Arduino playground : **Code** / **Messenger**

```
Messenger library for Arduino
by Thomas Ouellet Fredericks
contact: mrtoftrash@gmail.com
```

VERSION

- 1.2 2008/09/25: Modified header file so it compiles in Arduino 12

HISTORY

- 1.1 2008/09/19: Added an attach method for a callback function
- 1.0 2008/09/19: Initial release

DOWNLOAD

Latest version

DESCRIPTION

Messenger is a "toolkit" that facilitates the processing of ASCII messages. **Messenger** processes characters until it receives a carriage return (CR). It then considers the message complete and available. The message is split into many elements as defined by a separator. The separator is the space character, but can be any character other than NULL, LF or CR.

Each element can be retrieved as an integer(int) or as a character. Strings might be implemented in the future if I need them.

NOTES

Messenger is the new SimpleMessageSystem. SimpleMessageSystem was nice, but not simple enough :)

I wanted to switch to Firmata, but when I found out Firmata took at least half of my atmega168's memory, I went back to the drawing board and designed **Messenger** .

Please note that **Messenger** only provides methods for incoming messages. The internal Serial.print() functions work just fine for outgoing messages.

COMPATIBILITY

All of your host software code that you designed for SimpleMessageSystem should still be compatible with **Messenger**. You will simply have to modify your Arduino code.

CREATION

Messenger(byte separator)

```
-- or --
```

Messenger()

You can create an instance of Messenger by specifying a message separator. If you do not specify a separator, the space character will be selected.

METHODS

byte process(int serialData)

returns true if a message has been built and is available. Once a message is built, you should read it immediately with readInt() or readChar() because every call to process() erases the previous message if there is still some leftovers.

byte available()

returns true there are elements in the message. You must make a call to process() before using available().

int readInt()

returns the next element as an integer and removes it from the message. You must make a call to process() to build a message before trying to read it's elements. Returns 0 if the element was not an integer.

char readChar()

returns the next element as a character and removes it from the message. If the character was part of a word, the whole word is removed from the message. You must make a call to process() to build a message before trying to read it's elements.

void attach(callbackFuntion)

executes the callback function once a message is completed. This is the preferred way of working with Messenger.

EXAMPLE

```
// This example sets all the values of the digital pins with a list through a callback function

#include <Messenger.h>
// Instantiate Messenger object with the default separator (the space character)
Messenger message = Messenger();

// Create the callback function
void messageReady() {
    int pin = 0;
       // Loop through all the available elements of the message
       while ( message.available() ) {
        // Set the pin as determined by the message
          digitalWrite( pin, message.readInt() );
          pin=pin+1;
      }
}
```

```
void setup() {
  // Initiate Serial Communication
  Serial.begin(115200);
  // Attach the callback function to the Messenger
  message.attach(messageReady);
}


void loop() {
  // The following line is the most effective way of using Serial and Messenger's callback
  while ( Serial.available() )  message.process(Serial.read () );
}
```

(Printable View of http://www.arduino.cc/playground/Code/Messenger)