

**Wallace Library**  
**for the**  
**68HC11 Microcontroller / Wallace Robot**

Software Library Version  
May 25, 2005

Andrew J. Blaich  
School of Engineering



**DISCLAIMER:**

This software is provided “as is” and without any express or implied warranties, including, without limitation, the implied warranties of merchantability and fitness for a particular purpose.

**COMPATABILITY:**

This software has been created and tested using the following development systems:

Compiler(s):

GCC 68HC11 compiler version 2.2

Processor(s):

Motorola 68HC11 E1 operating at 2 MHz E-clock

Evaluation Board(s):

Axiom Manufacturing CMM11E1 -EVBU

Hardware:

Wallace Robot, School of Engineering, GVSU

**Table of Contents**

1	Overview .....	1
2	Specifications and I/O Usage .....	2
3	Function Summary.....	3
3.1	Configuration/Miscellaneous Functions .....	3
3.2	Motor Driver Functions .....	3
3.3	Infrared Transmitter/Receiver Functions .....	3
3.4	Analog/Digital I/O Functions .....	4
4	Function Documentation.....	5
4.1	Configuration/Miscellaneous Functions .....	6
4.2	Motor Driver Functions .....	7
4.3	Infrared Transmitter/Receiver Functions .....	10
4.4	Analog/Digital I/O Functions .....	12
5	Example .....	14

## 1 Overview

The Wallace library allows a programmer to quickly and easily access and control the various I/O features of the Wallace robot. To use the library functions, include the `wallace.h` header file in the source code and link with the `libwallace.a` library file. If using the `gcc6811` batch file, simply use the `-l wallace` option on the command line to link your source code with the library.

The library consists of a base set of I/O functions along with numerous macros to provide flexibility and ease of use. The I/O functions require that the `Startup()` function be called to initialize the I/O peripherals/parameters. This must occur prior to using any of the I/O functions. The `Cleanup()` function should be called to disable/un-initialize the I/O peripherals/parameters. Failure to call these two functions will result in sporadic and incorrect operation. This library can be used for developing boot code.

In order to use the library functions, the Wallace I/O boards must be connected to the microcontroller board in a specific configuration. **Failure to properly connect the I/O boards will result in incorrect operation.** Refer to the I/O Usage section for the specific I/O configuration requirements.

## 2 Specifications and I/O Usage

The Wallace library functions have been developed for a fixed I/O configuration. Failure to properly connect the I/O boards will result in incorrect operation of the library functions and possible hardware damage. The following is a list of the required I/O connections between the various I/O boards and the microcontroller board. The front of Wallace is considered the side opposite the reset and power switches.

Motor Driver	Microcontroller
J2: Connected to left motor	
J3: Connected to right motor	
J4: EN1	MCU_PORT: PA4 (digital output)
J4: EN0	MCU_PORT: PA5 (digital output)
J4: D1	MCU_PORT: PA6 (digital output)
J4: D0	MCU_PORT: PA7 (digital output)

Infrared Receiver	Microcontroller
J2: Data	KEYPAD: PD2 (digital input)

Infrared Transmitter	Microcontroller
J2: Data	KEYPAD: PD3 (digital output)

Analog/Digital I/O	Microcontroller
J3: LED1	KEYPAD: PD4 (digital output)
J3: LED2	KEYPAD: PD5 (digital output)
J3: SW1	MCU_PORT: PE1 (digital input)
J3: SW2	MCU_PORT: PE2 (digital input)
J2: POT	MCU_PORT: PE4 (analog input)
J2: PHOTO	MCU_PORT: PE5 (analog input)
J2: BATVOLT	MCU_PORT: PE6 (analog input)

## 3 Function Summary

### 3.1 Configuration/Miscellaneous Functions

```
void WallaceStartup(void)
void WallaceCleanup(void)
void WallaceDelay(unsigned msec)
```

### 3.2 Motor Driver Functions

```
void WallaceMotorsSet(int left_on_off, int left_for_rev, int
    right_on_off, int right_for_rev)
    WallaceMotorSetRightOn()
    WallaceMotorSetRightOff()
    WallaceMotorSetRightForward()
    WallaceMotorSetRightReverse()
    WallaceMotorSetRightOnForward()
    WallaceMotorSetRightOnReverse()
    WallaceMotorSetLeftOn()
    WallaceMotorSetLeftOff()
    WallaceMotorSetLeftForward()
    WallaceMotorSetLeftReverse()
    WallaceMotorSetLeftOnForward()
    WallaceMotorSetLeftOnReverse()
    WallaceMotorsSetOn()
    WallaceMotorsSetOff()
    WallaceMotorsSetForward()
    WallaceMotorsSetReverse()
    WallaceMotorsSetOnForward()
    WallaceMotorsSetOnReverse()
void WallaceMotorsGet(int *pleft_on_off, int *pleft_for_rev, int
    *pright_on_off, int *pright_for_rev)
void WallaceMotorsSetDutyCycle(int left_dc, int right_dc)
    WallaceMotorSetDutyCycleLeft(dc)
    WallaceMotorSetDutyCycleRight(dc)
void WallaceMotorsGetDutyCycle(int *pleft_dc, int *pright_dc)
```

### 3.3 Infrared Transmitter/Receiver Functions

```
void WallaceIRTransmitterSet(int on_off)
    WallaceIRTransmitterSetOn()
    WallaceIRTransmitterSetOff()
int WallaceIRTransmitterGet(void)
    WallaceIRTransmitterIsOn()
    WallaceIRTransmitterIsOff()
int WallaceIRReceiverGet(void)
    WallaceIRReceiverIsOn()
    WallaceIRReceiverIsOff()
```

### 3.4 Analog/Digital I/O Functions

```
int WallaceSwitchGet(int num)
    WallaceSwitchIs1On()
    WallaceSwitchIs1Off()
    WallaceSwitchIs2On()
    WallaceSwitchIs2Off()
int WallaceLEDSet(int num, int on_off)
    WallaceLEDSet1On()
    WallaceLEDSet1Off()
    WallaceLEDSet2On()
    WallaceLEDSet2Off()
    WallaceLEDSetBothOn()
    WallaceLEDSetBothOff()
int WallaceLEDGet(int num)
    WallaceLEDIs1On()
    WallaceLEDIs1Off()
    WallaceLEDIs2On()
    WallaceLEDIs2Off()
int WallaceAnalogGet(int num)
    WallaceAnalogGetPOT()
    WallaceAnalogGetPHOTO()
    WallaceAnalogGetBATVOLT()
```

## **4 Function Documentation**

## 4.1 Configuration/Miscellaneous Functions

---

**void WallaceStartup(void)**

**Description:**

Performs all necessary initialization for Wallace library functions. This function must be called prior to calling any other Wallace library function. This function need only be called once.

**Parameters:**

None

**Return Value:**

None

**Related Functions:**

WallaceCleanup

---

**void WallaceCleanup(void)**

**Description:**

Turns off all of the Wallace features and restores processor to original state. This function need only be called once at the very end of the program.

**Parameters:**

None

**Return Value:**

None

**Related Functions:**

WallaceStartup

---

**void WallaceDelay(unsigned msec)**

**Description:**

Causes the program to delay for the specified number of milliseconds. This function is guaranteed to wait at least the designated delay time. It is accurate to within several milliseconds.

**Parameters:**

msec                      Delay time in milliseconds

**Return Value:**

None

**Related Functions:**

None

## 4.2 Motor Driver Functions

---

```
void WallaceMotorsSet(int left_on_off, int left_for_rev, int
                    right_on_off, int right_for_rev)
```

**Description:**

Sets the status of the motors.

**Parameters:**

left_on_off	Left motor enable status. Valid values are ON and OFF. All other values are ignored.
left_for_rev	Left motor direction status. Valid values are FORWARD and REVERSE. All other values are ignored.
right_on_off	Right motor enable status. Valid values are ON and OFF. All other values are ignored.
right_for_rev	Right motor direction status. Valid values are FORWARD and REVERSE. All other values are ignored.

**Return Value:**

None

**Related Functions:**

WallaceMotorsGet  
WallaceMotorsSetDutyCycle  
WallaceMotorsGetDutyCycle

**Macros:**

WallaceMotorSetRightOn()  
WallaceMotorSetRightOff()  
WallaceMotorSetRightForward()  
WallaceMotorSetRightReverse()  
WallaceMotorSetRightOnForward()  
WallaceMotorSetRightOnReverse()  
WallaceMotorSetLeftOn()  
WallaceMotorSetLeftOff()  
WallaceMotorSetLeftForward()  
WallaceMotorSetLeftReverse()  
WallaceMotorSetLeftOnForward()  
WallaceMotorSetLeftOnReverse()  
WallaceMotorsSetOn()  
WallaceMotorsSetOff()  
WallaceMotorsSetForward()  
WallaceMotorsSetReverse()  
WallaceMotorsSetOnForward()  
WallaceMotorsSetOnReverse()

---

```
void WallaceMotorsGet(int *pleft_on_off, int *pleft_for_rev, int
                    *pright_on_off, int *pright_for_rev)
```

**Description:**

Gets the status of the motors.

**Parameters:**

pleft_on_off	Address to store left motor enable status.
pleft_for_rev	Address to store left motor direction status.
pright_on_off	Address to store right motor enable status.
pright_for_rev	Address to store right motor direction status.

**Return Value:**

None

**Related Functions:**

WallaceMotorsSet  
WallaceMotorsSetDutyCycle  
WallaceMotorsGetDutyCycle

---

```
void WallaceMotorsSetDutyCycle(int left_dc, int right_dc)
```

**Description:**

Changes the speed of the motors by setting the duty cycle of the motor enable signals.

**Parameters:**

left_dc	Left motor enable signal duty cycle. Valid values are from 1 to 100. All other values are ignored.
right_dc	Right motor enable signal duty cycle. Valid values are from 1 to 100. All other values are ignored.

**Return Value:**

None

**Related Functions:**

WallaceMotorsSet  
WallaceMotorsGet  
WallaceMotorsGetDutyCycle

**Macros:**

WallaceMotorSetDutyCycleLeft(dc)  
WallaceMotorSetDutyCycleRight(dc)

---

```
void WallaceMotorsGetDutyCycle(int *pleft_dc, int *pright_dc)
```

**Description:**

Gets the duty cycle of the motor enable signals.

**Parameters:**

pleft_dc	Address to store left motor enable signal duty cycle.
pright_dc	Address to store right motor enable signal duty cycle.

**Return Value:**

None

**Related Functions:**

WallaceMotorsSet  
WallaceMotorsGet  
WallaceMotorsSetDutyCycle

## 4.3 Infrared Transmitter/Receiver Functions

---

`void WallaceIRTransmitterSet(int on_off)`

**Description:**

Sets the status of the IR transmitter.

**Parameters:**

on\_off                      IR transmitter status. Valid values are ON and OFF. All other values are ignored.

**Return Value:**

None

**Related Functions:**

WallaceIRTransmitterGet  
WallaceIRReceiverGet

**Macros:**

WallaceIRTransmitterSetOn()  
WallaceIRTransmitterSetOff()

---

`int WallaceIRTransmitterGet(void)`

**Description:**

Gets the status of the IR transmitter.

**Parameters:**

None

**Return Value:**

ON                            IR transmitter is on.  
OFF                           IR transmitter is off.

**Related Functions:**

WallaceIRTransmitterSet  
WallaceIRReceiverGet

**Macros:**

WallaceIRTransmitterIsOn()  
WallaceIRTransmitterIsOff()

---

**int WallaceIRReceiverGet(void)**

**Description:**

Gets the status of the IR receiver.

**Parameters:**

None

**Return Value:**

ON                   IR receiver is on (IR signal detected).  
OFF                  IR receiver is off (IR signal not detected).

**Related Functions:**

WallaceIRTransmitterSet  
WallaceIRTransmitterGet

**Macros:**

WallaceIRReceiverIsOn()  
WallaceIRReceiverIsOff()

## 4.4 Analog/Digital I/O Functions

---

`int WallaceSwitchGet(int num)`

**Description:**

Gets the status of the specified switch.

**Parameters:**

num                      Switch number. Valid numbers are SW1 and SW2.

**Return Value:**

ON                        Switch is on (pressed).  
OFF                      Switch is off (not pressed).  
-1                        Invalid number specified.

**Related Functions:**

None

**Macros:**

WallaceSwitchIs1On()  
WallaceSwitchIs1Off()  
WallaceSwitchIs2On()  
WallaceSwitchIs2Off()

---

`int WallaceLEDSet(int num, int on_off)`

**Description:**

Sets the status of the specified LED.

**Parameters:**

num                      LED number. Valid numbers are LED1, LED2, and LED3.  
on\_off                    LED status. Valid values are ON and OFF. All other values are ignored.

**Return Value:**

-1                        Invalid number specified.  
0                         Otherwise.

**Related Functions:**

WallaceLEDGet

**Macros:**

WallaceLEDSet1On()  
WallaceLEDSet1Off()  
WallaceLEDSet2On()  
WallaceLEDSet2Off()  
WallaceLEDSetBothOn()  
WallaceLEDSetBothOff()

---

**int WallaceLEDGet(int num)****Description:**

Gets the status of the specified LED.

**Parameters:**

num                    LED number. Valid numbers are LED1 and LED2.

**Return Value:**

ON                    LED is turned on.  
OFF                   LED is turned off.  
-1                    Invalid number specified.

**Related Functions:**

WallaceLEDSet

**Macros:**

WallaceLEDIs1On()  
WallaceLEDIs1Off()  
WallaceLEDIs2On()  
WallaceLEDIs2Off()

---

**int WallaceAnalogGet(int num)****Description:**

Gets the ADC value for the specified analog signal.

**Parameters:**

num                    Analog number. Valid numbers are POT, PHOTO, and BATVOLT.

**Return Value:**

0..255                ADC value. The 8-bit value corresponds to a voltage range of 0...5 volts.  
-1                    Invalid number specified.

**Related Functions:**

None

**Macros:**

WallaceAnalogGetPOT()  
WallaceAnalogGetPHOTO()  
WallaceAnalogGetBATVOLT()

## 5 Example

```
#include <wallace.h>

int main(void)
{
    WallaceStartup();

    WallaceLEDSetBothOn();
    WallaceIRTransmitterSetOn();

    /* Wait for switch 1 to be on */
    while ( WallaceSwitchIs1On() == 0 );

    /* Toggle LED 2 */
    if ( WallaceLEDIs2On() )
        WallaceLEDSet2Off();
    else
        WallaceLEDSet2On();

    /* Change speed based on light sensor */
    if ( WallaceAnalogGetPHOTO() > 100 )
        WallaceMotorsSetDutyCycle(50, 50);
    else
        WallaceMotorsSetDutyCycle(100, 100);

    if ( WallaceIRReceiverIsOn() )
        WallaceMotorsSetOnReverse();
    else
        WallaceMotorsSetOnForward();
    WallaceDelay(1000);

    WallaceCleanup();

    return 0;
}
```