Reuseable Memory (ReMem) for Microcontrollers

Garrett Berg, Cloudform Design, garrett@cloudformdesign.com

LICENSE: This library is released under the FreeBSD License, if you need a copy go to: http://www.freebsd.org/copyright/freebsd-license.html

Copyright: Garrett Berg cloudformdesign.com, garrett@cloudformdesign.com

Introduction

Microcontrollers have no way to use dynamic memory allocation. It is a VERY bad idea to use **malloc** and **free** unless you know exactly what you are doing.

When creating the pthread functions that create dynamic linked lists, I ran into this issue. I wanted a library that could dynamically create data linked to pthreads, but then disapear when it wasn't needed.

The ReMem library (ReMem.h) is that library. **ReMem** is an object with it's own data space. You can then call **Remem.rmalloc** and **ReMem.free** to perform malloc and free operations. Memory that has been freed is reused with subsequent mallocs (at some cost to speed and an extra byte per allocated data).

Overview

- **ReMem ::**
    - primary object. Create with **ReMem MyMem = ReMem();**
- **ReMem.init(uint16_t MemorySize) ::**
    - Initializes the **ReMem** object with an amount of data == MemorySize.
- **ReMem.rmalloc(uint8_t data_size) ::**
    - Same as **malloc(data_size)** that you are familiar with. Allocates memory from inside the ReMem memory and returns the pointer to allocated data.
    - The maximum data size allowed is 127 bytes
    - Each allocation uses at least size + 1 bytes of data.

- **ReMem.free(void *ptr) ::**
    - Frees the pointer. Data will be re-used in subsequent mallocs
- **ReMem.defrag() ::**
    - Under development. Automatically used by threading module (no need for you to call it).
    - recounts available data.
    - TODO: reclaims space at end, combines large blocks of data for re-use

---

Document generated by eLyXer 1.2.3 (2011-08-31) on 2014-04-16T11:54:11.650750