FORCETOC

This is the current status of the Database. At present, the read and write functions for passing arrays into and out of the EEPROM is working, but only for byte format arrays. see #Code for the full explanation of each function.

EEPROM

Having finished the library of functions entirely, I have put the code up on Google Code, with a wiki and a zip of the latest version. The Google Code page is here.

Behaviour

Following is a dump of the serial output from the Arduino.

```
Checking for Database

Database not present: Creating Database
HeadID: 250
HeadData: 0

Printing DB
0
No Tags in Database
Adding Entry to DB
Printing DB
1
EntryID: 0 Entry Data: 0CCCCCCCCC
Entry already present at ID0
Adding Entry to DB
Printing DB
2
EntryID: 0 Entry Data: 0CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
Entry already present at ID0
Entry already present at ID1
Adding Entry to DB
Printing DB
3
EntryID: 0 Entry Data: 0CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
EntryID: 2 Entry Data: 2CCCCCCCCC
Printing DB
2
EntryID: 0 Entry Data: 2CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC

Printing all entry bays
```

```
EntryID: 0 Entry Data: 2CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
No entry at ID 2
HeadID: 250
HeadData: 2


Checking for Database

Database present
HeadID: 250
HeadData: 2

Printing DB
2
EntryID: 0 Entry Data: 2CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
Entry already present at ID0
Entry already present at ID1
Adding Entry to DB
Printing DB
3
EntryID: 0 Entry Data: 2CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
EntryID: 2 Entry Data: 0CCCCCCCCC
Entry already in Database
Printing DB
3
EntryID: 0 Entry Data: 2CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
EntryID: 2 Entry Data: 0CCCCCCCCC
Entry already in Database
Printing DB
3
EntryID: 0 Entry Data: 2CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
EntryID: 2 Entry Data: 0CCCCCCCCC
Printing DB
2
EntryID: 0 Entry Data: 0CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC

Printing all entry bays
EntryID: 0 Entry Data: 0CCCCCCCCC
EntryID: 1 Entry Data: 1CCCCCCCCC
No entry at ID 2
HeadID: 250
HeadData: 2
```

Code

```
/*
  EXDB - Database functions based on the EXROM library

  Copyright(c) 2010 Tom "TBSliver" Bloor. All rights reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/


#include <EXROM.h>
#define HEADER_P 0
#define HBYTE 250
#define EBYTE 251
#define MAXMEM 512 // Max memory allocated to the database


//DO NOT TOUCH
union Header
{
  byte raw[3];
  struct Data
  {
    byte hByte;
    int noTag;
  }
  data;
}
header;
//END OF DO NOT TOUCH AREA

union Entry
{
  byte raw[11]; // must be the same size as the Data struct.
  //remember that ints are 2 bytes, floats, longs etc. are 4.
  struct Data
  {
```

```
    //Needed for functionality
    byte eByte;
    union EData
    {
      byte raw[10]; //must be same size as the PData Struct.
      struct PData
      {
        //Customize this bit!
        byte anyData[10];
        //End of custom area
      }
      pData;
    }
    eData;
  }
  data;
}
entry;


//demo tags for testing
byte tag0[10] = {
  0,12,12,12,12,12,12,12,12,12};
byte tag1[10] = {
  1,12,12,12,12,12,12,12,12,12};
byte tag2[10] = {
  2,12,12,12,12,12,12,12,12,12};
byte tag3[10] = {
  3,12,12,12,12,12,12,12,12,12};
byte tag4[10] = {
  4,12,12,12,12,12,12,12,12,12};
byte tag5[10] = {
  5,12,12,12,12,12,12,12,12,12};
byte tag6[10] = {
  6,12,12,12,12,12,12,12,12,12};
/*
void setup()
{
  Serial.begin(9600);
  createHead();
  printDB();
  addEntry(tag0);
  printDB();
  addEntry(tag1);
  printDB();
  addEntry(tag2);
  printDB();
  delEntry(0);
  printDB();
```

```
  Serial.println("\r\nPrinting all entry bays");
  printEntry(0);
  printEntry(1);
  printEntry(2);
  Serial.println();
}

void loop()
{

}
*/
//Customize to print the data inside the PData struct.
void printEData()
{
  Serial.print(" Entry Data: ");
  for(int i=0;i<10;i++)
    Serial.print(entry.data.eData.pData.anyData[i], HEX);
}


/*------------------------------------------------------
                            DB
------------------------------------------------------*/

void printDB()
{
  Serial.println("Printing DB");
  readHead();
  Serial.println(header.data.noTag);
  for(int i=0;i<header.data.noTag;i++)
    printEntry(i);
  if(header.data.noTag == 0)
    Serial.println("No Tags in Database");
}

/*------------------------------------------------------
                          Entry
------------------------------------------------------*/

int entryPointer(int entryID) //returns pointer for specified entryID
{
  return HEADER_P + sizeof(header.raw) + (sizeof(entry.raw)*entryID);
}

void writeEntry(byte entryW[sizeof(entry.data.eData.raw)], int entryID)
{
  entry.data.eByte = EBYTE;
  for(int i=0;i<sizeof(entry.data.eData.raw);i++)
  {
    entry.data.eData.raw[i]=entryW[i];
```

```
  }
  EXROM.write(entryPointer(entryID), entry.raw, sizeof(entry.raw));
  readHead();
  header.data.noTag++;
  updateHead();
}

boolean readEntry(int entryID)
{
  EXROM.read(entryPointer(entryID), &entry.data.eByte);
  if(entry.data.eByte==EBYTE)
  {
    EXROM.read(entryPointer(entryID), entry.raw, sizeof(entry.raw));
    return true;
  }
  else
    return false;
}

void printEntry(int entryID)
{
  if(readEntry(entryID))
  {
    Serial.print("EntryID: ");
    Serial.print(entryID);
    printEData();
    Serial.println();
  }
  else
  {
    Serial.print("No entry at ID ");
    Serial.print(entryID);
    Serial.println();
  }
}

boolean compEntry(byte entryA[sizeof(entry.data.eData.raw)], byte
entryB[sizeof(entry.data.eData.raw)])
{
  boolean flag = false;
  for(int i=0;i<sizeof(entry.data.eData.raw);i++)
  {
    if (entryA[i]==entryB[i])
      flag = true;
    else
    {
      flag = false;
      break;
    }
```

```
  }
  return flag;
}

int findEntry(byte entryF[sizeof(entry.data.eData.raw)])
{
  int flag = -1;
  readHead();
  for(int i=0;i<header.data.noTag;i++)
  {
    readEntry(i);
    if (compEntry(entryF, entry.data.eData.raw))
    {
      flag = i;
      break;
    }
  }
  return flag;
}

boolean addEntry(byte entryA[sizeof(entry.data.eData.raw)])
{
  if(findEntry(entryA)==-1 && (entryPointer(header.data.noTag) +
sizeof(entry.raw)) <= MAXMEM)
  {
    readHead();
    for(int i=0; i <= header.data.noTag; i++)
    {
      if(!readEntry(i))
      {
        writeEntry(entryA, i);
        Serial.println("Adding Entry to DB");
        return true;
      }
      else
      {
        Serial.print("Entry already present at ID");
        Serial.println(i);
      }
    }

  }
  else if((entryPointer(header.data.noTag) + sizeof(entry.raw)) > MAXMEM)
  {
    Serial.println("Database Full");
    return false;
  }
  else
  {
    Serial.println("Entry already in Database");
    return false;
```

```
  }
}

void delEntry(int entryID)
{
  readHead();
  byte eEntry[sizeof(entry.raw)];
  for(int i=0;i<sizeof(eEntry);i++)
  {
    eEntry[i] = 255;
  }
  if(entryID==header.data.noTag)
  {
    EXROM.write(entryPointer(entryID), eEntry, sizeof(eEntry));
  }
  else
  {
    readEntry(header.data.noTag-1);
    EXROM.write(entryPointer(entryID), entry.raw, sizeof(entry.raw));
    EXROM.write(entryPointer(header.data.noTag-1), eEntry, sizeof(eEntry));
  }
  header.data.noTag--;
  updateHead();
}

/*------------------------------------------------------
                          Header
------------------------------------------------------*/

//Writes a new header
void writeHead()
{
  header.data.hByte=HBYTE;
  header.data.noTag=0;
  EXROM.write(HEADER_P, header.raw, sizeof(header.raw));
}

//Reads a header if there is one present at HEADER_P
boolean readHead()
{
  EXROM.read(HEADER_P, &header.data.hByte);
  if(header.data.hByte==HBYTE)
  {
    EXROM.read(HEADER_P, header.raw, sizeof(header.raw));
    return true;
  }
  else
    return false;
}
```

```
//Updates the header if there is one present, with whatever noTag is set.
//use readHead() first to get an accurate update, for example:

//readHead();
//header.data.noTag++;
//updateHead();

//This will update the header with an increment of one in noTag.
boolean updateHead()
{
  byte hCheck;
  EXROM.read(HEADER_P, &hCheck);
  if(hCheck==HBYTE)
  {
    EXROM.write(HEADER_P, header.raw, sizeof(header.raw));
    return true;
  }
  else
    return false;
}

//Creates a header if there isnt one present.
void createHead()
{
  Serial.println("Checking for Database");
  if(readHead())
  {
    Serial.println();
    Serial.println("Database present");
  }
  else
  {
    Serial.println();
    Serial.println("Database not present: Creating Database");
    writeHead();
  }
}

//Does what it says on the tin.
void printHead()
{
  Serial.print("HeadID: ");
  Serial.println(header.data.hByte, DEC);
  Serial.print("HeadData: ");
  Serial.print(header.data.noTag);
  Serial.println("\r\n");
}

/*------------------------------------------------------------
                              MAIN
------------------------------------------------------------*/
```

```
void setup()
{
  Serial.begin(9600);
  createHead();
  printHead();
  printDB();
  addEntry(tag0);
  printDB();
  addEntry(tag1);
  printDB();
  addEntry(tag2);
  printDB();
  delEntry(0);
  printDB();
  Serial.println("\r\nPrinting all entry bays");
  printEntry(0);
  printEntry(1);
  printEntry(2);
  printHead();
  Serial.println();
}

void loop()
{

}
```

Category:Madlab projects