legOS Documentation

0.1.7

legOS Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- MotorState
- Sensor
  - LightSensor
  - RotationSensor
- _process_data

MotorState Struct Reference

the motor status type. More...

Public Members

- unsigned **assembler**
  assures word alignment for assembler.
- unsigned char **delta**
  the speed setting.
- volatile unsigned char **sum**
  running sum.
- union { ... } **access**
  provides access from C and assembler.
- unsigned char **dir**
  output pattern when sum overflows.

Detailed Description

the motor status type.

The documentation for this struct was generated from the following files:

- direct-motor.h

direct-motor.h File Reference

direct motor access. More...

Compounds

- struct MotorState

Defines

- #define **MIN_SPEED**
  minimum motor speed.
- #define **MAX_SPEED**
  maximum motor speed.

Enumerations

- enum **MotorDirection** { off, fwd, rev, brake }
  the motor directions. More...

Functions

- void **dm_init** (void)
  initialize motors.
- void **dm_shutdown** (void)
  shutdown motors.
- const void **motor_a_dir** (MotorDirection dir)
  set motor A direction. More...
- const void **motor_b_dir** (MotorDirection dir)
  set motor B direction. More...
- const void **motor_c_dir** (MotorDirection dir)
  set motor C direction. More...
- const void **motor_a_speed** (unsigned char speed)
  set motor A speed. More...
- const void **motor_b_speed** (unsigned char speed)
  set motor B speed. More...

- const void **motor_c_speed** (unsigned char speed)
  set motor C speed. More...

Variables

- const unsigned char **dm_a_pattern** [4]
  motor drive patterns. More...

- const unsigned char dm_b_pattern [4]
- const unsigned char dm_c_pattern [4]
- MotorState **dm_a**
  motor A state.

- MotorState **dm_b**
  motor B state.

- MotorState **dm_c**
  motor C state.

Detailed Description

direct motor access.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Enumeration Type Documentation

enum MotorDirection

the motor directions.

Enumeration values:

- **off**

  freewheel.

- **fwd**

  forward.

- **rev**

reverse.

- **brake**

  hold current position.

Function Documentation

const void motor_a_dir (MotorDirection *dir*)

set motor A direction.

Parameters:
dir - the direction

const void motor_b_dir (MotorDirection *dir*)

set motor B direction.

Parameters:
dir - the direction

const void motor_c_dir (MotorDirection *dir*)

set motor C direction.

Parameters:
dir - the direction

const void motor_a_speed (unsigned char *speed*)

set motor A speed.

Parameters:
speed - the speed

const void motor_b_speed (unsigned char *speed*)

set motor B speed.

Parameters:
speed - the speed

const void motor_c_speed (unsigned char *speed*)

set motor C speed.

Parameters:
speed - the speed

Variable Documentation

const unsigned char dm_a_pattern[4]

motor drive patterns.

to be indexed with MotorDirections

See also:
MotorDirections

const unsigned char dm_b_pattern[4]

to be indexed with MotorDirections

See also:
MotorDirections

const unsigned char dm_c_pattern[4]

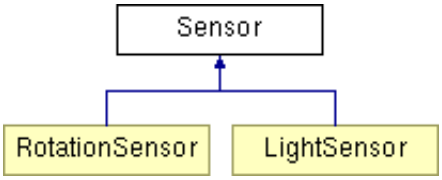
to be indexed with MotorDirections

See also:
MotorDirections

Sensor Class Reference

Raw Sensor class. More...

Class diagram for Sensor:



List of all members.

Public Members

- **Sensor** (unsigned* addr=s1, int active=0)
  Creates a sensor at the specified address. More...
- **~Sensor** ()
  Destructor. More...
- unsigned **value** ()
  Read raw sensor value.

Protected Members

- unsigned* **ptr**
  Pointer to raw sensor value.

Detailed Description

Raw Sensor class.

Member Function Documentation

Sensor::Sensor (unsigned * *addr* = s1, int *active* = 0)

Creates a sensor at the specified address.

Parameters:

addr - One of Sensor::s1, Sensor::s2, Sensor::s3 or Sensor::battery.
active - Flag to activate alimentation for active Sensors.

Sensor::~Sensor ()

Destructor.

Turns off alimentation.

The documentation for this class was generated from the following files:

- sensor.h

Sensor Member List

This is the complete list of members for Sensor, including all inherited members.

- ptr `[protected]`
- Sensor(unsigned* addr=s1, int active=0)
- value()
- ~Sensor()

sensor.h File Reference

Direct sensor access in C++. More...

Compounds

- class Sensor
- class LightSensor
- class RotationSensor

Variables

- unsigned* const **s1**
  RCX sensor 1 address.
- unsigned* const **s2**
  RCX sensor 2 address.
- unsigned* const **s3**
  RCX sensor 3 address.
- unsigned* const **battery**
  RCX battery sensor address.

Detailed Description

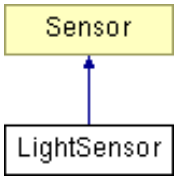
Direct sensor access in C++.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

LightSensor Class Reference

LEGO light sensor class. More...

Class diagram for LightSensor:



List of all members.

Public Members

- **LightSensor** (unsigned* addr=s1,int active=1)
  Creates a light sensor at the specified address. More...
- unsigned **value** ()
  Read light sensor value (0..100).

Detailed Description

LEGO light sensor class.

Member Function Documentation

LightSensor::LightSensor (unsigned * *addr* = s1, int *active* = 1)

Creates a light sensor at the specified address.

Parameters:

addr - One of Sensor::s1, Sensor::s2, Sensor::s3 or Sensor::battery.
active - Flag to activate sensor alimentation. With alimentation, the sensor measures reflectivity, without alimentation it samples ambient light level.

The documentation for this class was generated from the following files:

- sensor.h

LightSensor Member List

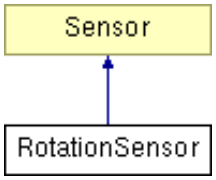
This is the complete list of members for LightSensor, including all inherited members.

- LightSensor(unsigned* addr=s1,int active=1)
- ptr `[protected]`
- Sensor(unsigned* addr=s1, int active=0)
- value()
- ~Sensor()

RotationSensor Class Reference

LEGO rotation sensor class. More...

Class diagram for RotationSensor:



List of all members.

Public Members

- **RotationSensor** (unsigned *addr=s1,int initial=0)
  Creates a rotation sensor at the specified address. More...
- **~RotationSensor** ()
  Decactivates rotation tracking.
- int **position** ()
  Read current rotational position.

Protected Members

- int* volatile **posPtr**
  Pointer to position value.

Detailed Description

LEGO rotation sensor class.

Member Function Documentation

RotationSensor::RotationSensor (unsigned * *addr* = s1, int *initial* = 0)

Creates a rotation sensor at the specified address.

Parameters:

addr - One of Sensor::s1, Sensor::s2 or Sensor::s3.
initial - Initial rotational position.

Rotation sensors are always active.

The documentation for this class was generated from the following files:

- sensor.h

RotationSensor Member List

This is the complete list of members for RotationSensor, including all inherited members.

- position()
- posPtr `[protected]`
- ptr `[protected]`
- RotationSensor(unsigned *addr=s1,int initial=0)
- Sensor(unsigned* addr=s1, int active=0)
- value()
- ~RotationSensor()
- ~Sensor()

_process_data Struct Reference

process data structure. More...

Public Members

- size_t* **sp_save**
  saved stack pointer.
- pstate_t **pstate**
  process state.
- priority_t **priority**
  process priority.
- struct _process_data* **next**
  next process in queue.
- struct _process_data* **prev**
  previous process in queue.
- struct _process_data* **parent**
  parent process.
- size_t* **stack_base**
  lower stack boundary.
- wakeup_t (* **wakeup**)(wakeup_t)
  event wakeup function.
- wakeup_t **wakeup_data**
  user data for wakeup fn.

Detailed Description

process data structure.

The documentation for this struct was generated from the following files:

- tm.h

tm.h File Reference

task management interna. More...

Functions

- void **tm_init** (void)
  init task management. More...
- void **tm_start** (void)
  start task management. More...
- void **tm_switcher** (void)
  the task switcher. More...
- size_t* **tm_scheduler** (size_t *old_sp)
  the process scheduler. More...
- int **tm_idle_task** (void)
  the idle task. More...

Detailed Description

task management interna.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void tm_init (void)

init task management.

called in single tasking mode before task setup.

void tm_start (void)

start task management.

called in single tasking mode after task setup

void tm_switcher (void)

the task switcher.

saves active context and passes sp to scheduler then restores new context from returned sp

size_t * tm_scheduler (size_t * *old_sp*)

the process scheduler.

Parameters:
old_sp - current task's current stack pointer

Returns:
new task's current stack pointer

actual context switches performed by tm_switcher (assembler wrapper)

int tm_idle_task (void)

the idle task.

infinite sleep instruction to conserve power.

legOS Compound List

Here are the classes, structs and unions with brief descriptions:

- LightSensor (LEGO light sensor class)
- MotorState (The motor status type)
- RotationSensor (LEGO rotation sensor class)
- Sensor (Raw Sensor class)
- _process_data (Process data structure)

legOS File List

Here is a list of all documented files with brief descriptions:

- include/tm.h (Header file for task management)
- include/sys/tm.h (Task management interna)
- include/unistd.h (Reduced UNIX standard library)

bitops.h File Reference

H8/300 bit operations. More...

Detailed Description

H8/300 bit operations.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

config.h File Reference

overall kernel configuration. More...

Defines

- #define **NO_EQUAL_PRIORITIES**
  -> faster scheduler.
- #define **NO_DIRECT_SOUND**
  no direct sound.

Detailed Description

overall kernel configuration.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

conio.c File Reference

console input / output. More...

Functions

- void **delay** (unsigned ms)
  uncalibrated delay loop. More...
- void **cputc_native** (char mask,int pos)
  display native mode segment mask. More...
- void **cputc_native_0** (char mask)
  display native mode segment mask at display position 0. More...
- void **cputc_native_1** (char mask)
  display native mode segment mask at display position 1. More...
- void **cputc_native_2** (char mask)
  display native mode segment mask at display position 2. More...
- void **cputc_native_3** (char mask)
  display native mode segment mask at display position 3. More...
- void **cputc_native_4** (char mask)
  display native mode segment mask at display position 4. More...
- void **cputc_native_5** (char mask)
  display native mode segment mask at display position 5. More...
- void **cputw** (unsigned word)
  display a hexword in the four leftmost positions. More...
- void **cputs** (char *s)
  display an ASCIIZ string. More...

Variables

- const char **hex_display_codes** []
  hex display codes.
- const char **ascii_display_codes** []
  ASCII display codes. More...

Detailed Description

console input / output.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Warning:
Display updates are realized exclusively by lcd_refresh()

Display positions
Digit display positions are denumerated from right to left, starting with 0 for the digit right to the running man. Digit 5 is only partially present on the RCXs display.

Native segment masks
In these bitmasks, bit 0 toggles the middle segment. Bit 1 toggles the top right segment, and the remaining segments are denumerated counterclockwise. The dot isn't encoded because it is desirable to seperate its positioning from the number display code.

Function Documentation

void delay (unsigned *ms*)

uncalibrated delay loop.

Parameters:
ms - approximate time in ms

void cputc_native (char *mask*, int *pos*)

display native mode segment mask.

Parameters:
mask - the segment mask.
pos - the desired display position.

this is a dispatcher for the fixed position routines.

void cputc_native_0 (char *mask*)

display native mode segment mask at display position 0.

Parameters:
mask - the mask to display

void cputc_native_1 (char *mask*)

display native mode segment mask at display position 1.

Parameters:
mask - the mask to display

void cputc_native_2 (char *mask*)

display native mode segment mask at display position 2.

Parameters:
mask - the mask to display

void cputc_native_3 (char *mask*)

display native mode segment mask at display position 3.

Parameters:
mask - the mask to display

void cputc_native_4 (char *mask*)

display native mode segment mask at display position 4.

Parameters:
mask - the mask to display

void cputc_native_5 (char *mask*)

display native mode segment mask at display position 5.

Parameters:
mask - the mask to display. only the middle segment is present on the display.

void cputw (unsigned *word*)

display a hexword in the four leftmost positions.

Parameters:
word - the hexword

position 0 is unaffected by this call.

void cputs (char * *s*)

display an ASCIIZ string.

Parameters:
s - the string

only the first 5 characters will be displayed.

Variable Documentation

const char ascii_display_codes[]

ASCII display codes.

This is a 7-bit ASCII table only.

---

conio.h File Reference

console input / output. More...

Functions

- void **delay** (unsigned d)
  uncalibrated delay loop. More...
- void **cputc_native_0** (char mask)
  display native mode segment mask at display position 0. More...
- void **cputc_native_1** (char mask)
  display native mode segment mask at display position 1. More...
- void **cputc_native_2** (char mask)
  display native mode segment mask at display position 2. More...
- void **cputc_native_3** (char mask)
  display native mode segment mask at display position 3. More...
- void **cputc_native_4** (char mask)
  display native mode segment mask at display position 4. More...
- void **cputc_native_5** (char mask)
  display native mode segment mask at display position 5. More...
- void **cputc_native** (char mask,int pos)
  display native mode segment mask. More...
- void **cputw** (unsigned word)
  display a hexword in the four leftmost positions. More...
- void **cputs** (char *s)
  display an ASCIIZ string. More...

Detailed Description

console input / output.

Author(s):

Markus L. Noga <noga@inrialpes.fr>

Warning:
Display updates are realized exclusively by lcd_refresh()

Display positions
Digit display positions are denumerated from right to left, starting with 0 for the digit right to the running man. Digit 5 is only partially present on the RCXs display.

Native segment masks
In these bitmasks, bit 0 toggles the middle segment. Bit 1 toggles the top right segment, and the remaining segments are denumerated counterclockwise. The dot isn't encoded because it is desirable

---

Function Documentation

void delay (unsigned *d*)

uncalibrated delay loop.

Parameters:
ms - approximate time in ms

void cputc_native_0 (char *mask*)

display native mode segment mask at display position 0.

Parameters:
mask - the mask to display

void cputc_native_1 (char *mask*)

display native mode segment mask at display position 1.

Parameters:
mask - the mask to display

void cputc_native_2 (char *mask*)

display native mode segment mask at display position 2.

Parameters:
mask - the mask to display

void cputc_native_3 (char *mask*)

display native mode segment mask at display position 3.

Parameters:
mask - the mask to display

void cputc_native_4 (char *mask*)

display native mode segment mask at display position 4.

Parameters:
mask - the mask to display

void cputc_native_5 (char *mask*)

display native mode segment mask at display position 5.

Parameters:
mask - the mask to display. only the middle segment is present on the display.

void cputc_native (char *mask*, int *pos*)

display native mode segment mask.

Parameters:
mask - the segment mask.
pos - the desired display position.

this is a dispatcher for the fixed position routines.

void cputw (unsigned *word*)

display a hexword in the four leftmost positions.

Parameters:
word - the hexword

position 0 is unaffected by this call.

void cputs (char * *s*)

display an ASCIIZ string.

Parameters:
s - the string

only the first 5 characters will be displayed.

direct-button.h File Reference

query button states. More...

Detailed Description

query button states.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

direct-ir.c File Reference

direct IR port access. More...

Defines

- #define **RX_BUF_SIZE**
  receive buffer size.
- #define **RX_OK**
  success in receiving.
- #define **RX_ERROR**
  failure in receiving.
- #define **TX_ACTIVE**
  transmission states.
- #define **TX_OK**
  transmission states.
- #define **TX_MISMATCH**
  transmission states.

Functions

- void **dir_rx_c** (void)
  C core for the rx byte received interrupt.
- void **dir_rx_handler** (void)
  assembler wrapper for the rx byte received interrupt.
- void **dir_rxerror_c** (void)
  C core for the rx error interrupt.
- void **dir_rxerror_handler** (void)
  assembler wrapper for the rx error interrupt.
- void **dir_tx_handler** (void)
  assembler tx byte sent interrupt. More...
- void **dir_txend_handler** (void)
  assembler tx end interrupt. More...

- void **dir_init** (void)
  initialize IR port.
- void **dir_shutdown** (void)
  shutdown IR port.
- size_t **dir_write** (void* const buf,size_t len)
  write to IR port, blocking. More...
- size_t **dir_read** (void* buf,size_t len)
  read from IR port, blocking. More...
- void **dir_fflush** (void)
  flush input buffer.

Variables

- unsigned char **dir_rx_buf** [RX_BUF_SIZE]
  rx buffer.
- unsigned char* **dir_rx_end**
  rx buffer end (last+1).
- unsigned char* volatile **dir_rx_write**
  ptr to next byte to write in rx irq.
- unsigned char* **dir_rx_read**
  ptr to next byte to read.
- volatile int **dir_rx_state**
  rx error flag.
- unsigned char* **dir_tx_end**
  tx buffer end (last+1).
- unsigned char* volatile **dir_tx_read**
  ptr to next byte to be sent.
- unsigned char* **dir_tx_verify**
  ptr to compare sent bytes with received ones. More...
- volatile int **dir_tx_state**
  transmission state.

Detailed Description

direct IR port access.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void dir_tx_handler (void)

assembler tx byte sent interrupt.

write next byte if there's one left, otherwise unhook irq.

void dir_txend_handler (void)

assembler tx end interrupt.

shutdown transmission

size_t dir_write (void * *const buf*, size_t *len*)

write to IR port, blocking.

Parameters:
buf - data to transmit
len - number of bytes to transmit

Returns:
number of bytes written, -1 indicates error.

size_t dir_read (void * *buf*, size_t *len*)

read from IR port, blocking.

Parameters:
buf - allocated receive buffer
len - number of bytes to read

Returns:
number of bytes read, -1 indicates error.

Variable Documentation

unsigned char* dir_tx_verify

ptr to compare sent bytes with received ones.

null: don’t compare.

---

direct-ir.h File Reference

direct IR port access interna. More...

Functions

- void **dir_init** (void)
  initialize IR port.
- void **dir_shutdown** (void)
  shutdown IR port.

Detailed Description

direct IR port access interna.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

direct-lcd.h File Reference

control the LCD display directly. More...

Defines

- #define **dlcd_show** (a)
  set a segment directly in the LCD buffer. More...
- #define **dlcd_hide** (a)
  clear a segment directly in the LCD buffer. More...
- #define **dlcd_store** (a)
  store the carry flag to a segment directly in the LCD buffer. More...
- #define **BYTE_OF** (a,b)
  helper macros.

Detailed Description

control the LCD display directly.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Warning:
Display updates are realized exclusively by lcd_refresh()

Define Documentation

#define dlcd_show(a)

set a segment directly in the LCD buffer.

Parameters:
a - the segment to set

#define dlcd_hide(a)

clear a segment directly in the LCD buffer.

Parameters:

a - the segment to clear

#define dlcd_store(a)

store the carry flag to a segment directly in the LCD buffer.

Parameters:

a - the segment to store to

this is highly useful in combination with bit_load(mask,bit)

---

direct-motor.c File Reference

direct motor access. More...

Functions

- void **dm_handler** (void)
  direct motor output handler. More...
- void **dm_init** (void)
  initialize motors.
- void **dm_shutdown** (void)
  shutdown motors.

Variables

- const unsigned char **dm_a_pattern** []
  motor drive patterns. More...
- const unsigned char dm_b_pattern []
- const unsigned char dm_c_pattern []
- MotorState **dm_a**
  motor A state.
- MotorState **dm_b**
  motor B state.
- MotorState **dm_c**
  motor C state.

Detailed Description

direct motor access.

Author(s):

Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void dm_handler (void)

direct motor output handler.

called by system timer in the 16bit timer OCIA irq

Variable Documentation

const unsigned char dm_a_pattern[]

motor drive patterns.

to be indexed with MotorDirections

See also:
MotorDirections

const unsigned char dm_b_pattern[]

to be indexed with MotorDirections

See also:
MotorDirections

const unsigned char dm_c_pattern[]

to be indexed with MotorDirections

See also:
MotorDirections

---

direct-sensor.c File Reference

direct sensor access. More...

Defines

- #define **DS_ALL_ACTIVE**
  all sensors active mode.
- #define **DS_ALL_PASSIVE**
  all sensors passive mode.
- #define **RANGE_SIZE**
  A/D values for the rotation sensor states.
- #define **IN_RANGE** (val)
  rotation sensor range matching function.

Enumerations

- enum **RotationState** { **STATE_0**, **STATE_1**, **STATE_2**, **STATE_3** }
  states for rotation state machine.

Functions

- void **ds_rotation_set** (unsigned* const sensor,int pos)
  set rotation to an absolute value. More...
- void **ds_rotation_handler** ()
  process rotation sensor on current A/D channel. More...
- void **ds_handler** (void)
  sensor A/D conversion IRQ handler.
- void **ds_init** (void)
  initialize sensor a/d conversion. More...
- void **ds_shutdown** (void)
  shutdown sensor a/d conversion. More...

Variables

- volatile unsigned char **ds_channel**
  current A/D channel.
- unsigned char **ds_activation**
  channel bitmask. 1-> active.
- unsigned char **ds_rotation**
  channel bitmask. 1-> rotation.
- volatile int **ds_rotations** [3]
  rotation sensor values.
- RotationState **rotation_state** [3]
  rotation state machine state.

Detailed Description

direct sensor access.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void ds_rotation_set (unsigned * *const sensor*, int *pos*)

set rotation to an absolute value.

Parameters:
sensor - the sensor address, can be &SENSOR_1, &SENSOR_2 or &SENSOR_3
pos - desired absolute position

axis should be inert during the function call

void ds_rotation_handler ()

process rotation sensor on current A/D channel.

See also:
ds_channel current channel (global input value)

void ds_init (void)

initialize sensor a/d conversion.

all sensors set to passive mode rotation tracking disabled

void ds_shutdown (void)

shutdown sensor a/d conversion.

all sensors set to passive mode

---

direct-sensor.h File Reference

direct sensor access interna. More...

Functions

- void **ds_init** (void)
  initialize sensor a/d conversion. More...
- void **ds_shutdown** (void)
  shutdown sensor a/d conversion. More...

Detailed Description

direct sensor access interna.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void ds_init (void)

initialize sensor a/d conversion.

all sensors set to passive mode rotation tracking disabled

void ds_shutdown (void)

shutdown sensor a/d conversion.

all sensors set to passive mode

direct-sound.c File Reference

direct sound access. More...

Functions

- void **ds_switcher** (void)
  the sound IRQ handler.
- void **ds_play** (unsigned char *sample, unsigned length)
  start playing sound. More...
- void **ds_stop** (void)
  stop playing sound.

Variables

- volatile unsigned* **ds_current**
  current byte playing & status.
- volatile unsigned char* **ds_buf_ptr**
  ptr to next byte to play.
- unsigned char * **ds_buf_end**
  ptrs to start/end of sample.
- unsigned **ds_loop_flag**
  flag to loop output.

Detailed Description

direct sound access.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void ds_play (unsigned char * *sample*, unsigned *length*)

start playing sound.

Parameters:

sample - an 1 bit / 8 kHz sample. util/sample-convert.pl will perform a conversion for standard wav files.
length - sample length in bytes

Warning:

output sounds horrible
cycles not implemented.

---

direct-sound.h File Reference

direct sound access. More...

Functions

- void **ds_play** (unsigned char *sample, unsigned length)
  start playing sound. More...
- void **ds_stop** (void)
  stop playing sound.

Detailed Description

direct sound access.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void ds_play (unsigned char * *sample*, unsigned *length*)

start playing sound.

Parameters:
sample - an 1 bit / 8 kHz sample. util/sample-convert.pl will perform a conversion for standard wav files.
length - sample length in bytes

Warning:
output sounds horrible
cycles not implemented.

h8.h File Reference

H8/3297 processor registers. More...

Detailed Description

H8/3297 processor registers.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

irq.h File Reference

RCX redirected IRQ vectors. More...

Detailed Description

RCX redirected IRQ vectors.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Lego Mindstorms RCX IRQ redirection vector table All redirected handlers can assume r6 to be saved All redirected handlers must return with rts, *not* rte.

Warning:
Incomplete.

kmain.c File Reference

Main kernel loop. More...

Functions

- int **main** (void)
  the user main(). More...
- void **kmain** (void)
  the beginning of everything. More...

Variables

- unsigned char* **firmware_string**
  firmware recognition string. More...

Detailed Description

Main kernel loop.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

int main (void)

the user main().

this is what you supply ;-)

void kmain (void)

the beginning of everything.

initially called by ROM

Variable Documentation

unsigned char* firmware_string

firmware recognition string.

the ROM checks for this string when validating new firmware

---

lcd.c File Reference

wrapper for ROM LCD number display functions. More...

Functions

- void **lcd_number** (int i,lcd_number_style n,lcd_comma_style c)
  show number on LCD display. More...

Detailed Description

wrapper for ROM LCD number display functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void lcd_number (int *i*, lcd_number_style *n*, lcd_comma_style *c*)

show number on LCD display.

Parameters:
i - the number
n - a number style
c - a comma style

lcd.h File Reference

ROM LCD control. More...

Defines

- #define **lcd_int** (i)
  display an integer in decimal.
- #define **lcd_unsigned** (u)
  display an unsigned value in decimal.
- #define **lcd_clock** (t)
  display a clock. More...
- #define **lcd_digit** (d)
  display a single digit right of the man symbol.

Enumerations

- enum **lcd_segment** { **man_stand**, **man_run**, **s1_select**, **s1_active**, **s2_select**, **s2_active**, **s3_select**, **s3_active**, **a_select**, **a_left**, **a_right**, **b_select**, **b_left**, **b_right**, **c_select**, **c_left**, **c_right**, unknown_1, circle, dot, dot_inv, **battery_x**, ir_half, **ir_full**, **everything** }
  LCD segment codes. More...
- enum **lcd_number_style** { digit, sign, unsign }
  LCD number display styles. More...
- enum **lcd_comma_style** { digit_comma, e0, e_1, e_2, e_3 }
  LCD comma display styles. More...

Functions

- void **lcd_show** (lcd_segment segment)
  show LCD segment. More...
- void **lcd_hide** (lcd_segment segment)
  hide LCD segment. More...
- void **lcd_number** (int i,lcd_number_style n,lcd_comma_style c)
  show number on LCD display. More...

- void **lcd_clear** (void)
  clear LCD display.
- void **lcd_refresh** (void)
  show LCD display contents to the world. More...

Detailed Description

ROM LCD control.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Warning:
Display updates are realized exclusively by lcd_refresh()

Define Documentation

#define lcd_clock(t)

display a clock.

passing an argument of 1015 will display 10.15

Enumeration Type Documentation

enum lcd_segment

LCD segment codes.

these are not to be confused with the codes defined in direct-lcd.h

circle and dot codes cycle. cycle state is preserved on powerdown.

each dot code should be invoked six times before using the other. mixing them will result in strange behaviour.

Enumeration values:

- **unknown_1**

  seemingly without effect. cycle reset?

- **circle**

  0..3 quarters: add one. 4 quarters: reset.

- **dot**

  0..4 dots: add a dot. 5 dots: reset.

- **dot_inv**

  0 dots: show 5. 1..4 dots: subtract one.

- **ir_half**

  the IR display values are mutually exclusive.

enum lcd_number_style

LCD number display styles.

note: signed and unsigned are taken by the C programming language

Enumeration values:

- **digit**

  single digit on the right.

- **sign**

  signed, no leading zeros.

- **unsign**

  unsigned, 0 displayed as 0000.

enum lcd_comma_style

LCD comma display styles.

Enumeration values:

- **digit_comma**

  single digit on the right.

- **e0**

whole.

- **e_1**

  10ths.

- **e_2**

  100ths.

- **e_3**

  1000ths, problematic with negatives.

Function Documentation

void lcd_show (lcd_segment *segment*)

show LCD segment.

Parameters:
segment - segment to show

void lcd_hide (lcd_segment *segment*)

hide LCD segment.

Parameters:
segment - segment to hide

void lcd_number (int *i*, lcd_number_style *n*, lcd_comma_style *c*)

show number on LCD display.

Parameters:
i - the number
n - a number style
c - a comma style

void lcd_refresh (void)

show LCD display contents to the world.

display updates are realized exclusively by calling this function.

mem.h File Reference

memory functions. More...

Detailed Description

memory functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

mm.c File Reference

dynamic memory management. More...

Functions

- size_t **mm_try_join** (size_t *ptr)
  check for free blocks after this one and join them if possible.
- void **mm_update_first_free** (size_t *start)
  update first free block pointer. More...
- void **mm_init** ()
  initialize memory management.
- void* **malloc** (size_t size)
  allocate a block of memory. More...
- void **free** (void *the_ptr)
  free a previously allocated block of memory. More...
- void* **calloc** (size_t nmemb, size_t size)
  allocate adjacent blocks of memory. More...
- void **mm_reaper** ()
  free all blocks allocated by the current process. More...

Variables

- size_t* **mm_first_free**
  first free block.
- sem_t **mm_semaphore**
  assures tasksafe operation.

Detailed Description

dynamic memory management.

Author(s):

Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void mm_update_first_free (size_t * *start*)

update first free block pointer.

Parameters:
start - pointer to owner field of a memory block to start with.

void * malloc (size_t *size*)

allocate a block of memory.

Parameters:
size - requested block size

Returns:
0 on error, else pointer to block.

void free (void * *the_ptr*)

free a previously allocated block of memory.

Parameters:
the_ptr - pointer to block

ever heard of free(software_paradigm)?

void * calloc (size_t *nmemb*, size_t *size*)

allocate adjacent blocks of memory.

Parameters:
nmemb - number of blocks
size - individual block size

Returns:
0 on error, else pointer to block

void mm_reaper ()

free all blocks allocated by the current process.

called by exit() and kmain().

mm.h File Reference

memory management interna. More...

Functions

- void **mm_init** ()
  initialize memory management.
- void **mm_reaper** ()
  free all blocks allocated by the current process. More...

Detailed Description

memory management interna.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void mm_reaper ()

free all blocks allocated by the current process.

called by exit() and kmain().

registers.h File Reference

registers cached by ROM functions. More...

Defines

- #define **ROM_PORT6**
  ROM-cached PORT6. More...
- #define **ROM_PORT6_DDR**
  ROM-cached PORT6_DDR.

Detailed Description

registers cached by ROM functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Define Documentation

#define ROM_PORT6

ROM-cached PORT6.

Port 6 is connected to both LCD and active sensor output. As lcd_refresh() is a ROM call, we need to update this location if active sensors are to remain active after a LCD refresh.

semaphore.c File Reference

POSIX 1003.1b semaphores for process synchronization. More...

Functions

- wakeup_t **sem_event_wait** (wakeup_t data)
  the semaphore event wakeup function for wait_event(). More...
- int **sem_wait** (sem_t * sem)
  wait on a semaphore. More...
- int **sem_trywait** (sem_t * sem)
  non-blocking check on a semaphore. More...
- int **sem_post** (sem_t * sem)
  increase semaphore count. More...

Detailed Description

POSIX 1003.1b semaphores for process synchronization.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

wakeup_t sem_event_wait (wakeup_t *data*)

the semaphore event wakeup function for wait_event().

Parameters:
data - pointer to the semaphore passed as a wakeup_t

int sem_wait (sem_t * *sem*)

wait on a semaphore.

Parameters:
sem - a valid semaphore

suspends the calling thread until the semaphore has non-zero count. It then atomically decreases the semaphore count.

implemented with wait_event().

int sem_trywait (sem_t * *sem*)

non-blocking check on a semaphore.

Parameters:
sem - a valid semaphore

a non-blocking variant of sem_wait. If the semaphore has non-zero count, the count is atomically decreased and sem_trywait immediately returns 0. If the semaphore count is zero, sem_trywait immediately returns with error EAGAIN.

this is IRQ handler safe.

int sem_post (sem_t * *sem*)

increase semaphore count.

Parameters:
sem - a valid semaphore

atomically increases the count of the semaphore. This function never blocks and can safely be used in asynchronous signal handlers.

---

semaphore.h File Reference

POSIX 1003.1b semaphores for process synchronization. More...

Defines

- #define **EAGAIN**
  an error code.

Typedefs

- typedef unsigned char **sem_t**
  the semaphore type.

Functions

- int **sem_wait** (sem_t * sem)
  wait on a semaphore. More...
- int **sem_trywait** (sem_t * sem)
  non-blocking check on a semaphore. More...
- int **sem_post** (sem_t * sem)
  increase semaphore count. More...

Detailed Description

POSIX 1003.1b semaphores for process synchronization.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

int sem_wait (sem_t * *sem*)

wait on a semaphore.

Parameters:
sem - a valid semaphore

suspends the calling thread until the semaphore has non-zero count. It then atomically decreases the semaphore count.

implemented with wait_event().

int sem_trywait (sem_t * *sem*)

non-blocking check on a semaphore.

Parameters:
sem - a valid semaphore

a non-blocking variant of sem_wait. If the semaphore has non-zero count, the count is atomically decreased and sem_trywait immediately returns 0. If the semaphore count is zero, sem_trywait immediately returns with error EAGAIN.

this is IRQ handler safe.

int sem_post (sem_t * *sem*)

increase semaphore count.

Parameters:
sem - a valid semaphore

atomically increases the count of the semaphore. This function never blocks and can safely be used in asynchronous signal handlers.

sound.h File Reference

ROM sound functions. More...

Functions

- void **sound_system** (unsigned nr)
  play one of the system sounds.
- int **sound_playing** ()
  is a sound playing? More...

Detailed Description

ROM sound functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Warning:
These functions will only work if ROM is allowed to handle the OCIA interrupt. legOS system time, motor control and task management depend upon handling it themselves.

Function Documentation

int sound_playing ()

is a sound playing?

Returns:
0=no, else yes

stdlib.h File Reference

reduced standard C library. More...

Functions

- void* **calloc** (size_t nmemb, size_t size)
  allocate adjacent blocks of memory. More...
- void* **malloc** (size_t size)
  allocate a block of memory. More...
- void **free** (void *ptr)
  free a previously allocated block of memory. More...

Detailed Description

reduced standard C library.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void * calloc (size_t *nmemb*, size_t *size*)

allocate adjacent blocks of memory.

Parameters:
nmemb - number of blocks
size - individual block size

Returns:
0 on error, else pointer to block

void * malloc (size_t *size*)

allocate a block of memory.

Parameters:
size - requested block size

Returns:
0 on error, else pointer to block.

void free (void * *ptr*)

free a previously allocated block of memory.

Parameters:
the_ptr - pointer to block

ever heard of free(software_paradigm)?

---

string.h File Reference

string functions. More...

Functions

- void **memcpy** (void* dest,void* src,size_t size)
  copy memory block from src to dest. More...
- void* **memset** (void* s,int c,size_t n)
  fill memory block with a byte value. More...
- void **mem_clear** (void* start,void* end)
  set memory block [start,end[to zero. More...
- char* **strcpy** (char *dest,const char *src)
  Copy null-terminated string from src to dest. More...
- int **strlen** (const char *s)
  Determine string length. More...
- int **strcmp** (const char *s1,const char *s2)
  Compare two strings. More...

Detailed Description

string functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void memcpy (void * *dest*, void * *src*, size_t *size*)

copy memory block from src to dest.

Parameters:
dest - destination
src - source
size - number of bytes to copy

Warning:
behaviour is undefined in case source and destination blocks overlap.

void * memset (void * *s*, int *c*, size_t *n*)

fill memory block with a byte value.

Parameters:
s - start
c - byte fill value
n - number of bytes to fill

void mem_clear (void * *start*, void * *end*)

set memory block [start,end[to zero.

Parameters:
start - start
end - end (non-inclusive).

Bugs and limitations:
FIXME: seems buggy. memset() usage advised.

char * strcpy (char * *dest*, const char * *src*)

Copy null-terminated string from src to dest.

Parameters:
src - source
dest - destination

Returns:
pointer to dest

int strlen (const char * *s*)

Determine string length.

Parameters:
s - string
s2 - second string

Returns:
string length

int strcmp (const char * *s1*, const char * *s2*)

Compare two strings.

Parameters:

s1 - first string
s2 - second string

Returns:

<0: s1<s2, ==0: s1==s2, >0: s1>s2

---

system.h File Reference

ROM system control functions. More...

Functions

- void **power_off** (void)
  enters software standby mode.
- void **power_init** (void)
  disables software standby mode so tm_idle_task() can use the sleep instruction.
- void **rom_reset** (void) __asm__ ("0x03ae") __attribute__ ((noreturn))
  erases legOS, returning control to ROM.

Detailed Description

ROM system control functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

systime.c File Reference

system time services. More...

Functions

- void **systime_handler** (void)
  system time handler for the 16bit timer OCIA irq. More...
- void **systime_init** (void)
  initialize system timer. More...
- void **systime_shutdown** (void)
  shutdown system timer. More...
- void **systime_set_switcher** (void* switcher)
  set task switcher vector. More...
- void **systime_set_timeslice** (unsigned char slice)
  set multitasking timeslice in ms. More...

Variables

- volatile time_t **sys_time**
  current system time in ms. More...
- unsigned char **tm_timeslice**
  task time slice.
- volatile unsigned char **tm_current_slice**
  current time remaining.
- void* **tm_switcher_vector**
  pointer to task switcher.

Detailed Description

system time services.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void systime_handler (void)

system time handler for the 16bit timer OCIA irq.

this is the pulse of the system. task switcher and motor driver calls are initiated here.

void systime_init (void)

initialize system timer.

task switcher initialized to empty handler motors turned off

void systime_shutdown (void)

shutdown system timer.

will also stop task switching and motors.

void systime_set_switcher (void * *switcher*)

set task switcher vector.

Parameters:
switcher - the switcher

void systime_set_timeslice (unsigned char *slice*)

set multitasking timeslice in ms.

Parameters:
slice - the timeslice. must be at least 5ms.

Variable Documentation

volatile time_t sys_time

current system time in ms.

Warning:
This is a 32 bit value which will overflow after 49.7 days of continuous operation.

time.h File Reference

Internal system time functions. More...

Functions

- void **systime_init** (void)
  initialize system timer. More...
- void **systime_shutdown** (void)
  shutdown system timer. More...
- void **systime_set_switcher** (void* switcher)
  set task switcher vector. More...
- void **systime_set_timeslice** (unsigned char slice)
  set multitasking timeslice in ms. More...

Detailed Description

Internal system time functions.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

void systime_init (void)

initialize system timer.

task switcher initialized to empty handler motors turned off

void systime_shutdown (void)

shutdown system timer.

will also stop task switching and motors.

void systime_set_switcher (void * *switcher*)

set task switcher vector.

Parameters:

switcher - the switcher

void systime_set_timeslice (unsigned char *slice*)

set multitasking timeslice in ms.

Parameters:

slice - the timeslice. must be at least 5ms.

---

tm.c File Reference

Task management. More...

Functions

- void **tm_switcher** (void)
  the task switcher. More...
- size_t* **tm_scheduler** (size_t *old_sp)
  the process scheduler. More...
- void **yield** (void)
  yield the rest of the current timeslice. More...
- int **tm_idle_task** (void)
  the idle task. More...
- void **tm_init** (void)
  init task management. More...
- void **tm_start** (void)
  start task management. More...
- pid_t **execi** (int (*code_start)(void),priority_t priority,size_t stack_size)
  execute a memory image. More...
- void **exit** (int code)
  exit task, returning code. More...
- wakeup_t **wait_event** (wakeup_t (*wakeup)(wakeup_t),wakeup_t data)
  suspend process until wakeup function is non-null. More...
- wakeup_t **tm_sleep_wakeup** (wakeup_t data)
  wakeup function for sleep. More...
- void **kill** (pid_t pid)
  kill a process. More...

Variables

- process_data **pd_single**
  single process process data.

- process_data* **cpid**
  ptr to current process data.
- process_data* **pd_idle**
  idle proces.
- unsigned **nb_tasks**
  number of tasks.

Detailed Description

Task management.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Contains the multitasking switcher and scheduler as well as library functions relating to task management.

Function Documentation

void tm_switcher (void)

the task switcher.

saves active context and passes sp to scheduler then restores new context from returned sp

size_t * tm_scheduler (size_t * *old_sp*)

the process scheduler.

Parameters:
old_sp - current task's current stack pointer

Returns:
new task's current stack pointer

actual context switches performed by tm_switcher (assembler wrapper)

void yield (void)

yield the rest of the current timeslice.

doesn't speed up the system clock.

int tm_idle_task (void)

the idle task.

infinite sleep instruction to conserve power.

void tm_init (void)

init task management.

called in single tasking mode before task setup.

void tm_start (void)

start task management.

called in single tasking mode after task setup

pid_t execi (int(* *code_start*)(void), priority_t *priority*, size_t *stack_size*)

execute a memory image.

Parameters:

code_start - start address of code to execute
priority - new task's priority
stack_size - stack size for new process

Returns:

-1: fail, else pid.

will return to caller in any case.

void exit (int *code*)

exit task, returning code.

Parameters:

code - The return code

FIXME: for now, scrap the code.

wakeup_t wait_event (wakeup_t(* *wakeup*)(wakeup_t), wakeup_t *data*)

suspend process until wakeup function is non-null.

Parameters:
wakeup - the wakeup function. called in task scheduler context.
data - argument passed to wakeup function by scheduler

Returns:
return value passed on from wakeup

wakeup_t tm_sleep_wakeup (wakeup_t *data*)

wakeup function for sleep.

Parameters:
data - time to wakeup encoded as a wakeup_t

void kill (pid_t *pid*)

kill a process.

Parameters:
pid - must be valid process ID, or undefined behaviour will result!

---

unistd.h File Reference

reduced UNIX standard library. More...

Functions

- pid_t **execi** (int (*code_start)(void),priority_t priority,size_t stack_size)
  execute a memory image. More...
- void **exit** (int code) __attribute__ ((noreturn))
  exit task, returning code. More...
- void **yield** (void)
  yield the rest of the current timeslice. More...
- wakeup_t **wait_event** (wakeup_t (*wakeup)(wakeup_t),wakeup_t data)
  suspend process until wakeup function is non-null. More...
- wakeup_t **tm_sleep_wakeup** (wakeup_t data)
  wakeup function for sleep. More...
- unsigned int **sleep** (unsigned int sec)
  delay execution allowing other tasks to run. More...
- unsigned int **msleep** (unsigned int msec)
  delay execution allowing other tasks to run. More...
- void **kill** (pid_t pid)
  kill a process. More...

Detailed Description

reduced UNIX standard library.

Author(s):
Markus L. Noga <noga@inrialpes.fr>

Function Documentation

pid_t execi (int(* *code_start*)(void), priority_t *priority*, size_t *stack_size*)

execute a memory image.

Parameters:

code_start - start address of code to execute
priority - new task's priority
stack_size - stack size for new process

Returns:

-1: fail, else pid.

will return to caller in any case.

void exit (int *code*)

exit task, returning code.

Parameters:

code - The return code

FIXME: for now, scrap the code.

void yield (void)

yield the rest of the current timeslice.

doesn't speed up the system clock.

wakeup_t wait_event (wakeup_t(* *wakeup*)(wakeup_t), wakeup_t *data*)

suspend process until wakeup function is non-null.

Parameters:

wakeup - the wakeup function. called in task scheduler context.
data - argument passed to wakeup function by scheduler

Returns:

return value passed on from wakeup

wakeup_t tm_sleep_wakeup (wakeup_t *data*)

wakeup function for sleep.

Parameters:

data - time to wakeup encoded as a wakeup_t

unsigned int sleep (unsigned int *sec*)

delay execution allowing other tasks to run.

Parameters:
sec - sleep duration in seconds

Returns:
number of seconds left if interrupted, else 0.

Bugs and limitations:
interruptions not implemented.

unsigned int msleep (unsigned int *msec*)

delay execution allowing other tasks to run.

Parameters:
msec - sleep duration in milliseconds

Returns:
number of milliseconds left if interrupted, else 0.

Bugs and limitations:
interruptions not implemented.

void kill (pid_t *pid*)

kill a process.

Parameters:
pid - must be valid process ID, or undefined behaviour will result!

---

legOS Compound Members

Here is a list of all documented class members with links to the classes they belong to:

- access : MotorState
- assembler : MotorState
- delta : MotorState
- dir : MotorState
- LightSensor() : LightSensor
- next : _process_data
- parent : _process_data
- position() : RotationSensor
- posPtr : RotationSensor
- prev : _process_data
- priority : _process_data
- pstate : _process_data
- ptr : Sensor
- RotationSensor() : RotationSensor
- Sensor() : Sensor
- sp_save : _process_data
- stack_base : _process_data
- sum : MotorState
- value() : LightSensor, Sensor
- wakeup : _process_data
- wakeup_data : _process_data
- ~RotationSensor() : RotationSensor
- ~Sensor() : Sensor

legOS File Members

Here is a list of all documented file members with links to the files they belong to:

- ascii_display_codes : conio.c
- battery : sensor.h
- BATTERY : direct-sensor.h
- brake : direct-motor.h
- BYTE_OF : direct-lcd.h
- calloc() : mm.c, stdlib.h
- circle : lcd.h
- cpid : tm.c
- cputc_native() : conio.c, conio.h
- cputc_native_0() : conio.c, conio.h
- cputc_native_1() : conio.c, conio.h
- cputc_native_2() : conio.c, conio.h
- cputc_native_3() : conio.c, conio.h
- cputc_native_4() : conio.c, conio.h
- cputc_native_5() : conio.c, conio.h
- cputs() : conio.c, conio.h
- cputw() : conio.c, conio.h
- DEFAULT_STACK_SIZE : tm.h
- delay() : conio.c, conio.h
- digit : lcd.h
- digit_comma : lcd.h
- dir_fflush() : direct-ir.c, direct-ir.h
- dir_init() : direct-ir.c, direct-ir.h
- dir_read() : direct-ir.c, direct-ir.h
- dir_rx_buf : direct-ir.c
- dir_rx_c() : direct-ir.c
- dir_rx_end : direct-ir.c
- dir_rx_handler() : direct-ir.c
- dir_rx_read : direct-ir.c
- dir_rx_state : direct-ir.c
- dir_rx_write : direct-ir.c
- dir_rxerror_c() : direct-ir.c
- dir_rxerror_handler() : direct-ir.c
- dir_shutdown() : direct-ir.c, direct-ir.h
- dir_tx_end : direct-ir.c
- dir_tx_handler() : direct-ir.c

- dir_tx_read : direct-ir.c
- dir_tx_state : direct-ir.c
- dir_tx_verify : direct-ir.c
- dir_txend_handler() : direct-ir.c
- dir_write() : direct-ir.c, direct-ir.h
- dlcd_hide : direct-lcd.h
- dlcd_show : direct-lcd.h
- dlcd_store : direct-lcd.h
- dm_a : direct-motor.c, direct-motor.h
- dm_a_pattern : direct-motor.c, direct-motor.h
- dm_b : direct-motor.c, direct-motor.h
- dm_b_pattern : direct-motor.c, direct-motor.h
- dm_c : direct-motor.c, direct-motor.h
- dm_c_pattern : direct-motor.c, direct-motor.h
- dm_handler() : direct-motor.c
- dm_init() : direct-motor.c, direct-motor.h
- dm_shutdown() : direct-motor.c, direct-motor.h
- dot : lcd.h
- dot_inv : lcd.h
- ds_activation : direct-sensor.c, direct-sensor.h
- DS_ALL_ACTIVE : direct-sensor.c
- DS_ALL_PASSIVE : direct-sensor.c
- ds_buf_end : direct-sound.c
- ds_buf_ptr : direct-sound.c
- ds_channel : direct-sensor.c
- ds_current : direct-sound.c
- ds_handler() : direct-sensor.c
- ds_init() : direct-sensor.c, direct-sensor.h
- ds_loop_flag : direct-sound.c
- ds_play() : direct-sound.c, direct-sound.h
- ds_rotation : direct-sensor.c, direct-sensor.h
- ds_rotation_handler() : direct-sensor.c
- ds_rotation_set() : direct-sensor.c, direct-sensor.h
- ds_rotations : direct-sensor.c, direct-sensor.h
- ds_shutdown() : direct-sensor.c, direct-sensor.h
- ds_stop() : direct-sound.c, direct-sound.h
- ds_switcher() : direct-sound.c
- e0 : lcd.h
- e_1 : lcd.h
- e_2 : lcd.h
- e_3 : lcd.h
- EAGAIN : semaphore.h
- execi() : tm.c, unistd.h

- exit() : tm.c, unistd.h
- firmware_string : kmain.c
- free() : mm.c, stdlib.h
- fwd : direct-motor.h
- hex_display_codes : conio.c
- IN_RANGE : direct-sensor.c
- ir_half : lcd.h
- kill() : tm.c, unistd.h
- kmain() : kmain.c
- lcd_clear() : lcd.h
- lcd_clock : lcd.h
- lcd_comma_style : lcd.h
- lcd_digit : lcd.h
- lcd_hide() : lcd.h
- lcd_int : lcd.h
- lcd_number() : lcd.c, lcd.h
- lcd_number_style : lcd.h
- lcd_refresh() : lcd.h
- lcd_segment : lcd.h
- lcd_show() : lcd.h
- lcd_unsigned : lcd.h
- LIGHT : direct-sensor.h
- LIGHT_MAX : direct-sensor.h
- LIGHT_RAW_BLACK : direct-sensor.h
- LIGHT_RAW_WHITE : direct-sensor.h
- main() : kmain.c
- malloc() : mm.c, stdlib.h
- MAX_SPEED : direct-motor.h
- mem_clear() : string.h
- memcpy() : string.h
- memset() : string.h
- MIN_SPEED : direct-motor.h
- mm_first_free : mm.c
- mm_init() : mm.c, mm.h
- mm_reaper() : mm.c, mm.h
- mm_semaphore : mm.c
- mm_try_join() : mm.c
- mm_update_first_free() : mm.c
- motor_a_dir() : direct-motor.h
- motor_a_speed() : direct-motor.h
- motor_b_dir() : direct-motor.h
- motor_b_speed() : direct-motor.h
- motor_c_dir() : direct-motor.h

- motor_c_speed() : direct-motor.h
- MotorDirection : direct-motor.h
- msleep() : unistd.h
- nb_tasks : tm.c
- NO_DIRECT_SOUND : config.h
- NO_EQUAL_PRIORITIES : config.h
- off : direct-motor.h
- P_DEAD : tm.h
- P_RUNNING : tm.h
- P_SLEEPING : tm.h
- P_WAITING : tm.h
- P_ZOMBIE : tm.h
- pd_idle : tm.c
- pd_single : tm.c
- pid_t : tm.h
- power_init() : system.h
- power_off() : system.h
- priority_t : tm.h
- process_data : tm.h
- pstate_t : tm.h
- RANGE_SIZE : direct-sensor.c
- rev : direct-motor.h
- ROM_PORT6 : registers.h
- ROM_PORT6_DDR : registers.h
- rom_reset() : system.h
- rotation_state : direct-sensor.c
- RotationState : direct-sensor.c
- RX_BUF_SIZE : direct-ir.c
- RX_ERROR : direct-ir.c
- RX_OK : direct-ir.c
- s1 : sensor.h
- s2 : sensor.h
- s3 : sensor.h
- sem_event_wait() : semaphore.c
- sem_post() : semaphore.c, semaphore.h
- sem_t : semaphore.h
- sem_trywait() : semaphore.c, semaphore.h
- sem_wait() : semaphore.c, semaphore.h
- SENSOR_1 : direct-sensor.h
- SENSOR_2 : direct-sensor.h
- SENSOR_3 : direct-sensor.h
- sign : lcd.h
- sleep() : unistd.h

- sound_playing() : sound.h
- sound_system() : sound.h
- strcmp() : string.h
- strcpy() : string.h
- strlen() : string.h
- sys_time : systime.c, time.h
- systime_handler() : systime.c
- systime_init() : systime.c, time.h
- systime_set_switcher() : systime.c, time.h
- systime_set_timeslice() : systime.c, time.h
- systime_shutdown() : systime.c, time.h
- time_t : time.h
- tm_current_slice : systime.c
- tm_idle_task() : tm.c, tm.h
- tm_init() : tm.c, tm.h
- tm_scheduler() : tm.c, tm.h
- tm_sleep_wakeup() : tm.c, unistd.h
- tm_start() : tm.c, tm.h
- tm_switcher() : tm.c, tm.h
- tm_switcher_vector : systime.c
- tm_timeslice : systime.c
- TX_ACTIVE : direct-ir.c
- TX_MISMATCH : direct-ir.c
- TX_OK : direct-ir.c
- unknown_1 : lcd.h
- unsign : lcd.h
- wait_event() : tm.c, unistd.h
- wakeup_t : tm.h
- yield() : tm.c, unistd.h