PROGMEM

[Utilities]

Description

Store data in flash (program) memory instead of SRAM. There’s a description of the various [types of memory](http://www.arduino.cc/playground/Learning/Memory) available on an Arduino board.

The PROGMEM keyword is a variable modifier, it should be used only with the datatypes defined in pgmspace.h. It tells the compiler "put this information into flash memory", instead of into SRAM, where it would normally go.

PROGMEM is part of the [pgmspace.h](http://www.nongnu.org/avr-libc/user-manual/group__avr__pgmspace.html) library. It is included automatically in modern versions of the IDE. However, if you are using an IDE version below 1.0 (2011), you’ll first need to include the library at the top of your sketch, like this:

#include <avr/pgmspace.h> While PROGMEM could be used on a single variable, it is really only worth the fuss if you have a larger block of data that needs to be stored, which is usually easiest in an array, (or another C++ data structure beyond our present discussion).

Using PROGMEM is also a two-step procedure. After getting the data into Flash memory, it requires special methods (functions), also defined in the [pgmspace.h](http://www.nongnu.org/avr-libc/user-manual/group__avr__pgmspace.html) library, to read the data from program memory back into SRAM, so we can do something useful with it.

Syntax

const dataType variableName[] PROGMEM = {data0, data1, data3…​};

Note that because PROGMEM is a variable modifier, there is no hard and fast rule about where it should go, so the Arduino compiler accepts all of the definitions below, which are also synonymous. However, experiments have indicated that, in various versions of Arduino (having to do with GCC version), PROGMEM may work in one location and not in another. The "string table" example below has been tested to work with Arduino 13. Earlier versions of the IDE may work better if PROGMEM is included after the variable name.

const dataType variableName[] PROGMEM = {}; // use this form  
const PROGMEM dataType variableName[] = {}; // or this one  
const dataType PROGMEM variableName[] = {}; // not this one

Parameters

dataType: Allowed data types: any variable type.  
variableName: the name for your array of data.

Example Code

The following code fragments illustrate how to read and write unsigned chars (bytes) and ints (2 bytes) to PROGMEM.

// save some unsigned ints

const PROGMEM uint16\_t charSet[] = { 65000, 32796, 16843, 10, 11234};

// save some chars

const char signMessage[] PROGMEM = {"I AM PREDATOR, UNSEEN COMBATANT. CREATED BY THE UNITED STATES DEPART"};

unsigned int displayInt;

char myChar;

void setup() {

Serial.begin(9600);

while (!Serial); // wait for serial port to connect. Needed for native USB

// put your setup code here, to run once:

// read back a 2-byte int

for (byte k = 0; k < 5; k++) {

displayInt = pgm\_read\_word\_near(charSet + k);

Serial.println(displayInt);

}

Serial.println();

// read back a char

for (byte k = 0; k < strlen\_P(signMessage); k++) {

myChar = pgm\_read\_byte\_near(signMessage + k);

Serial.print(myChar);

}

Serial.println();

}

void loop() {

// put your main code here, to run repeatedly:

}

**Arrays of strings**

It is often convenient when working with large amounts of text, such as a project with an LCD, to setup an array of strings. Because strings themselves are arrays, this is actually an example of a two-dimensional array.

These tend to be large structures so putting them into program memory is often desirable. The code below illustrates the idea.

/\*

PROGMEM string demo

How to store a table of strings in program memory (flash),

and retrieve them.

Information summarized from:

http://www.nongnu.org/avr-libc/user-manual/pgmspace.html

Setting up a table (array) of strings in program memory is slightly complicated, but

here is a good template to follow.

Setting up the strings is a two-step process. First, define the strings.

\*/

#include <avr/pgmspace.h>

const char string\_0[] PROGMEM = "String 0"; // "String 0" etc are strings to store - change to suit.

const char string\_1[] PROGMEM = "String 1";

const char string\_2[] PROGMEM = "String 2";

const char string\_3[] PROGMEM = "String 3";

const char string\_4[] PROGMEM = "String 4";

const char string\_5[] PROGMEM = "String 5";

// Then set up a table to refer to your strings.

const char \*const string\_table[] PROGMEM = {string\_0, string\_1, string\_2, string\_3, string\_4, string\_5};

char buffer[30]; // make sure this is large enough for the largest string it must hold

void setup() {

Serial.begin(9600);

while (!Serial); // wait for serial port to connect. Needed for native USB

Serial.println("OK");

}

void loop() {

/\* Using the string table in program memory requires the use of special functions to retrieve the data.

The strcpy\_P function copies a string from program space to a string in RAM ("buffer").

Make sure your receiving string in RAM is large enough to hold whatever

you are retrieving from program space. \*/

for (int i = 0; i < 6; i++) {

strcpy\_P(buffer, (char \*)pgm\_read\_word(&(string\_table[i]))); // Necessary casts and dereferencing, just copy.

Serial.println(buffer);

delay(500);

}

}

Notes and Warnings

Please note that variables must be either globally defined, OR defined with the static keyword, in order to work with PROGMEM.

The following code will NOT work when inside a function:

const char long\_str[] PROGMEM = "Hi, I would like to tell you a bit about myself.\n";

The following code WILL work, even if locally defined within a function:

const static char long\_str[] PROGMEM = "Hi, I would like to tell you a bit about myself.\n"

The F() macro

When an instruction like :

Serial.print("Write something on the Serial Monitor");

is used, the string to be printed is normally saved in RAM. If your sketch prints a lot of stuff on the Serial Monitor, you can easily fill the RAM. If you have free FLASH memory space, you can easily indicate that the string must be saved in FLASH using the syntax:

Serial.print(F("Write something on the Serial Monitor that is stored in FLASH"));