_int16\_t  
    batt\_Adafruit\_GFX.cpp, 290  
    batt\_Adafruit\_IS31FL3731.cpp, 354  
\_width  
    Adafruit\_GFX, 63  
~GFXcanvas1  
    GFXcanvas1, 86  
~GFXcanvas16  
    GFXcanvas16, 90  
~GFXcanvas8  
    GFXcanvas8, 94  
  
ACCENTchar  
    bitmaps.h, 184  
Achar  
    bitmaps.h, 184  
Adafruit\_GFX, 39  
    \_cp437, 63  
    \_height, 63  
    \_width, 63  
    Adafruit\_GFX, 43  
    charBounds, 43  
    cp437, 43  
    cursor\_x, 63  
    cursor\_y, 63  
    drawBitmap, 44, 45  
    drawChar, 46  
    drawCircle, 47  
    drawCircleHelper, 47  
    drawFastHLine, 47  
    drawFastVLine, 48  
    drawGrayscaleBitmap, 48, 49  
    drawLine, 50  
    drawPixel, 50  
    drawRect, 50  
    drawRGBBitmap, 51, 52  
    drawRoundRect, 52  
    drawTriangle, 53  
  
drawXBitmap, 53  
endWrite, 54  
fillCircle, 54  
fillCircleHelper, 54  
fillRect, 55  
fillRoundRect, 55  
fillScreen, 55  
fillTriangle, 56  
getCursorX, 56  
getCursorY, 56  
getRotation, 56  
getTextBounds, 57, 58  
gfxFont, 63  
HEIGHT, 64  
height, 58  
invertDisplay, 58  
rotation, 64  
setCursor, 58  
setFont, 59  
setRotation, 59  
setTextColor, 59  
setTextSize, 60  
setTextWrap, 60  
startWrite, 60  
textbgcolor, 64  
textcolor, 64  
textsize\_x, 64  
textsize\_y, 64  
WIDTH, 64  
width, 61  
wrap, 64  
write, 61  
writeFastHLine, 61  
writeFastVLine, 61  
writeFillRect, 62  
writeLine, 62  
writePixel, 62  
  
Adafruit\_GFX\_Button, 65  
    Adafruit\_GFX\_Button, 65  
    contains, 66  
    drawButton, 66  
    initButton, 66, 67  
    initButtonUL, 67, 68  
    isPressed, 69  
    justPressed, 69  
    justReleased, 69  
    press, 69  
Adafruit\_IS31FL3731, 69  
    \_frame, 73

\_i2caddr, 74  
Adafruit\_IS31FL3731, 71  
audioSync, 71  
begin, 71  
clear, 71  
displayFrame, 71  
drawPixel, 72  
readRegister8, 72  
selectBank, 72  
setFrame, 73  
setLEDPWM, 73  
writeRegister8, 73  
Adafruit\_IS31FL3731\_Wing, 74  
Adafruit\_IS31FL3731\_Wing, 74  
drawPixel, 75  
ADC\_3V3\_PIN  
  CorrectADCResponse.ino, 377  
ADC\_GND\_PIN  
  CorrectADCResponse.ino, 377  
ADC\_MAX\_GAIN  
  CorrectADCResponse.ino, 377  
ADC\_MIN\_GAIN  
  CorrectADCResponse.ino, 377  
ADC\_RANGE  
  CorrectADCResponse.ino, 377  
ADC\_READS\_COUNT  
  CorrectADCResponse.ino, 378  
ADC\_READS\_SHIFT  
  CorrectADCResponse.ino, 378  
ADC\_RESOLUTION\_BITS  
  CorrectADCResponse.ino, 378  
ADC\_TOP\_VALUE  
  CorrectADCResponse.ino, 378  
ADC\_UNITY\_GAIN  
  CorrectADCResponse.ino, 378  
AdcWakeup.ino  
  loop, 328  
  margin, 328  
  pin, 328  
  repetitions, 329  
  repetitionsIncrease, 328  
  setup, 328  
ADDR\_I2C\_DCDC  
  DCDC.h, 280  
afk\_counter  
  B1.ino, 122  
afk\_timer  
  B1.ino, 122  
alarm  
  HealthMonitor, 102  
Alarm\_Match  
  RTCZero, 106  
alarmMatch  
  SimpleRTCAlarm.ino, 389  
  SleepRTCAlarm.ino, 391  
ANALOG\_COMPARATOR\_WAKEUP  
  batt\_ArduinoLowPower.h, 326  
analogPin  
  PrimoDeepSleep.ino, 333  
analogReadCorrection  
  batt\_SAMD\_AnalogCorrection.cpp, 381  
  batt\_SAMD\_AnalogCorrection.h, 382  
ANPERSANchar  
  bitmaps.h, 184  
APOSTROPHEchar  
  bitmaps.h, 185  
ArduinoLowPowerClass, 75  
  attachInterruptWakeup, 76  
  deatchRTCInterrupt, 76  
  deepSleep, 76  
  idle, 76  
  sleep, 76, 77  
ASTERISKchar  
  bitmaps.h, 185  
ATchar  
  bitmaps.h, 185  
attachInterrupt  
  RTCZero, 106  
attachInterruptWakeup  
  ArduinoLowPowerClass, 76  
audioSync  
  Adafruit\_IS31FL3731, 71  
B1.ino  
  afk\_counter, 122  
  afk\_timer, 122  
  boost\_check, 122  
  button\_event, 122  
  buzzer\_error\_state, 122  
  buzzer\_state, 122  
  C\_CMD\_DIAGNOSTIC\_STOP, 122  
  C\_CMD\_DISPLAY\_ENDING\_OFF, 122  
  C\_CMD\_DISPLAY\_ENDING\_ON, 123  
  C\_CMD\_DISPLAY\_STARTING\_OFF, 123  
  C\_CMD\_DISPLAY\_STARTING\_ON, 123  
  C\_CMD\_LIVE\_OFF, 123  
  C\_CMD\_LIVE\_ON, 123  
  C\_CMD\_SOUND\_ENDING\_OFF, 123  
  C\_CMD\_SOUND\_ENDING\_ON, 123  
  C\_CMD\_SOUND\_FULL\_CHARGE\_OFF, 123  
  C\_CMD\_SOUND\_FULL\_CHARGE\_ON, 123  
  C\_CMD\_SOUND\_STARTING\_OFF, 123  
  C\_CMD\_SOUND\_STARTING\_ON, 124  
  C\_CMS\_SOUND\_CHARGE\_START\_OFF, 124  
  C\_CMS\_SOUND\_CHARGE\_START\_ON, 124  
  C\_CMS\_SOUND\_DEATH\_BATTERY\_OFF, 124  
  C\_CMS\_SOUND\_DEATH\_BATTERY\_ON, 124  
  C\_DIAGNOSTIC\_PASSWORD, 124  
  C\_IDLE\_TIMER\_COUNT, 124  
  C\_LIMIT\_COMSUPTION\_PROT, 124  
  C\_LIMIT\_OVERPOWER\_PROT, 124  
  C\_LIMIT\_UNDERVOLTAGE\_PROT, 124  
  C\_LIMIT\_VOLTAGE\_INPUT, 125  
  C\_LIVE\_CMD\_OFF\_LIVE\_VIEW, 125  
  C\_LIVE\_CMD\_OUT\_OFF, 125  
  C\_LIVE\_CMD\_OUT\_ON, 125  
  C\_LIVE\_CMD\_VOLT\_DOWN\_1, 125

C\_LIVE\_CMD\_VOLT\_DOWN\_10, 125  
C\_LIVE\_CMD\_VOLT\_UP\_1, 125  
C\_LIVE\_CMD\_VOLT\_UP\_10, 125  
C\_LOW\_BATTERY\_LEVEL, 125  
C\_MASK\_ACTIVATE, 125  
C\_MASK\_DEACTIVATE, 126  
C\_MAX\_VOLT, 126  
C\_MIN\_VOLT, 126  
C\_NUMB\_PRINTS\_STATIC\_DATA, 126  
C\_OFF, 126  
C\_ON, 126  
C\_PIN\_DEATH\_BATTERY\_CHECK, 126  
C\_PIN\_EN\_DCDC, 126  
C\_PIN\_FLAG\_BATTLOW, 126  
C\_PIN\_FLAG\_CHARG, 126  
C\_PIN\_I\_OUT, 127  
C\_PIN\_SCL\_2, 127  
C\_PIN\_SDA\_2, 127  
C\_PIN\_SHIPPING\_MOD, 127  
C\_PIN\_TEST\_MODE, 127  
C\_PIN\_USB\_CONNEXION, 127  
C\_PIN\_USB\_SEL, 127  
C\_PIN\_V\_IN, 127  
C\_PIN\_V\_OUT, 127  
C\_RETRY\_750\_COUNT, 127  
C\_SW\_ST\_CAPACITY, 128  
C\_SW\_ST\_DIAGNOSTIC, 128  
C\_SW\_ST\_ERROR, 128  
C\_SW\_ST\_RUN, 128  
C\_SW\_ST\_SLEEP, 128  
C\_SW\_ST\_START\_UP, 128  
C\_SW\_ST\_STOP, 128  
C\_SW\_ST\_USB, 128  
C\_TIME\_TO\_LIGHT\_DOWN, 128  
C\_TIMER\_ARMED, 128  
C\_TIMER\_DONE, 129  
C\_TIMER\_IDLE, 129  
C\_USB\_CONNECTED, 129  
C\_USB\_DISCONNECTED, 129  
C\_WATCH\_DOG\_TIME\_MS, 129  
capacity, 129  
capcaity\_ask\_off, 129  
change\_text\_answered, 129  
click\_events, 129  
cmd\_go\_sleep, 129  
consmpn\_event\_protection\_counter, 130  
cont\_idle\_timer, 130  
cont\_sec, 130  
counter\_prints\_static\_data, 130  
data, 130  
DCDC, 130  
delay\_live\_view, 130  
diag\_check, 130  
diagnostic\_msg, 130  
display\_error\_status, 130  
display\_status, 131  
enable\_charge\_sound, 131  
enable\_death\_battery\_sound, 131  
enable\_ending\_sound, 131  
enable\_ending\_text, 131  
enable\_full\_charge\_sound, 131  
enable\_live\_view, 131  
enable\_starting\_sound, 131  
enable\_starting\_text, 131  
error\_msg, 131  
flag\_arranque, 132  
flag\_cap\_one\_shot, 132  
flag\_diag\_header\_printed, 132  
flag\_display\_capacity, 132  
flag\_error, 132  
flag\_initialize, 132  
flag\_irq\_center\_button, 132  
flag\_irq\_death\_battery, 132  
flag\_irq\_singleshot, 132  
flag\_low\_battery, 132  
flag\_msg\_init, 133  
flag\_msg\_sleep, 133  
flag\_return, 133  
flag\_sleep, 133  
flag\_sound, 133  
flag\_test, 133  
flag\_timer\_low\_bright, 133  
flag\_update\_eeprom, 133  
flag\_usb\_change, 133  
flag\_waiting, 133  
flag\_work, 134  
go\_sleep, 134  
hw\_output, 134  
IRQ\_DeathBattery\_Handler, 121  
IRQ\_USB\_Handler, 121  
IRQCenterButtonHandler, 121  
ledmatrix, 134  
loop, 121  
low\_batt\_display, 134  
low\_batt\_sound, 134  
mask\_protection\_state, 134  
MAX\_NUM\_PROTECTION\_EVENTS, 134  
mode\_bright\_display, 134  
msg\_sleep, 134  
new\_text\_received, 135  
OP\_event\_protection\_counter, 135  
output\_mode, 135  
over\_consumption\_protection, 135  
over\_power\_protection, 135  
prev\_state, 135  
prev\_volt, 135  
print\_diagnostic\_static\_data, 136  
program\_tick, 136  
protection\_event\_delay, 136  
protection\_event\_delay\_flag, 136  
reset\_capacity\_off\_text, 136  
reset\_diag\_text, 136  
reset\_error\_text, 136  
reset\_init\_text, 136  
reset\_sleep\_text, 136  
sample\_IOut, 136

sample\_POut, 137  
 sample\_Vin, 137  
 sample\_VOut, 137  
 setup, 121  
 sound, 137  
 stage\_capacity\_1, 137  
 stage\_capacity\_2, 137  
 start\_string, 137  
 state\_to\_return, 137  
 sw\_output, 137  
 sw\_status, 137  
 t0, 138  
 t1, 138  
 test\_enable, 138  
 theory\_Vout, 138  
 timer\_arranque, 138  
 timer\_diagnostic\_querist, 138  
 timer\_display\_capacity, 138  
 timer\_display\_error, 138  
 timer\_idle, 138  
 timer\_irq\_button\_center, 138  
 timer\_low\_bright, 139  
 timer\_print\_diagnostic, 139  
 timer\_recover\_voltage, 139  
 timer\_sec\_count, 139  
 timer\_wait\_sleep, 139  
 trigger\_Display\_volt, 139  
 under\_voltage\_protection, 139  
 usb\_status, 139  
 user\_output, 140  
 UV\_event\_protection\_counter, 140  
 voltage\_input\_event\_protection\_counter, 140  
 voltage\_input\_protection, 121  
**BACKSLASHchar**  
 bitmaps.h, 185  
**batt\_Adafruit\_GFX.cpp**  
 \_swap\_int16\_t, 290  
 min, 291  
 pgm\_read\_bitmap\_ptr, 291  
 pgm\_read\_byte, 291  
 pgm\_read\_dword, 291  
 pgm\_read\_glyph\_ptr, 291  
 pgm\_read\_pointer, 291  
 pgm\_read\_word, 291  
**batt\_Adafruit\_IS31FL3731.cpp**  
 \_swap\_int16\_t, 354  
**batt\_Adafruit\_IS31FL3731.h**  
 ISSI\_ADDR\_DEFAULT, 358  
 ISSI\_BANK\_FUNCTIONREG, 358  
 ISSI\_COMMANDREGISTER, 358  
 ISSI\_CONF\_AUDIOMODE, 358  
 ISSI\_CONF\_AUTOFRAMEMODE, 358  
 ISSI\_CONF\_PICTUREMODE, 358  
 ISSI\_REG\_AUDIOSYNC, 359  
 ISSI\_REG\_CONFIG, 359  
 ISSI\_REG\_CONFIG\_AUDIOPLAYMODE, 359  
 ISSI\_REG\_CONFIG\_AUTOPLAYMODE, 359  
 ISSI\_REG\_CONFIG\_PICTUREMODE, 359  
**batt\_ArduinoLowPower.h**  
 ANALOG\_COMPARATOR\_WAKEUP, 326  
 GPIO\_WAKEUP, 326  
 irq\_mode, 326  
 LowPower, 326  
 NFC\_WAKEUP, 326  
 onOffFuncPtr, 326  
 OTHER\_WAKEUP, 326  
 RTC\_ALARM\_WAKEUP, 325  
 wakeup\_reason, 326  
**batt\_FlashAsEEPROM.cpp**  
 EEPROM, 370  
 FlashStorage, 370  
**batt\_FlashAsEEPROM.h**  
 EEPROM, 371  
 EEPROM\_EMULATION\_SIZE, 371  
**batt\_FlashStorage.h**  
 Flash, 375  
 FlashStorage, 375  
 PPCAT, 375  
 PPCAT\_NX, 375  
**batt\_glcdfont.c**  
 FONT5X7\_H, 319  
 PROGMEM, 319  
**batt\_RTCZero.cpp**  
 DEFAULT\_DAY, 394  
 DEFAULT\_HOUR, 394  
 DEFAULT\_MINUTE, 394  
 DEFAULT\_MONTH, 394  
 DEFAULT\_SECOND, 394  
 DEFAULT\_YEAR, 394  
 EPOCH\_TIME\_OFF, 394  
 EPOCH\_TIME\_YEAR\_OFF, 394  
 RTC\_callBack, 395  
 RTC\_Handler, 395  
**batt\_RTCZero.h**  
 voidFuncPtr, 401  
**batt\_SAMD\_AnalogCorrection.cpp**  
 analogReadCorrection, 381  
**batt\_SAMD\_AnalogCorrection.h**  
 analogReadCorrection, 382  
**batt\_SlowSoftI2CMaster.h**  
 BUFFER\_LENGTH, 289  
 DELAY, 289  
 I2C\_MAXWAIT, 289  
 I2C\_READ, 289  
 I2C\_WRITE, 289  
**Bchar**  
 bitmaps.h, 185  
**begin**  
 Adafruit\_IS31FL3731, 71  
 RTCZero, 106  
**bitmap**  
 GFXfont, 98  
**bitmapOffset**  
 GFXglyph, 99

bitmaps.h  
ACCENTchar, 184  
Achar, 184  
ANPERSANchar, 184  
APOSTROPHEchar, 185  
ASTERISKchar, 185  
ATchar, 185  
BACKSLASHchar, 185  
Bchar, 185  
C\_DISPLAY\_HEIGHT, 183  
C\_DISPLAY\_WIDTH, 183  
Cchar, 186  
Char2Bitmap, 184  
CIRCUMFLEXchar, 186  
CLOSECURLYBRAchar, 186  
CLOSEPARENTESISchar, 186  
CLOSESQUAREBRAQUETchar, 186  
COLONchar, 186  
COMMACHAR, 187  
Dchar, 187  
DISPLAY\_SIZE, 183  
DOLLARchar, 187  
DOTchar, 187  
Echar, 187  
EIGHTchar, 188  
EQUALchar, 188  
EXCLMARKchar, 188  
EYES2\_1, 188  
EYES2\_2, 188  
EYES\_1, 189  
EYES\_2, 189  
FACE\_1, 189  
FACE\_2, 189  
Fchar, 189  
FIVEchar, 190  
FOURchar, 190  
Frame\_Batt\_1, 190  
Frame\_Batt\_2, 190  
Frame\_Batt\_Low\_1, 190  
Frame\_Batt\_Low\_2, 191  
Frame\_Error\_1, 191  
Frame\_Error\_2, 191  
Frame\_USB\_1, 191  
Frame\_USB\_2, 191  
Gchar, 192  
GREATERTHANchar, 192  
HASTAGchar, 192  
Hchar, 192  
HYPHENchar, 192  
Ichar, 193  
Jchar, 193  
Kchar, 193  
Lchar, 193  
LESSTHANchar, 193  
Mchar, 194  
Nchar, 194  
NINEchar, 194  
Ochar, 194  
ONEchar, 194  
OPENCURLYBRAchar, 195  
OPENPARENTESISchar, 195  
Pchar, 195  
PERCENTchar, 195  
PLUSchar, 195  
POWERBAR\_LOCATION, 184  
Qchar, 196  
QUESTIONchar, 196  
QUOTMARKchar, 196  
Rchar, 196  
Schar, 196  
SEMICOLONchar, 197  
SEVENchar, 197  
SIXchar, 197  
SLASHchar, 197  
SPACEchar, 197  
SQUAREBRAQUETchar, 198  
SWUNGchar, 198  
Tchar, 198  
THREEchar, 198  
TWOchar, 198  
Uchar, 199  
UNDERSCOREchar, 199  
Vchar, 199  
VERTICALBARchar, 199  
Wchar, 199  
width\_bitmap, 200  
Xchar, 200  
Ychar, 200  
Zchar, 200  
ZEROchar, 200  
blink\_OFF  
    display.h, 225  
blink\_ON  
    display.h, 225  
BOOST\_ARRAY  
    DCDC.h, 280  
boost\_check  
    B1.ino, 122  
BUFFER\_LENGTH  
    batt\_SlowSoftI2CMaster.h, 289  
button\_event  
    B1.ino, 122  
button\_not\_pressed  
    Dpad.h, 162  
button\_pressed  
    Dpad.h, 162  
Buzzer.h  
    C\_BUZZER\_BUSSY, 256  
    C\_BUZZER\_NOT\_BUSSY, 256  
    C\_MODE\_DEFAULT, 256  
    C\_MODE\_HACK\_ALTERNATIVE, 256  
    C\_NOTES\_CHARGE\_IN, 256  
    C\_NOTES\_CHARGE\_OUT, 256  
    C\_NOTES\_DEATHBATTERY, 257  
    C\_NOTES\_DOWN\_1, 257  
    C\_NOTES\_END\_1, 257

C\_NOTES\_ERROR, 257  
 C\_NOTES\_FULL\_CHARGE, 257  
 C\_NOTES\_LOW\_BATTERY, 257  
 C\_NOTES\_OFF\_1, 257  
 C\_NOTES\_ON\_1, 257  
 C\_NOTES\_START\_1, 257  
 C\_NOTES\_UP\_1, 258  
 C\_PIN\_BUZZER, 258  
 C\_SOUND\_CHARGE\_IN, 258  
 C\_SOUND\_CHARGE\_OUT, 258  
 C\_SOUND\_DEATH\_BATTERY, 258  
 C\_SOUND\_DOWN, 258  
 C\_SOUND\_END, 258  
 C\_SOUND\_ERROR, 258  
 C\_SOUND\_FULL\_CHARGE, 258  
 C\_SOUND\_LOW\_BATTERY, 258  
 C\_SOUND\_MUTE, 259  
 C\_SOUND\_OFF, 259  
 C\_SOUND\_ON, 259  
 C\_SOUND\_START, 259  
 C\_SOUND\_UP, 259  
 getSound, 255  
 InitBuzzer, 255  
 NOTE\_B9, 255  
 playSound, 256  
 size\_sound\_charge\_in, 259  
 size\_sound\_charge\_out, 259  
 size\_sound\_death\_battery, 259  
 size\_sound\_down, 259  
 size\_sound\_end, 259  
 size\_sound\_error, 260  
 size\_sound\_full\_charge, 260  
 size\_sound\_low\_battery, 260  
 size\_sound\_off, 260  
 size\_sound\_on, 260  
 size\_sound\_start, 260  
 size\_sound\_up, 260  
 SOUND\_CHARGE\_IN, 260  
 SOUND\_CHARGE\_OUT, 260  
 SOUND\_DEATH\_BATTERY, 260  
 SOUND\_DOWN, 261  
 SOUND\_END, 261  
 SOUND\_ERROR, 261  
 SOUND\_FULL\_CHARGE, 261  
 SOUND\_LOW\_BATTERY, 261  
 SOUND\_OFF, 261  
 SOUND\_ON, 261  
 SOUND\_START, 261  
 SOUND\_UP, 261  
 timer\_buzzer, 261  
 buzzer\_error\_state  
   B1.ino, 122  
 buzzer\_state  
   B1.ino, 122  
 byteSwap  
   GFXcanvas16, 90  
  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/CODE/README  
     161  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     EXAMPLE/librarie.h, 161  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     BUTTONS/Dpad.h, 162, 164  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     BUTTONS/examples/TEST\_0/TEST\_0.ino,  
       167, 168  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     BUTTONS/README.md, 161  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/bitmaps.h, 181, 201  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/display.h, 221, 227  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_0/TEST\_0.ino,  
       169, 170  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_1/TEST\_1.ino,  
       233, 234  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_2/TEST\_2.ino,  
       238, 239  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_3/TEST\_3.ino,  
       241, 242  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_4/TEST\_4.ino,  
       244, 246  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_5/TEST\_5.ino,  
       246, 247  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_6/TEST\_6.ino,  
       248, 249  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_7/TEST\_7.ino,  
       250, 251  
 C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
   PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/000  
     DISPLAY/examples/TEST\_8/TEST\_8.ino,

251, 253  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/PROJECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
DISPLAY/README.md, 161  
Adafruit\_GFX\_Library/batt\_Adafruit\_GFX.cpp,  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/EXAMPLES/00m Dropbox/Javi Rodriguez/Workspace/2-  
BUZZER/Buzzer.h, 253, 262  
PROJECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
BUZZER/examples/TEST\_0/TEST\_0.ino, C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
171, 173  
Adafruit\_GFX\_Library/batt\_Adafruit\_GFX.h,  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
BUZZER/notes.h, 264, 278  
PROJECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
BUZZER/README.md, 161  
Adafruit\_GFX\_Library/batt\_gfxfont.h, 319  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
BUZZER/notes.h, 264, 278  
PROJECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
BUZZER/TEST\_0/TEST\_0.ino, C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
171, 173  
Adafruit\_GFX\_Library/batt\_gfxfont.h, 319  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
DCDC/exmples/TEST\_0/TEST\_0.ino, 173, LOWPOWER/batt\_ArduinoLowPower.h, 325,  
175 326  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
DCDC/exmples/TEST\_1/TEST\_1.ino, 235, LOWPOWER/examples/AdcWakeUp/AdcWakeUp.ino,  
236 328, 329  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
DCDC/README.md, 161  
LOWPOWER/examples/ExternalWakeUp/ExternalWakeUp.ino,  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
SENSING/examples/TEST\_0/TEST\_0.ino, 330, 331  
176, 178  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/EXAMPLES/00m Dropbox/Javi Rodriguez/Workspace/2-  
SENSING/HealthMonitor.h, 282, 283  
PROJECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
SENSING/README.md, 161  
LOWPOWER/examples/TianStandby/TianStandby.ino,  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
POWERBAR/examples/TEST\_0/TEST\_0.ino, 336  
178, 179  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
POWERBAR/README.md, 161  
LOWPOWER/examples/TimedWakeUp/TimedWakeUp.ino,  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
POWERBAR/power\_bar.h, 284, 286  
PROJECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
POWERBAR/README.md, 161  
LOGGING/diagnostic.h, 337, 342  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
POWERBAR/README.md, 161  
LOGGING/Eeprom\_LUT.h, 346, 351  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00  
SlowSoftI2CMaster/batt\_SlowSoftI2CMaster.cppC:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
286, 287  
C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/EXAMPLES/00m Dropbox/Javi Rodriguez/Workspace/2-  
SlowSoftI2CMaster/batt\_SlowSoftI2CMaster.h, PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/LIBRARIES/00



C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C\_CMD\_SOUND\_STARTING\_ON  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TESTBCP.h, 101  
Produccion/ARDUINO/Test\_Production/Test\_ProduCMS.i, SOUND\_CHARGE\_START\_OFF  
403, 407  
B1.ino, 124

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C\_CMS\_SOUND\_CHARGE\_START\_ON  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TESTBCP.h, 101  
Produccion/LABVIEW/CPU TEST/sketch\_jul14a/Sketch CMS SOUND, DEATH\_BATTERY\_OFF  
409, 411  
B1.ino, 124

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C\_CMS\_SOUND\_DEATH\_BATTERY\_ON  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TESTBCP.h, 101  
Diseño/Arduino/Test\_Shield\_Validacion\_DCDC/Test\_Condition\_EEPROMDCDC.ino,  
413  
Eeprom\_LUT.h, 347

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C\_COUNTER\_LIMIT  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TESTDISPLAY.h, 282  
Diseño/Arduino/Test\_Shield\_Validacion\_Display/Test DISPLAY VALIDATION BATTERY,  
414  
display.h, 225

C:/Users/Javi/Team Dropbox/Javi Rodriguez/Workspace/2-C\_DIAGNOSTIC\_PASSWORD  
PROYECTOS/Bitbucket/03-Battery Pack/bat\_sw/TESTINPUT.h, 174  
Diseño/Test\_Shield\_Validacion\_Input/Test\_Shield\_VALIDATION\_INPUT,  
414  
bitmaps.h, 183

C\_BOOST\_MODE  
DCDC.h, 280

C\_BUZZER\_BUSSY  
Buzzer.h, 256

C\_BUZZER\_NOT\_BUSSY  
Buzzer.h, 256

C\_CAPACITY  
Eeprom\_LUT.h, 347

C\_CLICK\_CENTER  
Dpad.h, 163

C\_CLICK\_DOWN  
Dpad.h, 163

C\_CLICK\_UP  
Dpad.h, 163

C\_CMD\_DIAGNOSTIC\_STOP  
B1.ino, 122

C\_CMD\_DISPLAY\_ENDING\_OFF  
B1.ino, 122

C\_CMD\_DISPLAY\_ENDING\_ON  
B1.ino, 123

C\_CMD\_DISPLAY\_STARTING\_OFF  
B1.ino, 123

C\_CMD\_DISPLAY\_STARTING\_ON  
B1.ino, 123

C\_CMD\_LIVE\_OFF  
B1.ino, 123

C\_CMD\_LIVE\_ON  
B1.ino, 123

C\_CMD\_SOUND\_ENDING\_OFF  
B1.ino, 123

C\_CMD\_SOUND\_ENDING\_ON  
B1.ino, 123

C\_CMD\_SOUND\_FULL\_CHARGE\_OFF  
B1.ino, 123

C\_CMD\_SOUND\_FULL\_CHARGE\_ON  
B1.ino, 123

C\_CMD\_SOUND\_STARTING\_OFF  
B1.ino, 123

C\_DISPLAY\_OFF  
display.h, 225

C\_DISPLAY\_ON  
display.h, 226

C\_DISPLAY\_ST\_BUSSY  
display.h, 226

C\_DISPLAY\_ST\_NOT\_BUSSY  
display.h, 226

C\_DISPLAY\_WIDTH  
bitmaps.h, 183

C\_HACK\_CHARGE\_SOUND  
Eeprom\_LUT.h, 347

C\_HACK\_DEATH\_BATTERY\_SOUND  
Eeprom\_LUT.h, 348

C\_HACK\_END\_DISPLAY  
Eeprom\_LUT.h, 348

C\_HACK\_END\_SOUND  
Eeprom\_LUT.h, 348

C\_HACK\_FULL\_CHARGE\_SOUND  
Eeprom\_LUT.h, 348

C\_HACK\_START\_DISPLAY  
Eeprom\_LUT.h, 348

C\_HACK\_START\_SOUND  
Eeprom\_LUT.h, 348

C\_HIGH\_BRIGHTNESS  
display.h, 226

C\_IDLE\_TIMER\_COUNT  
B1.ino, 124

C\_INIT\_STATE  
Eeprom\_LUT.h, 348

C\_INSTANT\_CURRENT  
Eeprom\_LUT.h, 348

C\_INSTANT\_POWER  
Eeprom\_LUT.h, 348

C\_INSTANT\_VOLTAGE  
Eeprom\_LUT.h, 348

C\_LIMIT\_COMSUPTION\_PROT  
B1.ino, 124

C\_LIMIT\_OVERPOWER\_PROT  
B1.ino, 124

C\_LIMIT\_UNDERVOLTAGE\_PROT  
B1.ino, 124

C\_LIMIT\_VOLTAGE\_INPUT  
B1.ino, 125

C\_LIVE\_CMD\_OFF\_LIVE\_VIEW  
B1.ino, 125

C\_LIVE\_CMD\_OUT\_OFF  
B1.ino, 125

C\_LIVE\_CMD\_OUT\_ON  
B1.ino, 125

C\_LIVE\_CMD\_VOLT\_DOWN\_1  
B1.ino, 125

C\_LIVE\_CMD\_VOLT\_DOWN\_10  
B1.ino, 125

C\_LIVE\_CMD\_VOLT\_UP\_1  
B1.ino, 125

C\_LIVE\_CMD\_VOLT\_UP\_10  
B1.ino, 125

C\_LOW\_BATTERY\_LEVEL  
B1.ino, 125

C\_LOW\_BRIGHTNESS  
display.h, 226

C\_LP\_CENTER  
Dpad.h, 163

C\_LP\_DOWN  
Dpad.h, 163

C\_LP\_UP  
Dpad.h, 163

C\_MASK\_ACTIVATE  
B1.ino, 125

C\_MASK\_DEACTIVATE  
B1.ino, 126

C\_MAX\_VOLT  
B1.ino, 126

C\_MIN\_VOLT  
B1.ino, 126

C\_MODE\_DEFAULT  
Buzzer.h, 256

C\_MODE\_HACK\_ALTERNATIVE  
Buzzer.h, 256

C\_MODE\_HIGH\_BRIGHT  
display.h, 226

C\_MODE\_LOW\_BRIGHT  
display.h, 226

C\_MODEL  
Eeprom\_LUT.h, 349

C\_NON\_BOOST\_MODE  
DCDC.h, 280

C\_NONE\_EVENT  
Dpad.h, 163

C\_NOTES\_CHARGE\_IN  
Buzzer.h, 256

C\_NOTES\_CHARGE\_OUT  
Buzzer.h, 256

C\_NOTES\_DEATHBATTERY  
Buzzer.h, 257

C\_NOTES\_DOWN\_1  
Buzzer.h, 257

C\_NOTES\_END\_1  
Buzzer.h, 257

C\_NOTES\_ERROR  
Buzzer.h, 257

C\_NOTES\_FULL\_CHARGE  
Buzzer.h, 257

C\_NOTES\_LOW\_BATTERY  
Buzzer.h, 257

C\_NOTES\_OFF\_1  
Buzzer.h, 257

C\_NOTES\_ON\_1  
Buzzer.h, 257

C\_NOTES\_START\_1  
Buzzer.h, 257

C\_NOTES\_UP\_1  
Buzzer.h, 258

C\_NUM\_ELEMENTS  
Eeprom\_LUT.h, 349

C\_NUMB\_PRINTS\_STATIC\_DATA  
B1.ino, 126

C\_NUMBER\_BLINK\_LOW\_BATTERY  
display.h, 226

C\_NUMBER\_CYCLES  
Eeprom\_LUT.h, 349

C\_OFF  
B1.ino, 126

C\_ON  
B1.ino, 126

C\_PIN\_BUTT\_CENTER  
Dpad.h, 163

C\_PIN\_BUTT\_DOWN  
Dpad.h, 163

C\_PIN\_BUTT\_UP  
Dpad.h, 163

C\_PIN\_BUZZER  
Buzzer.h, 258

C\_PIN\_DEATH\_BATTERY\_CHECK  
B1.ino, 126

C\_PIN\_EN\_DCDC  
B1.ino, 126

TEST\_0.ino, 174

TEST\_1.ino, 236

C\_PIN\_ENABLE\_LDO  
sketch\_jul14a.ino, 409

Test\_Production.ino, 405

C\_PIN\_FLAG\_BATTLOW  
B1.ino, 126

C\_PIN\_FLAG\_CHARG  
B1.ino, 126

C\_PIN\_I\_IN  
TEST\_0.ino, 174

C\_PIN\_I\_OUT  
B1.ino, 127

C\_PIN\_IOUT\_SENSE  
sketch\_jul14a.ino, 409

Test\_Production.ino, 405

C\_PIN\_ISENSE\_RAW  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_LED\_1  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_LED\_2  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_LED\_3  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_SCL\_2  
    B1.ino, 127  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_SDA\_2  
    B1.ino, 127  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_SHIPPING\_MOD  
    B1.ino, 127

C\_PIN\_SW1  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_SW2  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 405

C\_PIN\_TEST\_MODE  
    B1.ino, 127

C\_PIN\_USB\_CONNEXION  
    B1.ino, 127

C\_PIN\_USB\_SEL  
    B1.ino, 127

C\_PIN\_V\_IN  
    B1.ino, 127

C\_PIN\_V\_OUT  
    B1.ino, 127  
    TEST\_0.ino, 174

C\_PIN\_VCC  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 406

C\_PIN\_VCC\_2  
    sketch\_jul14a.ino, 410  
    Test\_Production.ino, 406

C\_PIXEL\_START\_STRING  
    display.h, 226

C\_PIXEL\_STOP\_STRING  
    display.h, 226

C\_POWER\_ERROR  
    Eeprom\_LUT.h, 349

C\_PWBAR\_Y\_AXE  
    display.h, 227

C\_RETRY\_750\_COUNT  
    B1.ino, 127

C\_RISE\_COUNT  
    HealthMonitor.h, 282

C\_SAMPLING\_TIME

    HealthMonitor.h, 282

C\_SERIAL\_NUMBER  
    Eeprom\_LUT.h, 349

C\_SOUND\_CHARGE\_IN  
    Buzzer.h, 258

C\_SOUND\_CHARGE\_OUT  
    Buzzer.h, 258

C\_SOUND\_DEATH\_BATTERY  
    Buzzer.h, 258

C\_SOUND\_DOWN  
    Buzzer.h, 258

C\_SOUND\_END  
    Buzzer.h, 258

C\_SOUND\_ERROR  
    Buzzer.h, 258

C\_SOUND\_FULL\_CHARGE  
    Buzzer.h, 258

C\_SOUND\_LOW\_BATTERY  
    Buzzer.h, 258

C\_SOUND\_MUTE  
    Buzzer.h, 259

C\_SOUND\_OFF  
    Buzzer.h, 259

C\_SOUND\_ON  
    Buzzer.h, 259

C\_SOUND\_START  
    Buzzer.h, 259

C\_SOUND\_UP  
    Buzzer.h, 259

C\_SW\_ST\_CAPACITY  
    B1.ino, 128

C\_SW\_ST\_DIAGNOSTIC  
    B1.ino, 128

C\_SW\_ST\_ERROR  
    B1.ino, 128

C\_SW\_ST\_RUN  
    B1.ino, 128

C\_SW\_ST\_SLEEP  
    B1.ino, 128

C\_SW\_ST\_START\_UP  
    B1.ino, 128

C\_SW\_ST\_STOP  
    B1.ino, 128

C\_SW\_ST\_USB  
    B1.ino, 128

C\_TEXT\_BITMAP\_WEIGHT  
    display.h, 227

C\_TEXT\_CHAR\_HEIGHT  
    display.h, 227

C\_TEXT\_CHAR\_OFFSET\_Y  
    display.h, 227

C THEORY\_VOLTAGE  
    Eeprom\_LUT.h, 349

C\_TIME\_HEALTH\_MONITORING  
    HealthMonitor.h, 282

C\_TIME\_TO\_LIGHT\_DOWN  
    B1.ino, 128

C\_TIMER\_ARMED

B1.ino, 128  
**C\_TIMER\_BTW\_FRAMES**  
 display.h, 227  
**C\_TIMER\_DEBOUNCE**  
 Dpad.h, 164  
**C\_TIMER\_DONE**  
 B1.ino, 129  
**C\_TIMER\_IDLE**  
 B1.ino, 129  
**C\_TIMER\_LONGPRESS**  
 Dpad.h, 164  
**C\_TIMER\_LONGPRESS\_LOOP**  
 Dpad.h, 164  
**C\_TIMER\_NOT\_EXPIRED**  
 MilliTimmer.h, 384  
**C\_USB\_CONNECTED**  
 B1.ino, 129  
**C\_USB\_DISCONNECTED**  
 B1.ino, 129  
**C\_VOLTAGE\_ERROR**  
 Eeprom\_LUT.h, 349  
**C\_VOLTAGE\_INPUT\_ERROR**  
 Eeprom\_LUT.h, 349  
**C\_WATCH\_DOG\_TIME\_MS**  
 B1.ino, 129  
**C\_WORK\_TIME**  
 Eeprom\_LUT.h, 349  
**capacity**  
 B1.ino, 129  
**capcaity\_ask\_off**  
 B1.ino, 129  
**category**  
 Element, 81  
**Cchar**  
 bitmaps.h, 186  
**change\_text\_answered**  
 B1.ino, 129  
**Char2Bitmap**  
 bitmaps.h, 184  
**charBounds**  
 Adafruit\_GFX, 43  
**check**  
 HealthMonitor, 101  
**CIRCUMFLEXchar**  
 bitmaps.h, 186  
**clear**  
 Adafruit\_IS31FL3731, 71  
**click\_events**  
 B1.ino, 129  
**CLOSECURLYBRAchar**  
 bitmaps.h, 186  
**CLOSEPARENTESISchar**  
 bitmaps.h, 186  
**CLOSESQUAREBRAQUETchar**  
 bitmaps.h, 186  
**cmd\_go\_sleep**  
 B1.ino, 129  
**COLONchar**  
 bitmaps.h, 186  
**bitmaps.h**, 186  
**COMMACHAR**  
 bitmaps.h, 187  
**commit**  
 EEPROMClass, 80  
**consumptn\_event\_protection\_counter**  
 B1.ino, 130  
**cont\_idle\_timer**  
 B1.ino, 130  
**cont\_sec**  
 B1.ino, 130  
**contains**  
 Adafruit\_GFX\_Button, 66  
**CorrectADCResponse.ino**  
 ADC\_3V3\_PIN, 377  
 ADC\_GND\_PIN, 377  
 ADC\_MAX\_GAIN, 377  
 ADC\_MIN\_GAIN, 377  
 ADC\_RANGE, 377  
 ADC\_READS\_COUNT, 378  
 ADC\_READS\_SHIFT, 378  
 ADC\_RESOLUTION\_BITS, 378  
 ADC\_TOP\_VALUE, 378  
 ADC\_UNITY\_GAIN, 378  
**loop**, 378  
 MAX\_TOP\_VALUE\_READS, 378  
 read3V3Level, 378  
 readGndLevel, 378  
 setup, 379  
**count**  
 TEST\_0.ino, 177  
**counter\_prints\_static\_data**  
 B1.ino, 130  
**cp437**  
 Adafruit\_GFX, 43  
**cursor\_x**  
 Adafruit\_GFX, 63  
**cursor\_y**  
 Adafruit\_GFX, 63  
**dac\_output**  
 TEST\_0.ino, 177  
**DAC\_PIN**  
 TEST\_0.ino, 176  
**data**  
 B1.ino, 130  
 EEPROM\_EMULATION, 79  
 sketch\_jul14a.ino, 411  
 Test\_Production.ino, 406  
**day**  
 SimpleRTC.ino, 387  
 SimpleRTCAlarm.ino, 389  
 SleepRTCAlarm.ino, 392  
**DCDC**  
 B1.ino, 130  
 TEST\_0.ino, 175  
 TEST\_1.ino, 236  
**DCDC.h**  
 ADDR\_I2C\_DCDC, 280

BOOST\_ARRAY, 280  
C\_BOOST\_MODE, 280  
C\_NON\_BOOST\_MODE, 280  
LenDCDCvalues, 280  
NON\_BOOST\_ARRAY, 280  
dcdc\_controller, 77  
    dcdc\_controller, 77  
    EnableDCDC, 78  
    encoderPos, 78  
    pin\_enable, 78  
    SetVoltage, 78  
    TPICvalue, 78  
Dchar  
    bitmaps.h, 187  
deatchRTCInterrupt  
    ArduinoLowPowerClass, 76  
deepSleep  
    ArduinoLowPowerClass, 76  
DEFAULT\_DAY  
    batt\_RTCZero.cpp, 394  
DEFAULT\_HOUR  
    batt\_RTCZero.cpp, 394  
DEFAULT\_MINUTE  
    batt\_RTCZero.cpp, 394  
DEFAULT\_MONTH  
    batt\_RTCZero.cpp, 394  
DEFAULT\_SECOND  
    batt\_RTCZero.cpp, 394  
DEFAULT\_YEAR  
    batt\_RTCZero.cpp, 394  
DELAY  
    batt\_SlowSoftI2CMaster.h, 289  
delay\_live\_view  
    B1.ino, 130  
detachInterrupt  
    RTCZero, 106  
diag\_check  
    B1.ino, 130  
diagnostic.h  
    \_dirty, 341  
    \_eeprom\_RAM, 341  
    EEPROM\_EMULATION\_SIZE, 338  
    eeprom\_index, 341  
    elements, 341  
    FlashStorage, 339  
    IncrementDiagnosticData, 339  
    Init\_diagnostic\_elements, 339  
    init\_flag, 341  
    isValid, 339  
    LogDiagnosticData, 339  
    MAX\_VOLTAGE, 341  
    MAX\_WATS, 341  
    MIN\_VOLTAGE, 341  
    MIN\_WATS, 342  
    power\_values, 342  
    PrintDiagnosticData, 340  
    PrintStaticData, 340  
    PrintStats, 340  
    ReadDiagnosticData, 340  
    ResetBateriaEeprom, 340  
    SaveEepromChasis, 340  
    Stats, 340  
    UpdateEepromBatery, 341  
    voltage\_values, 342  
    diagnostic\_msg  
        B1.ino, 130  
    digitalPin  
        PrimoDeepSleep.ino, 333  
    disableAlarm  
        RTCZero, 106  
    display.h  
        blink\_OFF, 225  
        blink\_ON, 225  
        C\_DELAY\_BLINK\_LOW\_BATTERY, 225  
        C\_DISPLAY\_OFF, 225  
        C\_DISPLAY\_ON, 226  
        C\_DISPLAY\_ST\_BUSSY, 226  
        C\_DISPLAY\_ST\_NOT\_BUSSY, 226  
        C\_HIGH\_BRIGHTNESS, 226  
        C\_LOW\_BRIGHTNESS, 226  
        C\_MODE\_HIGH\_BRIGHT, 226  
        C\_MODE\_LOW\_BRIGHT, 226  
        C\_NUMBER\_BLINK\_LOW\_BATTERY, 226  
        C\_PIXEL\_START\_STRING, 226  
        C\_PIXEL\_STOP\_STRING, 226  
        C\_PWBAR\_Y\_AXE, 227  
        C\_TEXT\_BITMAP\_WEIGHT, 227  
        C\_TEXT\_CHAR\_HEIGHT, 227  
        C\_TEXT\_CHAR\_OFFSET\_Y, 227  
        C\_TIMER\_BTW\_FRAMES, 227  
        DisplayBattCharging, 222  
        DisplayCap, 223  
        DisplayDiagnosticMode, 223  
        DisplayError, 223  
        DisplayLowBattery, 223  
        DisplayText, 223  
        DisplayUsbIn, 224  
        DisplayUsbOut, 224  
        DisplayVolt, 224  
        LedWork, 225  
        ScreenON, 225  
        SwitchScreenOff, 225  
    display\_error\_status  
        B1.ino, 130  
    DISPLAY\_SIZE  
        bitmaps.h, 183  
    display\_status  
        B1.ino, 131  
        TEST\_6.ino, 249  
        TEST\_7.ino, 251  
    DisplayBattCharging  
        display.h, 222  
    DisplayCap  
        display.h, 223  
    DisplayDiagnosticMode  
        display.h, 223

DisplayError  
    display.h, 223  
displayFrame  
    Adafruit\_IS31FL3731, 71  
DisplayLowBattery  
    display.h, 223  
DisplayText  
    display.h, 223  
DisplayUsbIn  
    display.h, 224  
DisplayUsbOut  
    display.h, 224  
DisplayVolt  
    display.h, 224  
DOLLARchar  
    bitmaps.h, 187  
doMyStuff  
    PrimoDeepSleep.ino, 332  
doMyStuffWithNFC  
    PrimoDeepSleep.ino, 332  
doOtherStuff  
    PrimoDeepSleep.ino, 332  
DOTchar  
    bitmaps.h, 187  
Dpad.h  
    button\_not\_pressed, 162  
    button\_pressed, 162  
    C\_CLICK\_CENTER, 163  
    C\_CLICK\_DOWN, 163  
    C\_CLICK\_UP, 163  
    C\_LP\_CENTER, 163  
    C\_LP\_DOWN, 163  
    C\_LP\_UP, 163  
    C\_NONE\_EVENT, 163  
    C\_PIN\_BUTT\_CENTER, 163  
    C\_PIN\_BUTT\_DOWN, 163  
    C\_PIN\_BUTT\_UP, 163  
    C\_TIMER\_DEBOUNCE, 164  
    C\_TIMER\_LONGPRESS, 164  
    C\_TIMER\_LONGPRESS\_LOOP, 164  
    ReadDirPad, 162  
drawBitmap  
    Adafruit\_GFX, 44, 45  
drawButton  
    Adafruit\_GFX\_Button, 66  
drawChar  
    Adafruit\_GFX, 46  
drawCircle  
    Adafruit\_GFX, 47  
drawCircleHelper  
    Adafruit\_GFX, 47  
drawFastHLine  
    Adafruit\_GFX, 47  
    GFXcanvas1, 86  
    GFXcanvas16, 90  
    GFXcanvas8, 94  
drawFastRawHLine  
    GFXcanvas1, 86  
    GFXcanvas16, 91  
    GFXcanvas8, 95  
drawFastRawVLine  
    GFXcanvas1, 86  
    GFXcanvas16, 91  
    GFXcanvas8, 95  
drawFastVLine  
    Adafruit\_GFX, 48  
    GFXcanvas1, 87  
    GFXcanvas16, 91  
    GFXcanvas8, 96  
drawGrayscaleBitmap  
    Adafruit\_GFX, 48, 49  
drawLine  
    Adafruit\_GFX, 50  
drawPixel  
    Adafruit\_GFX, 50  
    Adafruit\_IS31FL3731, 72  
    Adafruit\_IS31FL3731\_Wing, 75  
    GFXcanvas1, 87  
    GFXcanvas16, 92  
    GFXcanvas8, 96  
drawRect  
    Adafruit\_GFX, 50  
drawRGBBitmap  
    Adafruit\_GFX, 51, 52  
drawRoundRect  
    Adafruit\_GFX, 52  
drawTriangle  
    Adafruit\_GFX, 53  
drawXBitmap  
    Adafruit\_GFX, 53  
dummy  
    TimedWakeup.ino, 336  
Echar  
    bitmaps.h, 187  
EEPROM  
    batt\_FlashAsEEPROM.cpp, 370  
    batt\_FlashAsEEPROM.h, 371  
EEPROM\_EMULATION, 79  
    data, 79  
    valid, 79  
EEPROM\_EMULATION\_SIZE  
    batt\_FlashAsEEPROM.h, 371  
    diagnostic.h, 338  
eeprom\_index  
    diagnostic.h, 341  
Eeprom\_LUT.h  
    C\_CAPACITY, 347  
    C\_CONSUMPTION\_ERROR, 347  
    C\_HACK\_CHARGE\_SOUND, 347  
    C\_HACK\_DEATH\_BATTERY\_SOUND, 348  
    C\_HACK\_END\_DISPLAY, 348  
    C\_HACK\_END\_SOUND, 348  
    C\_HACK\_FULL\_CHARGE\_SOUND, 348  
    C\_HACK\_START\_DISPLAY, 348  
    C\_HACK\_START\_SOUND, 348  
    C\_INIT\_STATE, 348

C\_INSTANT\_CURRENT, 348  
C\_INSTANT\_POWER, 348  
C\_INSTANT\_VOLTAGE, 348  
C\_MODEL, 349  
C\_NUM\_ELEMENTS, 349  
C\_NUMBER\_CYCLES, 349  
C\_POWER\_ERROR, 349  
C\_SERIAL\_NUMBER, 349  
C THEORY\_VOLTAGE, 349  
C\_VOLTAGE\_ERROR, 349  
C\_VOLTAGE\_INPUT\_ERROR, 349  
C\_WORK\_TIME, 349  
POS\_EEPROM\_CONSUMPTION\_ERROR, 349  
POS\_EEPROM\_INIT, 350  
POS\_EEPROM\_MODEL, 350  
POS\_EEPROM\_NUMBER\_CYCLES, 350  
POS\_EEPROM\_POWER\_ARRAY, 350  
POS\_EEPROM\_POWER\_ERROR, 350  
POS\_EEPROM\_SERIAL\_NUMBER, 350  
POS\_EEPROM\_VOLTAGE\_ERROR, 350  
POS\_EEPROM\_VOLTAGE\_INPUT\_ERROR, 350  
POS\_EEPROM\_VOLTS\_ARRAY, 350  
POS\_EEPROM\_WORK\_TIME, 350  
EEPROMADDR  
    EepromBitBang.h, 352  
EepromBitBang.h  
    EEPROMADDR, 352  
    ReadArrayEEPROM, 352  
    readEEPROM, 352  
    ReadWordEEPROM, 352  
    SCL\_PIN, 352  
    SDA\_PIN, 352  
    si, 353  
    WriteArrayEEPROM, 352  
    writeEEPROM, 352  
    WriteWordEEPROM, 353  
EEPROMClass, 79  
    commit, 80  
    EEPROMClass, 80  
    isValid, 80  
    length, 80  
    read, 80  
    update, 80  
    write, 81  
EIGHTchar  
    bitmaps.h, 188  
Element, 81  
    category, 81  
    name, 81  
    value, 82  
elements  
    diagnostic.h, 341  
EmulateEEPROM.ino  
    loop, 365  
    setup, 365  
enable\_charge\_sound  
    B1.ino, 131  
enable\_death\_battery\_sound  
    B1.ino, 131  
enable\_ending\_sound  
    B1.ino, 131  
enable\_ending\_text  
    B1.ino, 131  
enable\_full\_charge\_sound  
    B1.ino, 131  
enable\_live\_view  
    B1.ino, 131  
enable\_starting\_sound  
    B1.ino, 131  
enable\_starting\_text  
    B1.ino, 131  
enableAlarm  
    RTCZero, 106  
EnableDCDC  
    dcdc\_controller, 78  
encoderPos  
    dcdc\_controller, 78  
endWrite  
    Adafruit\_GFX, 54  
Epoch.ino  
    loop, 385  
    print2digits, 385  
    rtc, 385  
    setup, 385  
EPOCH\_TIME\_OFF  
    batt\_RTCZero.cpp, 394  
EPOCH\_TIME\_YEAR\_OFF  
    batt\_RTCZero.cpp, 394  
EQUALchar  
    bitmaps.h, 188  
erase  
    FlashClass, 82  
error  
    SlowSoftI2CMaster, 112  
error\_msg  
    B1.ino, 131  
EXCLMARKchar  
    bitmaps.h, 188  
ExternalWakeups.ino  
    loop, 330  
    pin, 330  
    repetitions, 330  
    repetitionsIncrease, 330  
    setup, 330  
EYES2\_1  
    bitmaps.h, 188  
EYES2\_2  
    bitmaps.h, 188  
EYES\_1  
    bitmaps.h, 189  
EYES\_2  
    bitmaps.h, 189  
FACE\_1  
    bitmaps.h, 189  
FACE\_2  
    bitmaps.h, 189

Fchar  
    bitmaps.h, 189

fillCircle  
    Adafruit\_GFX, 54

fillCircleHelper  
    Adafruit\_GFX, 54

fillRect  
    Adafruit\_GFX, 55

fillRoundRect  
    Adafruit\_GFX, 55

fillScreen  
    Adafruit\_GFX, 55

    GFXcanvas1, 87

    GFXcanvas16, 92

    GFXcanvas8, 96

fillTriangle  
    Adafruit\_GFX, 56

first  
    GFXfont, 98

FIVEchar  
    bitmaps.h, 190

flag\_aranque  
    B1.ino, 132

flag\_cap\_one\_shot  
    B1.ino, 132

flag\_diag\_header\_printed  
    B1.ino, 132

flag\_display\_capacity  
    B1.ino, 132

flag\_error  
    B1.ino, 132

flag\_initialize  
    B1.ino, 132

flag\_irq\_center\_button  
    B1.ino, 132

flag\_irq\_death\_battery  
    B1.ino, 132

flag\_irq\_singleshot  
    B1.ino, 132

flag\_low\_battery  
    B1.ino, 132

flag\_msg\_init  
    B1.ino, 133

flag\_msg\_sleep  
    B1.ino, 133

flag\_return  
    B1.ino, 133

flag\_sleep  
    B1.ino, 133

flag\_sound  
    B1.ino, 133

flag\_test  
    B1.ino, 133

flag\_timer\_low\_bright  
    B1.ino, 133

flag\_update\_eeprom  
    B1.ino, 133

flag\_usb\_change

B1.ino, 133

flag\_waiting  
    B1.ino, 133

flag\_work  
    B1.ino, 134

Flash  
    batt\_FlashStorage.h, 375

FlashClass, 82

    erase, 82

    FlashClass, 82

    read, 83

    write, 83

FlashStorage  
    batt\_FlashAsEEPROM.cpp, 370

    batt\_FlashStorage.h, 375

    diagnostic.h, 339

    FlashStoreAndRetrieve.ino, 367

    StoreNameAndSurname.ino, 368

FlashStorageClass  
    FlashStorageClass< T >, 84

FlashStorageClass< T >, 83

    FlashStorageClass, 84

    read, 84

    write, 84

FlashStoreAndRetrieve.ino  
    FlashStorage, 367

    loop, 367

    setup, 367

FONT5X7\_H  
    batt\_glcdfont.c, 319

FOURchar  
    bitmaps.h, 190

Frame\_Batt\_1  
    bitmaps.h, 190

Frame\_Batt\_2  
    bitmaps.h, 190

Frame\_Batt\_Low\_1  
    bitmaps.h, 190

Frame\_Batt\_Low\_2  
    bitmaps.h, 191

Frame\_Error\_1  
    bitmaps.h, 191

Frame\_Error\_2  
    bitmaps.h, 191

Frame\_USB\_1  
    bitmaps.h, 191

Frame\_USB\_2  
    bitmaps.h, 191

Gchar  
    bitmaps.h, 192

getAlarmDay  
    RTCZero, 107

getAlarmHours  
    RTCZero, 107

getAlarmMinutes  
    RTCZero, 107

getAlarmMonth  
    RTCZero, 107

getAlarmSeconds  
    RTCZero, 107  
getAlarmYear  
    RTCZero, 107  
getBuffer  
    GFXcanvas1, 88  
    GFXcanvas16, 92  
    GFXcanvas8, 96  
getCounter  
    HealthMonitor, 101  
getCursorX  
    Adafruit\_GFX, 56  
getCursorY  
    Adafruit\_GFX, 56  
getDay  
    RTCZero, 107  
getEpoch  
    RTCZero, 107  
getHours  
    RTCZero, 107  
getMinutes  
    RTCZero, 107  
getMonth  
    RTCZero, 108  
getPixel  
    GFXcanvas1, 88  
    GFXcanvas16, 92  
    GFXcanvas8, 97  
getRawPixel  
    GFXcanvas1, 88  
    GFXcanvas16, 93  
    GFXcanvas8, 97  
getRotation  
    Adafruit\_GFX, 56  
getSample  
    HealthMonitor, 101  
getSeconds  
    RTCZero, 108  
getSound  
    Buzzer.h, 255  
getTextBounds  
    Adafruit\_GFX, 57, 58  
getY2kEpoch  
    RTCZero, 108  
getYear  
    RTCZero, 108  
GFXcanvas1, 84  
    ~GFXcanvas1, 86  
    drawFastHLine, 86  
    drawFastRawHLine, 86  
    drawFastRawVLine, 86  
    drawFastVLine, 87  
    drawPixel, 87  
    fillScreen, 87  
    getBuffer, 88  
    getPixel, 88  
    getRawPixel, 88  
    GFXcanvas1, 85  
GFXcanvas16, 89  
    ~GFXcanvas16, 90  
    byteSwap, 90  
    drawFastHLine, 90  
    drawFastRawHLine, 91  
    drawFastRawVLine, 91  
    drawFastVLine, 91  
    drawPixel, 92  
    fillScreen, 92  
    getBuffer, 92  
    getPixel, 92  
    getRawPixel, 93  
    GFXcanvas16, 90  
GFXcanvas8, 93  
    ~GFXcanvas8, 94  
    drawFastHLine, 94  
    drawFastRawHLine, 95  
    drawFastRawVLine, 95  
    drawFastVLine, 96  
    drawPixel, 96  
    fillScreen, 96  
    getBuffer, 96  
    getPixel, 97  
    getRawPixel, 97  
    GFXcanvas8, 94  
gfxdemo.ino  
    loop, 360  
    matrix, 361  
    setup, 361  
GFXfont, 97  
    bitmap, 98  
    first, 98  
    glyph, 98  
    last, 98  
    yAdvance, 98  
gfxFont  
    Adafruit\_GFX, 63  
GFXglyph, 99  
    bitmapOffset, 99  
    height, 99  
    width, 99  
    xAdvance, 99  
    xOffset, 99  
    yOffset, 100  
glyph  
    GFXfont, 98  
go\_sleep  
    B1.ino, 134  
GPIO\_WAKEUP  
    batt\_ArduinoLowPower.h, 326  
GREATERTHANchar  
    bitmaps.h, 192  
HASTAGchar  
    bitmaps.h, 192  
Hchar  
    bitmaps.h, 192  
HealthMonitor, 100  
    alarm, 102

check, 101  
 getCounter, 101  
 getSample, 101  
 HealthMonitor, 100  
 limit, 102  
 rampDOWNdec, 102  
 rampUPinc, 102  
 sense\_values, 102  
 setCounter, 102  
 threshold, 102  
 HealthMonitor.h  
     C\_COUNTER\_LIMIT, 282  
     C\_RISE\_COUNT, 282  
     C\_SAMPLING\_TIME, 282  
     C\_TIME\_HEALTH\_MONITORING, 282  
 HEIGHT  
     Adafruit\_GFX, 64  
 height  
     Adafruit\_GFX, 58  
     GFXglyph, 99  
 hours  
     SimpleRTC.ino, 387  
     SimpleRTCAlarm.ino, 389  
     SleepRTCAlarm.ino, 392  
 hw\_output  
     B1.ino, 134  
 HYPHENchar  
     bitmaps.h, 192  
 i  
     sketch\_jul14a.ino, 411  
     TEST\_0.ino, 172  
 i2c\_init  
     SlowSoftI2CMaster, 111  
 I2C\_MAXWAIT  
     batt\_SlowSoftI2CMaster.h, 289  
 I2C\_READ  
     batt\_SlowSoftI2CMaster.h, 289  
 i2c\_read  
     SlowSoftI2CMaster, 112  
 i2c\_rep\_start  
     SlowSoftI2CMaster, 112  
 i2c\_start  
     SlowSoftI2CMaster, 112  
 i2c\_start\_wait  
     SlowSoftI2CMaster, 112  
 i2c\_stop  
     SlowSoftI2CMaster, 112  
 I2C\_WRITE  
     batt\_SlowSoftI2CMaster.h, 289  
 i2c\_write  
     SlowSoftI2CMaster, 112  
 i\_sense  
     TEST\_0.ino, 175  
 Ichar  
     bitmaps.h, 193  
 idle  
     ArduinoLowPowerClass, 76  
     MilliTimer, 103  
     IncrementDiagnosticData  
         diagnostic.h, 339  
     Init\_diagnostic\_elements  
         diagnostic.h, 339  
     init\_flag  
         diagnostic.h, 341  
     initButton  
         Adafruit\_GFX\_Button, 66, 67  
     initButtonUL  
         Adafruit\_GFX\_Button, 67, 68  
 InitBuzzer  
     Buzzer.h, 255  
 invertDisplay  
     Adafruit\_GFX, 58  
 iout\_sense  
     sketch\_jul14a.ino, 411  
 Irq\_DeathBattery\_Handler  
     B1.ino, 121  
 irq\_mode  
     batt\_ArduinoLowPower.h, 326  
 Irq\_USB\_Handler  
     B1.ino, 121  
 IrqCenterButtonHandler  
     B1.ino, 121  
 isConfigured  
     RTCZero, 108  
 isPressed  
     Adafruit\_GFX\_Button, 69  
 ISSI\_ADDR\_DEFAULT  
     batt\_Adafruit\_IS31FL3731.h, 358  
 ISSI\_BANK\_FUNCTIONREG  
     batt\_Adafruit\_IS31FL3731.h, 358  
 ISSI\_COMMANDREGISTER  
     batt\_Adafruit\_IS31FL3731.h, 358  
 ISSI\_CONF\_AUDIOMODE  
     batt\_Adafruit\_IS31FL3731.h, 358  
 ISSI\_CONF\_AUTOFRAMEMODE  
     batt\_Adafruit\_IS31FL3731.h, 358  
 ISSI\_CONF\_PICTUREMODE  
     batt\_Adafruit\_IS31FL3731.h, 358  
 ISSI\_REG\_AUDIOSYNC  
     batt\_Adafruit\_IS31FL3731.h, 359  
 ISSI\_REG\_CONFIG  
     batt\_Adafruit\_IS31FL3731.h, 359  
 ISSI\_REG\_CONFIG\_AUDIOPLAYMODE  
     batt\_Adafruit\_IS31FL3731.h, 359  
 ISSI\_REG\_CONFIG\_AUTOPLAYMODE  
     batt\_Adafruit\_IS31FL3731.h, 359  
 ISSI\_REG\_CONFIG\_PICTUREMODE  
     batt\_Adafruit\_IS31FL3731.h, 359  
 ISSI\_REG\_PICTUREFRAME  
     batt\_Adafruit\_IS31FL3731.h, 359  
 ISSI\_REG\_SHUTDOWN  
     batt\_Adafruit\_IS31FL3731.h, 359  
 isValid  
     diagnostic.h, 339  
     EEPROMClass, 80  
 Jchar

bitmaps.h, 193  
justPressed  
    Adafruit\_GFX\_Button, 69  
justReleased  
    Adafruit\_GFX\_Button, 69  
  
Kchar  
    bitmaps.h, 193  
  
last  
    GFXfont, 98  
Lchar  
    bitmaps.h, 193  
ledmatrix  
    B1.ino, 134  
    manualanim.ino, 363  
    swirldemo.ino, 364  
    TEST\_0.ino, 170, 179  
    TEST\_1.ino, 234  
    TEST\_2.ino, 238  
    TEST\_3.ino, 242  
    TEST\_4.ino, 245  
    TEST\_5.ino, 247  
    TEST\_6.ino, 249  
    TEST\_7.ino, 251  
    TEST\_8.ino, 252  
LEDS\_IN\_POWERBAR  
    power\_bar.h, 285  
LedWork  
    display.h, 225  
LenDCDCvalues  
    DCDC.h, 280  
length  
    EEPROMClass, 80  
LESSTHANchar  
    bitmaps.h, 193  
limit  
    HealthMonitor, 102  
LogDiagnosticData  
    diagnostic.h, 339  
loop  
    AdcWakeup.ino, 328  
    B1.ino, 121  
    CorrectADCResponse.ino, 378  
    EmulateEEPROM.ino, 365  
    Epoch.ino, 385  
    ExternalWakeup.ino, 330  
    FlashStoreAndRetrieve.ino, 367  
    gfxdemo.ino, 360  
    manualanim.ino, 363  
    PrimoDeepSleep.ino, 332  
    SimpleRTC.ino, 386  
    SimpleRTCAlarm.ino, 389  
    sketch\_jul14a.ino, 409  
    SleepRTCAlarm.ino, 391  
    StoreNameAndSurname.ino, 368  
    swirldemo.ino, 364  
    TEST\_0.ino, 167, 170, 172, 174, 177, 179, 180  
    TEST\_1.ino, 234, 235, 237  
    TEST\_2.ino, 238, 240  
    TEST\_3.ino, 241, 243  
    TEST\_4.ino, 245  
    TEST\_5.ino, 247  
    TEST\_6.ino, 248  
    TEST\_7.ino, 250  
    TEST\_8.ino, 252  
    Test\_Production.ino, 404  
    TianStandby.ino, 335  
    TimedWakeup.ino, 336  
    low\_batt\_display  
        B1.ino, 134  
    low\_batt\_sound  
        B1.ino, 134  
    LowPower  
        batt\_ArduinoLowPower.h, 326  
    manualanim.ino  
        ledmatrix, 363  
        loop, 363  
        setup, 363  
        x, 363  
    margin  
        AdcWakeup.ino, 328  
    mask\_protection\_state  
        B1.ino, 134  
    MATCH\_DHHMMSS  
        RTCZero, 106  
    MATCH\_HHMMSS  
        RTCZero, 106  
    MATCH\_MMDDHHMMSS  
        RTCZero, 106  
    MATCH\_MMSS  
        RTCZero, 106  
    MATCH\_OFF  
        RTCZero, 106  
    MATCH\_SS  
        RTCZero, 106  
    MATCH\_YYMMDDHHMMSS  
        RTCZero, 106  
    matrix  
        gfxdemo.ino, 361  
    MAX\_NUM\_PROTECTION\_EVENTS  
        B1.ino, 134  
    MAX\_POWER\_DISPLAYED  
        power\_bar.h, 285  
    MAX\_TOP\_VALUE\_READS  
        CorrectADCResponse.ino, 378  
    MAX\_VOLTAGE  
        diagnostic.h, 341  
    MAX\_WATS  
        diagnostic.h, 341  
    Mchar  
        bitmaps.h, 194  
    MilliTimer, 103  
        idle, 103  
        MilliTimer, 103  
        poll, 103  
        remaining, 103

set, 103  
MilliTimer.h  
    C\_TIMER\_NOT\_EXPIRED, 384  
min  
    batt\_Adafrauit\_GFX.cpp, 291  
MIN\_VOLTAGE  
    diagnostic.h, 341  
MIN\_WATS  
    diagnostic.h, 342  
minutes  
    SimpleRTC.ino, 387  
    SimpleRTCAlarm.ino, 390  
    SleepRTCAlarm.ino, 392  
MIPS\_PIN  
    TianStandby.ino, 334  
MipsPM  
    TianStandby.ino, 335  
mode\_bright\_display  
    B1.ino, 134  
    TEST\_6.ino, 249  
    TEST\_7.ino, 251  
month  
    SimpleRTC.ino, 387  
    SimpleRTCAlarm.ino, 390  
    SleepRTCAlarm.ino, 392  
msg\_sleep  
    B1.ino, 134  
name  
    Element, 81  
    Person, 104  
Nchar  
    bitmaps.h, 194  
new\_text\_received  
    B1.ino, 135  
NFC\_WAKEUP  
    batt\_ArduinoLowPower.h, 326  
NINEchar  
    bitmaps.h, 194  
NON\_BOOST\_ARRAY  
    DCDC.h, 280  
NOTE\_A1  
    notes.h, 267  
NOTE\_A2  
    notes.h, 267  
NOTE\_A3  
    notes.h, 267  
NOTE\_A4  
    notes.h, 267  
NOTE\_A5  
    notes.h, 267  
NOTE\_A6  
    notes.h, 267  
NOTE\_A7  
    notes.h, 267  
NOTE\_A8  
    notes.h, 267  
NOTE\_A9  
    notes.h, 267  
NOTE\_AS1  
    notes.h, 268  
NOTE\_AS2  
    notes.h, 268  
NOTE\_AS3  
    notes.h, 268  
NOTE\_AS4  
    notes.h, 268  
NOTE\_AS5  
    notes.h, 268  
NOTE\_AS6  
    notes.h, 268  
NOTE\_AS7  
    notes.h, 268  
NOTE\_AS8  
    notes.h, 268  
NOTE\_AS9  
    notes.h, 268  
NOTE\_B0  
    notes.h, 268  
NOTE\_B1  
    notes.h, 269  
NOTE\_B2  
    notes.h, 269  
NOTE\_B3  
    notes.h, 269  
NOTE\_B4  
    notes.h, 269  
NOTE\_B5  
    notes.h, 269  
NOTE\_B6  
    notes.h, 269  
NOTE\_B7  
    notes.h, 269  
NOTE\_B8  
    notes.h, 269  
NOTE\_B9  
    Buzzer.h, 255  
NOTE\_C1  
    notes.h, 269  
NOTE\_C2  
    notes.h, 269  
NOTE\_C3  
    notes.h, 270  
NOTE\_C4  
    notes.h, 270  
NOTE\_C5  
    notes.h, 270  
NOTE\_C6  
    notes.h, 270  
NOTE\_C7  
    notes.h, 270  
NOTE\_C8  
    notes.h, 270  
NOTE\_C9  
    notes.h, 270  
NOTE\_CS1  
    notes.h, 270

NOTE\_CS2  
notes.h, 270  
NOTE\_CS3  
notes.h, 270  
NOTE\_CS4  
notes.h, 271  
NOTE\_CS5  
notes.h, 271  
NOTE\_CS6  
notes.h, 271  
NOTE\_CS7  
notes.h, 271  
NOTE\_CS8  
notes.h, 271  
NOTE\_CS9  
notes.h, 271  
NOTE\_D1  
notes.h, 271  
NOTE\_D2  
notes.h, 271  
NOTE\_D3  
notes.h, 271  
NOTE\_D4  
notes.h, 271  
NOTE\_D5  
notes.h, 272  
NOTE\_D6  
notes.h, 272  
NOTE\_D7  
notes.h, 272  
NOTE\_D8  
notes.h, 272  
NOTE\_D9  
notes.h, 272  
NOTE\_DS1  
notes.h, 272  
NOTE\_DS2  
notes.h, 272  
NOTE\_DS3  
notes.h, 272  
NOTE\_DS4  
notes.h, 272  
NOTE\_DS5  
notes.h, 272  
NOTE\_DS6  
notes.h, 273  
NOTE\_DS7  
notes.h, 273  
NOTE\_DS8  
notes.h, 273  
NOTE\_DS9  
notes.h, 273  
NOTE\_E1  
notes.h, 273  
NOTE\_E2  
notes.h, 273  
NOTE\_E3  
notes.h, 273  
NOTE\_E4  
notes.h, 273  
NOTE\_E5  
notes.h, 273  
NOTE\_E6  
notes.h, 273  
NOTE\_E7  
notes.h, 274  
NOTE\_E8  
notes.h, 274  
NOTE\_E9  
notes.h, 274  
NOTE\_F1  
notes.h, 274  
NOTE\_F2  
notes.h, 274  
NOTE\_F3  
notes.h, 274  
NOTE\_F4  
notes.h, 274  
NOTE\_F5  
notes.h, 274  
NOTE\_F6  
notes.h, 274  
NOTE\_F7  
notes.h, 274  
NOTE\_F8  
notes.h, 275  
NOTE\_F9  
notes.h, 275  
NOTE\_FS1  
notes.h, 275  
NOTE\_FS2  
notes.h, 275  
NOTE\_FS3  
notes.h, 275  
NOTE\_FS4  
notes.h, 275  
NOTE\_FS5  
notes.h, 275  
NOTE\_FS6  
notes.h, 275  
NOTE\_FS7  
notes.h, 275  
NOTE\_FS8  
notes.h, 275  
NOTE\_FS9  
notes.h, 276  
NOTE\_G1  
notes.h, 276  
NOTE\_G2  
notes.h, 276  
NOTE\_G3  
notes.h, 276  
NOTE\_G4  
notes.h, 276  
NOTE\_G5  
notes.h, 276

NOTE\_G6  
  notes.h, 276  
NOTE\_G7  
  notes.h, 276  
NOTE\_G8  
  notes.h, 276  
NOTE\_G9  
  notes.h, 276  
NOTE\_GS1  
  notes.h, 277  
NOTE\_GS2  
  notes.h, 277  
NOTE\_GS3  
  notes.h, 277  
NOTE\_GS4  
  notes.h, 277  
NOTE\_GS5  
  notes.h, 277  
NOTE\_GS6  
  notes.h, 277  
NOTE\_GS7  
  notes.h, 277  
NOTE\_GS8  
  notes.h, 277  
NOTE\_GS9  
  notes.h, 277  
notes.h  
  NOTE\_A1, 267  
  NOTE\_A2, 267  
  NOTE\_A3, 267  
  NOTE\_A4, 267  
  NOTE\_A5, 267  
  NOTE\_A6, 267  
  NOTE\_A7, 267  
  NOTE\_A8, 267  
  NOTE\_A9, 267  
  NOTE\_AS1, 268  
  NOTE\_AS2, 268  
  NOTE\_AS3, 268  
  NOTE\_AS4, 268  
  NOTE\_AS5, 268  
  NOTE\_AS6, 268  
  NOTE\_AS7, 268  
  NOTE\_AS8, 268  
  NOTE\_AS9, 268  
  NOTE\_B0, 268  
  NOTE\_B1, 269  
  NOTE\_B2, 269  
  NOTE\_B3, 269  
  NOTE\_B4, 269  
  NOTE\_B5, 269  
  NOTE\_B6, 269  
  NOTE\_B7, 269  
  NOTE\_B8, 269  
  NOTE\_C1, 269  
  NOTE\_C2, 269  
  NOTE\_C3, 270  
  NOTE\_C4, 270  
NOTE\_C5, 270  
NOTE\_C6, 270  
NOTE\_C7, 270  
NOTE\_C8, 270  
NOTE\_C9, 270  
NOTE\_CS1, 270  
NOTE\_CS2, 270  
NOTE\_CS3, 270  
NOTE\_CS4, 271  
NOTE\_CS5, 271  
NOTE\_CS6, 271  
NOTE\_CS7, 271  
NOTE\_CS8, 271  
NOTE\_CS9, 271  
NOTE\_D1, 271  
NOTE\_D2, 271  
NOTE\_D3, 271  
NOTE\_D4, 271  
NOTE\_D5, 272  
NOTE\_D6, 272  
NOTE\_D7, 272  
NOTE\_D8, 272  
NOTE\_D9, 272  
NOTE\_DS1, 272  
NOTE\_DS2, 272  
NOTE\_DS3, 272  
NOTE\_DS4, 272  
NOTE\_DS5, 272  
NOTE\_DS6, 273  
NOTE\_DS7, 273  
NOTE\_DS8, 273  
NOTE\_DS9, 273  
NOTE\_E1, 273  
NOTE\_E2, 273  
NOTE\_E3, 273  
NOTE\_E4, 273  
NOTE\_E5, 273  
NOTE\_E6, 273  
NOTE\_E7, 274  
NOTE\_E8, 274  
NOTE\_E9, 274  
NOTE\_F1, 274  
NOTE\_F2, 274  
NOTE\_F3, 274  
NOTE\_F4, 274  
NOTE\_F5, 274  
NOTE\_F6, 274  
NOTE\_F7, 274  
NOTE\_F8, 275  
NOTE\_F9, 275  
NOTE\_FS1, 275  
NOTE\_FS2, 275  
NOTE\_FS3, 275  
NOTE\_FS4, 275  
NOTE\_FS5, 275  
NOTE\_FS6, 275  
NOTE\_FS7, 275  
NOTE\_FS8, 275

NOTE\_FS9, 276  
NOTE\_G1, 276  
NOTE\_G2, 276  
NOTE\_G3, 276  
NOTE\_G4, 276  
NOTE\_G5, 276  
NOTE\_G6, 276  
NOTE\_G7, 276  
NOTE\_G8, 276  
NOTE\_G9, 276  
NOTE\_GS1, 277  
NOTE\_GS2, 277  
NOTE\_GS3, 277  
NOTE\_GS4, 277  
NOTE\_GS5, 277  
NOTE\_GS6, 277  
NOTE\_GS7, 277  
NOTE\_GS8, 277  
NOTE\_GS9, 277

Ochar  
    bitmaps.h, 194

ONEchar  
    bitmaps.h, 194

onOffFuncPtr  
    batt\_ArduinoLowPower.h, 326

onWakeUp  
    TianStandby.ino, 335

OP\_event\_protection\_counter  
    B1.ino, 135

OPENCURLYBRAchar  
    bitmaps.h, 195

OPENPARENTESISchar  
    bitmaps.h, 195

OTHER\_WAKEUP  
    batt\_ArduinoLowPower.h, 326

output\_mode  
    B1.ino, 135

over\_consumption\_protection  
    B1.ino, 135

over\_power\_protection  
    B1.ino, 135

Pchar  
    bitmaps.h, 195

PERCENTchar  
    bitmaps.h, 195

Person, 104  
    name, 104  
    surname, 104  
    valid, 104

pgm\_read\_bitmap\_ptr  
    batt\_Adafruit\_GFX.cpp, 291

pgm\_read\_byte  
    batt\_Adafruit\_GFX.cpp, 291

pgm\_read\_dword  
    batt\_Adafruit\_GFX.cpp, 291

pgm\_read\_glyph\_ptr  
    batt\_Adafruit\_GFX.cpp, 291

pgm\_read\_pointer  
    batt\_Adafruit\_GFX.cpp, 291

pgm\_read\_word  
    batt\_Adafruit\_GFX.cpp, 291

pin  
    AdcWakeUp.ino, 328  
    ExternalWakeUp.ino, 330

pin\_enable  
    dcdc\_controller, 78

playSound  
    Buzzer.h, 256

PLUSchar  
    bitmaps.h, 195

poll  
    MilliTimer, 103

POS\_EEPROM\_CONSUMPTION\_ERROR  
    Eeprom\_LUT.h, 349

POS\_EEPROM\_INIT  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_MODEL  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_NUMBER\_CYCLES  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_POWER\_ARRAY  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_POWER\_ERROR  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_SERIAL\_NUMBER  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_VOLTAGE\_ERROR  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_VOLTAGE\_INPUT\_ERROR  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_VOLTS\_ARRAY  
    Eeprom\_LUT.h, 350

POS\_EEPROM\_WORK\_TIME  
    Eeprom\_LUT.h, 350

power\_bar.h  
    LEDS\_IN\_POWERBAR, 285  
    MAX\_POWER\_DISPLAYED, 285  
    power\_by\_led, 285  
    PowerBar, 285  
    refresh\_timer, 285  
    UpdatePowerBar, 285

power\_by\_led  
    power\_bar.h, 285

power\_values  
    diagnostic.h, 342

PowerBar  
    power\_bar.h, 285

POWERBAR\_LOCATION  
    bitmaps.h, 184

PPCAT  
    batt\_FlashStorage.h, 375

PPCAT\_NX  
    batt\_FlashStorage.h, 375

press  
    Adafruit\_GFX\_Button, 69

prev\_state  
     B1.ino, 135  
 prev\_volt  
     B1.ino, 135  
 PrimoDeepSleep.ino  
     analogPin, 333  
     digitalPin, 333  
     doMyStuff, 332  
     doMyStuffWithNFC, 332  
     doOtherStuff, 332  
     loop, 332  
     setup, 332  
     StmEspPM, 332  
 print2digits  
     Epoch.ino, 385  
     SimpleRTC.ino, 386  
 print\_diagnostic\_static\_data  
     B1.ino, 136  
 PrintDiagnosticData  
     diagnostic.h, 340  
 PrintStaticData  
     diagnostic.h, 340  
 PrintStats  
     diagnostic.h, 340  
 PROGMEM  
     batt\_glcdfont.c, 319  
 program\_tick  
     B1.ino, 136  
 protection\_event\_delay  
     B1.ino, 136  
 protection\_event\_delay\_flag  
     B1.ino, 136  
  
 Qchar  
     bitmaps.h, 196  
 QUESTIONchar  
     bitmaps.h, 196  
 QUOTMARKchar  
     bitmaps.h, 196  
  
 rampDOWNdec  
     HealthMonitor, 102  
 rampUPinc  
     HealthMonitor, 102  
 Rchar  
     bitmaps.h, 196  
 read  
     EEPROMClass, 80  
     FlashClass, 83  
     FlashStorageClass< T >, 84  
 read3V3Level  
     CorrectADCResponse.ino, 378  
 ReadArrayEEPROM  
     EepromBitBang.h, 352  
 ReadDiagnosticData  
     diagnostic.h, 340  
 ReadDirPad  
     Dpad.h, 162  
 readEEPROM

    EepromBitBang.h, 352  
 readGndLevel  
     CorrectADCResponse.ino, 378  
 readRegister8  
     Adafruit\_IS31FL3731, 72  
 ReadWordEEPROM  
     EepromBitBang.h, 352  
 refresh\_timer  
     power\_bar.h, 285  
 remaining  
     MilliTimer, 103  
 repetitions  
     AdcWakeUp.ino, 329  
     ExternalWakeUp.ino, 330  
 repetitionsIncrease  
     AdcWakeUp.ino, 328  
     ExternalWakeUp.ino, 330  
 reset\_capacity\_off\_text  
     B1.ino, 136  
 reset\_diag\_text  
     B1.ino, 136  
 reset\_error\_text  
     B1.ino, 136  
 reset\_init\_text  
     B1.ino, 136  
     TEST\_6.ino, 249  
     TEST\_7.ino, 251  
 reset\_sleep\_text  
     B1.ino, 136  
 ResetBateriaEeprom  
     diagnostic.h, 340  
 rotation  
     Adafruit\_GFX, 64  
 rtc  
     Epoch.ino, 385  
     SimpleRTC.ino, 387  
     SimpleRTCAlarm.ino, 390  
     SleepRTCAlarm.ino, 392  
 RTC\_ALARM\_WAKEUP  
     batt\_ArduinoLowPower.h, 325  
 RTC\_callBack  
     batt\_RTCZero.cpp, 395  
 RTC\_Handler  
     batt\_RTCZero.cpp, 395  
 RTCZero, 104  
     Alarm\_Match, 106  
     attachInterrupt, 106  
     begin, 106  
     detachInterrupt, 106  
     disableAlarm, 106  
     enableAlarm, 106  
     getAlarmDay, 107  
     getAlarmHours, 107  
     getAlarmMinutes, 107  
     getAlarmMonth, 107  
     getAlarmSeconds, 107  
     getAlarmYear, 107  
     getDay, 107

getEpoch, 107  
getHours, 107  
getMinutes, 107  
getMonth, 108  
getSeconds, 108  
getY2kEpoch, 108  
getYear, 108  
isConfigured, 108  
MATCH\_DHHMMSS, 106  
MATCH\_HHMMSS, 106  
MATCH\_MMDDHHMMSS, 106  
MATCH\_MMSS, 106  
MATCH\_OFF, 106  
MATCH\_SS, 106  
MATCH\_YYMMDDHHMMSS, 106  
RTCZero, 106  
setAlarmDate, 108  
setAlarmDay, 108  
setAlarmEpoch, 108  
setAlarmHours, 108  
setAlarmMinutes, 109  
setAlarmMonth, 109  
setAlarmSeconds, 109  
setAlarmTime, 109  
setAlarmYear, 109  
setDate, 109  
setDay, 109  
setEpoch, 109  
setHours, 110  
setMinutes, 110  
setMonth, 110  
setSeconds, 110  
setTime, 110  
setY2kEpoch, 110  
setYear, 110  
standbyMode, 110

sample  
    TEST\_0.ino, 177

sample\_IOut  
    B1.ino, 136

sample\_iout\_sense  
    Test\_Production.ino, 406

sample\_POut  
    B1.ino, 137

sample\_vcc  
    Test\_Production.ino, 406

sample\_vcc\_2  
    Test\_Production.ino, 406

sample\_Vin  
    B1.ino, 137

sample\_VOut  
    B1.ino, 137

SaveEepromChasis  
    diagnostic.h, 340

Schar  
    bitmaps.h, 196

SCL\_PIN  
    EepromBitBang.h, 352

scl\_pin  
    Test\_Production.ino, 406

ScreenON  
    display.h, 225

SDA\_PIN  
    EepromBitBang.h, 352

sda\_pin  
    Test\_Production.ino, 406

seconds  
    SimpleRTC.ino, 387  
    SimpleRTCAlarm.ino, 390  
    SleepRTCAlarm.ino, 392

selectBank  
    Adafruit\_IS31FL3731, 72

SEMICOLONchar  
    bitmaps.h, 197

sense\_values  
    HealthMonitor, 102

sensor\_1  
    TEST\_0.ino, 177

set  
    MilliTimmer, 103

setAlarmDate  
    RTCZero, 108

setAlarmDay  
    RTCZero, 108

setAlarmEpoch  
    RTCZero, 108

setAlarmHours  
    RTCZero, 108

setAlarmMinutes  
    RTCZero, 109

setAlarmMonth  
    RTCZero, 109

setAlarmSeconds  
    RTCZero, 109

setAlarmTime  
    RTCZero, 109

setAlarmYear  
    RTCZero, 109

setCounter  
    HealthMonitor, 102

setCursor  
    Adafruit\_GFX, 58

setDate  
    RTCZero, 109

setDay  
    RTCZero, 109

setEpoch  
    RTCZero, 109

setFont  
    Adafruit\_GFX, 59

setFrame  
    Adafruit\_IS31FL3731, 73

setHours  
    RTCZero, 110

setLEDPWM  
    Adafruit\_IS31FL3731, 73

setMinutes  
     RTCZero, 110  
 setMonth  
     RTCZero, 110  
 setRotation  
     Adafruit\_GFX, 59  
 setSeconds  
     RTCZero, 110  
 setTextColor  
     Adafruit\_GFX, 59  
 setTextSize  
     Adafruit\_GFX, 60  
 setTextWrap  
     Adafruit\_GFX, 60  
 setTime  
     RTCZero, 110  
 setup  
     AdcWakeup.ino, 328  
     B1.ino, 121  
     CorrectADCResponse.ino, 379  
     EmulateEEPROM.ino, 365  
     Epoch.ino, 385  
     ExternalWakeups.ino, 330  
     FlashStoreAndRetrieve.ino, 367  
     gfxdemo.ino, 361  
     manualanim.ino, 363  
     PrimoDeepSleep.ino, 332  
     SimpleRTC.ino, 387  
     SimpleRTCAlarm.ino, 389  
     sketch\_jul14a.ino, 409  
     SleepRTCAlarm.ino, 391  
     StoreNameAndSurname.ino, 368  
     swirldemo.ino, 364  
     TEST\_0.ino, 167, 170, 172, 174, 177, 179, 180  
     TEST\_1.ino, 234, 235, 237  
     TEST\_2.ino, 238, 240  
     TEST\_3.ino, 241, 243  
     TEST\_4.ino, 245  
     TEST\_5.ino, 247  
     TEST\_6.ino, 248  
     TEST\_7.ino, 250  
     TEST\_8.ino, 252  
     Test\_Production.ino, 404  
     TianStandby.ino, 335  
     TimedWakeup.ino, 336  
 SetVoltage  
     dcdc\_controller, 78  
 setY2kEpoch  
     RTCZero, 110  
 setYear  
     RTCZero, 110  
 SEVENchar  
     bitmaps.h, 197  
 si  
     EepromBitBang.h, 353  
 SimpleRTC.ino  
     day, 387  
     hours, 387  
     loop, 386  
     minutes, 387  
     month, 387  
     print2digits, 386  
     rtc, 387  
     seconds, 387  
     setup, 387  
     year, 387  
 SimpleRTCAlarm.ino  
     alarmMatch, 389  
     day, 389  
     hours, 389  
     loop, 389  
     minutes, 390  
     month, 390  
     rtc, 390  
     seconds, 390  
     setup, 389  
     year, 390  
 SIXchar  
     bitmaps.h, 197  
 size\_sound\_charge\_in  
     Buzzer.h, 259  
 size\_sound\_charge\_out  
     Buzzer.h, 259  
 size\_sound\_death\_battery  
     Buzzer.h, 259  
 size\_sound\_down  
     Buzzer.h, 259  
 size\_sound\_end  
     Buzzer.h, 259  
 size\_sound\_error  
     Buzzer.h, 260  
 size\_sound\_full\_charge  
     Buzzer.h, 260  
 size\_sound\_low\_battery  
     Buzzer.h, 260  
 size\_sound\_off  
     Buzzer.h, 260  
 size\_sound\_on  
     Buzzer.h, 260  
 size\_sound\_start  
     Buzzer.h, 260  
 size\_sound\_up  
     Buzzer.h, 260  
 sketch\_jul14a.ino  
     C\_PIN\_ENABLE\_LDO, 409  
     C\_PIN\_IOUT\_SENSE, 409  
     C\_PIN\_ISENSE\_RAW, 410  
     C\_PIN\_LED\_1, 410  
     C\_PIN\_LED\_2, 410  
     C\_PIN\_LED\_3, 410  
     C\_PIN\_SCL\_2, 410  
     C\_PIN\_SDA\_2, 410  
     C\_PIN\_SW1, 410  
     C\_PIN\_SW2, 410  
     C\_PIN\_VCC, 410  
     C\_PIN\_VCC\_2, 410

data, 411  
i, 411  
iout\_sense, 411  
loop, 409  
setup, 409  
vcc, 411  
vcc\_2, 411  
SLASHchar  
  bitmaps.h, 197  
sleep  
  ArduinoLowPowerClass, 76, 77  
SleepRTCAlarm.ino  
  alarmMatch, 391  
  day, 392  
  hours, 392  
  loop, 391  
  minutes, 392  
  month, 392  
  rtc, 392  
  seconds, 392  
  setup, 391  
  year, 392  
SlowSoftI2CMaster, 111  
  error, 112  
  i2c\_init, 111  
  i2c\_read, 112  
  i2c\_rep\_start, 112  
  i2c\_start, 112  
  i2c\_start\_wait, 112  
  i2c\_stop, 112  
  i2c\_write, 112  
  SlowSoftI2CMaster, 111  
sound  
  B1.ino, 137  
SOUND\_CHARGE\_IN  
  Buzzer.h, 260  
SOUND\_CHARGE\_OUT  
  Buzzer.h, 260  
SOUND\_DEATH\_BATTERY  
  Buzzer.h, 260  
SOUND\_DOWN  
  Buzzer.h, 261  
SOUND\_END  
  Buzzer.h, 261  
SOUND\_ERROR  
  Buzzer.h, 261  
SOUND\_FULL\_CHARGE  
  Buzzer.h, 261  
SOUND\_LOW\_BATTERY  
  Buzzer.h, 261  
SOUND\_OFF  
  Buzzer.h, 261  
SOUND\_ON  
  Buzzer.h, 261  
SOUND\_START  
  Buzzer.h, 261  
SOUND\_UP  
  Buzzer.h, 261  
SPACEchar  
  bitmaps.h, 197  
SQUAREBRAQUETchar  
  bitmaps.h, 198  
stage\_capacity\_1  
  B1.ino, 137  
stage\_capacity\_2  
  B1.ino, 137  
standbyMode  
  RTCZero, 110  
start\_string  
  B1.ino, 137  
startWrite  
  Adafruit\_GFX, 60  
state\_to\_return  
  B1.ino, 137  
Stats  
  diagnostic.h, 340  
StmEspPM  
  PrimoDeepSleep.ino, 332  
StoreNameAndSurname.ino  
  FlashStorage, 368  
  loop, 368  
  setup, 368  
surname  
  Person, 104  
sw\_output  
  B1.ino, 137  
sw\_status  
  B1.ino, 137  
sweep  
  swirldemo.ino, 364  
swirldemo.ino  
  ledmatrix, 364  
  loop, 364  
  setup, 364  
  sweep, 364  
SwitchScreenOff  
  display.h, 225  
SWUNGchar  
  bitmaps.h, 198  
t0  
  B1.ino, 138  
  TEST\_0.ino, 172  
t1  
  B1.ino, 138  
  TEST\_0.ino, 172  
Tchar  
  bitmaps.h, 198  
TEST\_0.ino  
  C\_PIN\_EN\_DCDC, 174  
  C\_PIN\_I\_IN, 174  
  C\_PIN\_V\_OUT, 174  
  count, 177  
  dac\_output, 177  
  DAC\_PIN, 176  
  DCDC, 175  
  i, 172

i\_sense, 175  
 ledmatrix, 170, 179  
 loop, 167, 170, 172, 174, 177, 179, 180  
 sample, 177  
 sensor\_1, 177  
 setup, 167, 170, 172, 174, 177, 179, 180  
 t0, 172  
 t1, 172  
 trigger\_timer, 177  
 v\_diff, 175  
 v\_sense, 175  
**TEST\_1.ino**  
 C\_PIN\_EN\_DCDC, 236  
 DCDC, 236  
 ledmatrix, 234  
 loop, 234, 235, 237  
 setup, 234, 235, 237  
**TEST\_2.ino**  
 ledmatrix, 238  
 loop, 238, 240  
 setup, 238, 240  
**TEST\_3.ino**  
 ledmatrix, 242  
 loop, 241, 243  
 setup, 241, 243  
**TEST\_4.ino**  
 ledmatrix, 245  
 loop, 245  
 setup, 245  
**TEST\_5.ino**  
 ledmatrix, 247  
 loop, 247  
 setup, 247  
**TEST\_6.ino**  
 display\_status, 249  
 ledmatrix, 249  
 loop, 248  
 mode\_bright\_display, 249  
 reset\_init\_text, 249  
 setup, 248  
**TEST\_7.ino**  
 display\_status, 251  
 ledmatrix, 251  
 loop, 250  
 mode\_bright\_display, 251  
 reset\_init\_text, 251  
 setup, 250  
**TEST\_8.ino**  
 ledmatrix, 252  
 loop, 252  
 setup, 252  
 timer, 252  
**test\_enable**  
 B1.ino, 138  
**Test\_Production.ino**  
 C\_PIN\_ENABLE\_LDO, 405  
 C\_PIN\_IOUT\_SENSE, 405  
 C\_PIN\_ISENSE\_RAW, 405  
 C\_PIN\_LED\_1, 405  
 C\_PIN\_LED\_2, 405  
 C\_PIN\_LED\_3, 405  
 C\_PIN\_SCL\_2, 405  
 C\_PIN\_SDA\_2, 405  
 C\_PIN\_SW1, 405  
 C\_PIN\_SW2, 405  
 C\_PIN\_VCC, 406  
 C\_PIN\_VCC\_2, 406  
 data, 406  
 loop, 404  
 sample\_iout\_sense, 406  
 sample\_vcc, 406  
 sample\_vcc\_2, 406  
 scl\_pin, 406  
 sda\_pin, 406  
 setup, 404  
 test\_start, 406  
**test\_start**  
 Test\_Production.ino, 406  
**textbgcolor**  
 Adafruit\_GFX, 64  
**textcolor**  
 Adafruit\_GFX, 64  
**textsize\_x**  
 Adafruit\_GFX, 64  
**textsize\_y**  
 Adafruit\_GFX, 64  
**theory\_Vout**  
 B1.ino, 138  
**THREEchar**  
 bitmaps.h, 198  
**threshold**  
 HealthMonitor, 102  
**TianStandby.ino**  
 loop, 335  
 MIPS\_PIN, 334  
 MipsPM, 335  
 onWakeUp, 335  
 setup, 335  
**TimedWakeUp.ino**  
 dummy, 336  
 loop, 336  
 setup, 336  
**timer**  
 TEST\_8.ino, 252  
**timer\_aranque**  
 B1.ino, 138  
**timer\_buzzer**  
 Buzzer.h, 261  
**timer\_diagnostic\_querist**  
 B1.ino, 138  
**timer\_display\_capacity**  
 B1.ino, 138  
**timer\_display\_error**  
 B1.ino, 138  
**timer\_idle**  
 B1.ino, 138

timer\_irq\_button\_center  
    B1.ino, 138  
timer\_low\_bright  
    B1.ino, 139  
timer\_print\_diagnostic  
    B1.ino, 139  
timer\_recover\_voltage  
    B1.ino, 139  
timer\_sec\_count  
    B1.ino, 139  
timer\_wait\_sleep  
    B1.ino, 139  
TPICTvalue  
    dcdc\_controller, 78  
trigger\_Display\_volt  
    B1.ino, 139  
trigger\_timer  
    TEST\_0.ino, 177  
TWOchar  
    bitmaps.h, 198  
Uchar  
    bitmaps.h, 199  
under\_voltage\_protection  
    B1.ino, 139  
UNDERSCOREchar  
    bitmaps.h, 199  
update  
    EEPROMClass, 80  
UpdateEepromBatery  
    diagnostic.h, 341  
UpdatePowerBar  
    power\_bar.h, 285  
usb\_status  
    B1.ino, 139  
user\_output  
    B1.ino, 140  
UV\_event\_protection\_counter  
    B1.ino, 140  
  
v\_diff  
    TEST\_0.ino, 175  
v\_sense  
    TEST\_0.ino, 175  
valid  
    EEPROM\_EMULATION, 79  
    Person, 104  
value  
    Element, 82  
vcc  
    sketch\_jul14a.ino, 411  
vcc\_2  
    sketch\_jul14a.ino, 411  
Vchar  
    bitmaps.h, 199  
VERTICALBARchar  
    bitmaps.h, 199  
voidFuncPtr  
    batt\_RTCZero.h, 401  
  
voltage\_input\_event\_protection\_counter  
    B1.ino, 140  
voltage\_input\_protection  
    B1.ino, 121  
voltage\_values  
    diagnostic.h, 342  
  
wakeup\_reason  
    batt\_ArduinoLowPower.h, 326  
Wchar  
    bitmaps.h, 199  
WIDTH  
    Adafruit\_GFX, 64  
width  
    Adafruit\_GFX, 61  
    GFXglyph, 99  
width\_bitmap  
    bitmaps.h, 200  
wrap  
    Adafruit\_GFX, 64  
write  
    Adafruit\_GFX, 61  
    EEPROMClass, 81  
    FlashClass, 83  
    FlashStorageClass< T >, 84  
WriteArrayEEPROM  
    EepromBitBang.h, 352  
writeEEPROM  
    EepromBitBang.h, 352  
writeFastHLine  
    Adafruit\_GFX, 61  
writeFastVLine  
    Adafruit\_GFX, 61  
writeFillRect  
    Adafruit\_GFX, 62  
writeLine  
    Adafruit\_GFX, 62  
writePixel  
    Adafruit\_GFX, 62  
writeRegister8  
    Adafruit\_IS31FL3731, 73  
WriteWordEEPROM  
    EepromBitBang.h, 353  
  
x  
    manualanim.ino, 363  
xAdvance  
    GFXglyph, 99  
Xchar  
    bitmaps.h, 200  
xOffset  
    GFXglyph, 99  
  
yAdvance  
    GFXfont, 98  
Ychar  
    bitmaps.h, 200  
year  
    SimpleRTC.ino, 387

SimpleRTCAuto.ino, [390](#)  
SleepRTCAuto.ino, [392](#)  
yOffset  
  GFXglyph, [100](#)

Zchar  
  bitmaps.h, [200](#)  
ZEROchar  
  bitmaps.h, [200](#)