Godafoss reference

random

colofon

This is simple 32-bit LCG random function, for demos and games.The random facilities of the standard library are not used becausethey eat up too much RAM. Do NOT use this for crypto work.The LCG used is the Microsoft Visual/Quick C/C++ variant as explainedon https://en.wikipedia.org/wiki/Linear_congruential_generator, butusing bits 8..23 rather than 16..30.

~~`uint16_t rand`~~

~~`uint32_t random32(){ ... }`~~

This function returns a 16-bit non-negative pseudo-random number.

~~`template< typename t > t random_i`~~

This function returns a 32-bit non-negative pseudo-random number.

This function returns a non-negative pseudo-random numberin the range [first .. last].This number is calculated from a number generated by random32by modulo arithmetic.This is simple and fast, but the distribution is ideal:the higher values in the range will be somewhat underrepresented.

~~`void random_seed( by_`~~

This function sets the start for the valuereturned by subsequent random calls.It can be used to re-start a random sequence, or (when you havea truely random source) to start a truely random random sequence.

background processing

colofon

The background class implements run-to-completion stylebackground processing.
A class that needs background processing must inherit from backgroundand implement the work function.
This work function will be calledwhen plain wait functions (the ones that allow background processing)are called.When an application contains background work,all plain wait functions can take longer than the specified time,up to the run time of the longest runtime of the work() functions.No background work will be done from wait calls made whilea work() function is running.For all background jobs: be careful to preserve the object,or your servicing will end.This is not UB: the background destructorremoves itself from the list of background jobs.When the application would terminate (exit from its main()),background::run() can be called instead, which will servethe background processing (it will never return).

```
struct background : publ
// ============================
```

ints specified by number of bits

colofon

```
template< uint64_t n > struct uint_bits {    typedef typename ...        fast;
```

uint_bits< N >::fast is the smallest 'fast'unsigned integer type that stores (at least) N bits.uint_bits< N >::least is the smallest (but not necesarrily fast)unsigned integer type that stores (at least) N bits.As both are unsigned they should be used for bit patterns, not for amounts.Note that both can be larger than requested, so they should not be usedfor modulo arithmetic (at least not without masking out excess bits).Use uint_bits< N >::fast for variables and parameters,use uint_bits< N >::least for arrays.

function and class attributes

colofon

```
#define GODAFOSS_INLINE  ...
```

GODAFOSS_INLINE forces a function to be inline.It is used when the function body is very simple, for instance whenit calls only one deeper function.This serves (only) to reduce code size and execution time.

```
#define G
```

GODAFOSS_NO_INLINE forces a function to be not inline.This is used to preserve low-level properties of a function,like the number of cylces taken by the function preable and postamble.This can be important to get predictable timing.

```
#define GODAFOSS_NO_RETURN  ...
```

GODAFOSS_NORETURN indicates that a function will not return.It is used for functions that contain a never-ending loop.This can reduce code size.

```
#define GODAFOSS_IN_RAM  ...
```

GODAFOSS_IN_RAM places the function body in RAM (instead of FLASH).On some targets, this is necesarry to get predicatable timing,or faster execution.

```
#define GODAFOSS_RUN_ONCE
```

GODAFOSS_RUN_ONCE causes the remainder of the function (the part afterthe macro) to be executed only once.

```
.
```

```
struct not_constructible {
```

Inheriting from not_constructible makes it impossible to create objectsof that class.

```
struct not_copyable {
```

Inheriting from not_copyable makes it impossible to copy an objectof that class.

hd44780

colofon

```
template<    pin_out_compatible    rs,    pin_out_compatible    e,    port_out_co
```

This template implements a

terminal

on an hd44780 character lcd.The rs, e and port must connect to the corresponding pins of the lcd.The lcd is used in 4-bit mode, so the port must connect to thed0..d3 of the lcd, the d4..d7 can be left unconnected.Only writes to the lcd are used.The _r/w pin must be connected to ground.The size of the lcd must be specifiedin characters in the x and y direction.Common sizes are 16x1, 16x2, 20x2 and 20x4.The timing is used for the waits as required by the hd44780 datasheet.

hx711

colofon

This template implements an interface to thehx711 24-Bit Analog-to-Digital Converter (ADC).This chip is intended to interface to a load cell (force sensor).

```
template<    pin_out_compatible   _sck,     pin_
```

The chip interface consist of a master-to-slave clock pin (sck),and a slave-to-master data pin (dout).The timing is used for the waits as required by the hx711 datasheet.The mode offers a choice between the A differential inputs witha gain of 128 or 64, and the B inputs with a gain of 32.The A inputs are meant to be used with a load cell.The datasheet suggest that the B inputs could be used to monitiorthe battery voltage.The mode is set at the initialization (the defauylt is a_128),and can be changed by the mode_set() function.The chip can be powered down. When a read is done the chipis first (automatically) powered up.