INAF - Osservatorio Astrofisico di Arcetri
Giano Project

Software structure of `server104`

E. Giani, C. Baffa, V.Biliotti

Generated by Doxygen 1.5.5 October 29, 2010

Arcetri internal report N° **3-2010**
Firenze, 10/2010

Abstract

We describe here the software structure of `server104`*, the controlling program of the embedded system of Giano infrared spectrometer.*

The embedded system is based on an industrial standard PC104 board, and it is controlled by a variant of Linux OS. On this framework runs the `server104` *program.*

Here we include the inner documentation produced by a software tools (Doxygen) which collect from the source code the huge amount of scattered documentation of this 20K line project.

Contents

1 Program Structure **1**

2 Data Structure Index **3**

2.1 Data Structures 3

3 File Index **5**

3.1 File List 5

4 Data Structure Documentation **7**

4.1 _s_FIFO Struct Reference 7

4.2 _SList Struct Reference 9

4.3 AnalogBoard Struct Reference 10

4.4 cmd_t Struct Reference 15

4.5 errTIM Struct Reference 16

4.6 event_struct Struct Reference 17

4.7 GPacket Struct Reference 19

4.8 multi_status Struct Reference 20

4.9 multiple Struct Reference 22

4.10 s_FIFO Struct Reference 24

4.11 sock_addr Struct Reference 25

4.12 sock_t Struct Reference 26

4.13 union_t Union Reference 28

5 File Documentation **29**

5.1 acq104.c File Reference 29

5.2 command.def File Reference 46

5.3 data104.c File Reference 52

5.4 define.h File Reference 55

5.5 exec104.c File Reference 87

5.6 gerrno.h File Reference 98

5.7 gversion.h File Reference 114
5.8 list104.c File Reference 116
5.9 lpt104.c File Reference 123
5.10 lpt104.h File Reference 130
5.11 mem104.c File Reference 137
5.12 packetio.c File Reference 157
5.13 serial.c File Reference 161
5.14 server104.c File Reference 167
5.15 server104.h File Reference 171
5.16 slist.h File Reference 184
5.17 sock104.c File Reference 187
5.18 socket.c File Reference 194
5.19 socket.h File Reference 200
5.20 spooler.c File Reference 203
5.21 string104.c File Reference 209
5.22 util104.c File Reference 212
5.23 win104.c File Reference 225

Chapter 1

Program Structure

Giano is an infrared spectrometer for the TNG telescope[1]. Its data acquisition system is an homebrew electronic and software compound loosely based on Fasti project[2].

The software component is composed of a low level daemon, `server104`, and a middle-ware, `gbridge`[3], which is in charge of external communication channels.

`server104` has been described in its general aspects in another document[4], and here we will give the inner informations needed to perform the maintenance on this software.

`server104` has two main mode of operation: the stand-alone menu mode, intended to be used for instrument maintenance and the daemon mode, intended as the normal operational mode.

The menu mode is structured as two different menu pages: one for low level operations and testing on electronics ad one for acquisition related tests, and can store results on local fits files.

The demon mode, after initialization, is structured as a loop which intercepts command coming from a pipe toward `gbridge`, and execute them. Data coming from the instrument are sent through a separate dedicated pipe, while warning ad error messages are sent back to the command pipe.

`server104` follows the Arcetri IR software conventions[5] and places its log files in the `/var/log/softir` directory, if it is available, or in the `/var/tmp` if it has not the right to write on the standard location.

[1] See http://www.tng.iac.es/ for details
[2] See http://www.arcetri.astro.it/irlab/fasti
[3] See "Gbridge: il software middle-ware di Giano", E.Giani, C.Baffa, Rapporto Interno di Arcetri N° 5/2010
[4] "Il programma `server104` ed il sistema embedded", C. Baffa, V. Biliotti, E. Giani, Rapporto Interno di Arcetri N° 2-2010
[5] See http://www.arcetri.astro.it/irlab/doc/std_places.pdf

Figure 1.1: Some humor about documentation...

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_s_FIFO (Structure containing a single submitted event data) 7
_SList (The SList struct is used for each element in the singly-linked list) 9
AnalogBoard (Structure with all analog electronics parameters) 10
cmd_t (Structure to associate command name to hex number) 15
errTIM (Structure containing the error message queue data) 16
event_struct (Structure containing the details of a submitted event) 17
GPacket (Header packet with info about destination and its type and length in bytes) 19
multi_status (Multiple read operation structure definition) 20
multiple (Multiple read sigle operation definition modification as discussed with E.Giani 8/2009) 22
s_FIFO (Structure containing the FIFO queue of submitted events) 24
sock_addr (Union with socket address) 25
sock_t (A data structure representing the socket) 26
union_t (Union defined in order to access both sock_t and cmd_t data from list) 28

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

acq104.c (Science data acquisition related routines) 29
command.def (Demon mode command definition) 46
data104.c (Routines to save data in fits format) 52
define.h (General symbols and board registers addresses and macros) 55
exec104.c (Main level operational routines) 87
gerrno.h (Defines the error codes and the tasks originating the errors) 98
gversion.h (Automatically generated GLOBAL version number) 114
list104.c (Functions to create and manipulate singly-linked lists) 116
lpt104.c (A Linux LPT user-space driver for driving the external LCD display) 123
lpt104.h (Definitions for lpt104) 130
mem104.c (Routines for I/O access to ISA bus) 137
packetio.c (Packet related routines, from gbridge) 157
serial.c (Serial handling for the HHameg Power supply) 161
server104.c (Main of the giano control program) 167
server104.h (Global variables and functions declarations) 171
slist.h (Singly-linked list type and functions) 184
sock104.c (High level I/O socket routines) 187
socket.c (Low level socket routines) 194
socket.h (Socket related include) 200
spooler.c (Low profile spooler for low priority task) 203
string104.c (Exended strings handling) 209
util104.c (Utility functions) 212
win104.c (Windowing routines for server104. Only active in menu mode) 225

Chapter 4

Data Structure Documentation

4.1 _s_FIFO Struct Reference

Structure containing a single submitted event data.

Data Fields

- time_t time_submitted
- time_t time_to_submit
- time_t delta_time
- int priority
- int event_index

4.1.1 Detailed Description

Structure containing a single submitted event data.

Definition at line 109 of file spooler.c.

4.1.2 Field Documentation

4.1.2.1 time_t _s_FIFO::time_submitted

Definition at line 110 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and f_sched().

4.1.2.2 time_t _s_FIFO::time_to_submit

Definition at line 111 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and f_sched().

4.1.2.3 time_t _s_FIFO::delta_time

Definition at line 112 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and f_sched().

4.1.2.4 int _s_FIFO::priority

Definition at line 113 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and f_sched().

4.1.2.5 int _s_FIFO::event_index

Definition at line 115 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), f_sched(), and s_sched().

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

- spooler.c (0.12)

4.2 _SList Struct Reference

The SList struct is used for each element in the singly-linked list.

`#include <slist.h>`

Data Fields

- void * data
- SList * next

4.2.1 Detailed Description

The SList struct is used for each element in the singly-linked list.

holds the element's data, which can be a pointer to any kind of data, or any integer value

Definition at line 82 of file slist.h.

4.2.2 Field Documentation

4.2.2.1 void* _SList::data

Definition at line 84 of file slist.h.

Referenced by close_all(), compare_cmd(), compare_socket(), compare_socktype(), dprint_slist(), slist_append(), slist_data_from_command(), slist_data_from_descriptor(), slist_data_from_socktype(), slist_free(), slist_nth_data(), slist_remove(), and slist_remove_disconnected().

4.2.2.2 SList* _SList::next

contains the link to the next element in the list.

Definition at line 86 of file slist.h.

Referenced by close_all(), dprint_slist(), MainLoop(), slist_append(), slist_delete_node(), slist_find_custom(), slist_free(), slist_length(), slist_nth_data(), slist_remove(), and slist_remove_disconnected().

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

- slist.h (2.3)

4.3 AnalogBoard Struct Reference

Structure with all analog electronics parameters.

`#include <server104.h>`

Data Fields

- int Id
- int Status
- int Filtro
- int Sensore
- int Link
- int Sequencer
- int V_reset
- int V_bias
- int Offset12
- int Offset34
- int VccOn
- int ErrVlevel
- int ErrOffset
- int ErrAddr
- int ErrAmplifier_
- int ErrAmplifier
- int ErrADConv
- char seqfile [80]
- int Prog_resnum
- int Prog_dryres
- int Prog_resclk
- int Prog_readclk
- int Prog_readdel
- int Prog_dit
- int Prog_fsync
- int Prog_lsync
- double Time_cycle

4.3.1 Detailed Description

Structure with all analog electronics parameters.

Definition at line 189 of file server104.h.

4.3.2 Field Documentation

4.3.2.1 int AnalogBoard::Id

analogic board identification number

Definition at line 190 of file server104.h.

Referenced by analog_boardId(), and initvar104().

4.3.2.2 int AnalogBoard::Status

analogic board status ST_IDLE/ST_DUMMY/ST_INTEG

Definition at line 191 of file server104.h.

Referenced by initvar104().

4.3.2.3 int AnalogBoard::Filtro

ADC filter used 1-2

Definition at line 192 of file server104.h.

Referenced by initvar104(), whichfilter(), and write_pk_param().

4.3.2.4 int AnalogBoard::Sensore

Sensor status on (1) /off (0)

Definition at line 193 of file server104.h.

Referenced by analog_boardStatus(), and initvar104().

4.3.2.5 int AnalogBoard::Link

optical link status: ok = 0x1111

Definition at line 194 of file server104.h.

Referenced by initvar104(), and link_status().

4.3.2.6 int AnalogBoard::Sequencer

sequencer status: 1 (running) 0 (idle)

Definition at line 195 of file server104.h.

Referenced by analog_boardStatus(), initvar104(), and sequencer_status().

4.3.2.7 int AnalogBoard::V_reset

Value of V_reset

Definition at line 196 of file server104.h.

Referenced by dlanalog(), initvar104(), program_DAC_Vlevel(), and write_pk_param().

4.3.2.8 int AnalogBoard::V_bias

Value of V_bias

Definition at line 197 of file server104.h.

Referenced by dlanalog(), initvar104(), program_DAC_Vlevel(), and write_pk_param().

4.3.2.9 int AnalogBoard::Offset12

offset 1 e 2

Definition at line 198 of file server104.h.

Referenced by dlanalog(), initvar104(), program_DAC_Vlevel(), and write_pk_param().

4.3.2.10 int AnalogBoard::Offset34

offset 3 e 4

Definition at line 199 of file server104.h.

Referenced by dlanalog(), initvar104(), program_DAC_Vlevel(), and write_pk_param().

4.3.2.11 int AnalogBoard::VccOn

VccOPTO, 5V, VA, -VA alimentation status

Definition at line 200 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.12 int AnalogBoard::ErrVlevel

error in Vreset/Vbias DA converter

Definition at line 201 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.13 int AnalogBoard::ErrOffset

error in offset12/offset34 DA converter

Definition at line 202 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.14 int AnalogBoard::ErrAddr

error in the address decoder

Definition at line 203 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.15 int AnalogBoard::ErrAmplifier_

error in negative output amplifier

Definition at line 204 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.16 int AnalogBoard::ErrAmplifier

error in positive output amplifier

Definition at line 205 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.17 int AnalogBoard::ErrADConv

error in the ADC converter

Definition at line 206 of file server104.h.

Referenced by analog_boardStatus().

4.3.2.18 char AnalogBoard::seqfile[80]

sequencer filename (max 80 chars)

Definition at line 207 of file server104.h.

4.3.2.19 int AnalogBoard::Prog_resnum

number of reset performed before first read

Definition at line 208 of file server104.h.

Referenced by compute_cycle(), compute_scan_time(), initvar104(), MenuLoop(), syntetize_multi(), syntetize_program(), and write_pk_param().

4.3.2.20 int AnalogBoard::Prog_dryres

number of reset with dummy read performed before first read

Definition at line 209 of file server104.h.

Referenced by compute_scan_time(), syntetize_program(), and write_pk_param().

4.3.2.21 int AnalogBoard::Prog_resclk

duration of pixel reset

Definition at line 210 of file server104.h.

Referenced by initvar104(), and write_pk_param().

4.3.2.22 int AnalogBoard::Prog_readclk

duration of pixel select during read

Definition at line 211 of file server104.h.

Referenced by compute_pix_readclk(), initvar104(), MenuLoop(), read_init_file(), read_pk_param(), syntetize_multi(), syntetize_program(), and write_pk_param().

4.3.2.23 int AnalogBoard::Prog_readdel

delay of conversion during read

Definition at line 212 of file server104.h.

Referenced by compute_pix_readclk(), initvar104(), MenuLoop(), read_init_file(), read_pk_param(), syntetize_multi(), syntetize_program(), and write_pk_param().

4.3.2.24 int AnalogBoard::Prog_dit

integration time in 1/100 of seconds

Definition at line 213 of file server104.h.

Referenced by handle_multi_acquisition(), initvar104(), MenuLoop(), and syntetize_program().

4.3.2.25 int AnalogBoard::Prog_fsync

Delay between end of frame and Fsync of next and Delay between end of Fsync and start of Lsync

Definition at line 214 of file server104.h.

Referenced by initvar104(), syntetize_multi(), and syntetize_program().

4.3.2.26 int AnalogBoard::Prog_lsync

Duration of Lsync and Delay between end of Lsync and start of Vclk

Definition at line 216 of file server104.h.

Referenced by initvar104(), syntetize_multi(), and syntetize_program().

4.3.2.27 double AnalogBoard::Time_cycle

total cycle time in seconds

Definition at line 218 of file server104.h.

Referenced by compute_cycle().

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

- server104.h (2.18)

4.4 cmd_t Struct Reference

structure to associate command name to hex number

`#include <slist.h>`

Data Fields

- int hex
- char name [30]

4.4.1 Detailed Description

structure to associate command name to hex number

Definition at line 66 of file slist.h.

4.4.2 Field Documentation

4.4.2.1 int cmd_t::hex

the command hexadecimal identify number

Definition at line 67 of file slist.h.

4.4.2.2 char cmd_t::name[30]

the command name in ASCII format

Definition at line 68 of file slist.h.

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

- slist.h (2.3)

4.5 errTIM Struct Reference

Structure containing the error message queue data.

Data Fields

- time_t err_time
- time_t err_min
- time_t err_max
- time_t last_clr

4.5.1 Detailed Description

Structure containing the error message queue data.

Definition at line 138 of file spooler.c.

4.5.2 Field Documentation

4.5.2.1 time_t errTIM::err_time

Definition at line 139 of file spooler.c.

Referenced by s_wrapper().

4.5.2.2 time_t errTIM::err_min

Definition at line 140 of file spooler.c.

Referenced by s_wrapper().

4.5.2.3 time_t errTIM::err_max

Definition at line 141 of file spooler.c.

Referenced by s_wrapper().

4.5.2.4 time_t errTIM::last_clr

Definition at line 142 of file spooler.c.

Referenced by s_wrapper().

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

- spooler.c (0.12)

4.6 event_struct Struct Reference

Structure containing the details of a submitted event.

Data Fields

- int priority
- time_t time_submitted
- int operation
- int arg_1
- char msg [80]
- float arg_2
- time_t min_duration
- time_t max_duration

4.6.1 Detailed Description

Structure containing the details of a submitted event.

Definition at line 91 of file spooler.c.

4.6.2 Field Documentation

4.6.2.1 int event_struct::priority

Definition at line 95 of file spooler.c.

Referenced by add_f_entry(), and add_s_entry().

4.6.2.2 time_t event_struct::time_submitted

Definition at line 96 of file spooler.c.

Referenced by add_f_entry(), and add_s_entry().

4.6.2.3 int event_struct::operation

Definition at line 97 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and s_wrapper().

4.6.2.4 int event_struct::arg_1

Definition at line 98 of file spooler.c.

Referenced by add_f_entry(), and add_s_entry().

4.6.2.5 char event_struct::msg[80]

Definition at line 99 of file spooler.c.

4.6.2.6 float event_struct::arg_2

Definition at line 100 of file spooler.c.

Referenced by add_f_entry(), and add_s_entry().

4.6.2.7 time_t event_struct::min_duration

Definition at line 102 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and s_wrapper().

4.6.2.8 time_t event_struct::max_duration

Definition at line 103 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), and s_wrapper().

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

- spooler.c (0.12)

4.7 GPacket Struct Reference

the header packet with info about destination and its type and length in bytes

`#include <socket.h>`

Data Fields

- unsigned short header [HDR_NUM]
- unsigned char payload [PCK_SIZE]
- unsigned short pck_sn
- unsigned short ack_sn

4.7.1 Detailed Description

the header packet with info about destination and its type and length in bytes

The strcuture describes the protocol packet to store the data to exchange between applications

Definition at line 130 of file socket.h.

4.7.2 Field Documentation

4.7.2.1 unsigned short GPacket::header[HDR_NUM]

Definition at line 134 of file socket.h.

Referenced by packetread(), and packetread_gbridge().

4.7.2.2 unsigned char GPacket::payload[PCK_SIZE]

the real data to exchange

Definition at line 136 of file socket.h.

Referenced by packetdecode(), packetread(), and packetread_gbridge().

4.7.2.3 unsigned short GPacket::pck_sn

the packet serial number

Definition at line 138 of file socket.h.

4.7.2.4 unsigned short GPacket::ack_sn

the ack packet serial number

Definition at line 140 of file socket.h.

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

- socket.h (2.5)

4.8 multi_status Struct Reference

multiple read operation structure definition

`#include <server104.h>`

Data Fields

- struct multiple multic [1024]
- int multic_item
- int integ_number
- double integ_total
- char multiprogram [128]

4.8.1 Detailed Description

multiple read operation structure definition

Definition at line 244 of file server104.h.

4.8.2 Field Documentation

4.8.2.1 struct multiple multi_status::multic[1024] [read]

multiple read operation structure

Definition at line 245 of file server104.h.

Referenced by handle_multi_acquisition(), multidummy(), multifile(), multivalid(), and syntetize_multi().

4.8.2.2 int multi_status::multic_item

number of item presents in multic

Definition at line 246 of file server104.h.

Referenced by handle_multi_acquisition(), multidummy(), multifile(), multivalid(), and syntetize_multi().

4.8.2.3 int multi_status::integ_number

number of integrations to be performed

Definition at line 247 of file server104.h.

Referenced by handle_multi_acquisition(), multidummy(), and multifile().

4.8.2.4 double multi_status::integ_total

total integration duration (sec)

Definition at line 248 of file server104.h.

Referenced by handle_multi_acquisition(), multidummy(), multifile(), and syntetize_multi().

4.8.2.5 char multi_status::multiprogram[128]

multiprogram file name

Definition at line 249 of file server104.h.

Referenced by initvar104(), MenuLoop(), multidummy(), and updatemenu().

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

- server104.h (2.18)

4.9 multiple Struct Reference

multiple read sigle operation definition modification as discussed with E.Giani 8/2009

`#include <server104.h>`

Data Fields

- char operation
- double duration
- int original_row
- int seq_row

4.9.1 Detailed Description

multiple read sigle operation definition modification as discussed with E.Giani 8/2009

Definition at line 232 of file server104.h.

4.9.2 Field Documentation

4.9.2.1 char multiple::operation

operation to be performed

Definition at line 233 of file server104.h.

Referenced by handle_multi_acquisition(), multidummy(), multifile(), multivalid(), and syntetize_multi().

4.9.2.2 double multiple::duration

integration time BEFORE operation

Definition at line 234 of file server104.h.

Referenced by handle_multi_acquisition(), multidummy(), multifile(), multivalid(), and syntetize_multi().

4.9.2.3 int multiple::original_row

row of original file

Definition at line 235 of file server104.h.

Referenced by multidummy(), multifile(), multivalid(), and syntetize_multi().

4.9.2.4 int multiple::seq_row

end location of operation in sequencer

Definition at line 236 of file server104.h.

Referenced by handle_multi_acquisition(), and syntetize_multi().

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

- server104.h (2.18)

4.10 s_FIFO Struct Reference

Structure containing the FIFO queue of submitted events.

Data Fields

- int top_fifo
- int bot_fifo
- struct _s_FIFO fifo [FIFO_WIDTH]

4.10.1 Detailed Description

Structure containing the FIFO queue of submitted events.

Definition at line 122 of file spooler.c.

4.10.2 Field Documentation

4.10.2.1 int s_FIFO::top_fifo

Definition at line 123 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), empty_fifo(), f_sched(), isfull_fifo(), isvoid_fifo(), and s_sched().

4.10.2.2 int s_FIFO::bot_fifo

Definition at line 124 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), empty_fifo(), f_sched(), isfull_fifo(), isvoid_fifo(), and s_sched().

4.10.2.3 struct _s_FIFO s_FIFO::fifo[FIFO_WIDTH] [read]

Definition at line 125 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), empty_fifo(), f_sched(), and s_sched().

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

- spooler.c (0.12)

4.11 sock_addr Struct Reference

union with socket address

`#include <socket.h>`

Data Fields

- struct sockaddr s
- struct sockaddr_in s_in
- struct sockaddr_un s_un

4.11.1 Detailed Description

union with socket address

Definition at line 85 of file socket.h.

4.11.2 Field Documentation

4.11.2.1 struct sockaddr sock_addr::s [read]

Definition at line 86 of file socket.h.

Referenced by local_server_new(), tcp_server_accept(), tcp_server_new(), and tcp_socket_connect().

4.11.2.2 struct sockaddr_in sock_addr::s_in [read]

Definition at line 87 of file socket.h.

Referenced by accept_connection(), close_listen_socket(), tcp_server_new(), and tcp_socket_connect().

4.11.2.3 struct sockaddr_un sock_addr::s_un [read]

Definition at line 88 of file socket.h.

Referenced by local_server_new().

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

- socket.h (2.5)

4.12 sock_t Struct Reference

A data structure representing the socket.

`#include <socket.h>`

Data Fields

- int index
- int sockfd
- int sockstatus
- int socktype
- short ioevents
- Callback_t iofunc
- Callback_t errfunc
- sock_addr sa

4.12.1 Detailed Description

A data structure representing the socket.

Definition at line 93 of file socket.h.

4.12.2 Field Documentation

4.12.2.1 int sock_t::index

index inside list memory pool

Definition at line 94 of file socket.h.

Referenced by accept_connection().

4.12.2.2 int sock_t::sockfd

The socket descriptor: an integer value

Definition at line 96 of file socket.h.

Referenced by accept_connection(), close_all(), close_giolamp(), close_listen_socket(), close_socket(), close_sockettype(), get_socket_descriptor(), local_server_new(), MainLoop(), multiswitch(), open_giolamp(), tcp_server_accept(), tcp_server_new(), tcp_socket_connect(), and update_event_sources().

4.12.2.3 int sock_t::sockstatus

the socket status (S104SockStatus representing the status of the connection socket status (CONNECTED/DISCONNECTED)

Definition at line 100 of file socket.h.

Referenced by accept_connection(), close_all(), close_listen_socket(), close_socket(), close_sockettype(), MainLoop(), multical(), open_giolamp(), and tcp_server_new().

4.12.2.4 int sock_t::socktype

the socket type (S104SockType) representing the type of data exchanged on the communication channel

Definition at line 104 of file socket.h.

Referenced by accept_connection(), close_all(), MainLoop(), and open_giolamp().

4.12.2.5 short sock_t::ioevents

bitwise combination representing a condition to watch for on an event source

Definition at line 108 of file socket.h.

Referenced by accept_connection(), open_giolamp(), and update_event_sources().

4.12.2.6 Callback_t sock_t::iofunc

The callback function used to handle the I/O operation on the connected channel.

Definition at line 112 of file socket.h.

Referenced by accept_connection(), and MainLoop().

4.12.2.7 Callback_t sock_t::errfunc

The callback function used to handle the error condition on the connected channel. callback function to handle I/O error

Definition at line 116 of file socket.h.

Referenced by accept_connection(), MainLoop(), and open_giolamp().

4.12.2.8 sock_addr sock_t::sa

sa is the remote host for clients, local host for servers

Definition at line 118 of file socket.h.

Referenced by accept_connection(), close_listen_socket(), local_server_new(), tcp_server_accept(), tcp_-server_new(), and tcp_socket_connect().

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

- socket.h (2.5)

4.13 union_t Union Reference

Union defined in order to access both sock_t and cmd_t data from list.

`#include <slist.h>`

Data Fields

- char data [MEM_CHUNK_SIZE]
- sock_t sock
- cmd_t cmd

4.13.1 Detailed Description

Union defined in order to access both sock_t and cmd_t data from list.

Definition at line 92 of file slist.h.

4.13.2 Field Documentation

4.13.2.1 char union_t::data[MEM_CHUNK_SIZE]

Definition at line 93 of file slist.h.

Referenced by accept_connection().

4.13.2.2 sock_t union_t::sock

Definition at line 94 of file slist.h.

Referenced by accept_connection().

4.13.2.3 cmd_t union_t::cmd

Definition at line 95 of file slist.h.

The documentation for this union was generated from the following file:

- slist.h (2.3)

Chapter 5

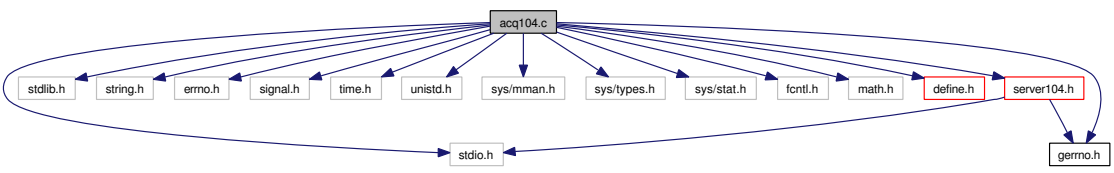
File Documentation

5.1 acq104.c File Reference

Science data acquisition related routines.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <math.h>
#include "define.h"
#include "server104.h"
#include "gerrno.h"
```

Include dependency graph for acq104.c:



Defines

- #define MIN(a, b) ((a) < (b) ? (a) : (b))
- #define MAX(a, b) ((a) > (b) ? (a) : (b))
- #define MAX_ERR 32
- #define Read_Pix()
- #define Clipper(a, b)

Functions

- int stop_CI_broadcast (void)
- int reset_fpga (void)
- int reset_fifo (int channel)
- int alloc_array_mem (int nword)
- int fifo_overflow (int channel)
- int frame_ready (void)
- int quadrant_ready (int channel)
- int fifo_ready (int channel, int local_verbosity)
- int compute_pix_readclk (int channel)
- int compute_scan_time (void)
- int check_group (void)
- int random_action (int duration, int percent_idle)
- int compute_cycle (int channel)
- int syntetize_program (int channel, int dummy_flag)
- int verify_sequencer (int channel)
- int program_sequencer (int channel, va_list args)
- int check_data (int framenum)
- int dump_data (int rows, int f_head)
- int read_testimage (int channel, int q2read, int nframes)
- int read_quadrant (int channel, int nframe)
- int reprogram_vlevel (int channel)
- int sys_running_status (int channel)
- int chkabort (void)
- int dummy_acquisition (int channel)
- int handle_oneboard_acquisition (int channel, int integ)
- int read_image (int nframe, int nready, int lettura)
- int handle_image_acquisition (int channel, int integ)
- void delay_transfer (void)
- int integriamo (void)
- int dummy_image (void)
- int syntetize_multi (int channel, va_list args)
- int handle_multi_acquisition (int channel)

Variables

- static int flag_ovf

5.1.1 Detailed Description

Science data acquisition related routines.

Definition in file acq104.c.

5.1.2 Define Documentation

5.1.2.1 #define MIN(a, b) ((a) < (b) ? (a) : (b))

Definition at line 175 of file acq104.c.

Referenced by frame_ready().

5.1.2.2 #define MAX(a, b) ((a) > (b) ? (a) : (b))

Definition at line 176 of file acq104.c.

Referenced by compute_scan_time(), handle_multi_acquisition(), MenuLoop(), packetdecode(), random_action(), and write_pk_param().

5.1.2.3 #define MAX_ERR 32

Max nuber of published errors

Definition at line 2199 of file acq104.c.

Referenced by read_image().

5.1.2.4 #define Read_Pix()

Value:

```
\
                tmp_pix = rmem104(curr_addr); \
                /* we count 0x000f occurences */ \
                if (0xf == tmp_pix) { \
                    cnt_15 ++; \
                } ;
```

Definition at line 2201 of file acq104.c.

Referenced by read_image().

5.1.2.5 #define Clipper(a, b)

Value:

```
\
   if ( a > b) { /* in correlate double sampling pixel would   */ \
                 /* result in values << - 500                  */ \
                 /* NOTE: sygnal are inverted (negated)        */ \
       a = 0;                                                    \
       b = 0;    /* to put pixel in ZERO state                 */ \
   } ;
```

Definition at line 2207 of file acq104.c.

Referenced by read_image().

5.1.3 Function Documentation

5.1.3.1 int stop_CI_broadcast (void)

stop at the ent of current operation

Stop all the channels of sequencer at the current istruction.

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1833 of file mem104.c.

References Analog, errlog(), error_code(), gb_errno, GB_ERROR, iolog(), MYERR, MYFALSE, MYTRUE, Opera, sequencer_status(), wmem104(), WR_BROADCAST, WR_RAM_SEQA, and WR_STOP_CI.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), and stop_kindly().

5.1.3.2 int reset_fpga (void)

Sends a broadcast reset of fifos.

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 200 of file acq104.c.

References Curframe, Curgroup, Err104, fifo_ready(), flag_ovf, iolog(), MYERR, MYFALSE, MYTRUE, NOERR, SVBCheck, Verbose, wmem104(), and WR_RST_BRD.

Referenced by initdev(), and reset_fifo().

5.1.3.3 int reset_fifo (int *channel*)

Executes a reset of the fifo related to the board specified by the argument

Parameters:

channel the analogic board number to operate with.
If channel = 5, it sends a broadcast reset command.

Returns:

MYTRUE on success.

See also:

reset_fpga

Definition at line 246 of file acq104.c.

References Curframe, Curgroup, fifo_ready(), flag_ovf, iolog(), l_function(), MYFALSE, MYTRUE, REG_OFFSET, reset_fpga(), SVBCheck, Verbose, wmem104(), and WR_RST_FIFOA.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), and handle_oneboard_-acquisition().

5.1.3.4 int alloc_array_mem (int *nword*)

Allocate memory on heap for a vector of nword (type short).

Parameters:

nword the number of word (16-bit) of dynamic memory to allocate

Returns:

MYTRUE on success.

Definition at line 297 of file acq104.c.

References array, array_size, GB_EMEMALLOC, gb_errno, MYFALSE, and MYTRUE.

Referenced by dummy_image(), handle_image_acquisition(), handle_multi_acquisition(), handle_-oneboard_acquisition(), and read_testimage().

5.1.3.5 int fifo_overflow (int *channel*)

Determines if the fifo is in overflow status.

Parameters:

channel the analogic board number to operate with

Returns:

MYTRUE if fifo overflows else MYFALSE. If errors in address reading it returns MYERR.

Definition at line 337 of file acq104.c.

References Err104, FIFO_OVERFLOW, GB_EFOVERRUN, GB_ENVALBOARD, gb_errno, MYERR, MYFALSE, MYTRUE, NOERR, RD_LSW_FIFOA, REG_OFFSET, and rmem104().

Referenced by frame_ready(), quadrant_ready(), and read_testimage().

5.1.3.6 int frame_ready (void)

Checks if a full frame (4 quadrants) is present. It returns the number of frames present in the FIFO.

Warning:

we assume we have read only full frames.

Returns:

number of frames if success, MYFALSE if no data, MYNOTREADY if fifo is not ready, MYERR if error present

Definition at line 375 of file acq104.c.

References errlog(), fifo_overflow(), fifo_ready(), flag_ovf, GB_EFOVERRUN, gb_errno, H_COL, H_-ROW, iolog(), MIN, MYERR, MYFALSE, MYNOTREADY, MYTRUE, NEXTRA, NHEADER, and Verbose.

Referenced by handle_image_acquisition(), and handle_multi_acquisition().

5.1.3.7 int quadrant_ready (int *channel*)

Checks if a full quadrant image is present. It returns the number of channel frames present in the FIFO.

Returns:

number of frames if success, MYFALSE if no data, MYNOTREADY if fifo is not ready, MYERR if error present

Definition at line 454 of file acq104.c.

References errlog(), fifo_overflow(), fifo_ready(), flag_ovf, GB_EFOVERRUN, gb_errno, H_COL, H_-ROW, iolog(), MYERR, MYFALSE, MYNOTREADY, MYTRUE, NEXTRA, NHEADER, and Verbose.

Referenced by handle_oneboard_acquisition().

5.1.3.8 int fifo_ready (int *channel*, int *local_verbosity*)

Checks if the specified fifo is ready to be read. If it is ready it returns the number of data present in the FIFO.

Parameters:

channel the analogic board number to operate with

local_verbosity the output verbosity

Returns:

number of data if success, MYNOTREADY if fifo is not ready, MYERR if error present

Definition at line 508 of file acq104.c.

References Err104, error_code(), FIFO_LSW_DAT, FIFO_MSW_DAT, FIFO_ORDIGIT, FIFO_-OVERFLOW, FIFO_READY, flag_ovf, GB_ENVALBOARD, gb_errno, GB_WARNING, iolog(), MY-ERR, MYNOTREADY, MYTRUE, NOERR, Opera, RD_LSW_FIFOA, RD_MSW_FIFOA, REG_-OFFSET, and rmem104().

Referenced by frame_ready(), quadrant_ready(), read_testimage(), reset_fifo(), and reset_fpga().

5.1.3.9 int compute_pix_readclk (int *channel*)

Compute the pixel readclk value from Base_scan value using the $*$ formula:

Base_Scan = 2 $*$ (readclk + 16) $*$ 62.5 ns.

readclk can assume value from 0 - 127.

Parameters:

channel the channel to operate on

Returns:

MYTRUE on success

Definition at line 593 of file acq104.c.

References Analog, Base_scan, compute_scan_time(), errlog(), error_code(), GB_EBADARG, GB_-WARNING, MYTRUE, Opera, AnalogBoard::Prog_readclk, and AnalogBoard::Prog_readdel.

Referenced by MenuLoop(), read_init_file(), syntetize_multi(), and syntetize_program().

5.1.3.10 int compute_scan_time (void)

compute the detector reading time (scan time) from Base_scan Scan_time units is in seconds Modification of 10/2009, we compute also Cycle_time and Transfer_time. Cycle time come from Dit (it is assumed > than Scan_time).

Returns:

MYTRUE on success

Definition at line 635 of file acq104.c.

References Acq_type, Analog, Base_scan, Cycle_time, D_lsync, Dit, ETH_TRANS_TIME, H_COL, H_-ROW, iolog(), MAX, MYTRUE, NEXTRA, AnalogBoard::Prog_dryres, AnalogBoard::Prog_resnum, R_-endframe, R_fsync_lsync, R_lsync_vclk, READ_FROM_104, Read_image_timeout, Scan_time, Tint, and Transfer_time.

Referenced by check_group(), compute_cycle(), compute_pix_readclk(), initvar104(), MenuLoop(), multidummy(), multivalid(), packetdecode(), random_action(), read_init_file(), syntetize_multi(), syntetize_-program(), and write_pk_param().

5.1.3.11 int check_group (void)

Verifies if the quantities Tint/Scan_time/Cycle_time/Transfer_time are compatible with the programmed Ngroup/Acq_type;

General schema of integrations:

```
Single read
|<resets><--Tint--><Scan_time>|<resets><--Tint--><Scan_time>|...
|<---------Cycle_time-------->|<---------Cycle_time-------->|
---classical read---
                             |<READ_FROM_104><ETH_TRANS_TIME>|...
                             |<----------Transfer_time------>|
Double read
|<resets><Scan_time><-Tint-><Scan_time>|<resets><Scan_time><-Tint-><Scan_time>|...
|<-------------Cycle_time------------->|<--------------Cycle_time------------>|
---classical read---
                   |<READ_FROM_104><ETH_TRANS_TIME>|<READ_FROM_104><ETH_TRANS_TIME>|......
                   |<----------Transfer_time------>|<----------Transfer_time------>|

So we define 3 different regimes
  Cycle_status = 1  Cycle_time << Transfer_time  -> Max 4 reads (2 int in DR)
  Cycle_status = 2  Cycle_time  < Transfer_time  -> Max few reads
  Cycle_status > 2  Cycle_time  > Transfer_time  -> unlimited reads
```

Returns:

MYTRUE on success

Definition at line 726 of file acq104.c.

References Acq_type, compute_scan_time(), Cycle_status, Cycle_time, Dit, errlog(), error_code(), GB_-EBADARG, gb_errno, GB_WARNING, iolog(), MYFALSE, MYTRUE, Ngroup, Opera, Scan_time, and Transfer_time.

Referenced by MenuLoop(), and packetdecode().

5.1.3.12 int random_action (int *duration*, int *percent_idle*)

Perform a random succession of actions, with percent_idle % of idle time

Parameters:

duration duration, in hours, of this test.

percent_idle portion, in percent, of idle time (1-90%)

Returns:

MYTRUE on success

Definition at line 835 of file acq104.c.

References Base_scan, Channel, compute_scan_time(), Dit, dlgetc(), execute_cmd(), handle_image_-acquisition(), iolog(), MAX, Menu, MYFALSE, MYTRUE, Ngroup, program_sequencer(), Scan_time, Tint, and updatemenu().

Referenced by MenuLoop().

5.1.3.13 int compute_cycle (int *channel*)

compute the cycle time for a syntetic program. Time_cycle is in seconds.

Parameters:

channel the analogic board number to operate with

Returns:

MYTRUE on success

Definition at line 935 of file acq104.c.

References Acq_type, Analog, compute_scan_time(), errlog(), error_code(), GB_ENVALBOARD, gb_-errno, GB_WARNING, H_COL, H_ROW, MYERR, MYTRUE, NBOARD, Opera, AnalogBoard::Prog_-resnum, Scan_time, and AnalogBoard::Time_cycle.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initvar104(), and syntetize_program().

5.1.3.14 int syntetize_program (int *channel*, int *dummy_flag*)

Programs the sequencer memory with a syntetic program from the content of Analog[].Prog_$*$. If Acq_-type is 0 (== single read), it DO NOT program the first read.

Warning:

: we program an integration time which is a little bit less than the **real** integration time, as detector integrates also during the read operation.

Parameters:

channel the analogic board number to operate with

dummy_flag if MYTRUE board is programmed as dummy integrations

Returns:

MYTRUE on success

The general structure of a syntetic program is:

- **Prog_resnum** iteration of Short Reset each of pixel duration of **Prog_resclk** (still not available)
- one iteration of Reset+Read, of pixel duration of **Prog_readclk** and with read delay of **Prog_readdel**
- multiple iterations of integration, for a total duration of **Prog_dit** in tens of milliseconds
- one iteration of **Read**, of pixel duration of **Prog_readc** and with read delay of **Prog_readdel**
- one iteration of restart
- delays related to **Fsync** and **Lsync** are specified by **R_endframe**, **R_fsync_lsync**, **D_lsync**, **R_-lsync_vclk**.

Times specified by **Prog_resclk**, **Prog_readclk** and **Prog_readdel** are

(value+16)∗61ns id est 0.98-8.78 usec (value $<$ 2^7-1).

Time specified by of **Prog_dit** is (value+1)∗10 msec, id est 10-163850 msec (value $<$ 2^14-1).

Note : we approximate the delay between end of Fsync and start of Lsync with R_fsync_lsync (ignoring the D_lsync component) as usually we should have R_fsync_lsync (max 4msec) $>>$ D_lsync (max 0.016 msec)

Definition at line 987 of file acq104.c.

References Acq_type, ADCRAMSIZE, Analog, Channel, COM_INTEGDELAY, COM_READ, COM_RESETDRY, COM_RESETREAD, COM_RESTART, COM_SHORTRESET, compute_cycle(), compute_pix_readclk(), compute_scan_time(), D_lsync, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_ESEQVERIFY, GB_ESPROGLONG, GB_ESPROGSHORT, GB_WARNING, iolog(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, AnalogBoard::Prog_dit, AnalogBoard::Prog_dryres, AnalogBoard::Prog_fsync, AnalogBoard::Prog_lsync, AnalogBoard::Prog_readclk, AnalogBoard::Prog_-readdel, AnalogBoard::Prog_resnum, R_endframe, R_fsync_lsync, R_lsync_vclk, rmem104(), Tint, Ver-bose, wmem104(), WR_FSYNC_TIM, WR_LSYNC_TIM, WR_RAM_SEQA, WR_RAM_SEQB, WR_-RAM_SEQC, and WR_RAM_SEQD.

Referenced by dummy_acquisition(), and program_sequencer().

5.1.3.15 int verify_sequencer (int *channel*)

Performs sequencer program verication.

Parameters:

channel the analogic board number to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1225 of file acq104.c.

References Analog, errlog(), error_code(), GB_EFILEOPEN, GB_ENVALBOARD, gb_errno, GB_-ESEQVERIFY, GB_ESPROGFORMAT, GB_WARNING, iolog(), MYERR, MYTRUE, NBOARD, Opera, rmem104(), Verbose, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_-SEQD.

Referenced by program_sequencer().

5.1.3.16 int program_sequencer (int *channel*, va_list *args*)

Programs the sequencer memory the contents of file whose name sits in Analog[channel-1].seqfile.

It handles Quadrant by means of wrapper.

Parameters:

channel the analogic board number to operate with

args the va_list arguments of the function

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1296 of file acq104.c.

References ADCRAMSIZE, Analog, BUFLEN, errlog(), error_code(), execute_cmd(), GB_-ENVALBOARD, gb_errno, GB_ESPROGFORMAT, GB_ESPROGLONG, GB_WARNING, iolog(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, stop_idle(), syntetize_program(), Verbose, verify_-sequencer(), wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and zero_seq_mem().

Referenced by handle_image_acquisition(), handle_oneboard_acquisition(), MenuLoop(), packetdecode(), random_action(), and write_pk_param().

5.1.3.17 int check_data (int *framenum*)

Checks the autotest pattern.

Data are arranged as row of NHEADER+1024 16-bits words.

Usually NHEADER is equal to 4. The 4 words of row header are: 0xffff framenum rownum 0x0 Data are a shifting bit in a 17 state ring:

1 2 4 8 10 ... 8000 0 (hex) and so on

Parameters:

framenum the number of the frame to check

Returns:

the error counts.

Definition at line 1396 of file acq104.c.

References array, errlog(), error_code(), GB_EFDATA, GB_EFHEADER, gb_errno, GB_WARNING, iolog(), l_error(), l_function(), Ncol, NHEADER, Nrow, Opera, and Verbose.

Referenced by read_testimage().

5.1.3.18 int dump_data (int *rows*, int *f_head*)

Dump the first rows lines of data in the global 'array' prints only first 16 bytes of data.

If f_head it includes also the 4 words of row header.

Parameters:

rows the number of row to print as output

f_head if > 0 it includes the row header

Returns:

MYTRUE on success.

Definition at line 1497 of file acq104.c.

References array, iolog(), l_function(), MYFALSE, MYTRUE, Ncol, and Verbose.

Referenced by read_quadrant(), and read_testimage().

5.1.3.19 int read_testimage (int *channel*, int *q2read*, int *nframes*)

Reads an image from the ISA bus. (C.Baffa E.Giani).

If channel == 5 (broadcast) the image is read from the specified q2read channel.

Parameters:

channel the analogic board number

q2read analogic board number to read from

nframes the number of frames to acquire

Returns:

MYTRUE on success.

Definition at line 1555 of file acq104.c.

References alloc_array_mem(), array, check_data(), chkabort(), dump_data(), elapsed_time(), Err104, errlog(), error_code(), fifo_overflow(), fifo_ready(), flag_ovf, GB_EFOVERRUN, GB_EINTERNAL, GB_EMMEMIO, gb_errno, GB_ERROR, GB_ESAVEIMG, GB_ETIMEOUT, get_current_time(), iolog(), l_error(), l_function(), Menu, MYERR, MYFALSE, MYNOTREADY, MYTRUE, Ncol, NHEADER, NOERR, Nrow, Opera, RD_DAT_FIFOA, REG_OFFSET, rmem104(), save_data(), TIMEOUT_READ, Verbose, wmem104(), WR_RST_BRD, WR_RST_FIFOA, WR_TST_BRD, and WR_TST_FIFOA.

Referenced by MenuLoop(), and packetdecode().

5.1.3.20 int read_quadrant (int *channel*, int *nframe*)

Reads an image from the ISA bus. (E.Giani).o

This routine performa a raw read, do no attempt to sysncronyze or to understand data structure, and, if enabled data will be saved as is.

Parameters:

channel the analogic board number

nframe the single frame number

Returns:

MYTRUE on success.

Definition at line 1771 of file acq104.c.

References ACQ_TASK, array, array_size, Curframe, dump_data(), elapsed_time(), errlog(), error_code(), FRAME_READY, GB_EIMGSYNC, gb_errno, GB_ERROR, GB_ESAVEIMG, GB_WARNING, get_current_time(), H_COL, H_ROW, infolog(), iolog(), l_error(), l_function(), Menu, MYFALSE, MYTRUE, Ncol, NEXTRA, NHEADER, Nrow, Opera, RD_DAT_FIFOA, REG_OFFSET, rmem104(), save_data(), send_image(), and Verbose.

Referenced by handle_image_acquisition(), and handle_oneboard_acquisition().

5.1.3.21 int reprogram_vlevel (int *channel*)

Checks if alimentation log is empty.

If there are bytes, it means the alimentation has gone off and then on.

So we need to reprogram the voltage level Vreset, Vbias, Offset12 and Offset34.

Parameters:

channel the analogic board number to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1868 of file acq104.c.

References BRD_ADDR_STEP, Err104, execute_cmd(), iolog(), MYERR, MYTRUE, NOERR, program_DAC_Vlevel(), RD_LOG_STAT, rmem104(), Verbose, WR_OFF1_2, WR_OFF3_4, WR_RAM_SEQ, WR_VBIAS, and WR_VRESET.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), and handle_oneboard_acquisition().

5.1.3.22 int sys_running_status (int *channel*)

Calls the routine to get the board status and checks:

- if the optical link is active

- if the sensor is on or off

This routine is used during acquisition because it controls the sensor alimentation. If this is down, we have to abort acquisition.

It handles Quadrant by means of wrapper.

Parameters:

channel the board (channel) to work with

Returns:

MYTRUE if the sequencer board is running, MYFALSE if the sequencer is still because of something going wrong in alimentation, MYERR on errors.

Definition at line 1912 of file acq104.c.

References Analog, analog_boardStatus(), errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_-ERROR, GB_ESYSNOTOP, GB_WARNING, MYERR, MYFALSE, MYTRUE, NBOARD, NoiseMode, and Opera.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), and handle_oneboard_-acquisition().

5.1.3.23 int chkabort (void)

Checks if exists a request of aborting the acquisition process.

Returns:

MYTRUE if there is an abort acquisition request, else MYFALSE.

Definition at line 1956 of file acq104.c.

References dlgetc(), iolog(), mainloop_iterate(), Menu, MYFALSE, MYTRUE, Run, SAS_ABORT, and SAS_STOP.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), integriamo(), read_testimage(), and test_seq_mem().

5.1.3.24 int dummy_acquisition (int *channel*)

Programs and starts dummy acquisitions on channel `channel`

Parameters:

channel the channel to operate on

Returns:

MYTRUE on success else MYERR on errors

Definition at line 1989 of file acq104.c.

References Channel, enable_sensor(), errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_-WARNING, IDLE_RUN, IDLE_STOP, IdleStatus, iolog(), MYERR, MYTRUE, Opera, start_sequencer(), stop_idle(), syntetize_program(), and treset_program().

Referenced by MenuLoop(), and packetdecode().

5.1.3.25 int handle_oneboard_acquisition (int *channel*, int *integ*)

Handles channel mode acquisition from a single board.

Note:

As this routine is intended for laboratory test mode, we do not start dummy acquisitions.

Parameters:

channel the channel to acquire

integ the total number of integrations

Returns:

MYTRUE on success or MYFALSE on errors or if acquisition process aborts.

Definition at line 2039 of file acq104.c.

References abort_sequencer(), ACQ_TASK, alloc_array_mem(), Analog, analog_boardStatus(), chk-abort(), compute_cycle(), Curframe, Curgroup, dump_acquisition_status(), enable_sensor(), errlog(), error_code(), execute_cmd(), flag_ovf, FRAME_STARTED, GB_ENVALBOARD, gb_errno, GB_-ERROR, GB_ETIMEOUT, H_COL, H_ROW, infolog(), iolog(), l_function(), l_opera(), link_error(), Menu, MYERR, MYFALSE, MYNOTREADY, MYTRUE, NEXTRA, NHEADER, NoiseMode, Opera, printstatus(), program_sequencer(), quadrant_ready(), read_quadrant(), reprogram_vlevel(), reset_fifo(), Run, SAS_BUSY, SAS_RUNNING, SAS_STOP, start_sequencer(), stop_idle(), stop_sequencer(), sys_-running_status(), WR_RAM_SEQ, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_-RAM_SEQD.

Referenced by handle_image_acquisition(), and MenuLoop().

5.1.3.26 int read_image (int *nframe*, int *nready*, int *lettura*)

Reads the frame(s) stored in the buffer fifo

Reads the frame(s) stored in the buffer fifo. If single read, we read only the second one, and we assume the first is NOT executed.

We implement a rough form of resynchronization of data rows. We also handle multiple read, according to global Run status variable. The parameter lettura is the indicator of first(1)/second(2) read, while in multiple read it is the sequential number of read.

Parameters:

nframe the frame number

nready the number of full frames stored in the buffer fifo

lettura indicator of first(1)/second(2) or multiple read.

Returns:

Upon successful completion it returns MYTRUE, otherwise MYFALSE.

Definition at line 2235 of file acq104.c.

References ACQ_TASK, Acq_type, array, array_size, BASE_VAL, Clipper, Curframe, Curgroup, elapsed_time(), errlog(), error_code(), Extra_sub, FRAME_READY, GB_EBADARG, GB_EIMGSYNC,

GB_EINVALMEM, GB_ENVALOPT, GB_EPIXLESS, GB_EPIXMORE, gb_errno, GB_ERROR, GB_-ESAVEIMG, GB_ETIMEOUT, GB_WARNING, get_current_time(), H_COL, H_ROW, infolog(), iolog(), l_error(), l_function(), link_error(), MAX_ERR, Menu, MYFALSE, MYTRUE, N_COL, N_ROW, Ncol, NEXTRA, NHEADER, NLOOSE, Nrow, Opera, RD_DAT_FIFOA, Read_Pix, REG_OFFSET, rmem104(), Run, SAS_MULTI, save_data(), send_image(), SVBCheck, TIMEOUT_READ, and Verbose.

Referenced by handle_image_acquisition(), and handle_multi_acquisition().

5.1.3.27 int handle_image_acquisition (int *channel*, int *integ*)

This functions executes the initial setup and preliminary checks for the acquisition process.

It then starts the sequencer of all the analogic boards and waits for a complete frame.

When a **full** frame is ready to be read, it calls the routine read_image() to collect the data.

Parameters:

channel the board (channel) to operate with.

integ the number of integrations to be performed.

Returns:

MYTRUE on success, MYFALSE on error or abort.

Note:

if channel !=5 (full frame) it is called the function handle_oneboard_acquisition().

Definition at line 2670 of file acq104.c.

References abort_broadcast(), ACQ_TASK, Acq_type, alloc_array_mem(), Analog, Channel, chkabort(), compute_cycle(), Curframe, Curgroup, Dit, dump_acquisition_status(), elapsed_time(), enable_sensor(), errlog(), error_code(), execute_cmd(), flag_ovf, frame_ready(), FRAME_STARTED, gb_errno, GB_-ERROR, GB_ETIMEOUT, get_current_time(), handle_oneboard_acquisition(), infolog(), iolog(), l_-function(), l_opera(), link_error(), Menu, MYERR, MYFALSE, MYNOTREADY, MYTRUE, N_-COL, N_ROW, NoiseMode, Opera, printstatus(), program_sequencer(), read_image(), Read_image_-timeout, read_quadrant(), reprogram_vlevel(), reset_fifo(), Run, SAS_BUSY, SAS_RUNNING, SAS_-STOP, start_broadcast(), stop_broadcast(), stop_CI_broadcast(), stop_idle(), SynchroTest, sys_running_-status(), treset_program(), and Verbose.

Referenced by MenuLoop(), packetdecode(), and random_action().

5.1.3.28 void delay_transfer (void)

Definition at line 2933 of file acq104.c.

Referenced by dummy_image().

5.1.3.29 int integriamo (void)

Definition at line 2952 of file acq104.c.

References chkabort(), Dit, dump_acquisition_status(), elapsed_time(), get_current_time(), iolog(), MYTRUE, and Run.

Referenced by dummy_image().

5.1.3.30 int dummy_image (void)

Definition at line 2978 of file acq104.c.

References alloc_array_mem(), array, Channel, Curframe, delay_transfer(), errlog(), FRAME_STARTED, gb_errno, H_COL, H_ROW, infolog(), integriamo(), MYFALSE, N_COL, N_ROW, NEXTRA, Ngroup, NHEADER, Run, SAS_BUSY, SAS_RUNNING, SAS_STOP, and send_image().

Referenced by packetdecode().

5.1.3.31 int syntetize_multi (int *channel*, va_list *args*)

Programs the sequencer memory with a syntetic program from the content of multistatus struct.

The general structure of a syntetic program is specified by the content of multistatus.multic array, generated from a multisample definition file. Here we quote a sample file:

```
    ############################################################
    128 R # 128 reset. Ogni files comincia con un reset

    0. l    # prima lettura comprensiva di reset
    12.24 L # seconda lettura dopo 12.24 secondi - il tempo di scansione
    25.19 L # terza lettura

    11.00 A # Calibrazione: accensione lampada A, integrazione e lettura
    10.00 B # Calibrazione: accensione lampada B, integrazione e lettura
    30.00 A # Calibrazione: accensione lampada A, integrazione e lettura

    12.00 L # settima lettura
    25.00 L # lettura
    12.00 L # lettura
    25.00 L # lettura

    10.00 B # Calibrazione: accensione lampada B, integrazione e lettura
    Z # Fine programma e restart

*
```

For First read and for restart, it is legal not to specify a duration (it get a default value of zero).

Reads can either be specified by Uppercase L (normal read) or lovercase l (reset-read, normally menat for first read)

Parameters:

channel the analogic board number to operate with

args optional parameters. Can contine the multiprogram filename

Returns:

MYTRUE on success

Note : we approximate the delay between end of Fsync and start of Lsync with R_fsync_lsync (ignoring the D_lsync component) as usually we should have R_fsync_lsync (max 4msec) >> D_lsync (max 0.016 msec)

Definition at line 3076 of file acq104.c.

References ADCRAMSIZE, Analog, BUFLEN, Channel, COM_INTEGDELAY, COM_READ, COM_-RESETDRY, COM_RESETREAD, COM_RESTART, COM_SHORTRESET, compute_pix_readclk(),

compute_scan_time(), D_lsync, multiple::duration, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_ESEQVERIFY, GB_ESPROGLONG, GB_ESPROGSHORT, GB_WARNING, multi_status::integ_total, iolog(), MULTI_VALID, multi_status::multic, multi_status::multic_item, multistatus, MYERR, MYFALSE, MYTRUE, NBOARD, Opera, multiple::operation, multiple::original_row, AnalogBoard::Prog_fsync, AnalogBoard::Prog_lsync, AnalogBoard::Prog_readclk, AnalogBoard::Prog_readdel, AnalogBoard::Prog_resnum, R_endframe, R_fsync_lsync, R_lsync_vclk, rmem104(), Scan_time, multiple::seq_row, Verbose, wmem104(), WR_FSYNC_TIM, WR_LSYNC_TIM, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_SEQD.

Referenced by handle_multi_acquisition(), MenuLoop(), and packetdecode().

5.1.3.32 int handle_multi_acquisition (int *channel*)

This functions executes the initial setup and preliminary checks for the acquisition process in multi read mode .

It then starts the sequencer of all the analogic boards and waits for a complete frames.

When a **full** frame is ready to be read, it calls the routine read_image() to collect the data.

Parameters:

channel the board (channel) to operate with.

Returns:

MYTRUE on success, MYFALSE on error or abort.

Definition at line 3334 of file acq104.c.

References abort_broadcast(), ACQ_TASK, alloc_array_mem(), Analog, Channel, chkabort(), compute_cycle(), Curframe, Curgroup, Dit, dump_acquisition_status(), multiple::duration, elapsed_time(), enable_sensor(), errlog(), error_code(), execute_cmd(), flag_ovf, FRAME_MULTI, frame_ready(), GB_ENVALBOARD, gb_errno, GB_ERROR, GB_ETIMEOUT, get_current_time(), infolog(), multi_status::integ_number, multi_status::integ_total, iolog(), ismultiinteg(), l_function(), l_opera(), link_error(), MAX, Menu, multi_status::multic, multi_status::multic_item, multistatus, MYERR, MYFALSE, MYNOTREADY, MYTRUE, N_COL, N_ROW, Ngroup, NoiseMode, open_giolamp(), Opera, multiple::operation, printstatus(), AnalogBoard::Prog_dit, read_image(), reprogram_vlevel(), reset_fifo(), Run, SAS_MULTI, SAS_RUNNING, SAS_STOP, Scan_time, multiple::seq_row, sequencer_counter(), start_broadcast(), stop_broadcast(), stop_CI_broadcast(), stop_idle(), syntetize_multi(), sys_running_status(), treset_program(), and Verbose.

Referenced by MenuLoop(), and packetdecode().

5.1.4 Variable Documentation

5.1.4.1 int flag_ovf [static]

flag to signal fifo overflow

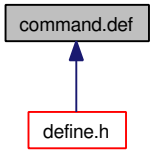
Definition at line 188 of file acq104.c.

Referenced by fifo_ready(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), quadrant_ready(), read_testimage(), reset_fifo(), and reset_fpga().

5.2 command.def File Reference

Demon mode command definition.

This graph shows which files directly or indirectly include this file:



Defines

- #define FILLMEM0 0x0101
- #define DUMPMEM 0x0102
- #define READPARM 0x0104
- #define WRITEPARM 0x0105
- #define LOADOBJ 0x0108
- #define LOADWAVE 0x0109
- #define DIT 0x0200
- #define GROUP 0x0201
- #define DOUBLE 0x0202
- #define QUADRANTS 0x0203
- #define NOISE 0x0205
- #define SYNCHRO 0x0206
- #define SVBTEST 0x0207
- #define SVBCHECK 0x0208
- #define SEQMEM 0x0209
- #define FIFOTST 0x020A
- #define EXPERT 0x020B
- #define RUN 0x0300
- #define START 0x0301
- #define STOP 0x0302
- #define ABORT 0x0303
- #define INTEGRA 0x0304
- #define FREERUN 0x0305
- #define MULTI 0x0306
- #define SOCKDS9 0x0309
- #define DRYRUN 0x0307
- #define FLIPSVB 0x0308
- #define REINIT 0x0310
- #define STATUS 0x0400
- #define READLOG 0x0410
- #define VERBOSE 0x0420
- #define MSGLEVEL 0x0430
- #define DUMMYACQ 0x0444
- #define NOP 0x04FF

5.2.1 Detailed Description

Demon mode command definition.

Definition in file command.def.

5.2.2 Define Documentation

5.2.2.1 #define FILLMEM0 0x0101

Command for filling sequencer memory with zeros

Definition at line 49 of file command.def.

Referenced by packetdecode().

5.2.2.2 #define DUMPMEM 0x0102

Command for dumping sequencer memory

Definition at line 50 of file command.def.

Referenced by packetdecode().

5.2.2.3 #define READPARM 0x0104

Command for reading waveform parameters

Definition at line 51 of file command.def.

Referenced by packetdecode().

5.2.2.4 #define WRITEPARM 0x0105

Command for writing waveform parameters

Definition at line 52 of file command.def.

Referenced by packetdecode().

5.2.2.5 #define LOADOBJ 0x0108

Command for loading a program into sequencer memory

Definition at line 53 of file command.def.

Referenced by packetdecode().

5.2.2.6 #define LOADWAVE 0x0109

Command for loading a waveform from file

Definition at line 54 of file command.def.

Referenced by packetdecode().

5.2.2.7 #define DIT 0x0200

Command for setting integration time (DIT)

Definition at line 55 of file command.def.

Referenced by packetdecode().

5.2.2.8 #define GROUP 0x0201

Command for setting group acquisitions number

Definition at line 56 of file command.def.

Referenced by packetdecode().

5.2.2.9 #define DOUBLE 0x0202

Command for setting single/doube acquisition mode

Definition at line 57 of file command.def.

Referenced by packetdecode().

5.2.2.10 #define QUADRANTS 0x0203

Command for setting active quadrant(s)

Definition at line 58 of file command.def.

Referenced by packetdecode().

5.2.2.11 #define NOISE 0x0205

Command for setting noise flag status

Definition at line 59 of file command.def.

Referenced by packetdecode().

5.2.2.12 #define SYNCHRO 0x0206

Command for starting synchro test acquisition

Definition at line 60 of file command.def.

Referenced by packetdecode().

5.2.2.13 #define SVBTEST 0x0207

Command for setting SVB test image mode

Definition at line 61 of file command.def.

Referenced by packetdecode().

5.2.2.14 #define SVBCHECK 0x0208

Command for setting Buffer board test image checking mode

Definition at line 62 of file command.def.

Referenced by packetdecode().

5.2.2.15 #define SEQMEM 0x0209

Command for starting Sequencer memory tests

Definition at line 63 of file command.def.

Referenced by packetdecode().

5.2.2.16 #define FIFOTST 0x020A

Command for starting Buffer board FIFO autotest

Definition at line 64 of file command.def.

Referenced by packetdecode().

5.2.2.17 #define EXPERT 0x020B

Command for setting Laboratory acquisition mode (expert)

Definition at line 65 of file command.def.

Referenced by packetdecode().

5.2.2.18 #define RUN 0x0300

Command for starting waveform generation and acquisistion

Definition at line 66 of file command.def.

Referenced by packetdecode().

5.2.2.19 #define START 0x0301

Command for acquisition with specific DIT e GROUP

Definition at line 67 of file command.def.

5.2.2.20 #define STOP 0x0302

Command for stopping of waveform generation

Definition at line 68 of file command.def.

Referenced by packetdecode().

5.2.2.21 #define ABORT 0x0303

Command for halting waveform generation at once

Definition at line 69 of file command.def.

Referenced by packetdecode().

5.2.2.22 #define INTEGRA 0x0304

Command for starting data acquisition

Definition at line 70 of file command.def.

5.2.2.23 #define FREERUN 0x0305

Command for starting Free run

Definition at line 71 of file command.def.

Referenced by packetdecode().

5.2.2.24 #define MULTI 0x0306

Command for statrting multi sampling acquisition

Definition at line 72 of file command.def.

Referenced by packetdecode().

5.2.2.25 #define SOCKDS9 0x0309

Command for specifyng the sockname for DS9

Definition at line 73 of file command.def.

5.2.2.26 #define DRYRUN 0x0307

Command for starting waveform generation

Definition at line 74 of file command.def.

Referenced by packetdecode().

5.2.2.27 #define FLIPSVB 0x0308

Command for starting a square wave test clock

Definition at line 75 of file command.def.

5.2.2.28 #define REINIT 0x0310

Command for re-initialization of the niosserver

Definition at line 76 of file command.def.

Referenced by packetdecode().

5.2.2.29 #define STATUS 0x0400

Command for asking ALL status variables

Definition at line 77 of file command.def.

Referenced by packetdecode().

5.2.2.30 #define READLOG 0x0410

Command for asking the internal log

Definition at line 78 of file command.def.

Referenced by packetdecode().

5.2.2.31 #define VERBOSE 0x0420

Command for setting NIOS verbosity level

Definition at line 79 of file command.def.

Referenced by packetdecode().

5.2.2.32 #define MSGLEVEL 0x0430

Command for setting the minimum message level verbosity

Definition at line 80 of file command.def.

Referenced by packetdecode().

5.2.2.33 #define DUMMYACQ 0x0444

Command for performing a syntetic frame acquisistion

Definition at line 81 of file command.def.

Referenced by packetdecode().

5.2.2.34 #define NOP 0x04FF

Command for No operations: used as keep-alive

Definition at line 82 of file command.def.

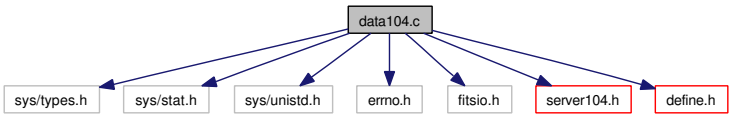
Referenced by embedsys_keepalive(), embedsys_keepalive_old(), and packetdecode().

5.3 data104.c File Reference

Routines to save data in fits format.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/unistd.h>
#include <errno.h>
#include "fitsio.h"
#include "server104.h"
#include "define.h"
```

Include dependency graph for data104.c:



Functions

- int print_fits_error (char *source, int status)
- void write_primary_HU (fitsfile *fptr, int *status)
- void write_primary_DU (fitsfile *fptr, long *dim, int *status, unsigned short *data)
- int save_data (int row, int col, unsigned short *data)

5.3.1 Detailed Description

Routines to save data in fits format.

If the symbol FITSPRESENT is defined, this portion depends on cfitsio library. This portion is only useful during the menu mode execution.

Definition in file data104.c.

5.3.2 Function Documentation

5.3.2.1 int print_fits_error (char * *source*, int *status*)

Print out cfitsio error messages.

Parameters:

source the routine name (NOT USED!!)

status the error status returned by cfitsio functions

Returns:

0 if ok, else > 0

Definition at line 58 of file data104.c.

References errlog(), error_code(), GB_CFTISIO_ERR, GB_EINTERNAL, gb_errno, GB_ERROR, and Opera.

Referenced by save_data().

5.3.2.2 void write_primary_HU (fitsfile * *fptr*, int * *status*)

Writes the primary fits header unit.

Parameters:

fptr pointer to fits file

status pointer to function returned status

Returns:

no value

Definition at line 92 of file data104.c.

References Tint.

Referenced by save_data().

5.3.2.3 void write_primary_DU (fitsfile * *fptr*, long * *dim*, int * *status*, unsigned short * *data*)

Writes the primary fits data unit.

Parameters:

fptr pointer to fits file

dim array with image dimension (dim[0] = ncol, dim[1] = nrow)

status pointer to function returned status

data the data array

Returns:

no value

Definition at line 119 of file data104.c.

Referenced by save_data().

5.3.2.4 int save_data (int *row*, int *col*, unsigned short * *data*)

Save the data present in *array in a minimal fits form

Parameters:

row the number of image rows

col the number of columns

data the data array

Returns:

0 if no error(s) , else > 0

Definition at line 156 of file data104.c.

References file_n, gid, iolog(), print_fits_error(), strdup_printf(), uid, write_primary_DU(), and write_primary_HU().

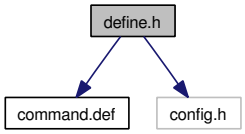
Referenced by read_image(), read_quadrant(), and read_testimage().

5.4 define.h File Reference

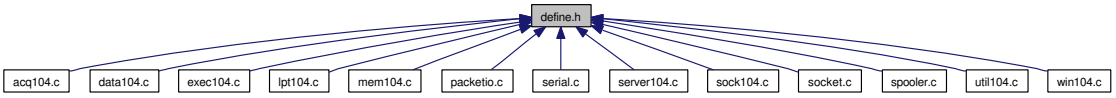
General symbols and board registers addresses and macros.

```
#include "command.def"
#include "config.h"
```

Include dependency graph for define.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define NBOARD 4
- #define BUFLEN 80
- #define N_ROW 2048
- #define N_COL 2048
- #define H_ROW 1024
- #define H_COL 1024
- #define NEXTRA 16
- #define NLOOSE 3
- #define NHEADER 4
- #define BASE_VAL 400
- #define REMOTE_HOST "falpala"
- #define REMOTE_USER "egiani"
- #define HAMEG_SERIAL "/dev/ttyS1"
- #define DEFAULT_TINT 100.0
- #define READ_FROM_104 7.1
- #define ETH_TRANS_TIME 1.3
- #define EMBEDDED_CMD_PORT 8082
- #define EMBEDDED_DATA_PORT 8083
- #define KEEPALIVE_INTVL 300000
- #define SOCK_TIMEOUT 3000
- #define SYS_TIMEOUT SOCK_TIMEOUT
- #define TIMEOUT_READ 8000
- #define NON_BLOCK 1
- #define BLOCK 0
- #define POLLFDN 6
- #define NGROUP_MAX 5

- #define NGROUP_FREERUN 10000
- #define MYTRUE 1
- #define MYFALSE 0
- #define MYERR -1
- #define MYNOTREADY -9
- #define MIN_ARRAY_SCANTIME 3.0
- #define MULTI_VALID "ABCabcLlRrZ"
- #define COMANDO 0x0010
- #define MESSAGGIO 0x0020
- #define INFO 0x0030
- #define ERROR 0xFF00
- #define ACK 0x0006
- #define EMBEDSERVER 0x1001
- #define GBSERVER 0x1002
- #define GUICLIENT 0x1003
- #define PRIVEMBED 0x1005
- #define MAGICMASK 0xA50F
- #define ISA_BASE 0x0D0000
- #define ISA_START ISA_BASE
- #define ISA_WINSIZE 0xFFFF
- #define ADCRAMSIZE 0x800
- #define ALIMEN_DEPTH 256
- #define RD_IDENT 0x000000
- #define RD_FIFO_STAT 0x000002
- #define WR_KPV_LINKA 0x000020
- #define WR_KPV_LINKB 0x000022
- #define WR_KPV_LINKC 0x000024
- #define WR_KPV_LINKD 0x000026
- #define WR_KPV_BRD 0x000028
- #define RD_KPV_LINKA 0x000030
- #define RD_KPV_LINKB 0x000032
- #define RD_KPV_LINKC 0x000034
- #define RD_KPV_LINKD 0x000036
- #define RD_DELAY_0 0x000040
- #define WR_TST_FIFOA 0x000060
- #define WR_TST_FIFOB 0x000062
- #define WR_TST_FIFOC 0x000064
- #define WR_TST_FIFOD 0x000066
- #define WR_TST_BRD 0x000068
- #define WR_RST_FIFOA 0x00006A
- #define WR_RST_FIFOB 0x00006C
- #define WR_RST_FIFOC 0x00006E
- #define WR_RST_FIFOD 0x000070
- #define WR_RST_BRD 0x000072
- #define RD_LSW_FIFOA 0x000080
- #define RD_LSW_FIFOB 0x000082
- #define RD_LSW_FIFOC 0x000084
- #define RD_LSW_FIFOD 0x000086
- #define RD_MSW_FIFOA 0x000088
- #define RD_MSW_FIFOB 0x00008A

- #define RD_MSW_FIFOC 0x00008C
- #define RD_MSW_FIFOD 0x00008E
- #define RD_FRM_CNTA 0x000090
- #define RD_FRM_CNTB 0x000092
- #define RD_FRM_CNTC 0x000094
- #define RD_FRM_CNTD 0x000096
- #define RD_DAT_FIFOA 0x000098
- #define RD_DAT_FIFOB 0x00009A
- #define RD_DAT_FIFOC 0x00009C
- #define RD_DAT_FIFOD 0x00009E
- #define WR_RAM_SEQ 0x008000
- #define WR_RAM_SEQA WR_RAM_SEQ
- #define WR_RAM_SEQB 0x009000
- #define WR_RAM_SEQC 0x00A000
- #define WR_RAM_SEQD 0x00B000
- #define RD_RAM_SEQ WR_RAM_SEQ
- #define WR_BROADCAST 0x00C000
- #define RD_BOARD 0x000800
- #define WR_SEQABORT 0x000802
- #define RD_LOG_STAT 0x000804
- #define RD_LOG_FIFO 0x000806
- #define WR_RST_LOG 0x000808
- #define WR_VRESET 0x00080A
- #define WR_VBIAS 0x00080C
- #define RD_PCOUNTER 0x000810
- #define WR_OFF1_2 0x000812
- #define WR_OFF3_4 0x000814
- #define RD_NBCONV 0x000816
- #define WR_SEQSTART 0x000818
- #define WR_SEQSTOP 0x00081A
- #define WR_SENSON 0x00081C
- #define WR_SENSOFF 0x00081E
- #define WR_FILTER1 0x000820
- #define WR_FILTER2 0x000822
- #define RD_ANALOGID 0x000824
- #define WR_FSYNC_TIM 0x000826
- #define WR_LSYNC_TIM 0x000828
- #define WR_STOP_CI 0x00082A
- #define WR_RESCLK 0x00082C
- #define FIFO_OVERFLOW 0x8000
- #define FIFO_READY 0x4000
- #define FIFO_ORDIGIT 0x2000
- #define FIFO_LSW_DAT 0x1FFF
- #define FIFO_MSW_DAT 0x03FF
- #define SENSOR_STATUS 0x0010
- #define BRD_ADDR_STEP 0x1000
- #define FILTER_ACTIVE 0x8000
- #define SEQUENCER_RUN 0x4000
- #define SENSOR_STATUS 0x0010
- #define READING_PIXEL 0x4000

- #define INTEGRATING 0x0800
- #define REG_OFFSET 0x0002
- #define LINK_STATUS 0x00F0
- #define LINK_VB_ERR 0x0F00
- #define LINK_VB_TRERR 0xF000
- #define LINK_ERROR 0x0800
- #define LINK_ERROR_RATE 0x07FF
- #define VCC_OPTO 0x0001
- #define VCC_5V 0x0002
- #define VCC_9V 0x0004
- #define VCC_3V 0x0008
- #define VCC_ON 0x000F
- #define VCC_SENSOR SENSOR_STATUS
- #define ERR_VLEVEL 0x0020
- #define ERR_OFFSET 0x0040
- #define ERR_ADDR 0x0080
- #define ERR_ADCONV 0x0100
- #define ERR_AMPL_ 0x0200
- #define ERR_AMPL 0x0400
- #define GB_ERRMASK 0x07FF
- #define COM_SHORTRESET 0x0000
- #define COM_RESETREAD 0x0000
- #define COM_RESETDRY 0x0081
- #define COM_READ 0x4000
- #define COM_INTEGDELAY 0x8000
- #define COM_RESTART 0xC000
- #define PROG_DIT 0x0001
- #define BOARD_DIT 0x0002
- #define NUM_SHORTRES 0x0003
- #define NUM_DRYRES 0x0004
- #define TRESET 0x0005
- #define NUM_LETTURE 0x0006
- #define PIX_LETTURA 0x0007
- #define LETTURA_PIX 0x0008
- #define PIXEL_WIDTH 0x0009
- #define BASE_SCAN 0x000A
- #define FSYNC_LSYNC 0x000B
- #define LSYNC_VCLK 0x000C
- #define D_LSYNC 0x000D
- #define VCLK_SCAN 0x000E
- #define VCLK_RESET 0x000F
- #define VCLK_CLK1 0x0010
- #define RESET_SCAN 0x0020
- #define SCAN_LSYNC 0x0030
- #define ENDFRAME 0x0040

Enumerations

- enum S104MemErr {
  NOERR = MYFALSE,
  GENERIC = MYTRUE,
  BADADDRESS,
  NOMEMORYMAPPED }
- enum NopStatus {
  NOP_IDLE,
  NOP_SENT }
- enum S104AState {
  SAS_IDLE = 1,
  SAS_BUSY = 2,
  SAS_RUNNING = 4,
  SAS_ABORT = 8,
  SAS_STOP = 16,
  SAS_FREERUN = 32,
  SAS_DUMMY = 64,
  SAS_TEST = 128,
  SAS_MULTI = 256,
  SAS_ERR = 512 }
- enum IdleState {
  IDLE_STOP = 0,
  IDLE_RUN = 1 }
- enum DataType {
  D_SIGNED = 0,
  D_UNSIGNED }
- enum BoardStatus {
  ST_IDLE = 0,
  ST_DUMMY = 1,
  ST_INTEG = 2 }
- enum S104Info {
  FRAME_STARTED = 0x0001,
  FRAME_MULTI = 0x0002,
  FRAME_READY = 0x0003,
  FRAME_FINISHED = 0x0004,
  FRAME_STOP = 0x0005,
  FRAME_ABORT = 0x0006,
  FRAME_INTEG = 0x0008,
  IDLE_ACQ = 0x0009 }

5.4.1 Detailed Description

General symbols and board registers addresses and macros.

Definition in file define.h.

5.4.2 Define Documentation

5.4.2.1 #define NBOARD 4

The number of quadrants in the array

Definition at line 64 of file define.h.

Referenced by abort_sequencer(), analog_boardId(), analog_boardStatus(), compute_cycle(), disable_sensor(), enable_sensor(), execute_cmd(), initvar104(), MenuLoop(), program_DAC_Vlevel(), program_sequencer(), sensor_onoff(), sequencer_counter(), sequencer_status(), start_sequencer(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_seq_mem(), treset_program(), verify_sequencer(), whichfilter(), write_pk_param(), and zero_seq_mem().

5.4.2.2 #define BUFLEN 80

A shared I/O buffer leght

Definition at line 65 of file define.h.

Referenced by program_sequencer(), and syntetize_multi().

5.4.2.3 #define N_ROW 2048

The number of array rows

Definition at line 67 of file define.h.

Referenced by dummy_image(), handle_image_acquisition(), handle_multi_acquisition(), read_image(), and send_image().

5.4.2.4 #define N_COL 2048

The number of array columns

Definition at line 68 of file define.h.

Referenced by dummy_image(), handle_image_acquisition(), handle_multi_acquisition(), main(), read_image(), and send_image().

5.4.2.5 #define H_ROW 1024

The number of rows in one quadrant

Definition at line 70 of file define.h.

Referenced by compute_cycle(), compute_scan_time(), dummy_image(), frame_ready(), handle_oneboard_acquisition(), initvar104(), packetdecode(), quadrant_ready(), read_image(), read_quadrant(), and send_image().

5.4.2.6 #define H_COL 1024

The number of cols in one quadrant

Definition at line 71 of file define.h.

Referenced by compute_cycle(), compute_scan_time(), dummy_image(), frame_ready(), handle_oneboard_acquisition(), initvar104(), main(), packetdecode(), quadrant_ready(), read_image(), read_quadrant(), and send_image().

5.4.2.7 #define NEXTRA 16

The number of extra pixels for noise reduction

Definition at line 73 of file define.h.

Referenced by compute_scan_time(), dummy_image(), frame_ready(), handle_oneboard_acquisition(), quadrant_ready(), read_image(), read_quadrant(), and send_image().

5.4.2.8 #define NLOOSE 3

The number of extra pixels we discard to ease syncronization

Definition at line 74 of file define.h.

Referenced by read_image().

5.4.2.9 #define NHEADER 4

Number of words in the header of each row

Definition at line 75 of file define.h.

Referenced by check_data(), dummy_image(), frame_ready(), handle_oneboard_acquisition(), quadrant_ready(), read_image(), read_quadrant(), read_testimage(), and send_image().

5.4.2.10 #define BASE_VAL 400

Value added to pixel subtractions to avoid negative numbers

Definition at line 76 of file define.h.

Referenced by read_image().

5.4.2.11 #define REMOTE_HOST "falpala"

Default remote host. For debug purpose

Definition at line 78 of file define.h.

Referenced by procname().

5.4.2.12 #define REMOTE_USER "egiani"

Default remote user. For debug purpose

Definition at line 79 of file define.h.

Referenced by procname().

5.4.2.13 #define HAMEG_SERIAL "/dev/ttyS1"

Serial device to control Hameg power supply

Definition at line 86 of file define.h.

Referenced by init_hameg(), off_power(), and read_power().

5.4.2.14 #define DEFAULT_TINT 100.0

Default for integration time

Definition at line 89 of file define.h.

5.4.2.15 #define READ_FROM_104 7.1

Read time from buffer, seconds

Definition at line 91 of file define.h.

Referenced by compute_scan_time().

5.4.2.16 #define ETH_TRANS_TIME 1.3

Transfer time by ethernet seconds

Definition at line 92 of file define.h.

Referenced by compute_scan_time().

5.4.2.17 #define EMBEDDED_CMD_PORT 8082

Server104 listen port for command

Definition at line 94 of file define.h.

Referenced by accept_connection(), and MainLoop().

5.4.2.18 #define EMBEDDED_DATA_PORT 8083

Server104 listen port for data

Definition at line 95 of file define.h.

Referenced by accept_connection(), and MainLoop().

5.4.2.19 #define KEEPALIVE_INTVL 300000

Keep alive interval specified in milliseconds

Definition at line 97 of file define.h.

Referenced by embedsys_keepalive().

5.4.2.20 #define SOCK_TIMEOUT 3000

Socket timeout interval in msec.

Definition at line 98 of file define.h.

Referenced by checksum_error(), MainLoop(), sock_read(), and sock_write().

5.4.2.21 #define SYS_TIMEOUT SOCK_TIMEOUT

interval in millisecond

Definition at line 99 of file define.h.

5.4.2.22 #define TIMEOUT_READ 8000

Timeout for pc104 reading of a frame

Definition at line 100 of file define.h.

Referenced by read_image(), read_log(), and read_testimage().

5.4.2.23 #define NON_BLOCK 1

Socket connection is unblocked type

Definition at line 102 of file define.h.

Referenced by accept_connection(), local_server_new(), open_giolamp(), and tcp_server_new().

5.4.2.24 #define BLOCK 0

Socket connection is blocked type

Definition at line 103 of file define.h.

5.4.2.25 #define POLLFDN 6

Default number of **pollfds** structures

Definition at line 104 of file define.h.

Referenced by MainLoop(), and update_event_sources().

5.4.2.26 #define NGROUP_MAX 5

max valid group number

Definition at line 106 of file define.h.

5.4.2.27 #define NGROUP_FREERUN 10000

group to acquire before FREERUN stops

Definition at line 107 of file define.h.

Referenced by packetdecode().

5.4.2.28 #define MYTRUE 1

True, to avoid conflicts with system definition

Definition at line 109 of file define.h.

Referenced by _wmem104_byte(), _wmem104_word(), abort_broadcast(), abort_sequencer(), add_f_entry(), add_s_entry(), alloc_array_mem(), analog_boardId(), analog_boardStatus(), check_group(), chkabort(), command_sock(), compute_cycle(), compute_pix_readclk(), compute_scan_time(), daemon_is_already_running(), disable_sensor(), dlanalog(), dummy_acquisition(), dump_data(), dump_init_file(), dump_seq_mem(), empty_events_h(), empty_fifo(), enable_sensor(), errlog(), execute_cmd(), exercise_seq_mem(), f_sched(), fifo_overflow(), fifo_ready(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init_power(), initdev(), initvar104(), initwin(), integriamo(), iolog(), isfull_fifo(), ismultiinteg(), isvoid_fifo(), l_get_port(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), main(), MainLoop(), map_isa_ram(), MenuLoop(), multical(), multifile(), multiswitch(), multivalid(), off_power(), packetdecode(), printanalog(), printstatus(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), random_action(), read_image(), read_init_file(), read_log(), read_pk_param(), read_power(), read_quadrant(), read_testimage(), reprogram_vlevel(), reset_fifo(), reset_fpga(), s_sched(), s_wrapper(), sawtooth_dac_test(), send_image(), sensor_onoff(), sequencer_running(), sequencer_status(), serial_command(), serial_decode(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_idle(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_board_id(), test_read(), test_seq_mem(), test_write(), treset_program(), unmap_isa_ram(), verify_sequencer(), whichfilter(), write_pk_param(), write_seqfile(), and zero_seq_mem().

5.4.2.29 #define MYFALSE 0

False, to avoid conflicts with system definition

Definition at line 110 of file define.h.

Referenced by abort_broadcast(), abort_sequencer(), add_f_entry(), add_s_entry(), alloc_array_mem(), check_group(), chkabort(), close_embed_sockets(), command_sock(), daemon_is_already_running(), disable_sensor(), dlanalog(), dummy_image(), dump_data(), dump_init_file(), enable_sensor(), errlog(), f_sched(), fifo_overflow(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init_power(), initdev(), initvar104(), iolog(), isfull_fifo(), ismultiinteg(), isvoid_fifo(), l_error(), l_function(), l_get_port(), l_header(), l_init_port(), l_opera(), l_operation(), l_position(), l_stdmessage(), l_w_ch_pos(), l_w_string(), lcd_init(), main(), MainLoop(), MenuLoop(), multical(), multifile(), multiswitch(), multivalid(), off_power(), packetdecode(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), random_action(), read_image(), read_init_file(), read_pk_param(), read_power(), read_quadrant(), read_testimage(), reset_fifo(), reset_fpga(), rmem104(), s_sched(), s_wrapper(), send_image(), sequencer_running(), serial_command(), serial_decode(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_idle(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_board_id(), test_seq_mem(), treset_program(), whichfilter(), write_pk_param(), write_seqfile(), and zero_seq_mem().

5.4.2.30 #define MYERR -1

error condition

Definition at line 111 of file define.h.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardId(), analog_boardStatus(), compute_cycle(), disable_sensor(), dlanalog(), dlascii(), dlhex(), dlint(), dummy_acquisition(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), fifo_overflow(), fifo_ready(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), link_error(), link_status(), map_isa_ram(), MenuLoop(), multifile(), open_giolamp(), packetdecode(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), read_log(), read_testimage(), reprogram_vlevel(), reset_fpga(), sawtooth_dac_test(), sensor_onoff(), sequencer_counter(), sequencer_status(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_idle(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_read(), test_seq_mem(), test_write(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), and zero_seq_mem().

5.4.2.31 #define MYNOTREADY -9

Not ready condition. May or may not be an error condition.

Definition at line 112 of file define.h.

Referenced by fifo_ready(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), quadrant_ready(), and read_testimage().

5.4.2.32 #define MIN_ARRAY_SCANTIME 3.0

Minimum array scan time in sec.

Definition at line 114 of file define.h.

5.4.2.33 #define MULTI_VALID "ABCabcLlRrZ"

valid values for multiple sample operation file * originally is = "ABCabcLlRrZ" - required lLRZ

Definition at line 132 of file define.h.

Referenced by multivalid(), and syntetize_multi().

5.4.2.34 #define COMANDO 0x0010

Packet mask of type command

Definition at line 180 of file define.h.

Referenced by dump_packet(), embedsys_keepalive(), embedsys_keepalive_old(), and packetdecode().

5.4.2.35 #define MESSAGGIO 0x0020

Packet mask of type message

Definition at line 181 of file define.h.

Referenced by dump_packet(), and iolog().

5.4.2.36 #define INFO 0x0030

Packet mask of type info

Definition at line 182 of file define.h.

Referenced by dump_packet(), and infolog().

5.4.2.37 #define ERROR 0xFF00

Packet mask of type error

Definition at line 183 of file define.h.

Referenced by dump_packet(), and errlog().

5.4.2.38 #define ACK 0x0006

Packet mask of type acknoledge

Definition at line 184 of file define.h.

Referenced by _packetsend(), dump_packet(), packetack(), packetdecode(), and packetsend_ack().

5.4.2.39 #define EMBEDSERVER 0x1001

Destination mask for embedded task

Definition at line 186 of file define.h.

Referenced by dump_packet().

5.4.2.40 #define GBSERVER 0x1002

Destination mask for gbridge task

Definition at line 187 of file define.h.

Referenced by dump_packet(), packetack_gbridge(), and packetsend_gbridge().

5.4.2.41 #define GUICLIENT 0x1003

Destination mask for gui task

Definition at line 188 of file define.h.

Referenced by dump_packet(), packetack_guilab(), packetdecode(), and packetsend_guilab().

5.4.2.42 #define PRIVEMBED 0x1005

Destination mask for embedded task (internal use)

Definition at line 189 of file define.h.

Referenced by dump_packet(), and packetdecode().

5.4.2.43 #define MAGICMASK 0xA50F

Mask used as first word of a packet

Definition at line 191 of file define.h.

Referenced by _packetsend(), and packetdecode().

5.4.2.44 #define ISA_BASE 0x0D0000

ADC_Board base address

Definition at line 194 of file define.h.

Referenced by exercise_seq_mem(), map_isa_ram(), read_log(), rmem104(), sawtooth_dac_test(), test_-read(), and wmem104().

5.4.2.45 #define ISA_START ISA_BASE

start of ISA bus Ram window

Definition at line 195 of file define.h.

5.4.2.46 #define ISA_WINSIZE 0xFFFF

width of ISA bus memory window

Definition at line 196 of file define.h.

Referenced by map_isa_ram(), MenuLoop(), rmem104(), unmap_isa_ram(), and wmem104().

5.4.2.47 #define ADCRAMSIZE 0x800

amount, in bytes of sequencer Ram

Definition at line 197 of file define.h.

Referenced by exercise_seq_mem(), program_sequencer(), syntetize_multi(), syntetize_program(), test_-seq_mem(), and zero_seq_mem().

5.4.2.48 #define ALIMEN_DEPTH 256

max depth of alimentation log

Definition at line 198 of file define.h.

Referenced by read_log().

5.4.2.49 #define RD_IDENT 0x000000

Board identification

Definition at line 202 of file define.h.

Referenced by exercise_seq_mem(), initdev(), and test_board_id().

5.4.2.50 #define RD_FIFO_STAT 0x000002

read status of the 4 FIFO

Definition at line 203 of file define.h.

Referenced by link_status().

5.4.2.51 #define WR_KPV_LINKA 0x000020

write word on optical link A

Definition at line 204 of file define.h.

5.4.2.52 #define WR_KPV_LINKB 0x000022

write word on optical link B

Definition at line 205 of file define.h.

5.4.2.53 #define WR_KPV_LINKC 0x000024

write word on optical link C

Definition at line 206 of file define.h.

5.4.2.54 #define WR_KPV_LINKD 0x000026

write word on optical link D

Definition at line 207 of file define.h.

5.4.2.55 #define WR_KPV_BRD 0x000028

write word on all optical links

Definition at line 208 of file define.h.

5.4.2.56 #define RD_KPV_LINKA 0x000030

write word on optical link A

Definition at line 209 of file define.h.

5.4.2.57 #define RD_KPV_LINKB 0x000032

write word on optical link B

Definition at line 210 of file define.h.

5.4.2.58 #define RD_KPV_LINKC 0x000034

write word on optical link C

Definition at line 211 of file define.h.

5.4.2.59 #define RD_KPV_LINKD 0x000036

write word on optical link D

Definition at line 212 of file define.h.

5.4.2.60 #define RD_DELAY_0 0x000040

read delay 0

Definition at line 213 of file define.h.

5.4.2.61 #define WR_TST_FIFOA 0x000060

Start test FIFO A

Definition at line 214 of file define.h.

Referenced by read_testimage().

5.4.2.62 #define WR_TST_FIFOB 0x000062

Start test FIFO B

Definition at line 215 of file define.h.

5.4.2.63 #define WR_TST_FIFOC 0x000064

Start test FIFO C

Definition at line 216 of file define.h.

5.4.2.64 #define WR_TST_FIFOD 0x000066

Start test FIFO D

Definition at line 217 of file define.h.

5.4.2.65 #define WR_TST_BRD 0x000068

Start Broadcast FIFO test

Definition at line 218 of file define.h.

Referenced by read_testimage().

5.4.2.66 #define WR_RST_FIFOA 0x00006A

Reset FIFO A

Definition at line 219 of file define.h.

Referenced by read_testimage(), and reset_fifo().

5.4.2.67 #define WR_RST_FIFOB 0x00006C

Reset FIFO B

Definition at line 220 of file define.h.

5.4.2.68 #define WR_RST_FIFOC 0x00006E

Reset FIFO C

Definition at line 221 of file define.h.

5.4.2.69 #define WR_RST_FIFOD 0x000070

Reset FIFO D

Definition at line 222 of file define.h.

5.4.2.70 #define WR_RST_BRD 0x000072

Broadcast FIFO reset

Definition at line 223 of file define.h.

Referenced by read_testimage(), and reset_fpga().

5.4.2.71 #define RD_LSW_FIFOA 0x000080

how many data present in FIFO A

Definition at line 224 of file define.h.

Referenced by fifo_overflow(), and fifo_ready().

5.4.2.72 #define RD_LSW_FIFOB 0x000082

how many data present in FIFO B

Definition at line 225 of file define.h.

5.4.2.73 #define RD_LSW_FIFOC 0x000084

how many data present in FIFO C

Definition at line 226 of file define.h.

5.4.2.74 #define RD_LSW_FIFOD 0x000086

how many data present in FIFO D

Definition at line 227 of file define.h.

5.4.2.75 #define RD_MSW_FIFOA 0x000088

how many data present in FIFO A

Definition at line 228 of file define.h.

Referenced by fifo_ready().

5.4.2.76 #define RD_MSW_FIFOB 0x00008A

how many data present in FIFO B

Definition at line 229 of file define.h.

5.4.2.77 #define RD_MSW_FIFOC 0x00008C

how many data present in FIFO C

Definition at line 230 of file define.h.

5.4.2.78 #define RD_MSW_FIFOD 0x00008E

how many data present in FIFO D

Definition at line 231 of file define.h.

5.4.2.79 #define RD_FRM_CNTA 0x000090

read the frame counter A

Definition at line 232 of file define.h.

5.4.2.80 #define RD_FRM_CNTB 0x000092

read the frame counter B

Definition at line 233 of file define.h.

5.4.2.81 #define RD_FRM_CNTC 0x000094

read the frame counter C

Definition at line 234 of file define.h.

5.4.2.82 #define RD_FRM_CNTD 0x000096

read the frame counter D

Definition at line 235 of file define.h.

5.4.2.83 #define RD_DAT_FIFOA 0x000098

read data from FIFO A

Definition at line 236 of file define.h.

Referenced by read_image(), read_quadrant(), and read_testimage().

5.4.2.84 #define RD_DAT_FIFOB 0x00009A

read data from FIFO B

Definition at line 237 of file define.h.

5.4.2.85 #define RD_DAT_FIFOC 0x00009C

read data from FIFO C

Definition at line 238 of file define.h.

5.4.2.86 #define RD_DAT_FIFOD 0x00009E

read data from FIFO D

Definition at line 239 of file define.h.

5.4.2.87 #define WR_RAM_SEQ 0x008000

Sequencer Ram base address

Definition at line 242 of file define.h.

Referenced by enable_sensor(), handle_oneboard_acquisition(), link_error(), program_DAC_filter(), reprogram_vlevel(), reset_logfifo(), sawtooth_dac_test(), sequencer_counter(), sequencer_status(), start_broadcast(), start_sequencer(), and whichfilter().

5.4.2.88 #define WR_RAM_SEQA WR_RAM_SEQ

Sequencer Ram on board A

Definition at line 243 of file define.h.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardId(), analog_boardStatus(), disable_sensor(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), handle_oneboard_acquisition(), link_error(), program_DAC_Vlevel(), program_sequencer(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_sequencer(), syntetize_multi(), syntetize_program(), test_seq_mem(), treset_program(), verify_sequencer(), and zero_seq_mem().

5.4.2.89 #define WR_RAM_SEQB 0x009000

Sequencer Ram on board B

Definition at line 244 of file define.h.

Referenced by abort_sequencer(), analog_boardId(), analog_boardStatus(), disable_sensor(), dump_seq_-mem(), enable_sensor(), exercise_seq_mem(), handle_oneboard_acquisition(), link_error(), program_-DAC_Vlevel(), program_sequencer(), start_sequencer(), stop_CI_sequencer(), stop_sequencer(), syntetize_multi(), syntetize_program(), test_seq_mem(), treset_program(), verify_sequencer(), and zero_seq_mem().

5.4.2.90 #define WR_RAM_SEQC 0x00A000

Sequencer Ram on board C

Definition at line 245 of file define.h.

Referenced by abort_sequencer(), analog_boardId(), analog_boardStatus(), disable_sensor(), dump_seq_-mem(), enable_sensor(), exercise_seq_mem(), handle_oneboard_acquisition(), link_error(), program_-DAC_Vlevel(), program_sequencer(), start_sequencer(), stop_CI_sequencer(), stop_sequencer(), syntetize_multi(), syntetize_program(), test_seq_mem(), treset_program(), verify_sequencer(), and zero_seq_mem().

5.4.2.91 #define WR_RAM_SEQD 0x00B000

Sequencer Ram on board D

Definition at line 246 of file define.h.

Referenced by abort_sequencer(), analog_boardId(), analog_boardStatus(), disable_sensor(), dump_seq_-mem(), enable_sensor(), exercise_seq_mem(), handle_oneboard_acquisition(), link_error(), program_-DAC_Vlevel(), program_sequencer(), start_sequencer(), stop_CI_sequencer(), stop_sequencer(), syntetize_multi(), syntetize_program(), test_seq_mem(), treset_program(), verify_sequencer(), and zero_seq_mem().

5.4.2.92 #define RD_RAM_SEQ WR_RAM_SEQ

Sequencer Ram base address

Definition at line 247 of file define.h.

Referenced by read_converters(), and read_log().

5.4.2.93 #define WR_BROADCAST 0x00C000

Broadcast address space start

Definition at line 249 of file define.h.

Referenced by abort_broadcast(), start_broadcast(), stop_broadcast(), and stop_CI_broadcast().

5.4.2.94 #define RD_BOARD 0x000800

read board status

Definition at line 252 of file define.h.

Referenced by analog_boardStatus(), sequencer_status(), and whichfilter().

5.4.2.95 #define WR_SEQABORT 0x000802

write word start/stop command

Definition at line 253 of file define.h.

Referenced by abort_broadcast(), and abort_sequencer().

5.4.2.96 #define RD_LOG_STAT 0x000804

read FIFO status of board log

Definition at line 254 of file define.h.

Referenced by read_log(), and reprogram_vlevel().

5.4.2.97 #define RD_LOG_FIFO 0x000806

read FIFO log on board

Definition at line 255 of file define.h.

Referenced by read_log().

5.4.2.98 #define WR_RST_LOG 0x000808

reset FIFO log on board

Definition at line 256 of file define.h.

Referenced by reset_logfifo().

5.4.2.99 #define WR_VRESET 0x00080A

to program Vreset on board

Definition at line 257 of file define.h.

Referenced by dlanalog(), initdev(), MenuLoop(), program_DAC_Vlevel(), read_pk_param(), reprogram_-vlevel(), sawtooth_dac_test(), and write_pk_param().

5.4.2.100 #define WR_VBIAS 0x00080C

to program Vbias on board

Definition at line 258 of file define.h.

Referenced by dlanalog(), initdev(), MenuLoop(), program_DAC_Vlevel(), read_pk_param(), reprogram_-vlevel(), sawtooth_dac_test(), and write_pk_param().

5.4.2.101 #define RD_PCOUNTER 0x000810

read the program counter on board

Definition at line 259 of file define.h.

Referenced by sequencer_counter().

5.4.2.102 #define WR_OFF1_2 0x000812

to program offset 1 2

Definition at line 260 of file define.h.

Referenced by dlanalog(), initdev(), MenuLoop(), program_DAC_Vlevel(), read_pk_param(), reprogram_-vlevel(), sawtooth_dac_test(), and write_pk_param().

5.4.2.103 #define WR_OFF3_4 0x000814

to program offset 3 4

Definition at line 261 of file define.h.

Referenced by dlanalog(), initdev(), MenuLoop(), program_DAC_Vlevel(), read_pk_param(), reprogram_-vlevel(), sawtooth_dac_test(), and write_pk_param().

5.4.2.104 #define RD_NBCONV 0x000816

board converter non buffered read

Definition at line 262 of file define.h.

Referenced by read_converters().

5.4.2.105 #define WR_SEQSTART 0x000818

start board sequencer

Definition at line 263 of file define.h.

Referenced by start_broadcast(), and start_sequencer().

5.4.2.106 #define WR_SEQSTOP 0x00081A

stop board sequencer

Definition at line 264 of file define.h.

Referenced by stop_broadcast(), and stop_sequencer().

5.4.2.107 #define WR_SENSON 0x00081C

enable board sensor

Definition at line 265 of file define.h.

Referenced by enable_sensor(), and write_pk_param().

5.4.2.108 #define WR_SENSOFF 0x00081E

disable board A sensor

Definition at line 266 of file define.h.

Referenced by disable_sensor(), and write_pk_param().

5.4.2.109 #define WR_FILTER1 0x000820

enable filter 1

Definition at line 267 of file define.h.

Referenced by program_DAC_filter(), read_pk_param(), and write_pk_param().

5.4.2.110 #define WR_FILTER2 0x000822

enable filter 2

Definition at line 268 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.111 #define RD_ANALOGID 0x000824

analogic board ID

Definition at line 269 of file define.h.

Referenced by analog_boardId(), and link_error().

5.4.2.112 #define WR_FSYNC_TIM 0x000826

fsync related delays

Definition at line 270 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.113 #define WR_LSYNC_TIM 0x000828

lsync related delays

Definition at line 271 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.114 #define WR_STOP_CI 0x00082A

stop on corrent istruction

Definition at line 272 of file define.h.

Referenced by stop_CI_broadcast(), and stop_CI_sequencer().

5.4.2.115 #define WR_RESCLK 0x00082C

to program the 4 reset times

Definition at line 273 of file define.h.

Referenced by treset_program().

5.4.2.116 #define FIFO_OVERFLOW 0x8000

Mask to detect fifo overflow

Definition at line 278 of file define.h.

Referenced by fifo_overflow(), and fifo_ready().

5.4.2.117 #define FIFO_READY 0x4000

Mask to detect fifo ready

Definition at line 279 of file define.h.

Referenced by fifo_ready().

5.4.2.118 #define FIFO_ORDIGIT 0x2000

Mask to detect data present in MSW fifo

Definition at line 280 of file define.h.

Referenced by fifo_ready().

5.4.2.119 #define FIFO_LSW_DAT 0x1FFF

Mask to detect less than 8192 data

Definition at line 281 of file define.h.

Referenced by fifo_ready().

5.4.2.120 #define FIFO_MSW_DAT 0x03FF

Mask to detect more than 8192 data

Definition at line 282 of file define.h.

Referenced by fifo_ready().

5.4.2.121 #define SENSOR_STATUS 0x0010

Mask to detect sensor status (0 = on, 1 = off)

sensor status (0 = on, 1 = off)

Definition at line 291 of file define.h.

5.4.2.122 #define BRD_ADDR_STEP 0x1000

step between board mem area addresses

Definition at line 286 of file define.h.

Referenced by program_DAC_filter(), read_converters(), read_log(), read_pk_param(), reprogram_vlevel(), reset_logfifo(), sawtooth_dac_test(), sequencer_counter(), sequencer_status(), whichfilter(), and write_pk_param().

5.4.2.123 #define FILTER_ACTIVE 0x8000

filter selection: 0=Filter1, 1=Filter2

Definition at line 289 of file define.h.

Referenced by whichfilter().

5.4.2.124 #define SEQUENCER_RUN 0x4000

sequencer running status (1 running)

Definition at line 290 of file define.h.

Referenced by analog_boardStatus(), and sequencer_status().

5.4.2.125 #define SENSOR_STATUS 0x0010

Mask to detect sensor status (0 = on, 1 = off)

sensor status (0 = on, 1 = off)

Definition at line 291 of file define.h.

5.4.2.126 #define READING_PIXEL 0x4000

program counter register mask to detect reading pixel condition

Definition at line 292 of file define.h.

5.4.2.127 #define INTEGRATING 0x0800

program counter register mask to detect integration condition

Definition at line 293 of file define.h.

5.4.2.128 #define REG_OFFSET 0x0002

offset between registers' addresses

Definition at line 294 of file define.h.

Referenced by fifo_overflow(), fifo_ready(), program_DAC_filter(), read_image(), read_quadrant(), read_testimage(), and reset_fifo().

5.4.2.129 #define LINK_STATUS 0x00F0

optical link status

Definition at line 295 of file define.h.

Referenced by link_status().

5.4.2.130 #define LINK_VB_ERR 0x0F00

optical link status on buffer board

Definition at line 296 of file define.h.

Referenced by link_status().

5.4.2.131 #define LINK_VB_TRERR 0xF000

error in data transfer on VB link

Definition at line 297 of file define.h.

Referenced by link_status().

5.4.2.132 #define LINK_ERROR 0x0800

optical link error

Definition at line 298 of file define.h.

Referenced by link_error().

5.4.2.133 #define LINK_ERROR_RATE 0x07FF

number of errors on the optical link

Definition at line 299 of file define.h.

Referenced by link_error().

5.4.2.134 #define VCC_OPTO 0x0001

Mask to select OPTO power status

Definition at line 302 of file define.h.

Referenced by print_alimentation().

5.4.2.135 #define VCC_5V 0x0002

Mask to select 5V power status

Definition at line 303 of file define.h.

Referenced by print_alimentation().

5.4.2.136 #define VCC_9V 0x0004

Mask to select 9V power status

Definition at line 304 of file define.h.

Referenced by print_alimentation().

5.4.2.137 #define VCC_3V 0x0008

Mask to select 3V power status

Definition at line 305 of file define.h.

Referenced by print_alimentation().

5.4.2.138 #define VCC_ON 0x000F

Mask to select all power status

Definition at line 306 of file define.h.

Referenced by analog_boardStatus().

5.4.2.139 #define VCC_SENSOR SENSOR_STATUS

Definition at line 307 of file define.h.

Referenced by analog_boardStatus(), and print_alimentation().

5.4.2.140 #define ERR_VLEVEL 0x0020

error in the DA converter of Vbias-Vreset

Definition at line 309 of file define.h.

Referenced by analog_boardStatus().

5.4.2.141 #define ERR_OFFSET 0x0040

error int the DA converter of offset

Definition at line 310 of file define.h.

Referenced by analog_boardStatus().

5.4.2.142 #define ERR_ADDR 0x0080

error in the address decoder

Definition at line 311 of file define.h.

Referenced by analog_boardStatus().

5.4.2.143 #define ERR_ADCONV 0x0100

error in the AD converter

Definition at line 312 of file define.h.

Referenced by analog_boardStatus().

5.4.2.144 #define ERR_AMPL_ 0x0200

error in the negative amplifier output

Definition at line 313 of file define.h.

Referenced by analog_boardStatus().

5.4.2.145 #define ERR_AMPL 0x0400

error in the positive amplifier output

Definition at line 314 of file define.h.

Referenced by analog_boardStatus().

5.4.2.146 #define GB_ERRMASK 0x07FF

global error mask

Definition at line 315 of file define.h.

Referenced by link_error().

5.4.2.147 #define COM_SHORTRESET 0x0000

Sequencer command: reset without read

Definition at line 318 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.148 #define COM_RESETREAD 0x0000

Sequencer command: reset with read

Definition at line 319 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.149 #define COM_RESETDRY 0x0081

Sequencer command: reset with dry (dummy) read

Definition at line 320 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.150 #define COM_READ 0x4000

Sequencer command: only read

Definition at line 321 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.151 #define COM_INTEGDELAY 0x8000

Sequencer command: reset with read

Definition at line 322 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.152 #define COM_RESTART 0xC000

Sequencer command: restart to location 0

Definition at line 323 of file define.h.

Referenced by syntetize_multi(), and syntetize_program().

5.4.2.153 #define PROG_DIT 0x0001

packet command to program DIT

Definition at line 326 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.154 #define BOARD_DIT 0x0002

packet command to program board level DIT

Definition at line 327 of file define.h.

Referenced by read_pk_param().

5.4.2.155 #define NUM_SHORTRES 0x0003

packet command to program short reset number

Definition at line 328 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.156 #define NUM_DRYRES 0x0004

packet command to program dry reset number

Definition at line 329 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.157 #define TRESET 0x0005

packet command to program reset duration

Definition at line 330 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.158 #define NUM_LETTURE 0x0006

packet command to program reads number

Definition at line 331 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.159 #define PIX_LETTURA 0x0007

packet command to program PIX_LETTURA

Definition at line 332 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.160 #define LETTURA_PIX 0x0008

packet command to program LETTURA_PIX

Definition at line 333 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.161 #define PIXEL_WIDTH 0x0009

packet command to program pixel clock duration

Definition at line 334 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.162 #define BASE_SCAN 0x000A

packet command to program pixel base scan time

Definition at line 335 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.163 #define FSYNC_LSYNC 0x000B

packet command to program FSYNC_LSYNC delay

Definition at line 336 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.164 #define LSYNC_VCLK 0x000C

packet command to program LSYNC_VCLK delay

Definition at line 337 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.165 #define D_LSYNC 0x000D

packet command to program LSYNC duration

Definition at line 338 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.166 #define VCLK_SCAN 0x000E

packet command to program VCLK_SCAN delay

Definition at line 339 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.167 #define VCLK_RESET 0x000F

packet command to program VCLK_RESET delay

Definition at line 340 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.168 #define VCLK_CLK1 0x0010

packet command to program VCLK_CLK1 delay

Definition at line 341 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.169 #define RESET_SCAN 0x0020

packet command to program RESET_SCAN delay

Definition at line 342 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.170 #define SCAN_LSYNC 0x0030

packet command to program SCAN_LSYNC delay

Definition at line 343 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.2.171 #define ENDFRAME 0x0040

packet command to program ENDFRAME delay

Definition at line 344 of file define.h.

Referenced by read_pk_param(), and write_pk_param().

5.4.3 Enumeration Type Documentation

5.4.3.1 enum S104MemErr

internal error conditions for memory access

Enumerator:

NOERR no error im access to mmaped memory
GENERIC generic error
BADADDRESS bad address specified in memeory access
NOMEMORYMAPPED mmap operation failed

Definition at line 117 of file define.h.

5.4.3.2 enum NopStatus

Status of keepalive command

Enumerator:

NOP_IDLE waiting timer expiration
NOP_SENT NOP package sent, waiting its ACK

Definition at line 126 of file define.h.

5.4.3.3 enum S104AState

Server104 Acquisition State

Enumerator:

SAS_IDLE system ready to start new acquisition
SAS_BUSY acquisition is in progress
SAS_RUNNING acquisition is running
SAS_ABORT acquisition is aborting/aborted
SAS_STOP acquisition stopped
SAS_FREERUN free run acquisition selected
SAS_DUMMY dummy acquisition selected
SAS_TEST memory or fifo autotest running
SAS_MULTI multiple acquisition
SAS_ERR error status during integration settings

Definition at line 136 of file define.h.

5.4.3.4 enum IdleState

Idle Acquisition State

Enumerator:

IDLE_STOP idle acquisitions NOT running

IDLE_RUN idle acquisitions running

Definition at line 150 of file define.h.

5.4.3.5 enum DataType

data type for plot program

Enumerator:

D_SIGNED external program plot uses signed data

D_UNSIGNED external program plot uses unsigned data

Definition at line 156 of file define.h.

5.4.3.6 enum BoardStatus

status of analog boards

Enumerator:

ST_IDLE analog board in idle (stopped) mode

ST_DUMMY analog board in dummy integration mode

ST_INTEG analog board in normal integration mode

Definition at line 162 of file define.h.

5.4.3.7 enum S104Info

Server104 status information

Enumerator:

FRAME_STARTED frame acqusition started

FRAME_MULTI multi-frame acquisition started

FRAME_READY the frame is ready to be read from fifo

FRAME_FINISHED frame(s) acquisition finished

FRAME_STOP frame acquisition stopped

FRAME_ABORT frame acquisition aborted

FRAME_INTEG frame integration info

IDLE_ACQ idle acquisition status info

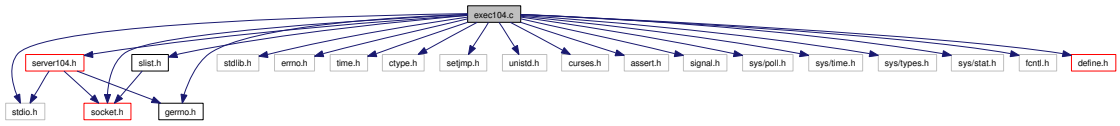
Definition at line 169 of file define.h.

5.5 exec104.c File Reference

Main level operational routines.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <ctype.h>
#include <setjmp.h>
#include <unistd.h>
#include <curses.h>
#include <assert.h>
#include <signal.h>
#include <sys/poll.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "socket.h"
#include "slist.h"
#include "define.h"
#include "server104.h"
#include "gerrno.h"
```

Include dependency graph for exec104.c:



Defines

- #define LOGFILETIMEOUT 10000
- #define POLLMS 12
- #define MAX(a, b) ((a) > (b) ? (a) : (b))

Functions

- int write_seqfile (char *filetype, int file_size, int npart, char *filename, char *payload, int chan)
- int MenuLoop (void)
- int write_pk_param (int payload_len, char *payload)

- int read_pk_param (int channel, va_list args)
- int packetread_gbridge (int sockfd, void *data)
- int packetsend (int destination, int cmd, int type, int datalen, char *payload)
- int packetack (int destination, int cmd, int datalen, unsigned short seqnum, char *payload)
- int packetsend_gbridge (int cmd, int type, int datalen, char *payload)
- int packetack_gbridge (int cmd, int datalen, int seqnum, char *payload)
- int packetsend_guilab (int cmd, int type, int datalen, char *payload)
- int packetack_guilab (int cmd, int datalen, int seqnum, char *payload)
- int packetdecode (unsigned short *header, unsigned char *_payload)
- int send_image (int nframe)
- int update_event_sources (void)
- void remove_event_source (int sockfd)
- void remove_event_sourcetype (int type)
- void remove_embed_sources ()
- void remove_disconnected_sources (SList *list)
- void remove_nop_timeout (void)
- void nop_timeout (int signal)
- int embedsys_keepalive (void)
- int embedsys_keepalive_old ()
- int MainLoop (void)

Variables

- struct pollfd * pfds
- static sigjmp_buf main_loop
- static volatile sig_atomic_t canjump

5.5.1 Detailed Description

Main level operational routines.

This file contains the main execution routines, Menu_loop for menu mode and MainLoop and packet_-decode for daemon mode. Here we have also few support routines.

Definition in file exec104.c.

5.5.2 Define Documentation

5.5.2.1 #define LOGFILETIMEOUT 10000

Definition at line 152 of file exec104.c.

Referenced by MainLoop().

5.5.2.2 #define POLLMS 12

Definition at line 153 of file exec104.c.

5.5.2.3 #define MAX(a, b) ((a) > (b) ? (a) : (b))

Definition at line 155 of file exec104.c.

5.5.3 Function Documentation

5.5.3.1 int write_seqfile (char * *filetype*, int *file_size*, int *filesection*, char * *filename*, char * *payload*, int *chan*)

Creates the specified sequencer file using the packet payload. The instructions inside the file are used to program the sequencer RAM. Return MYTRUE on success, or MYFALSE on error or if the sequencer file is splitted in different parts an not all of them have been already received.

Note:

This routine is called when the LOADWAVE command is received. The LOADWAVE command payload is built of:

filename string1 string2 ....stringN

where string1 strng2 ...stringN are the instruction lines of the sequencer file

Parameters:

filetype specify if a multiprogram or a sequence file will be * written

file_size the dimention of the file to create

filesection the part number of the file to create

filename the name of the file to create

payload the packet payload

chan the board to progrma (chan = 5 means all boards)

Returns:

MYTRUE on success, MYFALSE on error or if the sequencer file is splitted

Definition at line 1969 of file exec104.c.

References errlog(), GB_EBADARG, gb_errno, iolog(), MYFALSE, and MYTRUE.

Referenced by packetdecode().

5.5.3.2 int MenuLoop (void)

Main execution loop in menu mode. The real menu routine.

This routine init and display menu. It wait for a key pressed and call the appropriate execution routine. The update menu and restart the loop.

Returns:

MYTRUE if succesfull completion, MYFALSE on error.

Definition at line 176 of file exec104.c.

References abort_sequencer(), Acq_type, Analog, analog_boardId(), analog_boardStatus(), Base_-scan, Channel, check_group(), compute_pix_readclk(), compute_scan_time(), D_lsync, daemon_is_-already_running(), disable_sensor(), Dit, dlanalog(), dlascii(), dlhex(), dlint(), dummy_acquisition(), dump_seq_mem(), enable_sensor(), errlog(), execute_cmd(), exercise_seq_mem(), Extra_sub, finish(), gb_errno, handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init-menu(), iolog(), ISA_WINSIZE, Keepgoing, l_function(), l_header(), l_opera(), link_status(), MAX,

Menu, multical(), multidummy(), multifile(), multi_status::multiprogram, multistatus, multivalid(), MYERR, MYFALSE, MYTRUE, NBOARD, Ngroup, NoiseMode, Opera, printstatus(), AnalogBoard::Prog_dit, AnalogBoard::Prog_readclk, AnalogBoard::Prog_readdel, AnalogBoard::Prog_resnum, program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), R_endframe, R_fsync_lsync, R_lsync_vclk, random_action(), read_converters(), read_log(), read_testimage(), sawtooth_dac_test(), Scan_time, sensor_onoff(), start_sequencer(), stop_sequencer(), syntetize_multi(), test_board_id(), test_read(), test_seq_mem(), TEST_TASK, test_write(), Tint, updatemenu(), Verbose, WR_OFF1_2, WR_OFF3_4, WR_VBIAS, WR_VRESET, and zero_seq_mem().

Referenced by main().

5.5.3.3 int write_pk_param (int *payload_len*, char * *payload*)

Decodes a list of couples <addr> <values> to set SVB parameters.

Returns:

MYTRUE if all OK, MYFALSE otherwise.

Definition at line 710 of file exec104.c.

References Analog, Base_scan, BASE_SCAN, BRD_ADDR_STEP, Channel, compute_scan_time(), D_lsync, D_LSYNC, disable_sensor(), Dit, enable_sensor(), ENDFRAME, errlog(), error_code(), execute_cmd(), AnalogBoard::Filtro, FSYNC_LSYNC, GB_BADADDR, GB_EBADARG, GB_WARNING, iolog(), LETTURA_PIX, LSYNC_VCLK, MAX, MYFALSE, MYTRUE, NBOARD, NUM_DRYRES, NUM_LETTURE, NUM_SHORTRES, AnalogBoard::Offset12, AnalogBoard::Offset34, Opera, PIX_LETTURA, PIXEL_WIDTH, PROG_DIT, AnalogBoard::Prog_dryres, AnalogBoard::Prog_readclk, AnalogBoard::Prog_readdel, AnalogBoard::Prog_resclk, AnalogBoard::Prog_resnum, program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), R_endframe, R_fsync_lsync, R_lsync_vclk, RESET_SCAN, SCAN_LSYNC, Scan_time, Tint, TRESET, treset_program(), AnalogBoard::V_bias, AnalogBoard::V_reset, VCLK_CLK1, VCLK_RESET, VCLK_SCAN, WR_FILTER1, WR_FILTER2, WR_OFF1_2, WR_OFF3_4, WR_SENSOFF, WR_SENSON, WR_VBIAS, and WR_VRESET.

Referenced by packetdecode().

5.5.3.4 int read_pk_param (int *channel*, va_list *args*)

Sends a list of couples <addr> <values> with SVB parameters.

Parameters:

channel the board to operate with

args va_list

Returns:

MYTRUE if all OK, MYFALSE otherwise.

Definition at line 952 of file exec104.c.

References Analog, Base_scan, BASE_SCAN, BOARD_DIT, BRD_ADDR_STEP, D_lsync, D_LSYNC, ENDFRAME, errlog(), error_code(), FSYNC_LSYNC, GB_EBADARG, GB_WARNING, LETTURA_PIX, LSYNC_VCLK, MYFALSE, MYTRUE, NUM_DRYRES, NUM_LETTURE, NUM_SHORTRES, Opera, PIX_LETTURA, PIXEL_WIDTH, PROG_DIT, AnalogBoard::Prog_readclk,

AnalogBoard::Prog_readdel, R_endframe, R_fsync_lsync, R_lsync_vclk, RESET_SCAN, SCAN_-LSYNC, Tint, TRESET, VCLK_CLK1, VCLK_RESET, VCLK_SCAN, WR_FILTER1, WR_FILTER2, WR_OFF1_2, WR_OFF3_4, WR_VBIAS, and WR_VRESET.

Referenced by packetdecode().

5.5.3.5 int packetread_gbridge (int *sockfd*, void * *data*)

Reads a packet coming from gbridge application and decodes it.

Parameters:

sockfd the socket decriptor

data the passed argument

Returns:

$>$ 0 on success, -1 on unrecoverable errors or EOF, 0 on timeout.

Note:

This function is used when the program is executed in daemon mode.

Definition at line 1026 of file exec104.c.

References check_socket_status(), DISCONNECTED, GB_EBADARG, gb_errno, GB_IO_CLOSED, get_current_time(), GPacket::header, packetdecode(), packetread(), GPacket::payload, and timestamp.

Referenced by accept_connection(), and mainloop_iterate().

5.5.3.6 int packetsend (int *destination*, int *cmd*, int *type*, int *datalen*, char * *payload*)

Sends a packet to GUI or gbridge application. It uses _packetsend routine.

Parameters:

destination the task of destination: gbridge or gui

cmd the command to acknoledge

type the type of command to acknoledge

datalen the len in bytes of the packet

payload the data packet withou header

See also:

_packetsend

Definition at line 1070 of file exec104.c.

References _packetsend(), check_socket_status(), DISCONNECTED, EMBED_CMD, GB_EBADARG, gb_errno, GB_IO_CLOSED, and get_socket_descriptor().

Referenced by packetsend_gbridge(), and packetsend_guilab().

5.5.3.7 int packetack (int *destination*, int *cmd*, int *datalen*, unsigned short *seqnum*, char * *payload*)

Sends an ACK packet to the gui or gbridge application. It uses packetsend_ack() routine.

Parameters:

destination the task of destination: gbridge or gui

cmd the type of command to acknoledge

datalen the len in bytes of the packet

seqnum the packet number

payload the data packet withou header

Returns:

> 0 on success, -1 on unrecoverbale error(s), 0 on timeout.

See also:

packetsend_ack()

Definition at line 1112 of file exec104.c.

References ACK, check_socket_status(), DISCONNECTED, dump_packet(), EMBED_CMD, GB_-EBADARG, gb_errno, GB_IO_CLOSED, get_socket_descriptor(), and packetsend_ack().

Referenced by packetack_gbridge(), packetack_guilab(), and packetdecode().

5.5.3.8 int packetsend_gbridge (int *cmd*, int *type*, int *datalen*, char * *payload*)

Checks if command socket exist, build and send packet

Parameters:

cmd the command to execute

type the type of command (ACK, MESSAGE, etc.)

datalen the packet length in bytes

payload the data inside the packet without the header

Returns:

> 0 on success, -1 on unrecoverbale error(s), 0 on timeout.

Note:

This function is used when the program is executed in daemon mode.

Definition at line 1151 of file exec104.c.

References GBSERVER, and packetsend().

Referenced by embedsys_keepalive(), embedsys_keepalive_old(), errlog(), infolog(), and iolog().

5.5.3.9 int packetack_gbridge (int *cmd*, int *datalen*, int *seqnum*, char * *payload*)

Definition at line 1166 of file exec104.c.

References GBSERVER, and packetack().

Referenced by packetdecode().

5.5.3.10 int packetsend_guilab (int *cmd*, int *type*, int *datalen*, char * *payload*)

Definition at line 1181 of file exec104.c.

References GUICLIENT, and packetsend().

5.5.3.11 int packetack_guilab (int *cmd*, int *datalen*, int *seqnum*, char * *payload*)

Definition at line 1195 of file exec104.c.

References GUICLIENT, and packetack().

Referenced by packetdecode().

5.5.3.12 int packetdecode (unsigned short * *header*, unsigned char * *_payload*)

Decodes the packets coming from gbridge application.

Parameters:

header the packet header info

_payload the packet data (withou header)

Returns:

> 0 on success, < 0 on unrecoverable errors, 0 on timeout

Definition at line 1212 of file exec104.c.

References ABORT, ACK, ACQ_TASK, Acq_type, Channel, check_group(), close_giolamp(), COMANDO, compute_scan_time(), Curframe, Curgroup, disable_sensor(), Dit, DIT, DOUBLE, DRYRUN, dummy_acquisition(), dummy_image(), DUMMYACQ, dump_acquisition_status(), dump_packet(), dump_seq_mem(), DUMPMEM, enable_sensor(), errlog(), error_code(), execute_cmd(), EXPERT, ExpertMode, FIFOTST, FILLMEM0, FRAME_ABORT, FRAME_FINISHED, FRAME_STOP, FREERUN, GB_EBADARG, GB_EIDLERUN, gb_errno, GB_ERROR, GB_WARNING, GROUP, GUICLIENT, H_COL, H_ROW, handle_image_acquisition(), handle_multi_acquisition(), infolog(), iolog(), LOADOBJ, LOADWAVE, MAGICMASK, MAX, MSGLEVEL, MULTI, multifile(), MYERR, MYFALSE, MYTRUE, Ncol, Ngroup, NGROUP_FREERUN, NOISE, NoiseMode, NOP, NOP_IDLE, nopStatus, Nrow, Opera, packetack(), packetack_gbridge(), packetack_guilab(), GPacket::payload, PCK_SIZE, printstatus(), PRIVEMBED, program_sequencer(), QUADRANTS, read_log(), read_pk_param(), read_testimage(), READLOG, READPARM, REINIT, reinitdev(), RUN, Run, SAS_ABORT, SAS_BUSY, SAS_DUMMY, SAS_ERR, SAS_FREERUN, SAS_IDLE, SAS_STOP, SAS_TEST, Scan_time, Sendmsg, SEQMEM, start_sequencer(), STATUS, STOP, stop_idle(), SVBCheck, SVBCHECK, SVBTest, SVBTEST, SYNCHRO, SynchroTest, syntetize_multi(), test_seq_mem(), TEST_TASK, Tint, Verbose, VERBOSE, write_pk_param(), write_seqfile(), WRITEPARM, and zero_seq_mem().

Referenced by packetread_gbridge().

5.5.3.13 int send_image (int *nframe*)

Definition at line 2055 of file exec104.c.

References Acquired, array, Channel, check_socket_status(), close_socket(), DISCONNECTED, EMBED_DATA, errlog(), error_code(), GB_EMEMALLOC, gb_errno, GB_ERROR, GB_IO_EBADFD, get_socket_descriptor(), H_COL, H_ROW, IMG_HEADER, iolog(), mainloop_iterate(), MYFALSE, MYTRUE, N_COL, N_ROW, NEXTRA, Ngroup, NHEADER, Opera, Run, SAS_ABORT, sock_write(), and SynchroTest.

Referenced by dummy_image(), read_image(), and read_quadrant().

5.5.3.14 int update_event_sources (void)

Updates the array of pollfd structures with the event sources to check during poll.

If the number of event source is greater then the allocated array of pollfd structures, we realloc it.

Returns:

0 on success, -1 on error.

Note:

This function is used when the program is executed in daemon mode.

Definition at line 2197 of file exec104.c.

References sock_t::ioevents, pfds, POLLFDN, slist, slist_length(), slist_nth_data(), and sock_t::sockfd.

Referenced by MainLoop().

5.5.3.15 void remove_event_source (int *sockfd*)

Delete the selected data structure from the single linked list and remove from the main event loop the I/O source associated to data.

Descriptor closing is done inside I/O routines when unrecoverable error conditions are found on descriptor.

Updating of event sources is done inside the main loop.

Parameters:

sockfd the socket descriptor

Returns:

no value

Note:

This function is used when the program is executed in daemon mode.

Definition at line 2237 of file exec104.c.

References slist, slist_data_from_descriptor(), and slist_remove().

Referenced by remove_event_sourcetype().

5.5.3.16 void remove_event_sourcetype (int *type*)

Definition at line 2258 of file exec104.c.

References fd, get_socket_descriptor(), and remove_event_source().

Referenced by MainLoop(), and remove_embed_sources().

5.5.3.17 void remove_embed_sources ()

Definition at line 2268 of file exec104.c.

References EMBED_CMD, EMBED_DATA, and remove_event_sourcetype().

Referenced by MainLoop().

5.5.3.18 void remove_disconnected_sources (SList * *list*)

Removes all the event sources in DISCONNECTED status.

Definition at line 2281 of file exec104.c.

References slist, and slist_remove_disconnected().

Referenced by MainLoop().

5.5.3.19 void remove_nop_timeout (void)

Remove the timer installed when the NOP command is sent.

Definition at line 2294 of file exec104.c.

References get_current_time(), iolog(), NOP_IDLE, nopStatus, and timestamp.

Referenced by embedsys_keepalive().

5.5.3.20 void nop_timeout (int *signal*)

Signal handler for SIGALRM signal. SIGALRM signal is delivered upon the expiration of the timer setted by the setitimer() function.

Parameters:

signal the number of the signal received

Definition at line 2318 of file exec104.c.

References canjump, errlog(), iolog(), main_loop, NOP_IDLE, and nopStatus.

Referenced by catch_signals().

5.5.3.21 int embedsys_keepalive (void)

Send a NOP command packet to embedded server when KEEPALIVE_INTVL msec. are elapsed since last received packet.

Returns:

0 on success, < 0 on errors.

Definition at line 2343 of file exec104.c.

References COMANDO, elapsed_time(), iolog(), KEEPALIVE_INTVL, NOP, NOP_SENT, nopStatus, packetsend_gbridge(), remove_nop_timeout(), Run, SAS_IDLE, and timestamp.

Referenced by MainLoop().

5.5.3.22 int embedsys_keepalive_old ()

Sends teo NOP packets to gbridge each 10 sec, onl when the system is in idle state.

Definition at line 2395 of file exec104.c.

References COMANDO, elapsed_time(), NOP, packetsend_gbridge(), Run, SAS_IDLE, and timestamp.

5.5.3.23 int MainLoop (void)

Main execution loop in daemon mode.

Note:

This function is used when the program is executed in daemon mode.

Definition at line 2420 of file exec104.c.

References accept_connection(), canjump, check_sockettype_status(), close_all(), close_embed_sockets(), close_listen_socket(), CONNECTED, DISCONNECTED, dprint_slist(), dump_init_file(), elapsed_time(), EMBED_CMD, EMBED_DATA, EMBEDDED_CMD_PORT, EMBEDDED_DATA_PORT, embedsys_keepalive(), sock_t::errfunc, gb_errno, get_current_time(), init_power(), sock_t::iofunc, iolog(), Keepgoing, l_function(), l_header(), l_opera(), l_stdmessage(), LISTENING, LOGFILETIMEOUT, main_loop, MYFALSE, MYTRUE, _SList::next, packetread(), pfds, POLLFDN, read_power(), remove_disconnected_sources(), remove_embed_sources(), remove_event_sourcetype(), Run, SAS_IDLE, slist, slist_free(), slist_length(), slist_nth_data(), SOCK_TIMEOUT, sock_t::sockfd, sock_t::sockstatus, sock_t::socktype, strerror_104(), tcp_server_new(), Testmode, timestamp, and update_event_sources().

Referenced by main().

5.5.4 Variable Documentation

5.5.4.1 struct pollfd* pfds

Definition at line 157 of file exec104.c.

Referenced by MainLoop(), and update_event_sources().

5.5.4.2 sigjmp_buf main_loop [static]

Definition at line 159 of file exec104.c.

Referenced by MainLoop(), and nop_timeout().

5.5.4.3 volatile sig_atomic_t canjump [static]

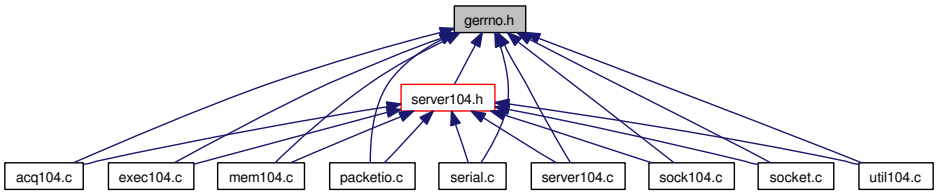
Definition at line 160 of file exec104.c.

Referenced by MainLoop(), and nop_timeout().

5.6 gerrno.h File Reference

Defines the error codes and the tasks originating the errors.

This graph shows which files directly or indirectly include this file:



Defines

- #define GB_WARNING 0x0000
- #define GB_ERROR 0x8000
- #define INITDEV_TASK 0x0800
- #define INTERNAL_TASK 0x1000
- #define PROGRAM_TASK 0x3000
- #define ACQ_TASK 0x4000
- #define SOCKETIO_TASK 0x5000
- #define PROTOCOL_TASK 0x6000
- #define TEST_TASK 0x7000
- #define GB_EMMEMIO 0x300
- #define GB_BADADDR 0x301
- #define GB_NOMMAPMEM 0x302
- #define GB_EMEMALLOC 0x310
- #define GB_EINVALMEM 0x311
- #define GB_EBADARG 0x320
- #define GB_ENVALBOARD 0X321
- #define GB_ELAMP 0x322
- #define GB_ESEQPROG 0X330
- #define GB_ESPROGFORMAT 0X331
- #define GB_ESPROGLONG 0x332
- #define GB_ESPROGSHORT 0x333
- #define GB_ESEQVERIFY 0x334
- #define GB_ELINK 0x340
- #define GB_ENOLINK 0x341
- #define GB_ELINKVB 0x342
- #define GB_ETRLINKVB 0x343
- #define GB_EBUFFVERIFY 0x344
- #define GB_EBUFFID 0x345
- #define GB_ESYSNOTOP 0x346
- #define GB_ESEQNOTOP GB_ESYSNOTOP
- #define GB_ESENSNOTOP GB_ESYSNOTOP
- #define GB_EVLEVEL GB_ESYSNOTOP
- #define GB_EDECODER GB_ESYSNOTOP
- #define GB_EADCONV GB_ESYSNOTOP

- #define GB_ENOSENSOR GB_ESYSNOTOP
- #define GB_ESEQRUNNING GB_ESYSNOTOP
- #define GB_ENOVCC GB_ESYSNOTOP
- #define GB_EIDLERUN 0x347
- #define GB_PROGERR 0x348
- #define GB_EHAMEG 0x349
- #define GB_EIMGSYNC 0x350
- #define GB_EPIXLESS 0x351
- #define GB_EPIXMORE 0x352
- #define GB_EHEADDESYNC 0x353
- #define GB_EFOVERRUN 0x354
- #define GB_ETIMEOUT 0x355
- #define GB_EFHEADER 0x356
- #define GB_EFDATA 0x357
- #define GB_RANGE_ROW 0x360
- #define GB_MATCH_ROW 0x361
- #define GB_ACQ_PROT_ERR 0x362
- #define GB_ACQ_TIMEOUT 0x367
- #define GB_ENVALOPT 0x380
- #define GB_EFILEOPEN 0x381
- #define GB_EINTERNAL 0x382
- #define GB_GBRIDGE_KILL 0x383
- #define GB_NSPACE_INVAL 0x384
- #define GB_CFTISIO_ERR 0x385
- #define GB_FITS_KEY_EIO 0x386
- #define GB_DS9_NOXPA 0x387
- #define GB_ESLIST 0x388
- #define GB_ESAVEIMG 0x389
- #define GB_PROT_ERR 0x401
- #define GB_CMD_NOTACK 0x402
- #define GB_CHKSUM_ERR 0x403
- #define GB_PROT_EFORMAT 0x404
- #define GB_EINVALMASK 0x405
- #define GB_FITGB_EPCK0 0x406
- #define GB_FITGB_EPCK1 0x407
- #define GB_FITGB_EPCK2 0x408
- #define GB_PROT_ERR 0x401
- #define GB_EPCKPROTOCOL GB_PROT_ERR
- #define GB_CMD_NOTACK 0x402
- #define GB_CHKSUM_ERR 0x403
- #define GB_PROT_EFORMAT 0x404
- #define GB_EINVALMASK 0x405
- #define GB_FITGB_EPCK0 0x406
- #define GB_FITGB_EPCK1 0x407
- #define GB_FITGB_EPCK2 0x408
- #define GB_IO_EBADFD 0x420
- #define GB_IO_EOF 0x421
- #define GB_IO_TIME_READ 0x422
- #define GB_IO_TIME_WRITE 0x423
- #define GB_IO_TIME_CONNECT 0x424

- #define GB_IO_CONNECT_ERR 0x425
- #define GB_IO_NONBLOCK_ERR 0x426
- #define GB_ECOMMEMBED 0x427
- #define GB_NOP_TIMEOUT GB_ECOMMEMBED
- #define GB_IO_POLLERR 0x428
- #define GB_IO_POLLHUP 0x429
- #define GB_IO_POLLNVAL 0x42A
- #define GB_IO_CLOSED 0x42B

Variables

- int gb_errno

5.6.1 Detailed Description

Defines the error codes and the tasks originating the errors.

Definition in file gerrno.h.

5.6.2 Define Documentation

5.6.2.1 #define GB_WARNING 0x0000

error code mask to signal a warning

Definition at line 50 of file gerrno.h.

Referenced by abort_sequencer(), analog_boardStatus(), check_data(), check_group(), compute_cycle(), compute_pix_readclk(), disable_sensor(), dummy_acquisition(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), fifo_ready(), init_hameg(), init_power(), initdev(), l_get_port(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), multiswitch(), off_power(), open_serial(), packetdecode(), program_DAC_Vlevel(), program_sequencer(), read_converters(), read_image(), read_init_file(), read_pk_param(), read_power(), read_quadrant(), restart_daemon(), sawtooth_dac_test(), sensor_onoff(), sequencer_counter(), sequencer_status(), serial_command(), serial_decode(), setowner(), start_sequencer(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_board_id(), treset_program(), verify_sequencer(), whichfilter(), write_pk_param(), and zero_seq_mem().

5.6.2.2 #define GB_ERROR 0x8000

error code mask to signal a error

Definition at line 51 of file gerrno.h.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardStatus(), disable_sensor(), enable_sensor(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initdev(), link_error(), main(), packetdecode(), print_fits_error(), program_DAC_filter(), read_converters(), read_image(), read_log(), read_quadrant(), read_testimage(), rmem104(), send_image(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_kindly(), stop_sequencer(), sys_running_status(), test_seq_mem(), whichfilter(), wmem104(), and zero_seq_mem().

5.6.2.3 #define INITDEV_TASK 0x0800

Initialization device task

Definition at line 53 of file gerrno.h.

Referenced by initdev(), l_opera(), and main().

5.6.2.4 #define INTERNAL_TASK 0x1000

Internal task (generic)

Definition at line 54 of file gerrno.h.

Referenced by initvar104(), and l_opera().

5.6.2.5 #define PROGRAM_TASK 0x3000

Programming electronics task

Definition at line 55 of file gerrno.h.

Referenced by init_hameg(), init_power(), l_opera(), off_power(), open_serial(), read_power(), serial_-command(), and serial_decode().

5.6.2.6 #define ACQ_TASK 0x4000

Test and real acquisition task

Definition at line 56 of file gerrno.h.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), l_opera(), packetdecode(), read_image(), and read_quadrant().

5.6.2.7 #define SOCKETIO_TASK 0x5000

Socket creation task

Definition at line 57 of file gerrno.h.

Referenced by l_opera().

5.6.2.8 #define PROTOCOL_TASK 0x6000

Communication protocol handling task

Definition at line 58 of file gerrno.h.

Referenced by l_opera().

5.6.2.9 #define TEST_TASK 0x7000

Menu test mode task

Definition at line 59 of file gerrno.h.

Referenced by l_opera(), MenuLoop(), packetdecode(), and sawtooth_dac_test().

5.6.2.10 #define GB_EMMEMIO 0x300

error in I/O access to mmaped memory

Definition at line 63 of file gerrno.h.

Referenced by read_testimage(), and strerror_104().

5.6.2.11 #define GB_BADADDR 0x301

memory bad address

Definition at line 64 of file gerrno.h.

Referenced by exercise_seq_mem(), rmem104(), strerror_104(), wmem104(), and write_pk_param().

5.6.2.12 #define GB_NOMMAPMEM 0x302

no memory mmapped

Definition at line 65 of file gerrno.h.

Referenced by rmem104(), strerror_104(), test_seq_mem(), wmem104(), and zero_seq_mem().

5.6.2.13 #define GB_EMEMALLOC 0x310

memory allocation error

Definition at line 67 of file gerrno.h.

Referenced by alloc_array_mem(), create_filepath(), local_server_new(), open_log_file(), read_converters(), send_image(), strerror_104(), tcp_server_new(), and tcp_socket_connect().

5.6.2.14 #define GB_EINVALMEM 0x311

invalid memory pointer

Definition at line 68 of file gerrno.h.

Referenced by read_image(), and strerror_104().

5.6.2.15 #define GB_EBADARG 0x320

invalid argument

Definition at line 70 of file gerrno.h.

Referenced by check_group(), compute_pix_readclk(), mainloop_iterate(), packetack(), packetdecode(), packetread_gbridge(), packetsend(), read_image(), read_pk_param(), strerror_104(), tcp_server_new(), tcp_socket_connect(), write_pk_param(), and write_seqfile().

5.6.2.16 #define GB_ENVALBOARD 0X321

invalid board (quadrant) number

Definition at line 71 of file gerrno.h.

Referenced by abort_sequencer(), analog_boardId(), analog_boardStatus(), compute_cycle(), disable_-sensor(), dummy_acquisition(), dump_seq_mem(), enable_sensor(), fifo_overflow(), fifo_ready(), handle_multi_acquisition(), handle_oneboard_acquisition(), program_DAC_Vlevel(), program_-sequencer(), read_converters(), sawtooth_dac_test(), sensor_onoff(), sequencer_counter(), sequencer_-status(), start_sequencer(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), strerror_104(), syntetize_multi(), syntetize_program(), sys_running_status(), test_seq_mem(), treset_program(), verify_-sequencer(), whichfilter(), and zero_seq_mem().

5.6.2.17 #define GB_ELAMP 0x322

Error in switching lamp on/off

Definition at line 72 of file gerrno.h.

Referenced by multiswitch(), and strerror_104().

5.6.2.18 #define GB_ESEQPROG 0X330

error in programming sequencer

Definition at line 74 of file gerrno.h.

Referenced by strerror_104().

5.6.2.19 #define GB_ESPROGFORMAT 0X331

wrong format in sequencer program

Definition at line 75 of file gerrno.h.

Referenced by program_sequencer(), strerror_104(), and verify_sequencer().

5.6.2.20 #define GB_ESPROGLONG 0x332

sequencer program too long

Definition at line 76 of file gerrno.h.

Referenced by program_sequencer(), strerror_104(), syntetize_multi(), and syntetize_program().

5.6.2.21 #define GB_ESPROGSHORT 0x333

sequencer program too short

Definition at line 77 of file gerrno.h.

Referenced by strerror_104(), syntetize_multi(), and syntetize_program().

5.6.2.22 #define GB_ESEQVERIFY 0x334

sequencer memory verify error

Definition at line 78 of file gerrno.h.

Referenced by exercise_seq_mem(), strerror_104(), syntetize_multi(), syntetize_program(), test_seq_-mem(), and verify_sequencer().

5.6.2.23 #define GB_ELINK 0x340

errors in optical link

Definition at line 80 of file gerrno.h.

Referenced by link_error(), and strerror_104().

5.6.2.24 #define GB_ENOLINK 0x341

no optical link active

Definition at line 81 of file gerrno.h.

Referenced by analog_boardStatus(), link_error(), program_DAC_filter(), read_log(), strerror_104(), test_-seq_mem(), and whichfilter().

5.6.2.25 #define GB_ELINKVB 0x342

errors in optical link on buffer board

Definition at line 82 of file gerrno.h.

Referenced by link_error(), link_status(), and strerror_104().

5.6.2.26 #define GB_ETRLINKVB 0x343

errors in transfer on VB board link

Definition at line 83 of file gerrno.h.

Referenced by link_error(), link_status(), and strerror_104().

5.6.2.27 #define GB_EBUFFVERIFY 0x344

buffer board verify error

Definition at line 85 of file gerrno.h.

Referenced by test_board_id().

5.6.2.28 #define GB_EBUFFID 0x345

invalid buffer board ID

Definition at line 86 of file gerrno.h.

Referenced by initdev(), and strerror_104().

5.6.2.29 #define GB_ESYSNOTOP 0x346

analog board not operative

Definition at line 88 of file gerrno.h.

Referenced by strerror_104(), and sys_running_status().

5.6.2.30 #define GB_ESEQNOTOP GB_ESYSNOTOP

sequencer not operative

Definition at line 89 of file gerrno.h.

Referenced by abort_broadcast(), abort_sequencer(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_kindly(), and stop_sequencer().

5.6.2.31 #define GB_ESENSNOTOP GB_ESYSNOTOP

error in sensor operation

Definition at line 90 of file gerrno.h.

Referenced by disable_sensor(), and enable_sensor().

5.6.2.32 #define GB_EVLEVEL GB_ESYSNOTOP

failure in Vlevel programming

Definition at line 91 of file gerrno.h.

Referenced by analog_boardStatus().

5.6.2.33 #define GB_EDECODER GB_ESYSNOTOP

error in address decoder

Definition at line 92 of file gerrno.h.

Referenced by analog_boardStatus().

5.6.2.34 #define GB_EADCONV GB_ESYSNOTOP

error in AD converter

Definition at line 93 of file gerrno.h.

Referenced by analog_boardStatus().

5.6.2.35 #define GB_ENOSENSOR GB_ESYSNOTOP

the sensor is not enabled

Definition at line 94 of file gerrno.h.

Referenced by start_sequencer().

5.6.2.36 #define GB_ESEQRUNNING GB_ESYSNOTOP

sequencer not running

Definition at line 95 of file gerrno.h.

5.6.2.37 #define GB_ENOVCC GB_ESYSNOTOP

failure in analog board alimentation

Definition at line 96 of file gerrno.h.

Referenced by analog_boardStatus().

5.6.2.38 #define GB_EIDLERUN 0x347

idle acquisitions running

Definition at line 97 of file gerrno.h.

Referenced by packetdecode().

5.6.2.39 #define GB_PROGERR 0x348

Error during acquisition param programming

Definition at line 98 of file gerrno.h.

Referenced by multivalid().

5.6.2.40 #define GB_EHAMEG 0x349

error in Hameg power supply

Definition at line 99 of file gerrno.h.

Referenced by init_power().

5.6.2.41 #define GB_EIMGSYNC 0x350

image desynchronization: too few pixels

Definition at line 102 of file gerrno.h.

Referenced by read_image(), read_quadrant(), and strerror_104().

5.6.2.42 #define GB_EPIXLESS 0x351

image desynchronization: few pixels

Definition at line 103 of file gerrno.h.

Referenced by read_image().

5.6.2.43 #define GB_EPIXMORE 0x352

image desynchronization: too many pixels

Definition at line 104 of file gerrno.h.

Referenced by read_image(), and strerror_104().

5.6.2.44 #define GB_EHEADDESYNC 0x353

image header desynchronization

Definition at line 105 of file gerrno.h.

Referenced by strerror_104().

5.6.2.45 #define GB_EFOVERRUN 0x354

fifo or frame data overrun

Definition at line 107 of file gerrno.h.

Referenced by fifo_overflow(), frame_ready(), quadrant_ready(), read_testimage(), and strerror_104().

5.6.2.46 #define GB_ETIMEOUT 0x355

timeout during I/O operations

Definition at line 109 of file gerrno.h.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), read_image(), read_log(), read_testimage(), and strerror_104().

5.6.2.47 #define GB_EFHEADER 0x356

error in image header

Definition at line 111 of file gerrno.h.

Referenced by check_data(), and strerror_104().

5.6.2.48 #define GB_EFDATA 0x357

error in image data autotest

Definition at line 112 of file gerrno.h.

Referenced by check_data(), and strerror_104().

5.6.2.49 #define GB_RANGE_ROW 0x360

error in interval row numbers

Definition at line 114 of file gerrno.h.

5.6.2.50 #define GB_MATCH_ROW 0x361

error in matching row number

Definition at line 115 of file gerrno.h.

5.6.2.51 #define GB_ACQ_PROT_ERR 0x362

error in image protocol

Definition at line 116 of file gerrno.h.

5.6.2.52 #define GB_ACQ_TIMEOUT 0x367

Definition at line 117 of file gerrno.h.

5.6.2.53 #define GB_ENVALOPT 0x380

invalid case option

Definition at line 120 of file gerrno.h.

Referenced by read_image(), and strerror_104().

5.6.2.54 #define GB_EFILEOPEN 0x381

error in opening file

Definition at line 121 of file gerrno.h.

Referenced by open_log_file(), read_converters(), strerror_104(), and verify_sequencer().

5.6.2.55 #define GB_EINTERNAL 0x382

internal error

Definition at line 122 of file gerrno.h.

Referenced by dump_init_file(), initdev(), initvar104(), l_get_port(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), print_fits_error(), read_init_file(), read_testimage(), restart_daemon(), sawtooth_dac_test(), setowner(), and strerror_104().

5.6.2.56 #define GB_GBRIDGE_KILL 0x383

gbridge received SIGTERM signal

Definition at line 124 of file gerrno.h.

5.6.2.57 #define GB_NSPACE_INVAL 0x384

Definition at line 125 of file gerrno.h.

5.6.2.58 #define GB_CFTISIO_ERR 0x385

cfitsio error routine

Definition at line 126 of file gerrno.h.

Referenced by print_fits_error().

5.6.2.59 #define GB_FITS_KEY_EIO 0x386

can't access keyword file

Definition at line 127 of file gerrno.h.

5.6.2.60 #define GB_DS9_NOXPA 0x387

Definition at line 128 of file gerrno.h.

5.6.2.61 #define GB_ESLIST 0x388

internal slist error

Definition at line 129 of file gerrno.h.

Referenced by open_giolamp(), and strerror_104().

5.6.2.62 #define GB_ESAVEIMG 0x389

error in saving data on disk

Definition at line 130 of file gerrno.h.

Referenced by read_image(), read_quadrant(), and read_testimage().

5.6.2.63 #define GB_PROT_ERR 0x401

error in trasmission protocol

Definition at line 143 of file gerrno.h.

5.6.2.64 #define GB_CMD_NOTACK 0x402

command not confirmed by embed

Definition at line 145 of file gerrno.h.

Referenced by serial_decode().

5.6.2.65 #define GB_CHKSUM_ERR 0x403

protocol checksum error

Definition at line 146 of file gerrno.h.

Referenced by packetread(), and strerror_104().

5.6.2.66 #define GB_PROT_EFORMAT 0x404

not conformed format

not conformed format unused

Definition at line 147 of file gerrno.h.

Referenced by strerror_104().

5.6.2.67 #define GB_EINVALMASK 0x405

Definition at line 148 of file gerrno.h.

Referenced by strerror_104().

5.6.2.68 #define GB_FITGB_EPCK0 0x406

Definition at line 149 of file gerrno.h.

5.6.2.69 #define GB_FITGB_EPCK1 0x407

Definition at line 150 of file gerrno.h.

5.6.2.70 #define GB_FITGB_EPCK2 0x408

Definition at line 151 of file gerrno.h.

5.6.2.71 #define GB_PROT_ERR 0x401

error in trasmission protocol

Definition at line 143 of file gerrno.h.

5.6.2.72 #define GB_EPCKPROTOCOL GB_PROT_ERR

alias for GB_PROT_ERR

Definition at line 144 of file gerrno.h.

5.6.2.73 #define GB_CMD_NOTACK 0x402

command not confirmed by embed

Definition at line 145 of file gerrno.h.

5.6.2.74 #define GB_CHKSUM_ERR 0x403

protocol checksum error

Definition at line 146 of file gerrno.h.

5.6.2.75 #define GB_PROT_EFORMAT 0x404

not conformed format

not conformed format unused

Definition at line 147 of file gerrno.h.

5.6.2.76 #define GB_EINVALMASK 0x405

Definition at line 148 of file gerrno.h.

5.6.2.77 #define GB_FITGB_EPCK0 0x406

Definition at line 149 of file gerrno.h.

5.6.2.78 #define GB_FITGB_EPCK1 0x407

Definition at line 150 of file gerrno.h.

5.6.2.79 #define GB_FITGB_EPCK2 0x408

Definition at line 151 of file gerrno.h.

5.6.2.80 #define GB_IO_EBADFD 0x420

bad file number

Definition at line 154 of file gerrno.h.

Referenced by send_image().

5.6.2.81 #define GB_IO_EOF 0x421

end of file

Definition at line 155 of file gerrno.h.

Referenced by checksum_error(), sock_read(), sock_write(), and strerror_104().

5.6.2.82 #define GB_IO_TIME_READ 0x422

timeout in reading

Definition at line 156 of file gerrno.h.

Referenced by serial_rdline(), sock_read(), and strerror_104().

5.6.2.83 #define GB_IO_TIME_WRITE 0x423

timeout in writing

Definition at line 157 of file gerrno.h.

Referenced by sock_write(), and strerror_104().

5.6.2.84 #define GB_IO_TIME_CONNECT 0x424

timeout during socket connection

Definition at line 158 of file gerrno.h.

Referenced by strerror_104(), tcp_server_accept(), and tcp_socket_connect().

5.6.2.85 #define GB_IO_CONNECT_ERR 0x425

socket connection error

Definition at line 159 of file gerrno.h.

Referenced by strerror_104(), and tcp_socket_connect().

5.6.2.86 #define GB_IO_NONBLOCK_ERR 0x426

error in setting non-blocking

Definition at line 160 of file gerrno.h.

Referenced by local_server_new(), strerror_104(), and tcp_server_new().

5.6.2.87 #define GB_ECOMMEMBED 0x427

embedded server not responding

Definition at line 161 of file gerrno.h.

5.6.2.88 #define GB_NOP_TIMEOUT GB_ECOMMEMBED

alias for GB_ECOMMEMBED

Definition at line 162 of file gerrno.h.

5.6.2.89 #define GB_IO_POLLERR 0x428

error in poll

Definition at line 163 of file gerrno.h.

Referenced by sock_read(), sock_write(), and strerror_104().

5.6.2.90 #define GB_IO_POLLHUP 0x429

POLLHUP condition

Definition at line 164 of file gerrno.h.

Referenced by sock_read(), sock_write(), and strerror_104().

5.6.2.91 #define GB_IO_POLLNVAL 0x42A

POLLNVAL condition

Definition at line 165 of file gerrno.h.

Referenced by sock_read(), sock_write(), and strerror_104().

5.6.2.92 #define GB_IO_CLOSED 0x42B

connection is closed

Definition at line 166 of file gerrno.h.

Referenced by packetack(), packetread_gbridge(), packetsend(), and strerror_104().

5.6.3 Variable Documentation

5.6.3.1 int gb_errno

global error code for gbridge/server104 applications. It is composed of two parts:

- the 15 bit signal the error severity GB_ERROR or GB_WARNING
- the next 4 msb represent the task originating the error
- the remaining bits represent the error code

error identifier

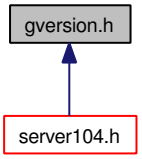
Definition at line 255 of file server104.h.

Referenced by abort_broadcast(), abort_sequencer(), accept_connection(), alloc_array_mem(), analog_boardId(), analog_boardStatus(), check_data(), check_group(), checksum_error(), compute_cycle(), create_filepath(), disable_sensor(), dummy_acquisition(), dummy_image(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), fifo_overflow(), fifo_ready(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init_hameg(), init_power(), initdev(), initvar104(), l_get_port(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), local_server_new(), main(), MainLoop(), mainloop_iterate(), MenuLoop(), multiswitch(), multivalid(), off_power(), open_giolamp(), open_log_file(), open_serial(), packetack(), packetdecode(), packetread(), packetread_gbridge(), packetsend(), print_fits_error(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), read_converters(), read_image(), read_init_file(), read_log(), read_power(), read_quadrant(), read_testimage(), restart_daemon(), rmem104(), sawtooth_dac_test(), send_image(), sensor_onoff(), sequencer_counter(), sequencer_status(), serial_command(), serial_decode(), serial_rdline(), set_close_on_exec(), set_descriptor_flags(), setowner(), sock_read(), sock_write(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), tcp_server_accept(), tcp_server_new(), tcp_socket_connect(), test_board_id(), test_seq_mem(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), write_seqfile(), and zero_seq_mem().

5.7 gversion.h File Reference

Automatically generated GLOBAL version number.

This graph shows which files directly or indirectly include this file:



Defines

- #define GVERSION 2.50252
- #define G_REVISION "2.50252"
- #define G_SOURCES "\ Makefile.am 1.15 Up-to-date \ acq104.c 2.38 Up-to-date \ autogen.sh 1.2 Up-to-date \ command.def 2.4 Up-to-date \ configure.in 2.5 Up-to-date \ data104.c 2.4 Up-to-date \ define.h 2.12 Up-to-date \ exec104.c 2.27 Up-to-date \ gerrno.h 2.7 Up-to-date \ list104.c 2.2 Up-to-date \ lpt104.c 2.5 Up-to-date \ lpt104.h 2.4 Up-to-date \ mem104.c 2.11 Up-to-date \ mk_-version.pl 2.1 Up-to-date \ multi.txt 1.3 Up-to-date \ packetio.c 2.3 Up-to-date \ sequenza.txt 1.2 Up-to-date \ serial.c 2.10 Up-to-date \ server104.c 2.8 Up-to-date \ server104.h 2.18 Up-to-date \ slist.h 2.3 Up-to-date \ sock104.c 2.8 Up-to-date \ socket.c 2.5 Up-to-date \ socket.h 2.5 Up-to-date \ spooler.c 0.12 Up-to-date \ string104.c 1.6 Up-to-date \ util104.c 2.22 Up-to-date \ version.sh 1.2 Up-to-date \ win104.c 2.8 Up-to-date \"

5.7.1 Detailed Description

Automatically generated GLOBAL version number.

Definition in file gversion.h.

5.7.2 Define Documentation

5.7.2.1 #define GVERSION 2.50252

Definition at line 4 of file gversion.h.

5.7.2.2 #define G_REVISION "2.50252"

Definition at line 5 of file gversion.h.

Referenced by initwin(), l_header(), lcd_init(), open_log_file(), and print_help().

5.7.2.3 #define G_SOURCES "\ Makefile.am 1.15 Up-to-date \ acq104.c 2.38 Up-to-date \ autogen.sh 1.2 Up-to-date \ command.def 2.4 Up-to-date \ configure.in 2.5 Up-to-date \ data104.c 2.4 Up-to-date \ define.h 2.12 Up-to-date \ exec104.c 2.27 Up-to-date \ gerrno.h 2.7 Up-to-date \ list104.c 2.2 Up-to-date \ lpt104.c 2.5 Up-to-date \ lpt104.h 2.4 Up-to-date \ mem104.c 2.11 Up-to-date \ mk_version.pl 2.1 Up-to-date \ multi.txt 1.3 Up-to-date \ packetio.c 2.3 Up-to-date \ sequenza.txt 1.2 Up-to-date \ serial.c 2.10 Up-to-date \ server104.c 2.8 Up-to-date \ server104.h 2.18 Up-to-date \ slist.h 2.3 Up-to-date \ sock104.c 2.8 Up-to-date \ socket.c 2.5 Up-to-date \ socket.h 2.5 Up-to-date \ spooler.c 0.12 Up-to-date \ string104.c 1.6 Up-to-date \ util104.c 2.22 Up-to-date \ version.sh 1.2 Up-to-date \ win104.c 2.8 Up-to-date \"

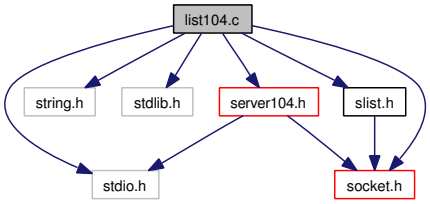
Definition at line 13 of file gversion.h.

5.8 list104.c File Reference

Functions to create and manipulate singly-linked lists.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "slist.h"
#include "socket.h"
#include "server104.h"
```

Include dependency graph for list104.c:



Functions

- static int compare_socktype (void *a, void *b)
- static int compare_socket (void *a, void *b)
- static int compare_cmd (void *a, void *b)
- SList * slist_new (void)
- SList * slist_remove (SList *head, sock_t *data)
- SList * slist_delete_node (SList *head, SList *node)
- void slist_free (SList *head)
- SList * slist_append (SList *head, void *data)
- SList * slist_find_custom (SList *head, CompareFunc func, void *data)
- int slist_length (SList *list)
- void * slist_nth_data (SList *list, int n)
- SList * slist_data_from_value (SList *head, CompareFunc func, int value)
- int mempool_index (void)
- int mempool_alloc (void)
- SList * slist_remove_disconnected (SList *head)
- sock_t * slist_data_from_descriptor (SList *head, int sockfd)
- sock_t * slist_data_from_socktype (SList *head, int socktype)
- cmd_t * slist_data_from_command (SList *head, int cmd)
- void dprint_slist (SList *list, int level)

5.8.1 Detailed Description

Functions to create and manipulate singly-linked lists.

Definition in file list104.c.

5.8.2 Function Documentation

5.8.2.1 static int compare_socktype (void * a, void * b) [static]

Callback function used by slist_find_custom(). Compares the type of the socket passed as second argument (i.e. **b**) with the corresponding element of the data structure of the node.

Parameters:

a the data node

b the socket type to compare

Returns:

0 if the match exists, else < 0

Definition at line 66 of file list104.c.

References _SList::data.

Referenced by slist_data_from_socktype().

5.8.2.2 static int compare_socket (void * a, void * b) [static]

Callback function used by slist_find_custom(). Compares the socket descriptor value passed as second argument (i.e. **b**) with the corresponding element of the socket data structure (sock_t) of the node.

Parameters:

a the data node

b the socket descriptor to compare

Returns:

0 if the match exists, else < 0

See also:

sock_t

Definition at line 94 of file list104.c.

References _SList::data, and fd.

Referenced by slist_data_from_descriptor().

5.8.2.3 static int compare_cmd (void * a, void * b) [static]

Callback function used by slist_find_custom(). Compares the command hex value passed as second argument (i.e. **b**) with the corresponding element of the socket data structure (cmd_t) of the node.

Parameters:

a the data node

b the socket descriptor to compare

Returns:

0 if the match exists, else < 0

See also:

cmd_t

Definition at line 122 of file list104.c.

References _SList::data.

Referenced by slist_data_from_command().

5.8.2.4 SList* slist_new (void)

Allocate space for one SList element. It is called by the slist_append() function and so is rarely used on its own.

Returns:

a pointer to the newly-allocated SList, NULL otherwise.

Definition at line 147 of file list104.c.

Referenced by slist_append().

5.8.2.5 SList* slist_remove (SList * *head*, sock_t * *data*)

Remove a node from the list whose data element is equal to tha passed as second argument (**data**).

It frees the memeory allocated for the node data and for the node.

Parameters:

head a SList

data the data of the node to be deleted

Returns:

a pointer to the head of the singly-linked list.

Definition at line 169 of file list104.c.

References _SList::data, MEM_CHUNK_SIZE, and _SList::next.

Referenced by remove_event_source().

5.8.2.6 SList* slist_delete_node (SList * *head*, SList * *node*)

Definition at line 199 of file list104.c.

References _SList::next.

5.8.2.7 void slist_free (SList * *head*)

Frees all of the memory used by a SList.

Parameters:

head a SList

Definition at line 231 of file list104.c.

References _SList::data, MEM_CHUNK_SIZE, and _SList::next.

Referenced by MainLoop().

5.8.2.8 SList* slist_append (SList * *head*, void * *data*)

Adds a new element to the beginning of the list

Parameters:

head a SList

data the data for the new element

Returns:

the new start of the SList.

Definition at line 252 of file list104.c.

References _SList::data, _SList::next, and slist_new().

Referenced by accept_connection(), and open_giolamp().

5.8.2.9 SList* slist_find_custom (SList * *head*, CompareFunc *func*, void * *data*)

Finds an element in a SList, using a supplied function to find the desired element. It iterates over the list, calling the given function which should return 0 when the desired element is found. The function takes two arguments, the SList element's data as the first argument and the given user data.

Parameters:

head the SList

func the function to call for each element.

data user data passed to the function

Returns:

the found Slist element, or NULL if it is not found

Note:

This function iterates over the whole list to count its elements.

Definition at line 293 of file list104.c.

References _SList::next.

Referenced by slist_data_from_value().

5.8.2.10 int slist_length (SList * *list*)

Gets he number of elements in a SList.

Parameters:

list the SList to operate on

Returns:

the number of elements in the SList.

Note:

This function iterates over the whole list to count its elements.

Definition at line 318 of file list104.c.

References _SList::next.

Referenced by MainLoop(), and update_event_sources().

5.8.2.11 void* slist_nth_data (SList * *list*, int *n*)

Gets the data of the element at the given position.

Parameters:

list a SList

n the position of the element

Returns:

the element's data, or NULL if the position is off the end of the Slist

Definition at line 343 of file list104.c.

References _SList::data, and _SList::next.

Referenced by MainLoop(), and update_event_sources().

5.8.2.12 SList* slist_data_from_value (SList * *head*, CompareFunc *func*, int *value*)

Definition at line 350 of file list104.c.

References slist_find_custom().

Referenced by slist_data_from_command(), slist_data_from_descriptor(), and slist_data_from_socktype().

5.8.2.13 int mempool_index (void)

Looks for the first free entry inside the memory pool table and returns its position.

Returns:

the position of the first free entry, < 0 on error

Definition at line 371 of file list104.c.

References iolog(), MEM_CHUNK_SIZE, MEM_POOL_ITEM, and mempool.

Referenced by accept_connection().

5.8.2.14 int mempool_alloc (void)

Allocates the memory used as a pool for list data.

Returns:

0 on success, < 0 on error.

Definition at line 397 of file list104.c.

References iolog(), MEM_CHUNK_SIZE, MEM_POOL_ITEM, and mempool.

Referenced by main().

5.8.2.15 SList* slist_remove_disconnected (SList * *head*)

Removes all list nodes with sockstatus equal disconnected.

Returns:

the new head of the list.

Definition at line 438 of file list104.c.

References _SList::data, DISCONNECTED, MEM_CHUNK_SIZE, and _SList::next.

Referenced by remove_disconnected_sources().

5.8.2.16 sock_t* slist_data_from_descriptor (SList * *head*, int *sockfd*)

It returns the socket structure of type sock_t whose socket descriptor element is equal to the second argument.

Parameters:

head a SList

sockfd the socket type (EMBED_CMD or EMBED_DATA)

Returns:

the structure of type sock_t with socket descriptor element equal to sockfd

See also:

sock_t

Definition at line 477 of file list104.c.

References compare_socket(), _SList::data, and slist_data_from_value().

Referenced by check_socket_status(), close_socket(), and remove_event_source().

5.8.2.17 sock_t* slist_data_from_socktype (SList * *head*, int *socktype*)

It returns the socket structure of type sock_t whose socket type element is equal to the second argument.

Parameters:

head a SList

socktype the socket type (EMBED_CMD or EMBED_DATA)

Returns:

the structure of type sock_t with socket ype element equal to socktype

See also:

sock_t

Definition at line 503 of file list104.c.

References compare_socktype(), _SList::data, and slist_data_from_value().

Referenced by close_sockettype(), and get_socket_descriptor().

5.8.2.18 cmd_t* slist_data_from_command (SList * *head*, int *cmd*)

It returns the command structure of type cmd_t whose command value is equal to the second argument of the function.

Parameters:

head a SList

cmd the command hex value

Returns:

the structure of type cmd_t with command info in string format.

See also:

cmd_t

Definition at line 529 of file list104.c.

References compare_cmd(), _SList::data, and slist_data_from_value().

5.8.2.19 void dprint_slist (SList * *list*, int *level*)

Debug printing of SList elements.

Parameters:

list a SList

level the verbosity level

Definition at line 550 of file list104.c.

References _SList::data, dump_sock_status(), iolog(), and _SList::next.

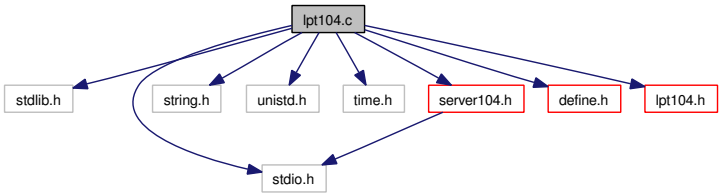
Referenced by MainLoop().

5.9 lpt104.c File Reference

A Linux LPT user-space driver for driving the external LCD display.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include "server104.h"
#include "define.h"
#include "lpt104.h"
```

Include dependency graph for lpt104.c:



Defines

- #define _XOPEN_SOURCE
- #define __USE_XOPEN

Functions

- int l_get_port (unsigned int base_addr, unsigned int port_len)
- void l_senddata (int port, unsigned char flags, unsigned char data)
- void l_output (int port, int data)
- int l_init_port (int port)
- int l_position (int x, int y)
- int l_w_ch_pos (unsigned char ich, int x, int y)
- int l_w_string (char *buff1, int x, int y)
- void l_clear (void)
- void l_home (void)
- int lcd_init (void)
- void l_stdmessage (void)
- void l_header (void)
- void l_function (char *funct_name)
- void l_opera (void)
- void l_operation (char *operation)
- void l_error (char *error)

Variables

- static int l_width = LCD_DEFAULT_WIDTH
- static int l_height = LCD_DEFAULT_HEIGHT
- static int l_x = 0
- static int l_y = 0
- static int l_port = BASE_LPT0

5.9.1 Detailed Description

A Linux LPT user-space driver for driving the external LCD display.

Definition in file lpt104.c.

5.9.2 Define Documentation

5.9.2.1 #define _XOPEN_SOURCE

definition for glibc2 which needs this

Definition at line 56 of file lpt104.c.

5.9.2.2 #define __USE_XOPEN

definition for glibc2 which needs this

Definition at line 57 of file lpt104.c.

5.9.3 Function Documentation

5.9.3.1 int l_get_port (unsigned int *base_addr*, unsigned int *port_len*)

Allocate the LPT port I/O space

Parameters:

base_addr is the initial address of LPT port

port_len number of I/O addresses reserved.

Returns:

MYTRUE if success, MYFALSE otherwise.

Definition at line 100 of file lpt104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), MYFALSE, MYTRUE, Opera, and Verbose.

Referenced by l_init_port().

5.9.3.2 void l_senddata (int *port*, unsigned char *flags*, unsigned char *data*)

Send data to selected LPT port. Handle data/control selection and strobe EN.

Parameters:

port LPT port address

flags select if data (RS_DATA) or commands

data characters or commands to be outputted to LCD.

Definition at line 130 of file lpt104.c.

References EN1, FAULT, INMASK, iolog(), OUTMASK, port_in(), port_out(), RS, and Verbose.

Referenced by l_clear(), l_home(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), and lcd_init().

5.9.3.3 void l_output (int *port*, int *data*)

Low level send data routine to selected LPT port. Do not handle data/control selection It only output data and strobe LE

Parameters:

port LPT port address

data characters or commands to be outputted to LCD.

Definition at line 171 of file lpt104.c.

References iolog(), LE, OUTMASK, port_out(), and Verbose.

5.9.3.4 int l_init_port (int *port*)

Initialize the LPT port in LCD compatible mode.

Parameters:

port LPT port address

Returns:

MYTRUE if success, MYFALSE otherwise.

Definition at line 194 of file lpt104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), l_get_port(), l_-senddata(), MYFALSE, MYTRUE, Opera, and Verbose.

Referenced by lcd_init().

5.9.3.5 int l_position (int *x*, int *y*)

Set position on LCD display.

Parameters:

x column position to program

y row position to program

Returns:

MYTRUE if success, MYFALSE otherwise.

Definition at line 226 of file lpt104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), l_height, l_port, l_senddata(), l_width, l_x, l_y, MYFALSE, MYTRUE, Opera, and Verbose.

Referenced by l_w_ch_pos(), and l_w_string().

5.9.3.6 int l_w_ch_pos (unsigned char *ich*, int *x*, int *y*)

Write a char in a position on LCD display

Parameters:

ich char to be printed

x column position to program

y row position to program

Returns:

MYTRUE if success, MYFALSE otherwise.

Definition at line 268 of file lpt104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), l_height, l_port, l_position(), l_senddata(), l_width, l_x, l_y, MYFALSE, MYTRUE, Opera, and Verbose.

5.9.3.7 int l_w_string (char * *buff1*, int *x*, int *y*)

Write a string in a position on LCD display

Parameters:

buff1 string to be printed

x column position to program

y row position to program

Returns:

MYTRUE if success, MYFALSE otherwise.

Definition at line 304 of file lpt104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), l_height, l_port, l_position(), l_senddata(), l_width, l_x, l_y, MYFALSE, MYTRUE, Opera, and Verbose.

Referenced by l_error(), l_function(), l_header(), l_opera(), l_operation(), l_stdmessage(), and lcd_init().

5.9.3.8 void l_clear (void)

Clear LCD display

Definition at line 345 of file lpt104.c.

References l_home(), l_port, and l_senddata().

Referenced by l_header(), l_stdmessage(), and lcd_init().

5.9.3.9 void l_home (void)

set current position on LCD to home (0,0)

Definition at line 358 of file lpt104.c.

References l_port, l_senddata(), l_x, and l_y.

Referenced by l_clear().

5.9.3.10 int lcd_init (void)

Init LPT port and init LCD device. Print hello msg

Returns:

MYTRUE on success

Definition at line 373 of file lpt104.c.

References errlog(), error_code(), G_REVISION, GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), l_clear(), l_init_port(), l_port, l_senddata(), l_w_string(), Lcd_present, MYFALSE, MYTRUE, Opera, and Verbose.

Referenced by initdev().

5.9.3.11 void l_stdmessage (void)

Print header and initialize the row 1-3

routines specific to server104 program the display use is as follow:

- 1 - header: 'server104, version, old'
- 2 - function in use
- 3 - operation, if any
- 4 - errors / no errors.

header call clears all function call clear operation operation clears only its line. error clears only its line.

Definition at line 443 of file lpt104.c.

References iolog(), l_clear(), l_header(), l_w_string(), l_width, Lcd_present, MYFALSE, and Prog_age.

Referenced by MainLoop().

5.9.3.12 void l_header (void)

print the top line on LCD

Definition at line 516 of file lpt104.c.

References G_REVISION, l_clear(), l_w_string(), l_width, Lcd_present, and MYFALSE.

Referenced by l_stdmessage(), MainLoop(), and MenuLoop().

5.9.3.13 void l_function (char * *funct_name*)

print the function in use on LCD

Parameters:

funct_name the name of the function running

Definition at line 555 of file lpt104.c.

References l_w_string(), Lcd_present, and MYFALSE.

Referenced by abort_broadcast(), abort_sequencer(), check_data(), disable_sensor(), dump_data(), enable_sensor(), exercise_seq_mem(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), MainLoop(), MenuLoop(), read_converters(), read_image(), read_quadrant(), read_testimage(), reset_fifo(), sawtooth_dac_test(), start_broadcast(), start_sequencer(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), test_board_id(), test_seq_mem(), and zero_seq_mem().

5.9.3.14 void l_opera (void)

print the operation status on LCD according to Opera variable

Definition at line 575 of file lpt104.c.

References ACQ_TASK, INITDEV_TASK, INTERNAL_TASK, l_w_string(), Lcd_present, MYFALSE, Opera, PROGRAM_TASK, PROTOCOL_TASK, Run, SAS_DUMMY, SOCKETIO_TASK, and TEST_TASK.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initdev(), MainLoop(), and MenuLoop().

5.9.3.15 void l_operation (char * *operation*)

print the operation status on LCD

Parameters:

operation the operation status

Definition at line 631 of file lpt104.c.

References l_w_string(), Lcd_present, and MYFALSE.

5.9.3.16 void l_error (char * *error*)

print the error status on LCD

Parameters:

error The error message to be published

Definition at line 653 of file lpt104.c.

References iolog(), l_w_string(), Lcd_present, and MYFALSE.

Referenced by check_data(), dump_seq_mem(), exercise_seq_mem(), read_converters(), read_image(), read_log(), read_quadrant(), read_testimage(), test_board_id(), and test_seq_mem().

5.9.4 Variable Documentation

5.9.4.1 int l_width = LCD_DEFAULT_WIDTH [static]

Width of display

Definition at line 75 of file lpt104.c.

Referenced by l_header(), l_position(), l_stdmessage(), l_w_ch_pos(), and l_w_string().

5.9.4.2 int l_height = LCD_DEFAULT_HEIGHT [static]

Height of display

Definition at line 76 of file lpt104.c.

Referenced by l_position(), l_w_ch_pos(), and l_w_string().

5.9.4.3 int l_x = 0 [static]

current column position on LCD display

Definition at line 79 of file lpt104.c.

Referenced by l_home(), l_position(), l_w_ch_pos(), and l_w_string().

5.9.4.4 int l_y = 0 [static]

current row position on LCD display

Definition at line 80 of file lpt104.c.

Referenced by l_home(), l_position(), l_w_ch_pos(), and l_w_string().

5.9.4.5 int l_port = BASE_LPT0 [static]

address of LPT port to LCD display

Definition at line 83 of file lpt104.c.

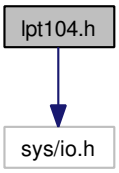
Referenced by l_clear(), l_home(), l_position(), l_w_ch_pos(), l_w_string(), and lcd_init().

5.10 lpt104.h File Reference

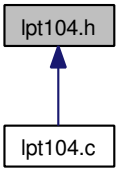
Definitions for lpt104.

`#include <sys/io.h>`

Include dependency graph for lpt104.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define LCD_DEFAULT_WIDTH 20
- #define LCD_DEFAULT_HEIGHT 4
- #define LCD_DEFAULT_CELLWIDTH 5
- #define LCD_DEFAULT_CELLHEIGHT 8
- #define BACKLIGHT_OFF 0
- #define BACKLIGHT_ON 1
- #define NUM_CCs 8
- #define BASE_LPT0 0x378
- #define nSTRB 0x01
- #define STRB 0x01
- #define nLF 0x02
- #define LF 0x02
- #define INIT 0x04
- #define nSEL 0x08
- #define SEL 0x08
- #define ENIRQ 0x10
- #define ENBI 0x20
- #define OUTMASK 0x0B
- #define nFAULT 0x08
- #define FAULT 0x08
- #define SELIN 0x10
- #define PAPEREND 0x20
- #define nACK 0x40
- #define ACK 0x40

- #define BUSY 0x80
- #define IRQ 0x02
- #define INMASK 0x84
- #define EN1 STRB
- #define EN2 SEL
- #define EN3 LF
- #define RW LFi
- #define RS INIT
- #define BL SEL
- #define LE LF

Enumerations

- enum CGmode {
  standard,
  vbar,
  hbar,
  bignum,
  bigchar }

Functions

- static int port_in (unsigned short port)
- static void port_out (unsigned short port, unsigned char val)
- static int port_access (unsigned short port)
- static int port_access_multiple (unsigned short port, int count)

5.10.1 Detailed Description

Definitions for lpt104.

Definition in file lpt104.h.

5.10.2 Define Documentation

5.10.2.1 #define LCD_DEFAULT_WIDTH 20

Default Whidth

Definition at line 33 of file lpt104.h.

5.10.2.2 #define LCD_DEFAULT_HEIGHT 4

Default height

Definition at line 34 of file lpt104.h.

5.10.2.3 #define LCD_DEFAULT_CELLWIDTH 5

Definition at line 35 of file lpt104.h.

5.10.2.4 #define LCD_DEFAULT_CELLHEIGHT 8

Definition at line 36 of file lpt104.h.

5.10.2.5 #define BACKLIGHT_OFF 0

Definition at line 39 of file lpt104.h.

5.10.2.6 #define BACKLIGHT_ON 1

Definition at line 40 of file lpt104.h.

5.10.2.7 #define NUM_CCs 8

number of custom characters

Definition at line 45 of file lpt104.h.

5.10.2.8 #define BASE_LPT0 0x378

base i/o address of lpt0

Definition at line 115 of file lpt104.h.

5.10.2.9 #define nSTRB 0x01

pin 1; negative logic

Definition at line 120 of file lpt104.h.

5.10.2.10 #define STRB 0x01

Definition at line 121 of file lpt104.h.

5.10.2.11 #define nLF 0x02

pin 14

Definition at line 122 of file lpt104.h.

5.10.2.12 #define LF 0x02

Definition at line 123 of file lpt104.h.

5.10.2.13 #define INIT 0x04

pin 16; the only positive logic output line

Definition at line 124 of file lpt104.h.

5.10.2.14 #define nSEL 0x08

pin 17

Definition at line 125 of file lpt104.h.

5.10.2.15 #define SEL 0x08

Definition at line 126 of file lpt104.h.

5.10.2.16 #define ENIRQ 0x10

Enable IRQ via ACK line (don't enable this without setting up interrupt stuff too)

Definition at line 127 of file lpt104.h.

5.10.2.17 #define ENBI 0x20

Enable bi-directional port (is nice to play with! I first didn't know a SPP could do this)

Definition at line 129 of file lpt104.h.

5.10.2.18 #define OUTMASK 0x0B

SEL, LF and STRB are hardware inverted

Definition at line 131 of file lpt104.h.

Referenced by l_output(), and l_senddata().

5.10.2.19 #define nFAULT 0x08

pin 15

Definition at line 137 of file lpt104.h.

5.10.2.20 #define FAULT 0x08

Definition at line 138 of file lpt104.h.

Referenced by l_senddata().

5.10.2.21 #define SELIN 0x10

pin 13

Definition at line 139 of file lpt104.h.

5.10.2.22 #define PAPEREND 0x20

pin 12

Definition at line 140 of file lpt104.h.

5.10.2.23 #define nACK 0x40

pin 10

Definition at line 141 of file lpt104.h.

5.10.2.24 #define ACK 0x40

Definition at line 142 of file lpt104.h.

5.10.2.25 #define BUSY 0x80

pin 11

Definition at line 143 of file lpt104.h.

5.10.2.26 #define IRQ 0x02

Definition at line 144 of file lpt104.h.

5.10.2.27 #define INMASK 0x84

BUSY input and the IRQ indicator are inverted

Definition at line 145 of file lpt104.h.

Referenced by l_senddata().

5.10.2.28 #define EN1 STRB

enable Display 1

Definition at line 149 of file lpt104.h.

Referenced by l_senddata().

5.10.2.29 #define EN2 SEL

Definition at line 150 of file lpt104.h.

5.10.2.30 #define EN3 LF

Definition at line 151 of file lpt104.h.

5.10.2.31 #define RW LFi

read/write

Definition at line 152 of file lpt104.h.

5.10.2.32 #define RS INIT

Definition at line 153 of file lpt104.h.

Referenced by l_senddata().

5.10.2.33 #define BL SEL

Definition at line 154 of file lpt104.h.

5.10.2.34 #define LE LF

Latch Enable

Definition at line 155 of file lpt104.h.

Referenced by l_output().

5.10.3 Enumeration Type Documentation

5.10.3.1 enum CGmode

Enumerator:

standard only char 0 is used for heartbeat

vbar vertical bars

hbar horizontal bars

bignum big numbers

bigchar big characters

Definition at line 47 of file lpt104.h.

5.10.4 Function Documentation

5.10.4.1 static int port_in (unsigned short *port*) [inline, static]

Read a byte from port

Definition at line 164 of file lpt104.h.

Referenced by l_senddata().

5.10.4.2 static void port_out (unsigned short *port*, unsigned char *val*) [inline, static]

Write a byte 'val' to port

Definition at line 169 of file lpt104.h.

Referenced by l_output(), and l_senddata().

5.10.4.3 static int port_access (unsigned short *port*) [inline, static]

Get access to a specific port

Definition at line 173 of file lpt104.h.

Referenced by port_access_multiple().

5.10.4.4 static int port_access_multiple (unsigned short *port*, int *count*) [inline, static]

Get access to multiple ports at once

Definition at line 190 of file lpt104.h.

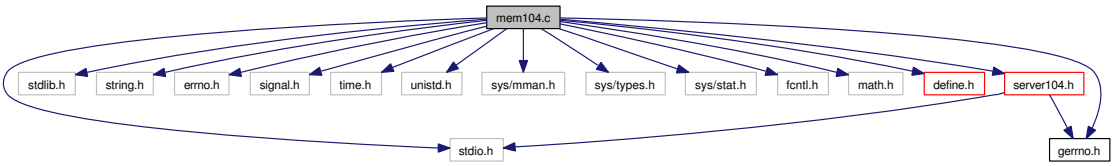
References port_access().

5.11 mem104.c File Reference

Routines for I/O access to ISA bus.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <time.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <math.h>
#include "define.h"
#include "server104.h"
#include "gerrno.h"
```

Include dependency graph for mem104.c:



Functions

- int map_isa_ram (void)
- int unmap_isa_ram (void)
- unsigned short _rmem104_word (int location)
- int _wmem104_word (int location, unsigned short value)
- unsigned short _rmem104_byte (int location)
- int _wmem104_byte (int location, unsigned short value)
- unsigned short rmem104 (int location)
- int wmem104 (int location, unsigned short value)
- int zero_seq_mem (int channel, va_list args)
- int analog_boardStatus (int channel)
- int analog_boardId (int channel)
- int test_read (int start, int iter)
- int test_write (int addr, int data)
- int test_board_id (int iter)
- int test_seq_mem (int channel, va_list args)

- int dump_seq_mem (void)
- int exercise_seq_mem (int channel, va_list args)
- int reset_logfifo (int channel)
- int read_log (int channel)
- void read_converters (int channel, int freq, int n_measure)
- int sequencer_status (int channel)
- int sequencer_running (int channel)
- int sequencer_counter (int channel)
- int start_sequencer (int channel)
- int stop_sequencer (int channel)
- int stop_CI_sequencer (int channel)
- int abort_sequencer (int channel)
- int start_broadcast (void)
- int stop_broadcast (void)
- int stop_CI_broadcast (void)
- int abort_broadcast (void)
- int stop_kindly (int channel)
- int stop_idle (int channel)
- int enable_sensor (int channel)
- int disable_sensor (int channel)
- int sensor_onoff (int channel)
- int whichfilter (int channel)
- int program_DAC_filter (int filtro)
- int link_status (int channel)
- int sawtooth_dac_test (int channel, int ndac)
- int program_DAC_Vlevel (int channel, va_list args)
- int execute_cmd (int(*func)(), int channel,...)
- int link_error (int channel)
- int treset_program (int channel)

Variables

- static unsigned short * isa_ram
- static int fd

5.11.1 Detailed Description

Routines for I/O access to ISA bus.

Definition in file mem104.c.

5.11.2 Function Documentation

5.11.2.1 int map_isa_ram (void)

Maps the ISA bus Ram window.

Returns:

MYFALSE if error.

Definition at line 94 of file mem104.c.

References fd, ISA_BASE, isa_ram, ISA_WINSIZE, MYERR, and MYTRUE.

Referenced by initdev().

5.11.2.2 int unmap_isa_ram (void)

Unmap the ISA bus Ram window.

Returns:

MYFALSE if error.

Definition at line 123 of file mem104.c.

References fd, isa_ram, ISA_WINSIZE, and MYTRUE.

Referenced by finish(), and reinitdev().

5.11.2.3 unsigned short _rmem104_word (int *location*)

Reads and returns the content of ISA BUS at address = location as a 16-bit value.

Parameters:

location the register offset address

Returns:

0-65535 on success, set Err104 if error.

See also:

Error104

Definition at line 147 of file mem104.c.

References isa_ram.

Referenced by rmem104().

5.11.2.4 int _wmem104_word (int *location*, unsigned short *value*)

Writes a 16-bit value to the location of ISA BUS at address = location

Parameters:

location the register offset address

value the 16-bit unsigned value to write

Returns:

MYTRUE on success, set Err104 if error.

See also:

Erro104

Definition at line 171 of file mem104.c.

References isa_ram, and MYTRUE.

Referenced by wmem104().

5.11.2.5 unsigned short _rmem104_byte (int *location*)

Reads and returns the content of ISA BUS at address = location.

It reads a byte each time.

Parameters:

location register offset address

Returns:

0-65535 on success, set Err104 if error.

See also:

Error104

Definition at line 197 of file mem104.c.

References isa_ram.

Referenced by rmem104().

5.11.2.6 int _wmem104_byte (int *location*, unsigned short *value*)

Writes value to the location of ISA BUS at address = location.

It writes one byte each time.

Parameters:

location register offset address

value unsigned short value to write

Returns:

MYTRUE on success, set Err104 if error.

See also:

Error104

Definition at line 228 of file mem104.c.

References isa_ram, and MYTRUE.

Referenced by wmem104().

5.11.2.7 unsigned short rmem104 (int *location*)

Reads a value from the specified location. Wrapper function for 8-bit and 16-bit low level routines.

Parameters:

location register offset address

Returns:

the read value.
If any error, it sets the global error flag Err104.

See also:

Error104

Definition at line 258 of file mem104.c.

References _rmem104_byte(), _rmem104_word(), BADADDRESS, Err104, errlog(), error_code(), GB_BADADDR, gb_errno, GB_ERROR, GB_NOMMAPMEM, ISA_BASE, isa_ram, ISA_WINSIZE, MYFALSE, NOERR, NOMEMORYMAPPED, and Opera.

Referenced by analog_boardId(), analog_boardStatus(), exercise_seq_mem(), fifo_overflow(), fifo_ready(), initdev(), link_error(), link_status(), read_converters(), read_image(), read_log(), read_quadrant(), read_testimage(), reprogram_vlevel(), sequencer_counter(), sequencer_status(), syntetize_multi(), syntetize_program(), test_board_id(), test_read(), test_seq_mem(), verify_sequencer(), whichfilter(), and wmem104().

5.11.2.8 int wmem104 (int *location*, unsigned short *value*)

Writes the value into the specified location.

Wrapper function for 8-bit and 16-bit low level routines.

Parameters:

location register offset address

value unsigned short value to write

Returns:

MYTRUE on success. Sets the global variable Err104 if error.

See also:

Error104

Definition at line 295 of file mem104.c.

References _wmem104_byte(), _wmem104_word(), BADADDRESS, Err104, errlog(), error_code(), GB_BADADDR, gb_errno, GB_ERROR, GB_NOMMAPMEM, iolog(), ISA_BASE, isa_ram, ISA_WINSIZE, MYERR, NOERR, NOMEMORYMAPPED, Opera, rmem104(), and Verbose.

Referenced by abort_broadcast(), abort_sequencer(), disable_sensor(), enable_sensor(), exercise_seq_mem(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), read_testimage(), reset_fifo(), reset_fpga(), reset_logfifo(), sawtooth_dac_test(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_sequencer(), syntetize_multi(), syntetize_program(), test_seq_mem(), test_write(), treset_program(), and zero_seq_mem().

5.11.2.9 int zero_seq_mem (int *channel*, va_list *args*)

Zero sequencer memory.

It works on only one board at a time.

It handles channel by means of wrapper

Parameters:

channel the analogic board number

args the va_list of arguments used by the function

Returns:

MYTRUE on success.

Definition at line 336 of file mem104.c.

References ADCRAMSIZE, analog_boardStatus(), Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_ERROR, GB_NOMMAPMEM, GB_WARNING, iolog(), isa_ram, l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_-RAM_SEQC, and WR_RAM_SEQD.

Referenced by MenuLoop(), packetdecode(), and program_sequencer().

5.11.2.10 int analog_boardStatus (int *channel*)

Reads the analog board status register and set the corresponding global variables.

It handles channel by means of wrapper.

Parameters:

channel the analogic board number to operate with

Returns:

MYTRUE on success, MYERR if the optical link is down or if errors in I/O memory operations.

Definition at line 416 of file mem104.c.

References Analog, Err104, ERR_ADCONV, ERR_ADDR, ERR_AMPL, ERR_AMPL_, ERR_OFFSET, ERR_VLEVEL, AnalogBoard::ErrADConv, AnalogBoard::ErrAddr, AnalogBoard::ErrAmplifier, AnalogBoard::ErrAmplifier_, errlog(), AnalogBoard::ErrOffset, error_code(), AnalogBoard::ErrVlevel, GB_EADCONV, GB_EDECODER, GB_ENOLINK, GB_ENOVCC, GB_ENVALBOARD, gb_errno, GB_ERROR, GB_EVLEVEL, GB_WARNING, link_error(), link_status(), MYERR, MYTRUE, NBOARD, NOERR, Opera, RD_BOARD, rmem104(), AnalogBoard::Sensore, AnalogBoard::Sequencer, SEQUENCER_RUN, VCC_ON, VCC_SENSOR, AnalogBoard::VccOn, Verbose, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_SEQD.

Referenced by disable_sensor(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), handle_-oneboard_acquisition(), initdev(), MenuLoop(), printstatus(), program_DAC_Vlevel(), sensor_onoff(), sys_running_status(), updatemenu(), and zero_seq_mem().

5.11.2.11 int analog_boardId (int *channel*)

Reads the analog board identification number ant stores it in the corresponding variable of the global structure.

It works on only one board at a time.

It handles channel by means of wrapper.

Parameters:

channel the analogic board number

Returns:

MYTRUE on success.

Definition at line 550 of file mem104.c.

References Analog, Err104, GB_ENVALBOARD, gb_errno, AnalogBoard::Id, MYERR, MYTRUE, NBOARD, NOERR, RD_ANALOGID, rmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_-SEQC, and WR_RAM_SEQD.

Referenced by initdev(), MenuLoop(), and printstatus().

5.11.2.12 int test_read (int *start*, int *iter*)

Tests reading a generic address.\ It reads at chunks of 8 addresses.

Parameters:

start the start address

iter the number of test ripetitions

Returns:

MYTRUE on success.

Definition at line 593 of file mem104.c.

References Err104, iolog(), ISA_BASE, Menu, MYERR, MYTRUE, NOERR, rmem104(), and Verbose.

Referenced by dump_seq_mem(), and MenuLoop().

5.11.2.13 int test_write (int *addr*, int *data*)

Tests writing to a generic address.

Parameters:

addr the address to write

data the data to write

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 637 of file mem104.c.

References iolog(), MYERR, MYTRUE, and wmem104().

Referenced by MenuLoop().

5.11.2.14 int test_board_id (int *iter*)

Tests reading board ID.

Reads iter times board ID and publish results.

Parameters:

iter the number of test ripetitions. If iter=0 continous.

Returns:

MYTRUE on success.

Definition at line 658 of file mem104.c.

References dlgetc(), dlmsg(), dlnomsg(), elapsed_time(), errlog(), error_code(), GB_EBUFFVERIFY, gb_errno, GB_WARNING, get_current_time(), iolog(), l_error(), l_function(), MYFALSE, MYTRUE, Opera, RD_IDENT, rmem104(), and VBuff_id.

Referenced by MenuLoop().

5.11.2.15 int test_seq_mem (int *channel*, va_list *args*)

Tests the sequencer memory RAM.

Parameters:

channel the analogic board number to operate with

args the va_list arguments

Returns:

MYTRUE on success.

Definition at line 715 of file mem104.c.

References ADCRAMSIZE, Analog, Channel, chkabort(), errlog(), GB_ENOLINK, GB_ENVALBOARD, gb_errno, GB_ERROR, GB_ESEQVERIFY, GB_NOMMAPMEM, iolog(), isa_ram, l_error(), l_function(), link_error(), link_status(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, rmem104(), Verbose, wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_SEQD.

Referenced by initdev(), MenuLoop(), and packetdecode().

5.11.2.16 int dump_seq_mem (void)

NON funziona con il wrapper perche' il caso Channel=5 non ha senso

Returns:

MYTRUE on success.

Definition at line 875 of file mem104.c.

References analog_boardStatus(), Channel, dlkey(), errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, iolog(), l_error(), Menu, MYERR, MYTRUE, Opera, test_read(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_SEQD.

Referenced by MenuLoop(), and packetdecode().

5.11.2.17 int exercise_seq_mem (int *channel*, va_list *args*)

Execises sequencer memory.

It handles channel by means of wrapper.

Parameters:

channel the analogic board number to operate with

args the va_list arguments

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 941 of file mem104.c.

References ADCRAMSIZE, analog_boardStatus(), Channel, dlgetc(), errlog(), error_code(), GB_BADADDR, gb_errno, GB_ESEQVERIFY, GB_WARNING, iolog(), ISA_BASE, l_error(), l_function(), link_error(), MYERR, MYTRUE, Opera, RD_IDENT, rmem104(), wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_SEQD.

Referenced by MenuLoop().

5.11.2.18 int reset_logfifo (int *channel*)

Resets the log fifos.

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1063 of file mem104.c.

References BRD_ADDR_STEP, wmem104(), WR_RAM_SEQ, and WR_RST_LOG.

Referenced by read_log().

5.11.2.19 int read_log (int *channel*)

Reads the alimentation log for the selected board.

It handles channel by means of wrapper.

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1083 of file mem104.c.

References ALIMEN_DEPTH, Analog, BRD_ADDR_STEP, elapsed_time(), Err104, errlog(), error_code(), execute_cmd(), GB_ENOLINK, gb_errno, GB_ERROR, GB_ETIMEOUT, get_current_time(), iolog(), ISA_BASE, l_error(), link_status(), MYERR, MYTRUE, NOERR, Opera, print_alimentation(), RD_LOG_FIFO, RD_LOG_STAT, RD_RAM_SEQ, reset_logfifo(), rmem104(), TIMEOUT_READ, and Verbose.

Referenced by MenuLoop(), and packetdecode().

5.11.2.20 void read_converters (int *channel*, int *freq*, int *n_measure*)

It reads the converter in timely manner and write results in a file.

It DO NOT handles channel by means of wrapper

Parameters:

channel the board number to operate with

freq the period time to acquire data

n_measure the number of measures to repeat

Returns:

no value.

Definition at line 1213 of file mem104.c.

References BRD_ADDR_STEP, dlgetc(), elapsed_time(), Err104, errlog(), error_code(), GB_EFILEOPEN, GB_EMEMALLOC, GB_ENVALBOARD, gb_errno, GB_ERROR, GB_WARNING, get_current_time(), iolog(), l_error(), l_function(), NOERR, Opera, RD_NBCONV, RD_RAM_SEQ, rmem104(), and strdup_printf().

Referenced by MenuLoop().

5.11.2.21 int sequencer_status (int *channel*)

Read the sequencer status corresponding to the board passed as argument. It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1348 of file mem104.c.

References Analog, BRD_ADDR_STEP, Err104, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, MYERR, MYTRUE, NBOARD, NOERR, Opera, RD_BOARD, rmem104(), AnalogBoard::Sequencer, SEQUENCER_RUN, and WR_RAM_SEQ.

Referenced by abort_broadcast(), abort_sequencer(), sequencer_running(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_kindly(), and stop_sequencer().

5.11.2.22 int sequencer_running (int *channel*)

Gets the running status of the sequncer.

Parameters:

channel the channel to work with

Returns:

MYTRUE if the sequencer is running, MYFALSE if the sequencer keeps still, MYERR on errors

Definition at line 1383 of file mem104.c.

References Analog, execute_cmd(), MYFALSE, MYTRUE, and sequencer_status().

Referenced by stop_idle(), and stop_kindly().

5.11.2.23 int sequencer_counter (int *channel*)

Read the sequencer program counter of the board passed as argument.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYERR on error else the program counter

Definition at line 1428 of file mem104.c.

References BRD_ADDR_STEP, Err104, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, MYERR, NBOARD, NOERR, Opera, RD_PCOUNTER, rmem104(), and WR_RAM_SEQ.

Referenced by handle_multi_acquisition().

5.11.2.24 int start_sequencer (int *channel*)

Start the sequencer on the board passed as argument.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYFALSE if the sequencer is still, MYERR on errors.

Definition at line 1462 of file mem104.c.

References Analog, Channel, enable_sensor(), errlog(), error_code(), GB_ENOSENSOR, GB_-ENVALBOARD, gb_errno, GB_ERROR, GB_ESEQNOTOP, GB_WARNING, iolog(), l_function(), MY-ERR, MYFALSE, MYTRUE, NBOARD, Opera, sequencer_status(), SVBTest, wmem104(), WR_-RAM_SEQ, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and WR_-SEQSTART.

Referenced by dummy_acquisition(), handle_oneboard_acquisition(), MenuLoop(), and packetdecode().

5.11.2.25 int stop_sequencer (int *channel*)

Stop the sequencer on the board passed as argument.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1553 of file mem104.c.

References Analog, Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_ERROR, GB_-ESEQNOTOP, GB_WARNING, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, sequencer_status(), wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_-RAM_SEQD, and WR_SEQSTOP.

Referenced by handle_oneboard_acquisition(), and MenuLoop().

5.11.2.26 int stop_CI_sequencer (int *channel*)

Stop the sequencer at the current istruction on the board passed as argument.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1617 of file mem104.c.

References Analog, Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, sequencer_status(), wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and WR_-STOP_CI.

Referenced by stop_kindly().

5.11.2.27 int abort_sequencer (int *channel*)

Abort the sequencer on the board passed as argument.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYFALSE if the sequencer is already running, MYERR on errors.

Definition at line 1679 of file mem104.c.

References Analog, Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_ERROR, GB_-ESEQNOTOP, GB_WARNING, IDLE_STOP, IdleStatus, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, sequencer_status(), wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and WR_SEQABORT.

Referenced by handle_oneboard_acquisition(), MenuLoop(), and stop_idle().

5.11.2.28 int start_broadcast (void)

Start the all 4 channel of sequencer

If NOBROADCAST is defined, It can handle channel by means of wrapper.

Returns:

MYTRUE on success, MYFALSE if the sequencer is still, MYERR on errors.

Definition at line 1742 of file mem104.c.

References Analog, errlog(), error_code(), gb_errno, GB_ERROR, GB_ESEQNOTOP, iolog(), l_-function(), MYERR, MYFALSE, MYTRUE, Opera, sequencer_status(), SVBTest, wmem104(), WR_-BROADCAST, WR_RAM_SEQ, and WR_SEQSTART.

Referenced by handle_image_acquisition(), and handle_multi_acquisition().

5.11.2.29 int stop_broadcast (void)

Stop all the channels of sequencer at the end of waveform.

Returns:

MYTRUE on success, MYERR on error.

Definition at line 1792 of file mem104.c.

References Analog, errlog(), error_code(), gb_errno, GB_ERROR, GB_ESEQNOTOP, iolog(), MYERR, MYFALSE, MYTRUE, Opera, sequencer_status(), wmem104(), WR_BROADCAST, WR_RAM_SEQA, and WR_SEQSTOP.

Referenced by handle_image_acquisition(), and handle_multi_acquisition().

5.11.2.30 int stop_CI_broadcast (void)

Stop all the channels of sequencer at the current istruction.

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 1833 of file mem104.c.

References Analog, errlog(), error_code(), gb_errno, GB_ERROR, iolog(), MYERR, MYFALSE, MYTRUE, Opera, sequencer_status(), wmem104(), WR_BROADCAST, WR_RAM_SEQA, and WR_-STOP_CI.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), and stop_kindly().

5.11.2.31 int abort_broadcast (void)

Stop the all 4 board of sequencer at once

Returns:

MYTRUE on success, MYFALSE if the sequencer is still, MYERR on errors.

Definition at line 1872 of file mem104.c.

References Analog, errlog(), error_code(), gb_errno, GB_ERROR, GB_ESEQNOTOP, IDLE_STOP, IdleStatus, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, Opera, sequencer_status(), wmem104(), WR_BROADCAST, WR_RAM_SEQA, and WR_SEQABORT.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), and stop_idle().

5.11.2.32 int stop_kindly (int *channel*)

Stop the sequencer on the board passed as argument

It in intended to stop IDLE measures only!! It handles channel by itself for efficiency

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE if the sequencer is still, MYFALSE if the sequencer is runnning, MYERR on errors.

Definition at line 1926 of file mem104.c.

References Channel, errlog(), error_code(), execute_cmd(), GB_ENVALBOARD, gb_errno, GB_ERROR, GB_ESEQNOTOP, GB_WARNING, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, sequencer_running(), sequencer_status(), stop_CI_broadcast(), and stop_CI_sequencer().

Referenced by stop_idle().

5.11.2.33 int stop_idle (int *channel*)

Stop the idle scan on the board passed as argument

This function distinguish between laboratory and normal operation mode. In first case, sequencer running is aborted abruptly, while in the second case the function calls the stop_kindly() routine to stop sequencer properly, at the end of the current istruction execution.

Parameters:

channel the active channel (5 if all boards active)

Returns:

MYTRUE on success, MYFALSE if the sequencer is running, MYERR on error(s)

Definition at line 1999 of file mem104.c.

References abort_broadcast(), abort_sequencer(), Channel, ExpertMode, IDLE_STOP, IdleStatus, iolog(), MYERR, MYFALSE, MYTRUE, sequencer_running(), and stop_kindly().

Referenced by dummy_acquisition(), handle_image_acquisition(), handle_multi_acquisition(), handle_-oneboard_acquisition(), packetdecode(), and program_sequencer().

5.11.2.34 int enable_sensor (int *channel*)

Enables the sensor on the selected board.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success. MYFALSE if sensor is off, MYERR if errors.

Definition at line 2046 of file mem104.c.

References Analog, analog_boardStatus(), Channel, errlog(), error_code(), GB_ENVALBOARD, gb_-errno, GB_ERROR, GB_ESENSNOTOP, GB_WARNING, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, wmem104(), WR_RAM_SEQ, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and WR_SENSON.

Referenced by dummy_acquisition(), handle_image_acquisition(), handle_multi_acquisition(), handle_-oneboard_acquisition(), MenuLoop(), packetdecode(), sawtooth_dac_test(), sensor_onoff(), start_-sequencer(), and write_pk_param().

5.11.2.35 int disable_sensor (int *channel*)

Disables the sensor on the selected board.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYFALSE if it is still on, MYERR on errors.

Definition at line 2114 of file mem104.c.

References Analog, analog_boardStatus(), Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_ERROR, GB_ESENSNOTOP, GB_WARNING, iolog(), l_function(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and WR_SENSOFF.

Referenced by MenuLoop(), packetdecode(), sensor_onoff(), and write_pk_param().

5.11.2.36 int sensor_onoff (int *channel*)

It switches the board sensor on or off depending of the sensor flag value.

It handles channel by means of wrapper.

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYFALSE if the operation fails, MYERR on errors.

Definition at line 2181 of file mem104.c.

References Analog, analog_boardStatus(), disable_sensor(), enable_sensor(), errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, MYERR, MYTRUE, NBOARD, and Opera.

Referenced by MenuLoop().

5.11.2.37 int whichfilter (int *channel*)

Reads the board status register (bit 15) to determine which filter is active.

It handles channel by means of wrapper.

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYERR on errors

Definition at line 2215 of file mem104.c.

References Analog, BRD_ADDR_STEP, Err104, errlog(), error_code(), FILTER_ACTIVE, AnalogBoard::Filtro, GB_ENOLINK, GB_ENVALBOARD, gb_errno, GB_ERROR, GB_WARNING, iolog(), link_status(), MYERR, MYFALSE, MYTRUE, NBOARD, NOERR, Opera, RD_BOARD, rmem104(), Verbose, and WR_RAM_SEQ.

Referenced by initdev(), program_DAC_filter(), and updatemenu().

5.11.2.38 int program_DAC_filter (int *filtro*)

Programs the selected filter (1 or 2) for ALL the boards.

It handles ALWAYS ALL channels

Parameters:

filtro the filter to program

Returns:

MYTRUE on success, MYFALSE if programming operation fails, MYERR on errors

Definition at line 2271 of file mem104.c.

References Analog, BRD_ADDR_STEP, Channel, errlog(), error_code(), execute_cmd(), GB_ENOLINK, gb_errno, GB_ERROR, link_status(), MYERR, MYTRUE, Opera, REG_OFFSET, whichfilter(), wmem104(), WR_FILTER1, and WR_RAM_SEQ.

Referenced by MenuLoop(), and write_pk_param().

5.11.2.39 int link_status (int *channel*)

Gets the optical link status of the selected board.

It handles channel by means of wrapper. RD_FIFO_STATUS register:

bit 5: 1 -> optical link board A ok

bit 6: 1 -> optical link board B ok

bit 7: 1 -> optical link board C ok

bit 8: 1 -> optical link board D ok

Get the active bit of the optical link status for the selected board and set the corresponding bit into the link flag if the optical link is active.

```
* ----------------------------------------
* board        mask            Link flag
* ----------------------------------------
*   A          0x0010          0x0001
*   B          0x0020          0x0002
*   C          0x0040          0x0004
*   D          0x0080          0x0008
*
```

Parameters:

channel the board to operate on

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 2335 of file mem104.c.

References Analog, Err104, errlog(), error_code(), GB_ELINKVB, gb_errno, GB_ETRLINKVB, GB_-WARNING, iolog(), AnalogBoard::Link, LINK_STATUS, LINK_VB_ERR, LINK_VB_TRERR, MY-ERR, MYTRUE, NOERR, Opera, RD_FIFO_STAT, rmem104(), and Verbose.

Referenced by analog_boardStatus(), initdev(), link_error(), MenuLoop(), program_DAC_filter(), read_-log(), test_seq_mem(), updatemenu(), and whichfilter().

5.11.2.40 int sawtooth_dac_test (int *channel*, int *ndac*)

Outputs a triangular wave on one of the four DAC on the ∗ selected board.

It DO NOT handles channel by means of wrapper.

Parameters:

channel the analogic board number to operate with

ndac the DAC to program (Vreset, Vbias, Offset12, Offset34)

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 2423 of file mem104.c.

References BRD_ADDR_STEP, dlgetc(), enable_sensor(), errlog(), error_code(), GB_EINTERNAL, GB_ENVALBOARD, gb_errno, GB_WARNING, iolog(), ISA_BASE, l_function(), MYERR, MYTRUE, Opera, TEST_TASK, wmem104(), WR_OFF1_2, WR_OFF3_4, WR_RAM_SEQ, WR_VBIAS, and WR_VRESET.

Referenced by MenuLoop().

5.11.2.41 int program_DAC_Vlevel (int *channel*, va_list *args*)

Programs the value of V_reset, V_bias, Offset12 and Offset34.

It handles channel by means of wrapper.

Parameters:

channel the analogic board number

args the va_list of arguments

Returns:

MYTRUE on success, MYFALSE if programming operation fails, MYERR on errors.

Definition at line 2496 of file mem104.c.

References Analog, analog_boardStatus(), Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, iolog(), MYERR, MYFALSE, MYTRUE, NBOARD, AnalogBoard::Offset12, AnalogBoard::Offset34, Opera, AnalogBoard::V_bias, AnalogBoard::V_reset, Verbose, wmem104(), WR_OFF1_2, WR_OFF3_4, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, WR_VBIAS, and WR_VRESET.

Referenced by dlanalog(), initdev(), MenuLoop(), reprogram_vlevel(), and write_pk_param().

5.11.2.42 int execute_cmd (int(∗)() *func*, int *channel*, ...)

Wrapper function for analog board routines.

Parameters:

func pointer to the handler function

channel the analogic board number

Returns:

MYTRUE on success.

Definition at line 2593 of file mem104.c.

References MYTRUE, and NBOARD.

Referenced by dlanalog(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initdev(), MenuLoop(), packetdecode(), printstatus(), program_DAC_filter(), program_sequencer(), random_action(), read_log(), reprogram_vlevel(), sequencer_running(), stop_kindly(), updatemenu(), and write_pk_param().

5.11.2.43 int link_error (int *channel*)

A more accurate control on link errors.

Parameters:

channel the analogic board number to operate with

Returns:

MYTRUE on success, MYERR on errors.

Definition at line 2635 of file mem104.c.

References Analog, Err104, errlog(), error_code(), GB_ELINK, GB_ELINKVB, GB_ENOLINK, GB_ERRMASK, gb_errno, GB_ERROR, GB_ETRLINKVB, GB_WARNING, iolog(), LINK_ERROR, LINK_ERROR_RATE, link_status(), MYERR, MYTRUE, NOERR, Opera, RD_ANALOGID, rmem104(), Verbose, WR_RAM_SEQ, WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, and WR_RAM_SEQD.

Referenced by analog_boardStatus(), exercise_seq_mem(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), read_image(), and test_seq_mem().

5.11.2.44 int treset_program (int *channel*)

Program the treset field of Analog structure in the WR_RESCLK register of the appropriate board.

It handles channel by means of wrapper

Parameters:

channel the analogic board to operate with

Returns:

MYTRUE on success, MYFALSE if the sequencer is already running, MYERR on errors.

Definition at line 2720 of file mem104.c.

References Analog, Channel, errlog(), error_code(), GB_ENVALBOARD, gb_errno, GB_WARNING, iolog(), MYERR, MYFALSE, MYTRUE, NBOARD, Opera, wmem104(), WR_RAM_SEQA, WR_RAM_SEQB, WR_RAM_SEQC, WR_RAM_SEQD, and WR_RESCLK.

Referenced by dummy_acquisition(), handle_image_acquisition(), handle_multi_acquisition(), and write_pk_param().

5.11.3 Variable Documentation

5.11.3.1 unsigned short* isa_ram [static]

pointer to ISA bus Ram window

Definition at line 80 of file mem104.c.

Referenced by _rmem104_byte(), _rmem104_word(), _wmem104_byte(), _wmem104_word(), map_isa_ram(), rmem104(), test_seq_mem(), unmap_isa_ram(), wmem104(), and zero_seq_mem().

5.11.3.2 int fd [static]

handler to /dev/mem

Definition at line 81 of file mem104.c.

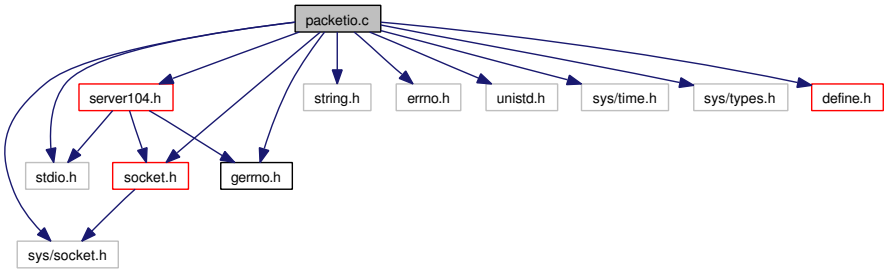
Referenced by compare_socket(), map_isa_ram(), open_serial(), remove_event_sourcetype(), and unmap_isa_ram().

5.12 packetio.c File Reference

Packet related routines, from gbridge.

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include "define.h"
#include "socket.h"
#include "server104.h"
#include "gerrno.h"
```

Include dependency graph for packetio.c:



Functions

- unsigned short calculate_checksum (unsigned short *header)
- static int checksum_error (int sockfd)
- int packetread (int sockfd, struct GPacket *packet)
- int _packetsend (int sockfd, int destination, int cmd, int type, int datalen, unsigned short seqnum, char *payload)
- int packetsend_ack (int sockfd, int destination, int cmd, int datalen, unsigned short seqnum, char *payload)

5.12.1 Detailed Description

Packet related routines, from gbridge.

Definition in file packetio.c.

5.12.2 Function Documentation

5.12.2.1 unsigned short calculate_checksum (unsigned short * *header*)

Compute the checksum of packet header

Parameters:

header the hedader of the packet

Returns:

the packet header checksum.

Definition at line 52 of file packetio.c.

References HDR_NUM.

Referenced by _packetsend(), and packetread().

5.12.2.2 static int checksum_error (int *sockfd*) [static]

This function handle the checksum packet error case.

Read a fixed amount of data from the buffer without removing it from the socket queue. It then looks for the MAGIC MASK packet value to individuate ,if exists, the next valid packet.

Parameters:

sockfd The socket handle of command packet

Returns:

> 0 on success, -1 on errors, 0 on timeout or if valid header not found.

Definition at line 79 of file packetio.c.

References close_socket(), elapsed_time(), gb_errno, GB_IO_EOF, get_current_time(), iolog(), PCK_-SIZE, and SOCK_TIMEOUT.

Referenced by packetread().

5.12.2.3 int packetread (int *sockfd*, struct GPacket * *packet*)

This function read the data inside the packet.

Each packet is composed of and header of 8 words at 16-bits and (eventually) a data area.

Parameters:

sockfd The socket handle of command packet

packet The packet structure

Returns:

the number of bytes read on success, < 0 on error, 0 on timeout.

Definition at line 159 of file packetio.c.

References calculate_checksum(), checksum_error(), close_socket(), GB_CHKSUM_ERR, gb_errno, HDR_SIZE, GPacket::header, iolog(), GPacket::payload, and sock_read().

Referenced by MainLoop(), and packetread_gbridge().

5.12.2.4 int _packetsend (int *sockfd*, int *destination*, int *cmd*, int *type*, int *datalen*, unsigned short *seqnum*, char * *payload*)

Send a structured packet to a socket.

If sending fails because of an error condition or EOF on socket, the socket is closed.

Parameters:

sockfd The socket handle of command packet

destination the target process of the paket

cmd the command field of the packet

type the type field of the packet

datalen the len of payload of the packet

seqnum the seqnum field of the packet

payload the payload of the packet

Returns:

$>$ 0 on success, -1 on errors, 0 on timeout.

Definition at line 231 of file packetio.c.

References ACK, calculate_checksum(), close_socket(), HDR_NUM, HDR_SIZE, iolog(), MAGIC-MASK, Pck_seqnum, PCK_SIZE, and sock_write().

Referenced by packetsend(), and packetsend_ack().

5.12.2.5 int packetsend_ack (int *sockfd*, int *destination*, int *cmd*, int *datalen*, unsigned short *seqnum*, char * *payload*)

Send the acknowledge of a packet to the socket.

If sending fails because of an error condition or EOF on socket, the socket is closed.

Parameters:

sockfd The socket handle of command packet

destination the target process of the paket

cmd the command field of the packet

datalen the len of payload of the packet

seqnum the seqnum field of the packet

payload the payload of the packet

Returns:

$>$ 0 on success, -1 on errors, 0 on timeout.

Definition at line 311 of file packetio.c.

References _packetsend(), and ACK.

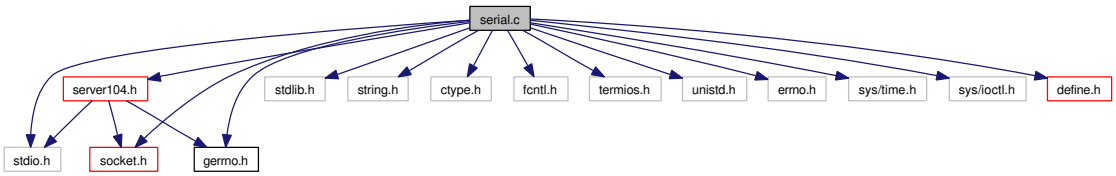
Referenced by packetack().

5.13 serial.c File Reference

Serial handling for the HHameg Power supply.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <errno.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include "socket.h"
#include "define.h"
#include "gerrno.h"
#include "server104.h"
```

Include dependency graph for serial.c:



Defines

- #define BAUDRATE B9600
- #define CR 13

Functions

- void close_serial (int fd)
- void print_named_ascii (size_t n_bytes, char *block)
- void flush_serial (int fd, double timewait)
- int open_serial (char serial[ ])
- int serial_wrline (int fd, char *wbuf)
- int serial_rdline (int fd, char *buf)
- int serial_decode (char *buf, int read_flag, int fd)
- int serial_command (int fd, char *buf)
- int init_hameg (void)
- int off_power (void)

- int read_power (void)
- int init_power (void)

Variables

- struct termios oldtio
- static int Hameg_fd
- static const char *const charname [33]

5.13.1 Detailed Description

Serial handling for the HHameg Power supply.

Definition in file serial.c.

5.13.2 Define Documentation

5.13.2.1 #define BAUDRATE B9600

the speed of our serial line

Definition at line 74 of file serial.c.

Referenced by open_serial().

5.13.2.2 #define CR 13

carriage return

Definition at line 76 of file serial.c.

Referenced by serial_rdline(), and serial_wrline().

5.13.3 Function Documentation

5.13.3.1 void close_serial (int *fd*)

Closes the serial port.

Parameters:

fd the serial file handler

Definition at line 103 of file serial.c.

References oldtio.

Referenced by init_power(), off_power(), open_serial(), and read_power().

5.13.3.2 void print_named_ascii (size_t *n_bytes*, char * *block*)

From GNU util od. Dump ascii strings as named chars

Returns:

no value

Definition at line 120 of file serial.c.

References charname.

5.13.3.3 void flush_serial (int *fd*, double *timewait*)

Flushes the serial port

Parameters:

fd the serial port file handler
timewait the time interval to wait for flush

Definition at line 153 of file serial.c.

References elapsed_time(), get_current_time(), iolog(), sock_read(), and Verbose.

Referenced by init_hameg(), off_power(), and read_power().

5.13.3.4 int open_serial (char *serial*[ ])

Opens the serial device.

Parameters:

serial the serial port name (COM1, COM2, etc.)

Returns:

the serial file handler if ok, < 0 if errors.

Definition at line 215 of file serial.c.

References BAUDRATE, close_serial(), errlog(), error_code(), fd, gb_errno, GB_WARNING, iolog(), oldtio, and PROGRAM_TASK.

Referenced by init_hameg(), off_power(), and read_power().

5.13.3.5 int serial_wrline (int *fd*, char * *wbuf*)

Writes a ascii command to serial port.

Parameters:

fd the serial file handler
wbuf the string to write

Returns:

0 on success, < 0 on errors.

Definition at line 274 of file serial.c.

References CR, and sock_write().

Referenced by init_hameg(), off_power(), read_power(), and serial_command().

5.13.3.6 int serial_rdline (int *fd*, char * *buf*)

Reads from serial port a line at time.

Parameters:

fd the serial file descriptor
buf the buffer to store read data

Returns:

the number of bytes read on success, 0 on timeout, < 0 on errors.

Definition at line 314 of file serial.c.

References CR, elapsed_time(), gb_errno, GB_IO_TIME_READ, get_current_time(), and sock_read().

Referenced by serial_command(), and serial_decode().

5.13.3.7 int serial_decode (char * *buf*, int *read_flag*, int *fd*)

Decodes the string received from the device

Parameters:

buf the device answer
read_flag flag of multi line answer (3)
fd the serial file descriptor

Returns:

0 on valid command else < 0

Definition at line 364 of file serial.c.

References errlog(), error_code(), GB_CMD_NOTACK, gb_errno, GB_WARNING, iolog(), MYFALSE, MYTRUE, PROGRAM_TASK, PW_amp, PW_stat, PW_volt, serial_rdline(), and Verbose.

Referenced by serial_command().

5.13.3.8 int serial_command (int *fd*, char * *buf*)

Send a command and sense if we are in error

Parameters:

fd the serial file descriptor
buf the command to send

Returns:

0 on valid command else < 0

Definition at line 425 of file serial.c.

References errlog(), error_code(), gb_errno, GB_WARNING, iolog(), MYFALSE, MYTRUE, PROGRAM_TASK, serial_decode(), serial_rdline(), serial_wrline(), and Verbose.

Referenced by init_hameg(), off_power(), and read_power().

5.13.3.9 int init_hameg (void)

Initialize the Power contrller and DO NOT close serial channel

Returns:

the channel ID on valid on success else < 0

Definition at line 488 of file serial.c.

References errlog(), error_code(), flush_serial(), gb_errno, GB_WARNING, Hameg_fd, HAMEG_SERIAL, iolog(), open_serial(), PROGRAM_TASK, serial_command(), and serial_wrline().

Referenced by init_power().

5.13.3.10 int off_power (void)

disable the Hameg power control

Returns:

MYTRUE on success

Definition at line 550 of file serial.c.

References close_serial(), errlog(), error_code(), flush_serial(), gb_errno, GB_WARNING, Hameg_fd, HAMEG_SERIAL, MYFALSE, MYTRUE, open_serial(), PROGRAM_TASK, serial_command(), and serial_wrline().

Referenced by finish(), and reinitdev().

5.13.3.11 int read_power (void)

read the Power controller voltage/current values in the global arrays PW_volt and PW_amp and CLOSE serial channel

Returns:

MYTRUE on success

Definition at line 590 of file serial.c.

References close_serial(), errlog(), error_code(), flush_serial(), gb_errno, GB_WARNING, Hameg_fd, HAMEG_SERIAL, MYFALSE, MYTRUE, open_serial(), PROGRAM_TASK, serial_command(), and serial_wrline().

Referenced by init_power(), and MainLoop().

5.13.3.12 int init_power (void)

Initialize the Power contrller and CLOSE serial channel

Returns:

MYTRUE on success

Definition at line 625 of file serial.c.

References close_serial(), errlog(), error_code(), GB_EHAMEG, gb_errno, GB_WARNING, Hameg_fd, init_hameg(), MYFALSE, MYTRUE, PROGRAM_TASK, PW_stat, and read_power().

Referenced by initdev(), and MainLoop().

5.13.4 Variable Documentation

5.13.4.1 struct termios oldtio

Definition at line 79 of file serial.c.

Referenced by close_serial(), and open_serial().

5.13.4.2 int Hameg_fd [static]

to hold the serial channel ID

Definition at line 81 of file serial.c.

Referenced by init_hameg(), init_power(), off_power(), and read_power().

5.13.4.3 const char* const charname[33] [static]

Initial value:

```
{
  "nul", "soh", "stx", "etx", "eot", "enq", "ack", "bel",
  "bs", "ht", "nl", "vt", "ff", "cr", "so", "si",
  "dle", "dc1", "dc2", "dc3", "dc4", "nak", "syn", "etb",
  "can", "em", "sub", "esc", "fs", "gs", "rs", "us",
  "sp"
}
```

list to dump communication in print_named_ascii

Definition at line 85 of file serial.c.

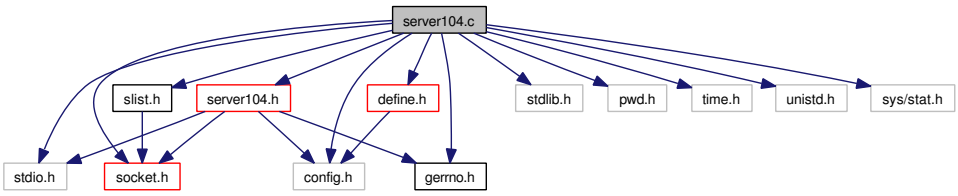
Referenced by print_named_ascii().

5.14 server104.c File Reference

Main of the giano control program.

```
#include <stdio.h>
#include <stdlib.h>
#include <pwd.h>
#include <time.h>
#include <unistd.h>
#include <sys/stat.h>
#include "socket.h"
#include "slist.h"
#include "define.h"
#include "server104.h"
#include "gerrno.h"
#include "config.h"
```

Include dependency graph for server104.c:



Functions

- void print_help (void)
- void setowner (void)
- void initvar104 (void)
- int main (int argc, char *argv[ ])

Variables

- char * optarg
- int optind
- int opterr
- int optopt
- char * mempool = NULL
- uid_t uid
- uid_t gid

5.14.1 Detailed Description

Main of the giano control program.

Definition in file server104.c.

5.14.2 Function Documentation

5.14.2.1 void print_help (void)

Print help for command line switches.

example of output:

```
   server104 Version 2.50232
   Compiled on Sep 30 2010 12:48:57
   Usage: server104  [-options]
   Options:
   -h          Output this help
   -v <level>  verbosity level (Default 1)
   -x          daemon mode
   -m          menu mode
   -b <n>      board number (1-4, 5 == all)
   -l          LCD not present
   -t          test mode
 *
```

Definition at line 95 of file server104.c.

References G_REVISION.

Referenced by main().

5.14.2.2 void setowner (void)

Sets the owner of the generated fits files.

Returns:

no value

Note:

This function is active only in menu (debug) mode.

Definition at line 122 of file server104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, gid, iolog(), Opera, and uid.

Referenced by main().

5.14.2.3 void initvar104 (void)

Initializes global variables to a sensible default value

STD_INI_FILE or ALT_INI_FILE now are handled in read_init_file

Returns:

no value

Definition at line 159 of file server104.c.

References Acq_type, Acquired, Analog, array, array_size, Base_scan, Channel, compute_cycle(), compute_scan_time(), current_group, D_lsync, Dit, ExpertMode, Extra_sub, file_n, AnalogBoard::Filtro, GB_EINTERNAL, gb_errno, gid, H_COL, H_ROW, AnalogBoard::Id, IDLE_STOP, IdleStatus, INTERNAL_TASK, iolog(), Keepgoing, Lcd_present, AnalogBoard::Link, Logfile, logmsg, logname, Menu, multidummy(), multi_status::multiprogram, multistatus, MYFALSE, MYTRUE, NBOARD, Ncol, Ngroup, NoiseMode, NOP_IDLE, nopStatus, Nrow, AnalogBoard::Offset12, AnalogBoard::Offset34, Opera, Pck_seqnum, AnalogBoard::Prog_dit, AnalogBoard::Prog_fsync, AnalogBoard::Prog_lsync, AnalogBoard::Prog_readclk, AnalogBoard::Prog_readdel, AnalogBoard::Prog_resclk, AnalogBoard::Prog_resnum, R_endframe, R_fsync_lsync, R_lsync_vclk, Run, SAS_IDLE, Scan_time, Sendmsg, AnalogBoard::Sensore, AnalogBoard::Sequencer, slist, ST_IDLE, AnalogBoard::Status, SVBCheck, SVBTest, SynchroTest, Testmode, Tint, uid, AnalogBoard::V_bias, AnalogBoard::V_reset, VBuff_id, Verbose, and Windowed.

Referenced by main().

5.14.2.4 int main (int *argc*, char * *argv*[ ])

Main routine.

Main handles command line switch, init the global variables and the hardware. Then it spawn the MenuLoop rountine, in interactive mode or the MainLoop routine in daemon mode. On exit call finish to have a clean exit.

Parameters:

argc The number of command line argouments

argv The command line argouments

Returns:

0 on regular exit, < 0 on bad error conditions

Definition at line 264 of file server104.c.

References catch_signals(), Channel, daemon_is_already_running(), errlog(), error_code(), finish(), gb_errno, GB_ERROR, H_COL, ignore_signals(), initdev(), INITDEV_TASK, initvar104(), initwin(), iolog(), Lcd_present, Logfile, logname, MainLoop(), mempool_alloc(), Menu, MenuLoop(), MYFALSE, MYTRUE, N_COL, Ncol, Nrow, open_log_file(), Opera, optarg, print_help(), read_init_file(), setowner(), syslog_message(), Testmode, and Verbose.

5.14.3 Variable Documentation

5.14.3.1 char* optarg

to access results of getopt function

Referenced by main().

5.14.3.2 int optind

5.14.3.3 int opterr

5.14.3.4 int optopt

to access results of getopt function

5.14.3.5 char* mempool = NULL

memory for linked lists

Definition at line 64 of file server104.c.

Referenced by accept_connection(), finish(), mempool_alloc(), and mempool_index().

5.14.3.6 uid_t uid

Definition at line 68 of file server104.c.

Referenced by initdev(), initvar104(), save_data(), and setowner().

5.14.3.7 uid_t gid

Definition at line 69 of file server104.c.

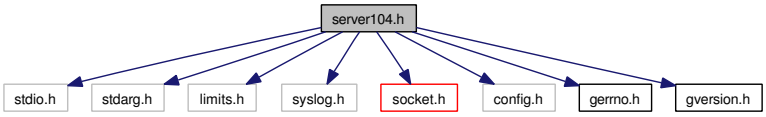
Referenced by initdev(), initvar104(), save_data(), and setowner().

5.15 server104.h File Reference

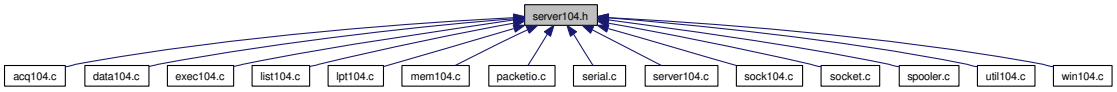
Global variables and functions declarations.

```
#include <stdio.h>
#include <stdarg.h>
#include <limits.h>
#include <syslog.h>
#include "socket.h"
#include "config.h"
#include "gerrno.h"
#include "gversion.h"
```

Include dependency graph for server104.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct AnalogBoard

  Structure with all analog electronics parameters.

- struct multiple

  multiple read sigle operation definition modification as discussed with E.Giani 8/2009

- struct multi_status

  multiple read operation structure definition

Variables

- int Verbose
- int Keepgoing
- int Menu
- int Windowed
- int Logfile
- int Sendmsg

- int Err104
- int VBuff_id
- int Acq_type
- int Extra_sub
- int Lcd_present
- int Opera
- int NoiseMode
- int SynchroTest
- int SVBTest
- int SVBCheck
- int gb_errno
- int Channel
- FILE * logmsg
- char * logname
- char * mempool
- int Acquired
- int Ncol
- int Nrow
- int Run
- unsigned short * array
- int array_size
- int Tint
- int Dit
- int Base_scan
- int R_endframe
- int R_fsync_lsync
- int D_lsync
- int R_lsync_vclk
- double Scan_time
- int Read_image_timeout
- int Ngroup
- int Curframe
- int Curgroup
- int Testmode
- int current_group
- time_t Prog_age
- time_t Logfile_age
- int file_n
- int Pck_seqnum
- int nopStatus
- double timestamp
- int serial_fd
- int IdleStatus
- int ExpertMode
- int Cycle_status
- double Cycle_time
- double Transfer_time
- struct AnalogBoard Analog [4]
- double PW_volt [4]
- double PW_amp [4]

- int PW_stat [4]
- struct multi_status multistatus
- uid_t uid
- gid_t gid
- sock_t giolamp

5.15.1 Detailed Description

Global variables and functions declarations.

Definition in file server104.h.

5.15.2 Variable Documentation

5.15.2.1 int Verbose

verbosity level 0->3

Definition at line 110 of file server104.h.

Referenced by add_f_entry(), add_s_entry(), analog_boardStatus(), check_data(), daemon_is_already_running(), dump_data(), f_sched(), flush_serial(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), initvar104(), iolog(), isfull_fifo(), isvoid_fifo(), l_get_port(), l_init_port(), l_output(), l_position(), l_senddata(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), main(), MenuLoop(), packetdecode(), printstatus(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), read_image(), read_init_file(), read_log(), read_quadrant(), read_testimage(), reprogram_vlevel(), reset_fifo(), reset_fpga(), s_sched(), s_wrapper(), serial_command(), serial_decode(), syntetize_multi(), syntetize_program(), test_read(), test_seq_mem(), updatemenu(), verify_sequencer(), whichfilter(), and wmem104().

5.15.2.2 int Keepgoing

flag of continuing operations

Definition at line 111 of file server104.h.

Referenced by initvar104(), MainLoop(), MenuLoop(), sock_read(), and stop_daemon().

5.15.2.3 int Menu

menu/daemon flag (MYFALSE=daemon)

Definition at line 112 of file server104.h.

Referenced by catch_signals(), chkabort(), dump_seq_mem(), errlog(), finish(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), ignore_signals(), infolog(), initdev(), initmenu(), initvar104(), iolog(), main(), MenuLoop(), printstatus(), random_action(), read_image(), read_quadrant(), read_testimage(), and test_read().

5.15.2.4 int Windowed

windows active (MYFALSE/MYTRUE)

Definition at line 113 of file server104.h.

Referenced by errlog(), initvar104(), initwin(), and iolog().

5.15.2.5 int Logfile

flag of logfile active

Definition at line 114 of file server104.h.

Referenced by errlog(), initvar104(), iolog(), main(), and printstatus().

5.15.2.6 int Sendmsg

minimum message level sent

Definition at line 115 of file server104.h.

Referenced by initvar104(), iolog(), packetdecode(), and printstatus().

5.15.2.7 int Err104

internal error status for memory routines

Definition at line 117 of file server104.h.

Referenced by analog_boardId(), analog_boardStatus(), fifo_overflow(), fifo_ready(), initdev(), link_error(), link_status(), read_converters(), read_log(), read_testimage(), reprogram_vlevel(), reset_fpga(), rmem104(), sequencer_counter(), sequencer_status(), test_read(), whichfilter(), and wmem104().

5.15.2.8 int VBuff_id

buffer board id

Definition at line 118 of file server104.h.

Referenced by initdev(), initvar104(), printstatus(), and test_board_id().

5.15.2.9 int Acq_type

single(==0)/double acquisition style

Definition at line 119 of file server104.h.

Referenced by check_group(), compute_cycle(), compute_scan_time(), handle_image_acquisition(), initvar104(), MenuLoop(), packetdecode(), printstatus(), read_image(), syntetize_program(), and updatemenu().

5.15.2.10 int Extra_sub

Extra pixels subtraction on/off

Definition at line 120 of file server104.h.

Referenced by initvar104(), MenuLoop(), printstatus(), read_image(), and updatemenu().

5.15.2.11 int Lcd_present

Flag of LCD device available

Definition at line 121 of file server104.h.

Referenced by initdev(), initvar104(), l_error(), l_function(), l_header(), l_opera(), l_operation(), l_stdmessage(), lcd_init(), and main().

5.15.2.12 int Opera

indentifier of the running operation

Definition at line 122 of file server104.h.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardStatus(), check_data(), check_group(), compute_cycle(), compute_pix_readclk(), disable_sensor(), dummy_acquisition(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), fifo_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initdev(), initvar104(), l_get_port(), l_init_port(), l_opera(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), main(), MenuLoop(), multiswitch(), packetdecode(), print_fits_error(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), read_converters(), read_image(), read_init_file(), read_log(), read_pk_param(), read_quadrant(), read_testimage(), restart_daemon(), rmem104(), sawtooth_dac_test(), send_image(), sensor_onoff(), sequencer_counter(), sequencer_status(), setowner(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_board_id(), test_seq_mem(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), write_pk_param(), and zero_seq_mem().

5.15.2.13 int NoiseMode

flag to abilitate Noise acquisition mode

Definition at line 123 of file server104.h.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initvar104(), MenuLoop(), packetdecode(), and sys_running_status().

5.15.2.14 int SynchroTest

flag to test acquisition desynchronization

Definition at line 124 of file server104.h.

Referenced by handle_image_acquisition(), initvar104(), packetdecode(), and send_image().

5.15.2.15 int SVBTest

flag to set SVB in test image mode

Definition at line 125 of file server104.h.

Referenced by initvar104(), packetdecode(), start_broadcast(), and start_sequencer().

5.15.2.16 int SVBCheck

flag to put SVB in checking image mode

Definition at line 126 of file server104.h.

Referenced by initvar104(), packetdecode(), read_image(), reset_fifo(), and reset_fpga().

5.15.2.17 int gb_errno

error identifier

Definition at line 127 of file server104.h.

Referenced by abort_broadcast(), abort_sequencer(), accept_connection(), alloc_array_mem(), analog_boardId(), analog_boardStatus(), check_data(), check_group(), checksum_error(), compute_cycle(), create_filepath(), disable_sensor(), dummy_acquisition(), dummy_image(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), fifo_overflow(), fifo_ready(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init_hameg(), init_power(), initdev(), initvar104(), l_get_port(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), local_server_new(), main(), MainLoop(), mainloop_iterate(), MenuLoop(), multiswitch(), multivalid(), off_power(), open_giolamp(), open_log_file(), open_serial(), packetack(), packetdecode(), packetread(), packetread_gbridge(), packetsend(), print_fits_error(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), read_converters(), read_image(), read_init_file(), read_log(), read_power(), read_quadrant(), read_testimage(), restart_daemon(), rmem104(), sawtooth_dac_test(), send_image(), sensor_onoff(), sequencer_counter(), sequencer_status(), serial_command(), serial_decode(), serial_rdline(), set_close_on_exec(), set_descriptor_flags(), setowner(), sock_read(), sock_write(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), tcp_server_accept(), tcp_server_new(), tcp_socket_connect(), test_board_id(), test_seq_mem(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), write_seqfile(), and zero_seq_mem().

5.15.2.18 int Channel

Channel to operate 1-4, 5 means all

Definition at line 130 of file server104.h.

Referenced by abort_sequencer(), disable_sensor(), dlanalog(), dummy_acquisition(), dummy_image(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), handle_image_acquisition(), handle_multi_acquisition(), initdev(), initvar104(), main(), MenuLoop(), packetdecode(), printfilter(), printstatus(), program_DAC_filter(), program_DAC_Vlevel(), random_action(), read_init_file(), send_image(), start_sequencer(), stop_CI_sequencer(), stop_idle(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), test_seq_mem(), treset_program(), updatemenu(), write_pk_param(), and zero_seq_mem().

5.15.2.19 FILE* logmsg

file pointer to message log file

Definition at line 133 of file server104.h.

Referenced by errlog(), finish(), initvar104(), iolog(), open_log_file(), and restart_daemon().

5.15.2.20 char* logname

log filename

Definition at line 134 of file server104.h.

Referenced by finish(), initvar104(), main(), open_log_file(), and restart_daemon().

5.15.2.21 char* mempool

Definition at line 135 of file server104.h.

5.15.2.22 int Acquired

number of images acquired

Definition at line 136 of file server104.h.

Referenced by dump_init_file(), initvar104(), printstatus(), read_init_file(), and send_image().

5.15.2.23 int Ncol

column number of active detector

Definition at line 137 of file server104.h.

Referenced by check_data(), dump_data(), initvar104(), main(), packetdecode(), printstatus(), read_image(), read_quadrant(), and read_testimage().

5.15.2.24 int Nrow

row number of active detector

Definition at line 138 of file server104.h.

Referenced by check_data(), initvar104(), main(), packetdecode(), printstatus(), read_image(), read_quadrant(), and read_testimage().

5.15.2.25 int Run

run status __

Definition at line 139 of file server104.h.

Referenced by chkabort(), dummy_image(), embedsys_keepalive(), embedsys_keepalive_old(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initvar104(), integriamo(), l_opera(), MainLoop(), packetdecode(), read_image(), and send_image().

5.15.2.26 unsigned short* array

data array

Definition at line 140 of file server104.h.

Referenced by alloc_array_mem(), check_data(), dummy_image(), dump_data(), initvar104(), read_image(), read_quadrant(), read_testimage(), and send_image().

5.15.2.27 int array_size

allocated array size

Definition at line 141 of file server104.h.

Referenced by alloc_array_mem(), initvar104(), read_image(), and read_quadrant().

5.15.2.28 int Tint

Physical integration time in tens of milliseconds

Note:

on **Tint** and **Dit:**
Tint is the delay programmed on the sequences generator. It is equivalent to the 4 Prog_dit and has a minimum of 10 milliseconds. **Dit** is the total integration time on detector and INCLUDES the detector scan time (roughly 1024∗(1024+NEXTRA)∗Base_scan). The minimum value is Scan_time + 10 milliseconds. The general formula should read:
Dit = Tint + (scan time)

Definition at line 155 of file server104.h.

Referenced by compute_scan_time(), initvar104(), MenuLoop(), packetdecode(), printstatus(), random_action(), read_init_file(), read_pk_param(), syntetize_program(), updatemenu(), write_pk_param(), and write_primary_HU().

5.15.2.29 int Dit

Total integration time in tens of milliseconds

Definition at line 156 of file server104.h.

Referenced by check_group(), compute_scan_time(), dump_init_file(), handle_image_acquisition(), handle_multi_acquisition(), initvar104(), integriamo(), MenuLoop(), packetdecode(), printstatus(), random_action(), read_init_file(), updatemenu(), and write_pk_param().

5.15.2.30 int Base_scan

Pixel reading time in tens of microseconds

Definition at line 157 of file server104.h.

Referenced by compute_pix_readclk(), compute_scan_time(), dump_init_file(), initvar104(), MenuLoop(), printstatus(), random_action(), read_init_file(), read_pk_param(), updatemenu(), and write_pk_param().

5.15.2.31 int R_endframe

Delay between end of frame and Fsync of next

Definition at line 158 of file server104.h.

Referenced by compute_scan_time(), initvar104(), MenuLoop(), printstatus(), read_pk_param(), syntetize_multi(), syntetize_program(), and write_pk_param().

5.15.2.32 int R_fsync_lsync

Delay between end of Fsync and start of Lsync

Definition at line 159 of file server104.h.

Referenced by compute_scan_time(), initvar104(), MenuLoop(), printstatus(), read_pk_param(), syntetize_multi(), syntetize_program(), and write_pk_param().

5.15.2.33 int D_lsync

Duration of Lsync

Definition at line 160 of file server104.h.

Referenced by compute_scan_time(), initvar104(), MenuLoop(), printstatus(), read_pk_param(), syntetize_multi(), syntetize_program(), and write_pk_param().

5.15.2.34 int R_lsync_vclk

Delay between end of Lsync and start of Vclk

Definition at line 161 of file server104.h.

Referenced by compute_scan_time(), initvar104(), MenuLoop(), printstatus(), read_pk_param(), syntetize_multi(), syntetize_program(), and write_pk_param().

5.15.2.35 double Scan_time

detector reading time (scan time) units of 1 sec.

Definition at line 163 of file server104.h.

Referenced by check_group(), compute_cycle(), compute_scan_time(), handle_multi_acquisition(), initvar104(), MenuLoop(), multidummy(), multivalid(), packetdecode(), printstatus(), random_action(), read_init_file(), syntetize_multi(), and write_pk_param().

5.15.2.36 int Read_image_timeout

detector reading timeout (units of 300msec)

Definition at line 164 of file server104.h.

Referenced by compute_scan_time(), and handle_image_acquisition().

5.15.2.37 int Ngroup

number of integrations per group

Definition at line 165 of file server104.h.

Referenced by check_group(), dummy_image(), handle_multi_acquisition(), initvar104(), MenuLoop(), packetdecode(), printstatus(), random_action(), send_image(), and updatemenu().

5.15.2.38 int Curframe

current frame number

Definition at line 166 of file server104.h.

Referenced by dummy_image(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), packetdecode(), read_image(), read_quadrant(), reset_fifo(), and reset_fpga().

5.15.2.39 int Curgroup

current group number (in DC 1 group = 2 frame)

Definition at line 167 of file server104.h.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), packetdecode(), read_image(), reset_fifo(), and reset_fpga().

5.15.2.40 int Testmode

Test mode: ignores all failing checks

Definition at line 168 of file server104.h.

Referenced by initdev(), initvar104(), main(), and MainLoop().

5.15.2.41 int current_group

Definition at line 169 of file server104.h.

Referenced by initvar104().

5.15.2.42 time_t Prog_age

Age of compilation in seconds

Definition at line 170 of file server104.h.

Referenced by l_stdmessage().

5.15.2.43 time_t Logfile_age

Age of logfile in seconds

Definition at line 171 of file server104.h.

5.15.2.44 int file_n

fits file naming number

Definition at line 172 of file server104.h.

Referenced by initdev(), initvar104(), and save_data().

5.15.2.45 int Pck_seqnum

communication sequence number

Definition at line 173 of file server104.h.

Referenced by _packetsend(), and initvar104().

5.15.2.46 int nopStatus

NOP command delivered status

Definition at line 174 of file server104.h.

Referenced by embedsys_keepalive(), initvar104(), nop_timeout(), packetdecode(), and remove_nop_-timeout().

5.15.2.47 double timestamp

packet timestamp

Definition at line 175 of file server104.h.

Referenced by embedsys_keepalive(), embedsys_keepalive_old(), MainLoop(), packetread_gbridge(), and remove_nop_timeout().

5.15.2.48 int serial_fd

serial line file descriptor

Definition at line 176 of file server104.h.

5.15.2.49 int IdleStatus

idle acquisitions running status

Definition at line 177 of file server104.h.

Referenced by abort_broadcast(), abort_sequencer(), dummy_acquisition(), initvar104(), and stop_idle().

5.15.2.50 int ExpertMode

expert (lab) or normal execution mode

Definition at line 178 of file server104.h.

Referenced by close_embed_sockets(), initvar104(), packetdecode(), and stop_idle().

5.15.2.51 int Cycle_status

regime indicator for acquisition: 1 = Cycle_time << Transfer_time 2 = Cycle_time <∼ Transfer time 2 > Cycle_time > Transfer_time

Definition at line 179 of file server104.h.

Referenced by check_group().

5.15.2.52 double Cycle_time

Total time to perfom an integration (sec)

Definition at line 183 of file server104.h.

Referenced by check_group(), and compute_scan_time().

5.15.2.53 double Transfer_time

Total time to perfom an integration transfer (sec)

Definition at line 184 of file server104.h.

Referenced by check_group(), and compute_scan_time().

5.15.2.54 struct AnalogBoard Analog[4]

array of structure of type AnalogBoard, one for each board

Definition at line 221 of file server104.h.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardId(), analog_boardStatus(), compute_cycle(), compute_pix_readclk(), compute_scan_time(), disable_sensor(), dlanalog(), dump_init_file(), enable_sensor(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), initvar104(), link_error(), link_status(), MenuLoop(), printanalog(), printfilter(), printlink(), printsensors(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), read_init_file(), read_log(), read_pk_param(), sensor_onoff(), sequencer_running(), sequencer_status(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_seq_mem(), treset_program(), updatemenu(), verify_sequencer(), whichfilter(), and write_pk_param().

5.15.2.55 double PW_volt[4]

array of voltages preset on Hameg power source They are 4 values the four analog boards alimentation tensions

Definition at line 224 of file server104.h.

Referenced by dump_init_file(), and serial_decode().

5.15.2.56 double PW_amp[4]

the four analog boards alimentation current

Definition at line 225 of file server104.h.

Referenced by dump_init_file(), and serial_decode().

5.15.2.57 int PW_stat[4]

the four analog boards alimentation status

Definition at line 226 of file server104.h.

Referenced by init_power(), and serial_decode().

5.15.2.58 struct multi_status multistatus

multiple read operation real structure

Referenced by handle_multi_acquisition(), initvar104(), MenuLoop(), multidummy(), multifile(), multivalid(), syntetize_multi(), and updatemenu().

5.15.2.59 uid_t uid

Definition at line 260 of file server104.h.

5.15.2.60 gid_t gid

Definition at line 261 of file server104.h.

5.15.2.61 sock_t giolamp

Definition at line 100 of file sock104.c.

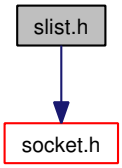
Referenced by multical().

5.16 slist.h File Reference

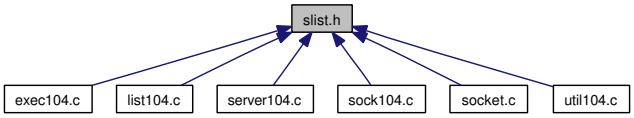
Singly-linked list type and functions.

`#include "socket.h"`

Include dependency graph for slist.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct cmd_t

  structure to associate command name to hex number

- struct _SList

  The SList struct is used for each element in the singly-linked list.

- union union_t

  Union defined in order to access both sock_t and cmd_t data from list.

Defines

- #define MEM_CHUNK_SIZE 256
- #define MEM_POOL_ITEM 128

Typedefs

- typedef int(∗ CompareFunc)(void ∗a, void ∗b)
- typedef struct _SList SList

Variables

- SList ∗ slist
- SList ∗ cmdList
- SList ∗ keyList

5.16.1 Detailed Description

Singly-linked list type and functions.

Definition in file slist.h.

5.16.2 Define Documentation

5.16.2.1 #define MEM_CHUNK_SIZE 256

Definition at line 45 of file slist.h.

Referenced by accept_connection(), mempool_alloc(), mempool_index(), slist_free(), slist_remove(), and slist_remove_disconnected().

5.16.2.2 #define MEM_POOL_ITEM 128

Definition at line 46 of file slist.h.

Referenced by mempool_alloc(), and mempool_index().

5.16.3 Typedef Documentation

5.16.3.1 typedef int(∗ CompareFunc)(void ∗a, void ∗b)

Specifies the type of a comparison function used to compare two values. The function should return 0 if the first and second value are equal, else a negative integer.

Parameters:

a a value

b a value to compare with

Returns:

negative value if a n.e b; zero if a = b.

Definition at line 62 of file slist.h.

5.16.3.2 typedef struct _SList SList

Singly-linked list : linked lists containing integer values or pointers to data, limited to iterating over the list in one direction

Definition at line 74 of file slist.h.

5.16.4 Variable Documentation

5.16.4.1 SList∗ slist

global list: each element describes a communication channel

Referenced by accept_connection(), check_socket_status(), close_all(), close_socket(), close_sockettype(), get_socket_descriptor(), initvar104(), MainLoop(), open_giolamp(), remove_disconnected_sources(), remove_event_source(), and update_event_sources().

5.16.4.2 SList* cmdList

global list: each element describes a command duple (name, hexval)

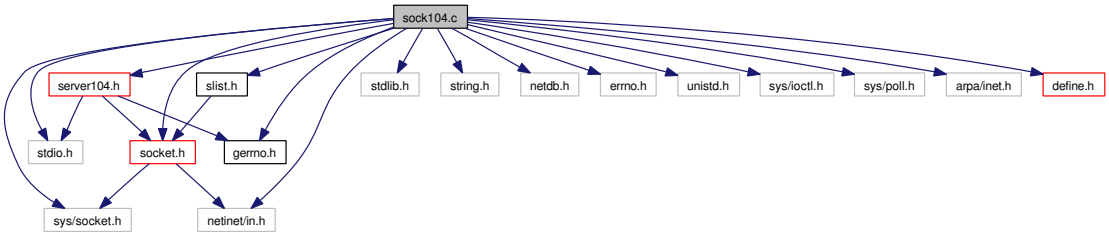
5.16.4.3 SList* keyList

5.17 sock104.c File Reference

High level I/O socket routines.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <errno.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/poll.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "socket.h"
#include "slist.h"
#include "define.h"
#include "server104.h"
#include "gerrno.h"
```

Include dependency graph for sock104.c:



Functions

- int update_event_sources (void)
- int get_socket_descriptor (int type)
- int check_socket_status (int sockfd)
- int check_sockettype_status (int socktype)
- int command_sock (void)
- int sockerr_func (int sockfd, void *data)
- int empty_socket (int socktype)
- void close_listen_socket (sock_t **listen_sock)
- void close_socket (int sockfd)
- void close_sockettype (int socktype)
- void close_embed_sockets ()

- void close_all (void)
- int mainloop_iterate (void)
- int accept_connection (sock_t *listen_sock, int timeout)
- int open_giolamp (void)
- void close_giolamp (void)
- int multiswitch (char c, int flag)

Variables

- sock_t giolamp

5.17.1 Detailed Description

High level I/O socket routines.

Author:

E.Giani

Definition in file sock104.c.

5.17.2 Function Documentation

5.17.2.1 int update_event_sources (void)

Updates the array of pollfd structures with the event sources to check during poll.

If the number of event source is greater then the allocated array of pollfd structures, we realloc it.

Returns:

0 on success, -1 on error.

Note:

This function is used when the program is executed in daemon mode.

Definition at line 2197 of file exec104.c.

References sock_t::ioevents, pfds, POLLFDN, slist, slist_length(), slist_nth_data(), and sock_t::sockfd.

Referenced by MainLoop().

5.17.2.2 int get_socket_descriptor (int *type*)

The function scans the linked list with the active connections and returns the socket descriptor associated with the connection type.

Parameters:

type the socket type: DATA or COMMAND

Returns:

Upon successful completion the function shall return the socket descriptor associated with the socket **type**. Otherwise -1 shall be returned

Definition at line 121 of file sock104.c.

References slist, slist_data_from_socktype(), and sock_t::sockfd.

Referenced by check_sockettype_status(), command_sock(), empty_socket(), mainloop_iterate(), packetack(), packetsend(), remove_event_sourcetype(), and send_image().

5.17.2.3 int check_socket_status (int *sockfd*)

The function shall return the connection status of the socket descriptor **sockfd**.

Parameters:

sockfd the socket descriptor

Returns:

the status of socket

Definition at line 145 of file sock104.c.

References DISCONNECTED, slist, and slist_data_from_descriptor().

Referenced by check_sockettype_status(), command_sock(), empty_socket(), mainloop_iterate(), packetack(), packetread_gbridge(), packetsend(), send_image(), and sockerr_func().

5.17.2.4 int check_sockettype_status (int *socktype*)

The function shall return the connection status of the socket descriptor whose type is **socktype**.

Parameters:

socktype the socket descripto type: DATA or COMMAND

Returns:

the status of socket descriptor of type **socktype:** CONNECTED , LISTENING, DISCONNECTED

Definition at line 169 of file sock104.c.

References check_socket_status(), and get_socket_descriptor().

Referenced by MainLoop().

5.17.2.5 int command_sock (void)

Definition at line 187 of file sock104.c.

References check_socket_status(), DISCONNECTED, EMBED_CMD, get_socket_descriptor(), MYFALSE, and MYTRUE.

Referenced by errlog(), and iolog().

5.17.2.6 int sockerr_func (int *sockfd*, void * *data*)

Callback function to handle I/O socket errors.

Parameters:

sockfd the socket descriptor

data a pointer to the corresponding socket structure (unused)

Returns:

0 on success, < 0 on errors.

Definition at line 214 of file sock104.c.

References check_socket_status(), close_socket(), DISCONNECTED, and iolog().

Referenced by accept_connection(), and open_giolamp().

5.17.2.7 int empty_socket (int *socktype*)

This function shall empty the queue of the socket descriptor of type **socktype**.

This call relies on the ioctl() system call to determine the number of bytes immediately available to be read on the socket descriptor. If there are data, the check is iterated for 1 sec.

Returns:

Unpon successful completion the function shall return 0. Otherwise -1.

Definition at line 244 of file sock104.c.

References check_socket_status(), DISCONNECTED, elapsed_time(), get_current_time(), get_socket_descriptor(), iolog(), and sock_read().

Referenced by close_embed_sockets().

5.17.2.8 void close_listen_socket (sock_t ** *listen_sock*)

The function shall close the listen socket opened by the server. **This** function is called after the successful completion of the accept() system call. In such way the server accepts only one connection per time.

Parameters:

listen_sock the listen socket structure

Definition at line 309 of file sock104.c.

References CLOSED, iolog(), LISTENING, sock_addr::s_in, sock_t::sa, sock_t::sockfd, and sock_t::sockstatus.

Referenced by MainLoop().

5.17.2.9 void close_socket (int *sockfd*)

The close_socket() function shall close the file descriptor indicated by **sockfd** and sets its status to DISCONNECTED.

The corresponding node in the linked list isn't removed.

Parameters:

sockfd the socket descriptor

Definition at line 338 of file sock104.c.

References CONNECTED, DISCONNECTED, iolog(), slist, slist_data_from_descriptor(), sock_t::sockfd, and sock_t::sockstatus.

Referenced by _packetsend(), checksum_error(), close_giolamp(), packetread(), send_image(), and sockerr_func().

5.17.2.10 void close_sockettype (int *socktype*)

The close_sockettype() function shall close the file descriptor whose type is indicated by **socktype** and sets its status to DISCONNECTED.

The corresponding node in the linked list isn't removed.

Parameters:

socktype the socket descriptor type: DATA or COMMAND

Definition at line 367 of file sock104.c.

References CONNECTED, DISCONNECTED, slist, slist_data_from_socktype(), sock_t::sockfd, and sock_t::sockstatus.

Referenced by close_embed_sockets().

5.17.2.11 void close_embed_sockets ()

The close_embed_sockets() function shall close the descriptor sockets corresponding to DATA and COMMAND type.

Internally it relies on close_sockettype() call.

Definition at line 392 of file sock104.c.

References close_sockettype(), EMBED_CMD, EMBED_DATA, empty_socket(), ExpertMode, and MYFALSE.

Referenced by MainLoop().

5.17.2.12 void close_all (void)

Definition at line 410 of file sock104.c.

References _SList::data, DISCONNECTED, iolog(), _SList::next, slist, sock_t::sockfd, sock_t::sockstatus, and sock_t::socktype.

Referenced by MainLoop().

5.17.2.13 int mainloop_iterate (void)

Runs a single iteration for the main loop. This involves checking to see if the COMMAND socket source is ready to be processed. This operation is done relying on the ioctl() system call.**If** there are data available to be read the callback associated to the event is called.

Returns:

Upon successful completion >= 0 shall be returned, otherwise -1.

Definition at line 439 of file sock104.c.

References check_socket_status(), DISCONNECTED, EMBED_CMD, GB_EBADARG, gb_errno, get_-socket_descriptor(), and packetread_gbridge().

Referenced by chkabort(), and send_image().

5.17.2.14 int accept_connection (sock_t * *listen_sock*, int *timeout*)

This function shall wait from a connection request on the opened listening socket. Once accpeted, it allocates the memory for the socket structure where the connection information is stored. **The** parameter timeout is the maximum time interval waited by accept() to complete a connection. A negative timeout configures a blocking accept() call (i.e. accept waits indefinitely), interruptible only by the SIGTERM

Parameters:

listen_sock the listing socket structure

timeout the maximum time interval to complete a connection.

Returns:

the connected socket status (>0) on success, < 0 on error(s).

Definition at line 488 of file sock104.c.

References CONNECTED, union_t::data, DISCONNECTED, EMBED_CMD, EMBED_DATA, EMBEDDED_CMD_PORT, EMBEDDED_DATA_PORT, sock_t::errfunc, gb_errno, sock_t::index, sock_t::ioevents, sock_t::iofunc, iolog(), MEM_CHUNK_SIZE, mempool, mempool_index(), NON_-BLOCK, packetread_gbridge(), sock_addr::s_in, sock_t::sa, set_descriptor_flags(), slist, slist_append(), union_t::sock, sockerr_func(), sock_t::sockfd, sock_t::sockstatus, sock_t::socktype, and tcp_server_-accept().

Referenced by MainLoop().

5.17.2.15 int open_giolamp (void)

Opens a connection to a terminal server to switch on/off the calibration lamps.

param hostname the terminal server name param port the port to progrma terminal server param nonBlock flag to use blockin or non-blocking connection

Returns:

the socket descriptor (> 0) on success, MYFALSE /MYERR (<= 0) on error(s)

Definition at line 552 of file sock104.c.

References CONNECTED, DISCONNECTED, EMBED_TS, sock_t::errfunc, gb_errno, GB_ESLIST, sock_t::ioevents, iolog(), MYERR, NON_BLOCK, set_descriptor_flags(), slist, slist_append(), sockerr_-func(), sock_t::sockfd, sock_t::sockstatus, sock_t::socktype, and tcp_socket_connect().

Referenced by handle_multi_acquisition(), and multiswitch().

5.17.2.16 void close_giolamp (void)

Closes the socket connection to terminal server.

Definition at line 615 of file sock104.c.

References close_socket(), DISCONNECTED, and sock_t::sockfd.

Referenced by multiswitch(), and packetdecode().

5.17.2.17 int multiswitch (char *c*, int *flag*)

Switch on and off the appropriate lamp, depending on operation **c** ad flag **flag**. NOTE: FOR NOW we IGNORE c <==========

Parameters:

c operation to test

flag flag of on (MYTRUE) or off operation

Returns:

MYTRUE if all OK, MYFALSE otherwise.

Definition at line 640 of file sock104.c.

References close_giolamp(), errlog(), error_code(), GB_ELAMP, gb_errno, GB_WARNING, iolog(), MYFALSE, MYTRUE, open_giolamp(), Opera, sock_read(), sock_write(), and sock_t::sockfd.

Referenced by multical().

5.17.3 Variable Documentation

5.17.3.1 sock_t giolamp

Definition at line 100 of file sock104.c.

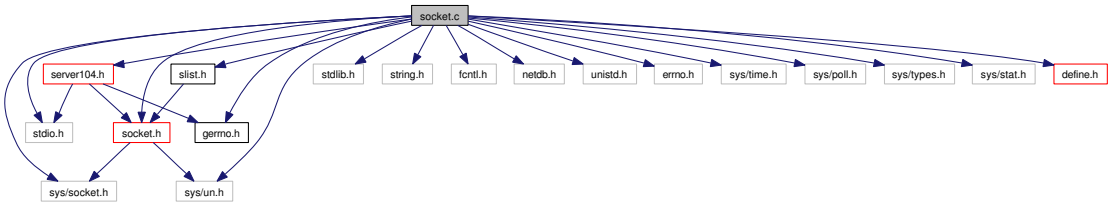
Referenced by multical().

5.18 socket.c File Reference

Low level socket routines.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>
#include <sys/time.h>
#include <sys/poll.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <sys/un.h>
#include "define.h"
#include "socket.h"
#include "gerrno.h"
#include "slist.h"
#include "server104.h"
```

Include dependency graph for socket.c:



Functions

- double get_current_time ()
- double elapsed_time (double time0)
- int set_close_on_exec (int fd)
- int set_descriptor_flags (int fd, int nonBLock)
- int tcp_socket_connect (char *hostname, int port, int nonBlock, sock_t *client_sock)
- int sock_read (int fd, void *buf, size_t nbytes, int timeout)
- int sock_write (int fd, void *buf, size_t nbytes, int timeout)
- sock_t * tcp_server_new (char *hostname, int port)
- int tcp_server_accept (int listen_sock, int timeout, sock_t *accept_sock)

- int local_server_accept (int listen_sock, int timeout, sock_t *sock)
- sock_t * local_server_new (char *file)

5.18.1 Detailed Description

Low level socket routines.

Author:

E.Giani

Definition in file socket.c.

5.18.2 Function Documentation

5.18.2.1 double get_current_time ()

Uses the function gettimeofday() to get the current time.

Returns:

the current time in millisec.

Definition at line 72 of file socket.c.

Referenced by checksum_error(), elapsed_time(), empty_socket(), flush_serial(), handle_image_acquisition(), handle_multi_acquisition(), integriamo(), MainLoop(), packetread_gbridge(), read_converters(), read_image(), read_log(), read_quadrant(), read_testimage(), remove_nop_timeout(), serial_rdline(), sock_read(), sock_write(), and test_board_id().

5.18.2.2 double elapsed_time (double *time0*)

Evaluates the time elapsed from a starting time

Parameters:

time0,: the starting time

Returns:

the time elapsed (in ms) from the init time0.

Definition at line 94 of file socket.c.

References get_current_time().

Referenced by checksum_error(), embedsys_keepalive(), embedsys_keepalive_old(), empty_socket(), flush_serial(), handle_image_acquisition(), handle_multi_acquisition(), integriamo(), MainLoop(), read_converters(), read_image(), read_log(), read_quadrant(), read_testimage(), serial_rdline(), sock_read(), sock_write(), and test_board_id().

5.18.2.3 int set_close_on_exec (int *fd*)

Sets the FD_CLOEXEC flag for this socket descriptor before calling any exec() call. If this flag is setted the descriptor is closed during exec.

Parameters:

fd the file descriptor

Returns:

0 on success , < 0 on error(s).

Definition at line 116 of file socket.c.

References gb_errno.

5.18.2.4 int set_descriptor_flags (int *fd*, int *nonBLock*)

Sets a file descriptor in blocking or non blocking mode

Parameters:

fd the file descriptor

nonBLock specifies if the O_NONBLOCK flag is enabled or not

Returns:

0 on success, -1 on errors.

Definition at line 145 of file socket.c.

References gb_errno, iolog(), and strerror_104().

Referenced by accept_connection(), local_server_new(), open_giolamp(), tcp_server_new(), and tcp_socket_connect().

5.18.2.5 int tcp_socket_connect (char * *hostname*, int *port*, int *nonBlock*, sock_t * *client_sock*)

Establishes a network connection with a specified server host and service. If the flag nonBlock is true, the connection is not blocking.

Parameters:

hostname the hostname of the server

port the server service port number

nonBlock the flag to configure blocking mode

client_sock a sock_t structure

Returns:

the connected socket on success, < 0 on error

See also:

sock_t

Definition at line 188 of file socket.c.

References GB_EBADARG, GB_EMEMALLOC, gb_errno, GB_IO_CONNECT_ERR, GB_IO_TIME_-CONNECT, sock_addr::s, sock_addr::s_in, sock_t::sa, set_descriptor_flags(), sock_t::sockfd, and strdup_-printf().

Referenced by open_giolamp().

5.18.2.6 int sock_read (int *fd*, void * *buf*, size_t *nbytes*, int *timeout*)

Reads a specified number of bytes from a file descriptor.

Parameters:

fd the file descriptor

buf the buffer to store data in

nbytes the nu,ber of bytes to read

timeout specifies a timeout in ms. A negative value means infinite timeout.

Returns:

the bytes read on success, -1 on unrecoverable errors, 0 on timeout.

Definition at line 326 of file socket.c.

References elapsed_time(), gb_errno, GB_IO_EOF, GB_IO_POLLERR, GB_IO_POLLHUP, GB_IO_-POLLNVAL, GB_IO_TIME_READ, get_current_time(), iolog(), Keepgoing, and SOCK_TIMEOUT.

Referenced by empty_socket(), flush_serial(), multiswitch(), packetread(), and serial_rdline().

5.18.2.7 int sock_write (int *fd*, void * *buf*, size_t *nbytes*, int *timeout*)

The function shall attempt to write the specified number of bytes from the buffer to the file associated with the open descriptor. If the file descriptor refers to a socket then it checks for peer end closing condition. This function leans on the `poll()` system call.

Parameters:

fd the file descriptor

buf the buffer to copy from

nbytes the number of bytes to write

timeout the timeout specified in ms. A negative value means an infinite timeout.

Returns:

Upon successful completion sock_write() shall return the number of bytes actually written to the file. This number shall be equal to **nbytes**. Otherwise, -1 shall be returned on unrecoverable errors. If the file descriptor refers to a socket, the condition of timeout shall return the bytes actually written and in this case this number shall never be greater than
bytes.

Definition at line 430 of file socket.c.

References elapsed_time(), gb_errno, GB_IO_EOF, GB_IO_POLLERR, GB_IO_POLLHUP, GB_IO_-POLLNVAL, GB_IO_TIME_WRITE, get_current_time(), and SOCK_TIMEOUT.

Referenced by _packetsend(), multiswitch(), send_image(), and serial_wrline().

5.18.2.8 sock_t* tcp_server_new (char * *hostname*, int *port*)

Parameters:

hostname the hostname (unused)

port the service port

Returns:

Upon successful completion the pointer to the structure sock_t. Otherwise NULL or error.

Definition at line 526 of file socket.c.

References GB_EBADARG, GB_EMEMALLOC, gb_errno, GB_IO_NONBLOCK_ERR, iolog(), LISTENING, NON_BLOCK, sock_addr::s, sock_addr::s_in, sock_t::sa, set_descriptor_flags(), sock_t::sockfd, and sock_t::sockstatus.

Referenced by MainLoop().

5.18.2.9 int tcp_server_accept (int *listen_sock*, int *timeout*, sock_t * *accept_sock*)

The function shall extract the first connection on the queue of pending connections, create a new socket and allocate a new file descriptor for that socket.

Parameters:

listen_sock a descriptor of a socket that has successful called listen()

timeout the timeout in ms. A negative value means infinite timeout.

accept_sock the file descriptor for the new socket

Returns:

the connected socket on success, < 0 on error.

Note:

This call is interrupted by signal even if the socket is in blocking mode.

Definition at line 639 of file socket.c.

References gb_errno, GB_IO_TIME_CONNECT, iolog(), sock_addr::s, sock_t::sa, and sock_t::sockfd.

Referenced by accept_connection(), and local_server_accept().

5.18.2.10 int local_server_accept (int *listen_sock*, int *timeout*, sock_t * *sock*)

Parameters:

listen_sock socket

timeout timeout

sock socket

Returns:

Upon successful completion the connected socket, -1 on error.

Definition at line 724 of file socket.c.

References tcp_server_accept().

5.18.2.11 sock_t* local_server_new (char * *file*)

Definition at line 737 of file socket.c.

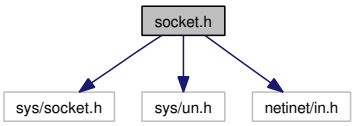
References GB_EMEMALLOC, gb_errno, GB_IO_NONBLOCK_ERR, iolog(), NON_BLOCK, sock_addr::s, sock_addr::s_un, sock_t::sa, set_descriptor_flags(), and sock_t::sockfd.

5.19 socket.h File Reference

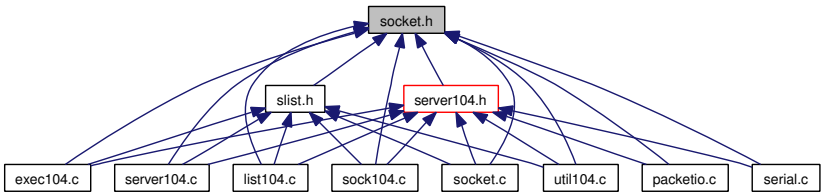
Socket related include.

```
#include <sys/socket.h>
#include <sys/un.h>
#include <netinet/in.h>
```

Include dependency graph for socket.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct sock_addr

  union with socket address

- struct sock_t

  A data structure representing the socket.

- struct GPacket

  the header packet with info about destination and its type and length in bytes

Defines

- #define IMG_HEADER 8
- #define HDR_NUM (HDR_SIZE/2)
- #define HDR_SIZE 16
- #define PCK_SIZE 1400

Typedefs

- typedef int(∗ Callback_t)(int sockfd, void ∗data)

Enumerations

- enum S104SockStatus {
  DISCONNECTED = -1,
  CLOSED = -1,
  CONNECTED = 1,
  LISTENING = 2 }
- enum S104SockType {
  EMBED_CMD = 1,
  EMBED_DATA,
  EMBED_TS }

5.19.1 Detailed Description

Socket related include.

Author:

E.Giani

Definition in file socket.h.

5.19.2 Define Documentation

5.19.2.1 #define IMG_HEADER 8

data packet header in bytes

Definition at line 122 of file socket.h.

Referenced by send_image().

5.19.2.2 #define HDR_NUM (HDR_SIZE/2)

command packet header in words

Definition at line 123 of file socket.h.

Referenced by _packetsend(), and calculate_checksum().

5.19.2.3 #define HDR_SIZE 16

command packet header in bytes

Definition at line 124 of file socket.h.

Referenced by _packetsend(), and packetread().

5.19.2.4 #define PCK_SIZE 1400

command packet pacyload in bytes

Definition at line 125 of file socket.h.

Referenced by _packetsend(), checksum_error(), and packetdecode().

5.19.3 Typedef Documentation

5.19.3.1 typedef int(∗ Callback_t)(int sockfd, void ∗data)

Specifies the type of a the function used to handle the I/O from/to different types of channel in a generic way.

Parameters:

sockfd the socket descriptor to operate on

data user-supplied data

Returns:

0 on success, < 0 on errors.

Definition at line 63 of file socket.h.

5.19.4 Enumeration Type Documentation

5.19.4.1 enum S104SockStatus

Specifies the status of the communication channel.

Enumerator:

DISCONNECTED The socket is not connected

CLOSED the socket is closed

CONNECTED the socket is connected

LISTENING the socket is listening on connection

Definition at line 66 of file socket.h.

5.19.4.2 enum S104SockType

Specifies the type of the communication channel.

Enumerator:

EMBED_CMD the connection is used to exchange command

EMBED_DATA the connection is used to exchange data

EMBED_TS connection to terminal server

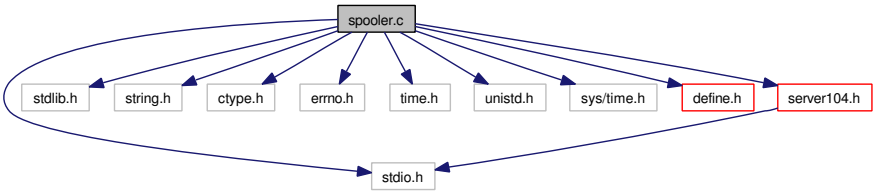
Definition at line 75 of file socket.h.

5.20 spooler.c File Reference

low profile spooler for low priority task

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <time.h>
#include <unistd.h>
#include <sys/time.h>
#include "define.h"
#include "server104.h"
```

Include dependency graph for spooler.c:



Data Structures

- struct event_struct

  Structure containing the details of a submitted event.

- struct _s_FIFO

  Structure containing a single submitted event data.

- struct s_FIFO

  Structure containing the FIFO queue of submitted events.

- struct errTIM

  Structure containing the error message queue data.

Defines

- #define FIFO_WIDTH 16
- #define FIFO_MASK 0xf

Enumerations

- enum PRIORITY_LEVEL {
  LOW_PRI = 1,
  HIGH_PRI }
- enum OPER_TYPE {
  LCD_ERR = 1,
  LCD_CLR,
  ERRLOG,
  DISK_FLUSH,
  KEEP_ALIVE }

Functions

- int isvoid_fifo (struct s_FIFO *p_fifo)
- int isfull_fifo (struct s_FIFO *p_fifo)
- int empty_fifo (struct s_FIFO *p_fifo)
- int empty_events_h (void)
- int add_s_entry (int priority, int operation, int arg_1, char msg[80], float arg_2, time_t min_duration, time_t max_duration)
- int add_f_entry (int priority, time_t time_to_submit, int delta_time, int operation, int arg_1, char msg[80], float arg_2, time_t min_duration, time_t max_duration)
- int s_wrapper (int event)
- int f_sched (void)
- int s_sched (int min_priority)

Variables

- int Verbose = 2
- static struct s_FIFO resched
- static struct s_FIFO hi_prio
- static struct s_FIFO lo_prio
- static struct errTIM errtim
- static struct event_struct events_h [2 *FIFO_WIDTH]
- int event_void [2 *FIFO_WIDTH]

5.20.1 Detailed Description

low profile spooler for low priority task

Basic structure: We have two different subsystems.

The first is the real scheduler. It is composed by a global structure containing the event-handling informations, and one FIFO ring for every level of priority, initially 2. At every call the spooler explore the rings in descending priority order. If no tasks are present on the current level, it switches to lower priority. If the lowest ring is empty, the spooler generates a 'idle state' event.

The second part is composed by the re-sceduler and by the ancillary functions. The rescheduler has two functions: the first maintain a list (with a format similar to the scheduler) of actions to be performed at a future time, the second chechs if some of these operations are ready to be performed, and then copy their

info in the scheduler FIFO ring. Other functions are reset of structures, insert of events, function wrappers, and all low level FIFO handling.

The event-handling informations structure is composed as follows: a priority number, the date of scheduling in unit time format, a pointer to the appropriate function wrapper, a small list of arguments, I.E. an integer, an array of chars, a float, an array of int, a minimum and a maximum duration. The FIFO rings contains only the scheduler time and a pointer to the appropriate item in the info structure. If the FIFO relates to the future scheduling time, it can contain also a re-scheduling interval, for periodic operations. For simplicity we use the same structure for both.

Definition in file spooler.c.

5.20.2 Define Documentation

5.20.2.1 #define FIFO_WIDTH 16

width of the FIFO.

Definition at line 85 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), empty_events_h(), empty_fifo(), f_sched(), isfull_fifo(), isvoid_fifo(), and s_sched().

5.20.2.2 #define FIFO_MASK 0xf

mask to enforce a FIFO_WIDTH addressing limit

Definition at line 86 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), f_sched(), isfull_fifo(), isvoid_fifo(), and s_sched().

5.20.3 Enumeration Type Documentation

5.20.3.1 enum PRIORITY_LEVEL

Enumerator:

LOW_PRI

HIGH_PRI

Definition at line 151 of file spooler.c.

5.20.3.2 enum OPER_TYPE

Enumerator:

LCD_ERR

LCD_CLR

ERRLOG

DISK_FLUSH

KEEP_ALIVE

Definition at line 157 of file spooler.c.

5.20.4 Function Documentation

5.20.4.1 int isvoid_fifo (struct s_FIFO * *p_fifo*)

check if FIFO is empty

Definition at line 177 of file spooler.c.

References s_FIFO::bot_fifo, FIFO_MASK, FIFO_WIDTH, MYFALSE, MYTRUE, s_FIFO::top_fifo, and Verbose.

Referenced by f_sched(), and s_sched().

5.20.4.2 int isfull_fifo (struct s_FIFO * *p_fifo*)

check if FIFO is full

Definition at line 194 of file spooler.c.

References s_FIFO::bot_fifo, FIFO_MASK, FIFO_WIDTH, MYFALSE, MYTRUE, s_FIFO::top_fifo, and Verbose.

Referenced by add_f_entry(), and add_s_entry().

5.20.4.3 int empty_fifo (struct s_FIFO * *p_fifo*)

empty the FIFO

Definition at line 211 of file spooler.c.

References s_FIFO::bot_fifo, s_FIFO::fifo, FIFO_WIDTH, MYTRUE, and s_FIFO::top_fifo.

5.20.4.4 int empty_events_h (void)

empty the events database

Definition at line 222 of file spooler.c.

References event_void, events_h, FIFO_WIDTH, and MYTRUE.

5.20.4.5 int add_s_entry (int *priority*, int *operation*, int *arg_1*, char *msg*[80], float *arg_2*, time_t *min_duration*, time_t *max_duration*)

Add a real call to the Hi/Lo/... fifos

Definition at line 236 of file spooler.c.

References event_struct::arg_1, event_struct::arg_2, s_FIFO::bot_fifo, _s_FIFO::delta_time, _s_FIFO::event_index, event_void, events_h, s_FIFO::fifo, FIFO_MASK, FIFO_WIDTH, hi_prio, HIGH_PRI, isfull_fifo(), lo_prio, LOW_PRI, event_struct::max_duration, event_struct::min_duration, MYFALSE, MYTRUE, event_struct::operation, _s_FIFO::priority, event_struct::priority, _s_FIFO::time_submitted, event_struct::time_submitted, _s_FIFO::time_to_submit, s_FIFO::top_fifo, and Verbose.

Referenced by f_sched(), and s_sched().

5.20.4.6 int add_f_entry (int *priority*, time_t *time_to_submit*, int *delta_time*, int *operation*, int *arg_1*, char *msg*[80], float *arg_2*, time_t *min_duration*, time_t *max_duration*)

Add a real call to the future events (resched) fifo

Definition at line 342 of file spooler.c.

References event_struct::arg_1, event_struct::arg_2, s_FIFO::bot_fifo, _s_FIFO::delta_time, _s_FIFO::event_index, event_void, events_h, s_FIFO::fifo, FIFO_MASK, FIFO_WIDTH, isfull_fifo(), event_struct::max_duration, event_struct::min_duration, MYFALSE, MYTRUE, event_struct::operation, _s_FIFO::priority, event_struct::priority, resched, _s_FIFO::time_submitted, event_struct::time_submitted, _s_FIFO::time_to_submit, s_FIFO::top_fifo, and Verbose.

5.20.4.7 int s_wrapper (int *event*)

execute the operation in the row event of array events_h

Definition at line 435 of file spooler.c.

References errTIM::err_max, errTIM::err_min, errTIM::err_time, errtim, events_h, errTIM::last_clr, LCD_CLR, LCD_ERR, event_struct::max_duration, event_struct::min_duration, MYFALSE, MYTRUE, event_struct::operation, and Verbose.

Referenced by s_sched().

5.20.4.8 int f_sched (void)

submit the first event scheduled for later execution which is ready.

Definition at line 492 of file spooler.c.

References add_s_entry(), s_FIFO::bot_fifo, _s_FIFO::delta_time, _s_FIFO::event_index, event_void, events_h, s_FIFO::fifo, FIFO_MASK, FIFO_WIDTH, isvoid_fifo(), MYFALSE, MYTRUE, _s_FIFO::priority, resched, _s_FIFO::time_submitted, _s_FIFO::time_to_submit, s_FIFO::top_fifo, and Verbose.

Referenced by s_sched().

5.20.4.9 int s_sched (int *min_priority*)

execute the highest priority and older item in Hi/Lo/... fifos min_priority is the minimum priority to be executed, if zero, all fifo are scanned (ah the blood!)

Definition at line 623 of file spooler.c.

References add_s_entry(), s_FIFO::bot_fifo, _s_FIFO::event_index, event_void, events_h, f_sched(), s_FIFO::fifo, FIFO_MASK, FIFO_WIDTH, hi_prio, HIGH_PRI, isvoid_fifo(), LCD_CLR, lo_prio, LOW_PRI, MYFALSE, MYTRUE, s_wrapper(), s_FIFO::top_fifo, and Verbose.

5.20.5 Variable Documentation

5.20.5.1 int Verbose = 2

Definition at line 87 of file spooler.c.

5.20.5.2 struct s_FIFO resched [static]

FIFO queue of periodic events

Definition at line 131 of file spooler.c.

Referenced by add_f_entry(), and f_sched().

5.20.5.3 struct s_FIFO hi_prio [static]

FIFO queue for higher priority tasks

Definition at line 132 of file spooler.c.

Referenced by add_s_entry(), and s_sched().

5.20.5.4 struct s_FIFO lo_prio [static]

< FIFO queue for lower priority tasks

Definition at line 133 of file spooler.c.

Referenced by add_s_entry(), and s_sched().

5.20.5.5 struct errTIM errtim [static]

Structure containing the error message queue data

Definition at line 145 of file spooler.c.

Referenced by s_wrapper().

5.20.5.6 struct event_struct events_h[2 ∗FIFO_WIDTH] [static]

database of events data

Definition at line 166 of file spooler.c.

Referenced by add_f_entry(), add_s_entry(), empty_events_h(), f_sched(), s_sched(), and s_wrapper().

5.20.5.7 int event_void[2 ∗FIFO_WIDTH]

pointers into events_h

Definition at line 167 of file spooler.c.

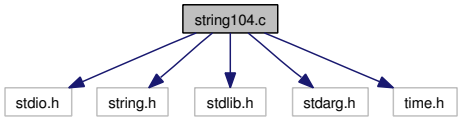
Referenced by add_f_entry(), add_s_entry(), empty_events_h(), f_sched(), and s_sched().

5.21 string104.c File Reference

Exended strings handling.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>
#include <time.h>
```

Include dependency graph for string104.c:



Functions

- char * strdup_printf (const char *format,...)
- char * gb_strndup (char *str, int n)
- char * gb_stpcpy (char *dest, const char *src)
- char * gb_strconcat (const char *string1,...)
- char * gb_strrstr (char *haystack, char *needle)
- char * strdup_vprintf (const char *format, va_list args1)

5.21.1 Detailed Description

Exended strings handling.

Author:

E.Giani

Definition in file string104.c.

5.21.2 Function Documentation

5.21.2.1 char* strdup_printf (const char * *format*, ...)

Similar to the standard C sprintf() function but safer, since it calculates the maximum space required and allocates memory to hold the result. The returned string should be freed with g_free() when no longer needed.

Parameters:

format a standard printf() format string

... the parameters to insert into the format string

Returns:

a newly-allocated string holding the result

Definition at line 54 of file string104.c.

Referenced by create_filepath(), open_log_file(), procname(), read_converters(), restart_daemon(), save_-data(), and tcp_socket_connect().

5.21.2.2 char* gb_strndup (char * *str*, int *n*)

Local definition of strndup() function which is a GNU extension

Duplicates the first `n` bytes of a string, returning a newly-allocated buffer n + 1 bytes long which will always be nul-terminated. If str is less than n bytes long the buffer is padded with nuls. If str is NULL it returns NULL.

Parameters:

str the string to duplicate

n the maximum number of bytes to copy from str

Returns:

a newly-allocated buffer containing the first n bytes of str or NULL if insufficient memory was available.

Note:

The returned value should be freed when no longer needed.

Definition at line 119 of file string104.c.

5.21.2.3 char* gb_stpcpy (char * *dest*, const char * *src*)

Local definition of stpcpy() function

The gb_stpcpy() function copies the string pointed to by `src` (including the terminating '\0' character) to the array pointed to by `dest`. The strings may not overlap, and the destination string dest must be large enough to receive the copy.

Parameters:

dest the destination string

src the string to copy

Returns:

a pointer to the end of the string `dest`, otherwise NULL

Definition at line 150 of file string104.c.

Referenced by gb_strconcat().

5.21.2.4 char* gb_strconcat (const char * *string1*, ...)

Concatenates all of the given strings into one long string.

Parameters:

string1 the first string to add, which must not be NULL

... a NULL-terminated list of strings to append to the string

Returns:

a newly-allocated string containing all the string arguments. NULL

Note:

The returned string should be freed when no longer needed, or NULL on errors

Definition at line 179 of file string104.c.

References gb_stpcpy().

5.21.2.5 char* gb_strrstr (char * *haystack*, char * *needle*)

Searches the string `haystack` for the last occurrence of the string `needle`.

Parameters:

haystack a nul-terminated string.

needle the nul-terminated string to search for.

Returns:

a pointer to the found occurrence, or NULL if not found.

Definition at line 227 of file string104.c.

5.21.2.6 char* strdup_vprintf (const char * *format*, va_list *args1*)

Similar to the standard C vsprintf() function but safer, since it calculates the maximum space required and allocates memory to hold the result.

Parameters:

format a standard printf() format string

args1 the list of arguments to insert in the output

Returns:

a newly-allocated string holding the result on success, NULL on error(s).

Note:

The returned string should be freed when no longer needed.

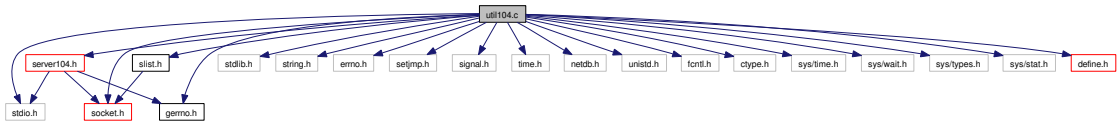
Definition at line 270 of file string104.c.

5.22 util104.c File Reference

Utility functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <setjmp.h>
#include <signal.h>
#include <time.h>
#include <netdb.h>
#include <unistd.h>
#include <fcntl.h>
#include <ctype.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "socket.h"
#include "slist.h"
#include "define.h"
#include "server104.h"
#include "gerrno.h"
```

Include dependency graph for util104.c:



Defines

- #define STD_INI_FILE "/home/softir/data/server104.ini"
- #define ALT_INI_FILE "/var/tmp/server104.ini"

Functions

- char * procname (char *program, int *vpid, int remote)
- int printanalog (int channel, va_list args)
- void printstatus (void)
- void printsensors (char *buf)

- void printlink (char *buf)
- void print_alimentation (int Astatus, float clockms)
- void printfilter (char *buf)
- int initdev (void)
- int reinitdev (void)
- void finish (int sig)
- int error_code (int errtype, int process, int error)
- char * strerror_104 (int _gb_errno)
- void syslog_message (int priority, char *fmt,...)
- char * create_filepath (char *destdir, char *optdir, char *name)
- int open_log_file (void)
- int daemon_is_already_running (void)
- void stop_daemon (int signal)
- void restart_daemon (int signal)
- int catch_signals (void)
- int ignore_signals (void)
- void dump_packet (char *func, int destination, int type, int cmd, int datalen, unsigned short seqnum, char *payload)
- char * dump_sock_status (int status)
- char * dump_acquisition_status (int status)
- int dump_init_file (void)
- int read_init_file (void)
- void multidummy (void)
- int ismultiinteg (char c)
- void my_bell (void)
- int multical (char c, int flag)
- int multivalid (void)
- int multifile (char *multi_program)

Variables

- static int facility = LOG_DAEMON|LOG_USER

5.22.1 Detailed Description

Utility functions.

Miscellaneous functions of medium level. We collect here general purpose procedure.

Definition in file util104.c.

5.22.2 Define Documentation

5.22.2.1 #define STD_INI_FILE "/home/softir/data/server104.ini"

Referenced by dump_init_file(), and read_init_file().

5.22.2.2 #define ALT_INI_FILE "/var/tmp/server104.ini"

Referenced by dump_init_file(), and read_init_file().

5.22.3 Function Documentation

5.22.3.1 char* procname (char * *program*, int * *vpid*, int *remote*)

Reads the process table using popen ad ps

Parameters:

program the program to look for

vpid the process id of the program

remote if the program should be executed remotely

Returns:

the program name on success, NULL otherwise.

Definition at line 145 of file util104.c.

References REMOTE_HOST, REMOTE_USER, and strdup_printf().

Referenced by daemon_is_already_running().

5.22.3.2 int printanalog (int *channel*, va_list *args*)

A routine to print status of one or all analog boards.

Parameters:

channel channel to operate on

args variable list of arguments

Definition at line 221 of file util104.c.

References Analog, iolog(), and MYTRUE.

Referenced by printstatus().

5.22.3.3 void printstatus (void)

A routine to print complete status. We try to put as much as possible of system status.

Warning:

This routines declare messages with status instead of printstatus.

Definition at line 302 of file util104.c.

References Acq_type, Acquired, analog_boardId(), analog_boardStatus(), Base_scan, Channel, D_lsync, Dit, execute_cmd(), Extra_sub, iolog(), Logfile, Menu, MYTRUE, Ncol, Ngroup, Nrow, printanalog(), R_endframe, R_fsync_lsync, R_lsync_vclk, Scan_time, Sendmsg, Tint, VBuff_id, and Verbose.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), MenuLoop(), and packetdecode().

5.22.3.4 void printsensors (char * *buf*)

Prints in a buffer the status of the sensors: if the sensor is on it prints the 'A' character, else a '.' .

Parameters:

buf the buffer to print to

Returns:

no value.

Definition at line 369 of file util104.c.

References Analog.

Referenced by updatemenu().

5.22.3.5 void printlink (char * *buf*)

Prints in a buffer the status of the optical link: if the link is ok it prints the 'A' character, else a '.' .

Parameters:

buf the buffer to print to

Returns:

no value.

Definition at line 397 of file util104.c.

References Analog.

Referenced by updatemenu().

5.22.3.6 void print_alimentation (int *Astatus*, float *clockms*)

Prints in a more comprensible mode the alimentation log.

Parameters:

Astatus the alimentation status word

clockms the time in millisec

Returns:

no value.

Definition at line 424 of file util104.c.

References iolog(), VCC_3V, VCC_5V, VCC_9V, VCC_OPTO, and VCC_SENSOR.

Referenced by read_log().

5.22.3.7 void printfilter (char * *buf*)

Prints in the menu the active filter on the selected board.

Parameters:

buf the buffer to print to

Returns:

no value.

Definition at line 467 of file util104.c.

References Analog, and Channel.

Referenced by updatemenu().

5.22.3.8 int initdev (void)

Performs device initialization.

The order of initialization is:

- Init Lcd display (if enabled)
- Map Isa memory
- Read and identify buffer board
- Reset buffer board and clear buffers
- Check of optical Link
- Memory test on analog boards
- Init of analog voltages
- Read analog ID and status
- Prepare data folder and init data number (if enabled)

Returns:

MYTRUE if all OK

Definition at line 512 of file util104.c.

References analog_boardId(), analog_boardStatus(), Channel, Err104, errlog(), error_code(), execute_cmd(), file_n, GB_EBUFFID, GB_EINTERNAL, gb_errno, GB_ERROR, GB_WARNING, gid, init_power(), INITDEV_TASK, iolog(), l_opera(), lcd_init(), Lcd_present, link_status(), map_isa_ram(), Menu, MYFALSE, MYTRUE, NOERR, Opera, program_DAC_Vlevel(), RD_IDENT, reset_fpga(), rmem104(), test_seq_mem(), Testmode, uid, VBuff_id, whichfilter(), WR_OFF1_2, WR_OFF3_4, WR_VBIAS, and WR_VRESET.

Referenced by main(), and reinitdev().

5.22.3.9 int reinitdev (void)

Re-initializes server104.

Returns:

MYTRUE on success, otherwise MYFALSE/MYERR

Definition at line 652 of file util104.c.

References dump_init_file(), initdev(), off_power(), read_init_file(), and unmap_isa_ram().

Referenced by packetdecode().

5.22.3.10 void finish (int *sig*)

end of all routine. Close all device and disallocate all, then exit.

Parameters:

sig the exit status to be trasmitted.

Definition at line 674 of file util104.c.

References dump_init_file(), finish_win(), logmsg, logname, mempool, Menu, off_power(), and unmap_isa_ram().

Referenced by catch_signals(), dlanalog(), dlkey(), dlmsg(), initwin(), main(), and MenuLoop().

5.22.3.11 int error_code (int *errtype*, int *process*, int *error*)

Builds the error code using info about the running operation and the code of the error.

Parameters:

errtype flag to signal if it is an error or a warning

process the running oepration originating of the error

error the error code

Returns:

the complessive error code

See also:

gerrno.h

Definition at line 772 of file util104.c.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardStatus(), check_data(), check_group(), compute_cycle(), compute_pix_readclk(), disable_sensor(), dummy_acquisition(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_seq_mem(), fifo_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init_hameg(), init_power(), initdev(), l_get_port(), l_init_port(), l_position(), l_w_ch_pos(), l_w_string(), lcd_init(), link_error(), link_status(), main(), multiswitch(), off_power(), open_serial(), packetdecode(), print_fits_error(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), read_converters(), read_image(), read_init_file(), read_log(), read_pk_param(), read_power(), read_quadrant(), read_testimage(), restart_daemon(),

rmem104(), sawtooth_dac_test(), send_image(), sensor_onoff(), sequencer_counter(), sequencer_status(), serial_command(), serial_decode(), setowner(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_-program(), sys_running_status(), test_board_id(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), write_pk_param(), and zero_seq_mem().

5.22.3.12 char* strerror_104 (int *_gb_errno*)

Generate the human readable error message from its argument.

Parameters:

_gb_errno,: the error code

Returns:

the string corresponding to the error number of the argument.

Definition at line 793 of file util104.c.

References GB_BADADDR, GB_CHKSUM_ERR, GB_EBADARG, GB_EBUFFID, GB_-EFDATA, GB_EFHEADER, GB_EFILEOPEN, GB_EFOVERRUN, GB_EHEADDESYNC, GB_-EIMGSYNC, GB_EINTERNAL, GB_EINVALMASK, GB_EINVALMEM, GB_ELAMP, GB_ELINK, GB_ELINKVB, GB_EMEMALLOC, GB_EMMEMIO, GB_ENOLINK, GB_ENVALBOARD, GB_ENVALOPT, GB_EPIXMORE, GB_ESEQPROG, GB_ESEQVERIFY, GB_ESLIST, GB_-ESPROGFORMAT, GB_ESPROGLONG, GB_ESPROGSHORT, GB_ESYSNOTOP, GB_ETIMEOUT, GB_ETRLINKVB, GB_IO_CLOSED, GB_IO_CONNECT_ERR, GB_IO_EOF, GB_IO_NONBLOCK_-ERR, GB_IO_POLLERR, GB_IO_POLLHUP, GB_IO_POLLNVAL, GB_IO_TIME_CONNECT, GB_IO_TIME_READ, GB_IO_TIME_WRITE, GB_NOMMAPMEM, and GB_PROT_EFORMAT.

Referenced by MainLoop(), and set_descriptor_flags().

5.22.3.13 void syslog_message (int *priority*, char * *fmt*, ...)

Opens a connection to the system logger for the program and generates a log message, which will be distributed by syslogd.

The priority argument is formed by ORing the facility and the level values (LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR LOG_WARNING, LOG_NOTICE, LOG_INFO, LOG_DEBUG)

Parameters:

priority priority of the message

fmt the real message

Returns:

no value

Definition at line 904 of file util104.c.

References facility.

Referenced by main(), open_log_file(), restart_daemon(), and stop_daemon().

5.22.3.14 char* create_filepath (char * *destdir*, char * *optdir*, char * *name*)

Checks the existence of the destination directory. If it does not exist or it hasn't the right permissions, it builds the file pathname using the optional directory name. Otherwise it uses the destination directory to build the filename.

Parameters:

destdir the base directory where the file has to be created

optdir the alternative base directory

name the name of the file to be created

Returns:

the complete file pathname on success, NULL on errors.

Definition at line 932 of file util104.c.

References GB_EMEMALLOC, gb_errno, and strdup_printf().

Referenced by open_log_file().

5.22.3.15 int open_log_file (void)

If logging is active, opens the log file

The logfile will be located in the the directory PACKAGE_LOG_DIR (/var/log/softir) if it exists and has the right permission. Otherwise the logfile will be opened in the PACKAGE_TEMP_DIR (/var/tmp) directory.

Returns:

0 on success, < 0 on errors.

Definition at line 999 of file util104.c.

References create_filepath(), errlog(), G_REVISION, GB_EFILEOPEN, GB_EMEMALLOC, gb_errno, iolog(), logmsg, logname, strdup_printf(), and syslog_message().

Referenced by main(), and restart_daemon().

5.22.3.16 int daemon_is_already_running (void)

Checks if the daemon is already running.

Returns:

MYFALSE if not running, else MYTRUE

Definition at line 1074 of file util104.c.

References iolog(), MYFALSE, MYTRUE, procname(), and Verbose.

Referenced by main(), and MenuLoop().

5.22.3.17 void stop_daemon (int *signal*)

Signal hadler for SIGTERM. It stops the daemon execution setting the global variable Keepgoing to zero.

Parameters:

signal the signal number id

Returns:

no value

Definition at line 1107 of file util104.c.

References iolog(), Keepgoing, and syslog_message().

Referenced by catch_signals().

5.22.3.18 void restart_daemon (int *signal*)

Restart the logging operation of the demon.

If gbridge daemon receives a SIGHUP signal, it responds by reopening the log file, saving the previous one with extension YYYYMMDDHHMMSS. Therefore we use the SIGHUP signal to rotate the log file.

Parameters:

signal the signal number id

Returns:

0 on success, < 0 on errors.

Definition at line 1131 of file util104.c.

References errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), logmsg, logname, open_log_file(), Opera, strdup_printf(), and syslog_message().

Referenced by catch_signals().

5.22.3.19 int catch_signals (void)

Registers the correct handlers for catching signals.

Returns:

0 on success, < 0 on error(s).

Definition at line 1188 of file util104.c.

References finish(), Menu, nop_timeout(), restart_daemon(), and stop_daemon().

Referenced by main().

5.22.3.20 int ignore_signals (void)

Registers the signals to ignore.

Returns:

0 on success, < 0 on error(s).

Definition at line 1236 of file util104.c.

References Menu.

Referenced by main().

5.22.3.21 void dump_packet (char * *func*, int *destination*, int *type*, int *cmd*, int *datalen*, unsigned short *seqnum*, char * *payload*)

Prints the header packet (and eventually the payload) in ASCII format.

Parameters:

func the calling function name

destination the target process of the paket

type the type field of the packet

cmd the command field of the packet

datalen the len of payload of the packet

seqnum the seqnum field of the packet

payload the payload of the packet

Definition at line 1278 of file util104.c.

References ACK, COMANDO, EMBEDSERVER, ERROR, GBSERVER, GUICLIENT, INFO, iolog(), MESSAGGIO, and PRIVEMBED.

Referenced by packetack(), and packetdecode().

5.22.3.22 char* dump_sock_status (int *status*)

Prints the status of the sockets in a human readable format.

Parameters:

status the socket descriptor status: CONNECTED, LISTENING, DISCONNECTED

Returns:

the socket status in string format ("Connected" or "Disconnected")

Definition at line 1362 of file util104.c.

References CONNECTED, and DISCONNECTED.

Referenced by dprint_slist().

5.22.3.23 char* dump_acquisition_status (int *status*)

This routine prints the acquisition status in human understandable format.

Parameters:

status the current status

Returns:

The ASCII string describing the status.

Definition at line 1385 of file util104.c.

References iolog(), SAS_ABORT, SAS_BUSY, SAS_DUMMY, SAS_FREERUN, SAS_IDLE, SAS_-RUNNING, SAS_STOP, and SAS_TEST.

Referenced by handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), integriamo(), and packetdecode().

5.22.3.24 int dump_init_file (void)

This function dumps present status of variable in a file suitable to be read at start-up.

It is meant to preserve system setting between different executions. Variable are dumped in STD_INI_FILE or ALT_INI_FILE.

Returns:

MYTRUE on success

initatialization file pointer

Definition at line 1445 of file util104.c.

References Acquired, ALT_INI_FILE, Analog, Base_scan, Channel, Dit, errlog(), error_code(), GB_-EINTERNAL, gb_errno, GB_WARNING, iolog(), MYFALSE, MYTRUE, Opera, PW_amp, PW_volt, and STD_INI_FILE.

Referenced by finish(), MainLoop(), and reinitdev().

5.22.3.25 int read_init_file (void)

Reads some status variable from a the file in STD_INI_FILE or ALT_INI_FILE.

It is meant to preserve system setting between different executions. It is the companion of dump_init_file. Lines beginning with # or / or ; or * are ignored

Returns:

MYTRUE on success

initatialization file pointer

Definition at line 1528 of file util104.c.

References Acquired, ALT_INI_FILE, Analog, Base_scan, Channel, compute_pix_readclk(), compute_-scan_time(), Dit, errlog(), error_code(), GB_EINTERNAL, gb_errno, GB_WARNING, iolog(), MY-FALSE, MYTRUE, Opera, AnalogBoard::Prog_readclk, AnalogBoard::Prog_readdel, Scan_time, STD_-INI_FILE, Tint, and Verbose.

Referenced by main(), and reinitdev().

5.22.3.26 void multidummy (void)

Produce a minimal multistatus dummy integration pattern. Intented to produce a harmless default

Returns:

MYTRUE on success, MYFALSE otherwise.

Definition at line 1666 of file util104.c.

References compute_scan_time(), multiple::duration, multi_status::integ_number, multi_status::integ_-total, iolog(), multi_status::multic, multi_status::multic_item, multi_status::multiprogram, multistatus, multiple::operation, multiple::original_row, and Scan_time.

Referenced by initvar104(), MenuLoop(), and multifile().

5.22.3.27 int ismultiinteg (char *c*)

Verifies if operation specified in c require an integration in a multisample operation.

Parameters:

c operation to test

Returns:

MYTRUE if true, MYFALSE otherwise.

Definition at line 1707 of file util104.c.

References MYFALSE, and MYTRUE.

Referenced by handle_multi_acquisition(), and multical().

5.22.3.28 void my_bell (void)

ring a terminal bell

Definition at line 1725 of file util104.c.

Referenced by multical().

5.22.3.29 int multical (char *c*, int *flag*)

Switch on and off the appropriate lamp, depending on operation c ad flag flag.

Parameters:

c operation to test

flag flag of on (MYTRUE) or off operation

Returns:

MYTRUE if all OK, MYFALSE otherwise.

Definition at line 1780 of file util104.c.

References DISCONNECTED, giolamp, iolog(), ismultiinteg(), multiswitch(), my_bell(), MYFALSE, MYTRUE, and sock_t::sockstatus.

Referenced by MenuLoop().

5.22.3.30 int multivalid (void)

Validate entries in multistatus.multic structure

Returns:

MYTRUE on success, MYFALSE otherwise.

Definition at line 1820 of file util104.c.

References compute_scan_time(), multiple::duration, errlog(), gb_errno, GB_PROGERR, iolog(), MULTI_VALID, multi_status::multic, multi_status::multic_item, multistatus, MYFALSE, MYTRUE, multiple::operation, multiple::original_row, and Scan_time.

Referenced by MenuLoop(), and multifile().

5.22.3.31 int multifile (char * *multi_program*)

Reads the file whose name is stored in multi_program decode it as a mutiple lectures file and stores results in the global array of structure multistatus If multi_program == "dummy", it call multidummy.

Parameters:

multi_program mutiple lectures program file

Returns:

MYTRUE on success, MYFALSE otherwise.

Definition at line 1858 of file util104.c.

References multiple::duration, errlog(), multi_status::integ_number, multi_status::integ_total, iolog(), multi_status::multic, multi_status::multic_item, multidummy(), multistatus, multivalid(), MYERR, MYFALSE, MYTRUE, multiple::operation, and multiple::original_row.

Referenced by MenuLoop(), and packetdecode().

5.22.4 Variable Documentation

5.22.4.1 int facility = LOG_DAEMON|LOG_USER [static]

to access the syslog facility

Definition at line 130 of file util104.c.

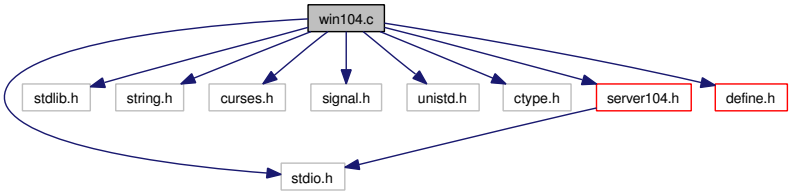
Referenced by syslog_message().

5.23 win104.c File Reference

Windowing routines for server104. Only active in menu mode.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curses.h>
#include <signal.h>
#include <unistd.h>
#include <ctype.h>
#include "server104.h"
#include "define.h"
```

Include dependency graph for win104.c:



Functions

- void initwin (void)
- void finisw_win (void)
- void initmenu (int page)
- void updatemenu (int page)
- void dlkey (char *message)
- void dlmsg (char *message)
- int dlgetc (void)
- void dlnomsg (void)
- int dlascii (char *messaggio, char *risp_default, char *buffer, int max_input)
- int dlint (char *messaggio, int risp_default, int min_value, int max_value, int *value)
- int dlhex (char *messaggio, int risp_default, int min_value, int max_value, int *value)
- int dlanalog (int flag)
- void iolog (int severity, const char *fmt,...)
- void errlog (int errcode, const char *fmt,...)
- void infolog (int cmd,...)
- void finish_win (void)

Variables

- WINDOW * bmsg1
- WINDOW * msg1

5.23.1 Detailed Description

Windowing routines for server104. Only active in menu mode.

Definition in file win104.c.

5.23.2 Function Documentation

5.23.2.1 void initwin (void)

windows initialization

Definition at line 79 of file win104.c.

References finish(), G_REVISION, MYTRUE, and Windowed.

Referenced by main().

5.23.2.2 void finisw_win (void)

Close all windows

Definition at line 184 of file win104.c.

5.23.2.3 void initmenu (int *page*)

menu init and display. May have more than one page, selected by argument page.

Parameters:

page the menu page to display

Definition at line 203 of file win104.c.

References Menu, and updatemenu().

Referenced by MenuLoop().

5.23.2.4 void updatemenu (int *page*)

menu update and display. May have more than one page, selected by argument page.

Parameters:

page the menu page to display

Definition at line 230 of file win104.c.

References Acq_type, Analog, analog_boardStatus(), Base_scan, Channel, Dit, execute_cmd(), Extra_sub, link_status(), multi_status::multiprogram, multistatus, Ngroup, printfilter(), printlink(), printsensors(), Tint, Verbose, and whichfilter().

Referenced by initmenu(), MenuLoop(), and random_action().

5.23.2.5 void dlkey (char * *message*)

Display a message and wait for a key pressed

Parameters:

message the message to display

Definition at line 388 of file win104.c.

References finish().

Referenced by dump_seq_mem().

5.23.2.6 void dlmsg (char * *message*)

Display a message

Parameters:

message the message to display

Definition at line 445 of file win104.c.

References bmsg1, finish(), and msg1.

Referenced by test_board_id().

5.23.2.7 int dlgetc (void)

test for a key pressed

Returns:

0 == no key, else the key

Definition at line 493 of file win104.c.

Referenced by chkabort(), exercise_seq_mem(), random_action(), read_converters(), sawtooth_dac_test(), and test_board_id().

5.23.2.8 void dlnomsg (void)

delete the message previously displayed bu dlmessage

Definition at line 517 of file win104.c.

References bmsg1, and msg1.

Referenced by test_board_id().

5.23.2.9 int dlascii (char * *messaggio*, char * *risp_default*, char * *buffer*, int *max_input*)

Display a message, and wait for a string, with a default

Parameters:

messaggio user message
risp_default default answer string
buffer pointer to the answer
max_input max input lenght

Returns:

lenght of user answer

Definition at line 544 of file win104.c.

References errlog(), and MYERR.

Referenced by MenuLoop().

5.23.2.10 int dlint (char * *messaggio*, int *risp_default*, int *min_value*, int *max_value*, int * *value*)

Display a message, and wait for a integer number, with a default

Parameters:

messaggio user message
risp_default default answer
min_value minimum acceptable value
max_value maximim acceptable value
value pointer to value returned

Returns:

0 on success $<$ 0 if failure

Definition at line 617 of file win104.c.

References errlog(), and MYERR.

Referenced by MenuLoop().

5.23.2.11 int dlhex (char * *messaggio*, int *risp_default*, int *min_value*, int *max_value*, int * *value*)

Display a message, and wait for a integer hex number, with a default

Parameters:

messaggio user message
risp_default default answer
min_value minimum acceptable value
max_value maximim acceptable value
value pointer to value returned

Returns:

0 on success $<$ 0 if failure

Definition at line 704 of file win104.c.

References errlog(), and MYERR.

Referenced by MenuLoop().

5.23.2.12 int dlanalog (int *flag*)

Routine which gets graphical input for VBias and VReset if flag=1 offset_1_2 e offset_3_4 if not.

Parameters:

flag flag selecting offset_1_2 (1) and offset_3_4 (!1)

Returns:

MYTRUE on success

Definition at line 790 of file win104.c.

References Analog, Channel, errlog(), execute_cmd(), finish(), iolog(), MYERR, MYFALSE, MYTRUE, AnalogBoard::Offset12, AnalogBoard::Offset34, program_DAC_Vlevel(), AnalogBoard::V_bias, AnalogBoard::V_reset, WR_OFF1_2, WR_OFF3_4, WR_VBIAS, and WR_VRESET.

Referenced by MenuLoop().

5.23.2.13 void iolog (int *severity*, const char * *fmt*, ...)

Routine which prints a log string in svbio window and in the logfile file

The meaning of the severity code is as follows:

- -1 = log&show but never transmitted on socket
- 0 = just a notice
- 1 = to show to the user
- 2 = to show and to log
- 3 = only to log If Menu != 0 it prints the string in svbio window, otherwise it sends to the command socket.

  If logfile is enabled, iolog also add the log message to logfile

Parameters:

severity the severity of message

fmt the log string

Definition at line 1031 of file win104.c.

References command_sock(), Logfile, logmsg, Menu, MESSAGGIO, MYFALSE, MYTRUE, packetsend_gbridge(), Sendmsg, Verbose, and Windowed.

Referenced by _packetsend(), abort_broadcast(), abort_sequencer(), accept_connection(), check_data(), check_group(), checksum_error(), chkabort(), close_all(), close_listen_socket(), close_socket(), compute_scan_time(), daemon_is_already_running(), disable_sensor(), dlanalog(), dprint_slist(),

dummy_acquisition(), dump_acquisition_status(), dump_data(), dump_init_file(), dump_packet(), dump_-seq_mem(), embedsys_keepalive(), empty_socket(), enable_sensor(), exercise_seq_mem(), fifo_ready(), finish_win(), flush_serial(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_acquisition(), init_hameg(), initdev(), initvar104(), integriamo(), l_error(), l_get_port(), l_init_port(), l_output(), l_position(), l_senddata(), l_stdmessage(), l_w_ch_pos(), l_w_string(), lcd_-init(), link_error(), link_status(), local_server_new(), main(), MainLoop(), mempool_alloc(), mempool_-index(), MenuLoop(), multical(), multidummy(), multifile(), multiswitch(), multivalid(), nop_timeout(), open_giolamp(), open_log_file(), open_serial(), packetdecode(), packetread(), print_alimentation(), printanalog(), printstatus(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), random_-action(), read_converters(), read_image(), read_init_file(), read_log(), read_quadrant(), read_testimage(), remove_nop_timeout(), reprogram_vlevel(), reset_fifo(), reset_fpga(), restart_daemon(), save_data(), sawtooth_dac_test(), send_image(), serial_command(), serial_decode(), set_descriptor_flags(), setowner(), sock_read(), sockerr_func(), start_broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_daemon(), stop_idle(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), tcp_server_accept(), tcp_server_new(), test_board_id(), test_read(), test_seq_mem(), test_write(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), write_pk_param(), write_-seqfile(), and zero_seq_mem().

5.23.2.14 void errlog (int *errcode*, const char * *fmt*, ...)

Routine which prints an error message in svbio window and in the logfile file

If Menu != 0 it prints the string in svbio window, otherwise it sends to the command socket.

If logfile is enabled, eerlog also add the log message to logfile

Parameters:

errcode the machine readable code of the error

fmt the log string

Definition at line 1136 of file win104.c.

References command_sock(), ERROR, Logfile, logmsg, Menu, MYFALSE, MYTRUE, packetsend_-gbridge(), and Windowed.

Referenced by abort_broadcast(), abort_sequencer(), analog_boardStatus(), check_data(), check_-group(), compute_cycle(), compute_pix_readclk(), disable_sensor(), dlanalog(), dlascii(), dlhex(), dlint(), dummy_acquisition(), dummy_image(), dump_init_file(), dump_seq_mem(), enable_sensor(), exercise_-seq_mem(), frame_ready(), handle_image_acquisition(), handle_multi_acquisition(), handle_oneboard_-acquisition(), init_hameg(), init_power(), initdev(), l_get_port(), l_init_port(), l_position(), l_w_ch_-pos(), l_w_string(), lcd_init(), link_error(), link_status(), main(), MenuLoop(), multifile(), multi-switch(), multivalid(), nop_timeout(), off_power(), open_log_file(), open_serial(), packetdecode(), print_-fits_error(), program_DAC_filter(), program_DAC_Vlevel(), program_sequencer(), quadrant_ready(), read_converters(), read_image(), read_init_file(), read_log(), read_pk_param(), read_power(), read_-quadrant(), read_testimage(), restart_daemon(), rmem104(), sawtooth_dac_test(), send_image(), sensor_-onoff(), sequencer_counter(), sequencer_status(), serial_command(), serial_decode(), setowner(), start_-broadcast(), start_sequencer(), stop_broadcast(), stop_CI_broadcast(), stop_CI_sequencer(), stop_kindly(), stop_sequencer(), syntetize_multi(), syntetize_program(), sys_running_status(), test_board_id(), test_-seq_mem(), treset_program(), verify_sequencer(), whichfilter(), wmem104(), write_pk_param(), write_-seqfile(), and zero_seq_mem().

5.23.2.15 void infolog (int *cmd*, ...)

Sends an informative message either to the user window, in menu mode, or to the command socket in daemon mode

Parameters:

cmd the command field of packet

Definition at line 1211 of file win104.c.

References INFO, Menu, and packetsend_gbridge().

Referenced by dummy_image(), handle_image_acquisition(), handle_multi_acquisition(), handle_-oneboard_acquisition(), packetdecode(), read_image(), and read_quadrant().

5.23.2.16 void finish_win (void)

End of all Menu related routine, it disallocateall

Definition at line 1261 of file win104.c.

References iolog().

Referenced by finish().

5.23.3 Variable Documentation

5.23.3.1 WINDOW* bmsg1

border window

Definition at line 67 of file win104.c.

Referenced by dlmsg(), and dlnomsg().

5.23.3.2 WINDOW* msg1

window for message

Definition at line 68 of file win104.c.

Referenced by dlmsg(), and dlnomsg().

Index

_SList, 9
    data, 9
    next, 9
_XOPEN_SOURCE
    lpt104.c, 124
__USE_XOPEN
    lpt104.c, 124
_packetsend
    packetio.c, 159
_rmem104_byte
    mem104.c, 140
_rmem104_word
    mem104.c, 139
_s_FIFO, 7
    delta_time, 7
    event_index, 8
    priority, 8
    time_submitted, 7
    time_to_submit, 7
_wmem104_byte
    mem104.c, 140
_wmem104_word
    mem104.c, 139

ABORT
    command.def, 49
abort_broadcast
    mem104.c, 150
abort_sequencer
    mem104.c, 148
accept_connection
    sock104.c, 192
ACK
    define.h, 66
    lpt104.h, 134
ack_sn
    GPacket, 19
acq104.c
    alloc_array_mem, 33
    check_data, 38
    check_group, 35
    chkabort, 41
    Clipper, 31
    compute_cycle, 36
    compute_pix_readclk, 34
    compute_scan_time, 35
    delay_transfer, 43
    dummy_acquisition, 41
    dummy_image, 43
    dump_data, 39
    fifo_overflow, 33
    fifo_ready, 34
    flag_ovf, 45
    frame_ready, 33
    handle_image_acquisition, 43
    handle_multi_acquisition, 45
    handle_oneboard_acquisition, 41
    integriamo, 43
    MAX, 31
    MAX_ERR, 31
    MIN, 31
    program_sequencer, 38
    quadrant_ready, 34
    random_action, 36
    read_image, 42
    Read_Pix, 31
    read_quadrant, 39
    read_testimage, 39
    reprogram_vlevel, 40
    reset_fifo, 32
    reset_fpga, 32
    stop_CI_broadcast, 32
    syntetize_multi, 44
    syntetize_program, 36
    sys_running_status, 40
    verify_sequencer, 37
acq104.c(2.38), 29
ACQ_TASK
    gerrno.h, 101
Acq_type
    server104.h, 174
Acquired
    server104.h, 177
ADCRAMSIZE
    define.h, 67
add_f_entry
    spooler.c, 206
add_s_entry
    spooler.c, 206
ALIMEN_DEPTH

define.h, 67
alloc_array_mem
acq104.c, 33
ALT_INI_FILE
util104.c, 213
Analog
server104.h, 182
analog_boardId
mem104.c, 142
analog_boardStatus
mem104.c, 142
AnalogBoard, 10
ErrADConv, 13
ErrAddr, 12
ErrAmplifier, 12
ErrAmplifier_, 12
ErrOffset, 12
ErrVlevel, 12
Filtro, 11
Id, 10
Link, 11
Offset12, 11
Offset34, 12
Prog_dit, 14
Prog_dryres, 13
Prog_fsync, 14
Prog_lsync, 14
Prog_readclk, 13
Prog_readdel, 13
Prog_resclk, 13
Prog_resnum, 13
Sensore, 11
seqfile, 13
Sequencer, 11
Status, 10
Time_cycle, 14
V_bias, 11
V_reset, 11
VccOn, 12
arg_1
event_struct, 17
arg_2
event_struct, 17
array
server104.h, 177
array_size
server104.h, 177

BACKLIGHT_OFF
lpt104.h, 132
BACKLIGHT_ON
lpt104.h, 132
BADADDRESS
define.h, 85
BASE_LPT0
lpt104.h, 132
BASE_SCAN
define.h, 83
Base_scan
server104.h, 178
BASE_VAL
define.h, 61
BAUDRATE
serial.c, 162
bigchar
lpt104.h, 135
bignum
lpt104.h, 135
BL
lpt104.h, 135
BLOCK
define.h, 63
bmsg1
win104.c, 231
BOARD_DIT
define.h, 82
BoardStatus
define.h, 86
bot_fifo
s_FIFO, 24
BRD_ADDR_STEP
define.h, 77
BUFLEN
define.h, 60
BUSY
lpt104.h, 134

calculate_checksum
packetio.c, 158
Callback_t
socket.h, 202
canjump
exec104.c, 96
catch_signals
util104.c, 220
CGmode
lpt104.h, 135
Channel
server104.h, 176
charname
serial.c, 166
check_data
acq104.c, 38
check_group
acq104.c, 35
check_socket_status
sock104.c, 189
check_sockettype_status

sock104.c, 189
checksum_error
packetio.c, 158
chkabort
acq104.c, 41
Clipper
acq104.c, 31
close_all
sock104.c, 191
close_embed_sockets
sock104.c, 191
close_giolamp
sock104.c, 192
close_listen_socket
sock104.c, 190
close_serial
serial.c, 162
close_socket
sock104.c, 190
close_sockettype
sock104.c, 191
CLOSED
socket.h, 202
cmd
union_t, 28
cmd_t, 15
hex, 15
name, 15
cmdList
slist.h, 186
COM_INTEGDELAY
define.h, 82
COM_READ
define.h, 81
COM_RESETDRY
define.h, 81
COM_RESETREAD
define.h, 81
COM_RESTART
define.h, 82
COM_SHORTRESET
define.h, 81
COMANDO
define.h, 65
command.def
ABORT, 49
DIT, 47
DOUBLE, 48
DRYRUN, 50
DUMMYACQ, 51
DUMPMEM, 47
EXPERT, 49
FIFOTST, 49
FILLMEM0, 47
FLIPSVB, 50
FREERUN, 50
GROUP, 48
INTEGRA, 50
LOADOBJ, 47
LOADWAVE, 47
MSGLEVEL, 51
MULTI, 50
NOISE, 48
NOP, 51
QUADRANTS, 48
READLOG, 51
READPARM, 47
REINIT, 50
RUN, 49
SEQMEM, 49
SOCKDS9, 50
START, 49
STATUS, 51
STOP, 49
SVBCHECK, 48
SVBTEST, 48
SYNCHRO, 48
VERBOSE, 51
WRITEPARM, 47
command.def(2.4), 46
command_sock
sock104.c, 189
compare_cmd
list104.c, 117
compare_socket
list104.c, 117
compare_socktype
list104.c, 117
CompareFunc
slist.h, 185
compute_cycle
acq104.c, 36
compute_pix_readclk
acq104.c, 34
compute_scan_time
acq104.c, 35
CONNECTED
socket.h, 202
CR
serial.c, 162
create_filepath
util104.c, 218
Curframe
server104.h, 179
Curgroup
server104.h, 180
current_group
server104.h, 180

Cycle_status
server104.h, 181
Cycle_time
server104.h, 181

D_SIGNED
define.h, 86
D_UNSIGNED
define.h, 86
D_LSYNC
define.h, 84
D_lsync
server104.h, 179
daemon_is_already_running
util104.c, 219
data
_SList, 9
union_t, 28
data104.c
print_fits_error, 52
save_data, 53
write_primary_DU, 53
write_primary_HU, 53
data104.c(2.4), 52
DataType
define.h, 86
DEFAULT_TINT
define.h, 62
define.h
ACK, 66
ADCRAMSIZE, 67
ALIMEN_DEPTH, 67
BADADDRESS, 85
BASE_SCAN, 83
BASE_VAL, 61
BLOCK, 63
BOARD_DIT, 82
BoardStatus, 86
BRD_ADDR_STEP, 77
BUFLEN, 60
COM_INTEGDELAY, 82
COM_READ, 81
COM_RESETDRY, 81
COM_RESETREAD, 81
COM_RESTART, 82
COM_SHORTRESET, 81
COMANDO, 65
D_SIGNED, 86
D_UNSIGNED, 86
D_LSYNC, 84
DataType, 86
DEFAULT_TINT, 62
EMBEDDED_CMD_PORT, 62
EMBEDDED_DATA_PORT, 62
EMBEDSERVER, 66
ENDFRAME, 84
ERR_ADCONV, 80
ERR_ADDR, 80
ERR_AMPL, 81
ERR_AMPL_, 81
ERR_OFFSET, 80
ERR_VLEVEL, 80
ERROR, 66
ETH_TRANS_TIME, 62
FIFO_LSW_DAT, 77
FIFO_MSW_DAT, 77
FIFO_ORDIGIT, 77
FIFO_OVERFLOW, 77
FIFO_READY, 77
FILTER_ACTIVE, 78
FRAME_ABORT, 86
FRAME_FINISHED, 86
FRAME_INTEG, 86
FRAME_MULTI, 86
FRAME_READY, 86
FRAME_STARTED, 86
FRAME_STOP, 86
FSYNC_LSYNC, 83
GB_ERRMASK, 81
GBSERVER, 66
GENERIC, 85
GUICLIENT, 66
H_COL, 60
H_ROW, 60
HAMEG_SERIAL, 62
IDLE_ACQ, 86
IDLE_RUN, 86
IDLE_STOP, 86
IdleState, 85
INFO, 65
INTEGRATING, 78
ISA_BASE, 67
ISA_START, 67
ISA_WINSIZE, 67
KEEPALIVE_INTVL, 62
LETTURA_PIX, 83
LINK_ERROR, 79
LINK_ERROR_RATE, 79
LINK_STATUS, 78
LINK_VB_ERR, 79
LINK_VB_TRERR, 79
LSYNC_VCLK, 83
MAGICMASK, 66
MESSAGGIO, 65
MIN_ARRAY_SCANTIME, 65
MULTI_VALID, 65
MYERR, 64
MYFALSE, 64

MYNOTREADY, 65
MYTRUE, 64
N_COL, 60
N_ROW, 60
NBOARD, 60
NEXTRA, 61
NGROUP_FREERUN, 63
NGROUP_MAX, 63
NHEADER, 61
NLOOSE, 61
NOERR, 85
NOMEMORYMAPPED, 85
NON_BLOCK, 63
NOP_IDLE, 85
NOP_SENT, 85
NopStatus, 85
NUM_DRYRES, 82
NUM_LETTURE, 83
NUM_SHORTRES, 82
PIX_LETTURA, 83
PIXEL_WIDTH, 83
POLLFDN, 63
PRIVEMBED, 66
PROG_DIT, 82
RD_ANALOGID, 76
RD_BOARD, 73
RD_DAT_FIFOA, 72
RD_DAT_FIFOB, 72
RD_DAT_FIFOC, 72
RD_DAT_FIFOD, 72
RD_DELAY_0, 69
RD_FIFO_STAT, 67
RD_FRM_CNTA, 71
RD_FRM_CNTB, 71
RD_FRM_CNTC, 71
RD_FRM_CNTD, 71
RD_IDENT, 67
RD_KPV_LINKA, 68
RD_KPV_LINKB, 68
RD_KPV_LINKC, 68
RD_KPV_LINKD, 69
RD_LOG_FIFO, 74
RD_LOG_STAT, 74
RD_LSW_FIFOA, 70
RD_LSW_FIFOB, 70
RD_LSW_FIFOC, 70
RD_LSW_FIFOD, 70
RD_MSW_FIFOA, 71
RD_MSW_FIFOB, 71
RD_MSW_FIFOC, 71
RD_MSW_FIFOD, 71
RD_NBCONV, 75
RD_PCOUNTER, 74
RD_RAM_SEQ, 73
READ_FROM_104, 62
READING_PIXEL, 78
REG_OFFSET, 78
REMOTE_HOST, 61
REMOTE_USER, 61
RESET_SCAN, 84
S104AState, 85
S104Info, 86
S104MemErr, 85
SAS_ABORT, 85
SAS_BUSY, 85
SAS_DUMMY, 85
SAS_ERR, 85
SAS_FREERUN, 85
SAS_IDLE, 85
SAS_MULTI, 85
SAS_RUNNING, 85
SAS_STOP, 85
SAS_TEST, 85
SCAN_LSYNC, 84
SENSOR_STATUS, 77, 78
SEQUENCER_RUN, 78
SOCK_TIMEOUT, 63
ST_DUMMY, 86
ST_IDLE, 86
ST_INTEG, 86
SYS_TIMEOUT, 63
TIMEOUT_READ, 63
TRESET, 82
VCC_3V, 80
VCC_5V, 79
VCC_9V, 79
VCC_ON, 80
VCC_OPTO, 79
VCC_SENSOR, 80
VCLK_CLK1, 84
VCLK_RESET, 84
VCLK_SCAN, 84
WR_BROADCAST, 73
WR_FILTER1, 76
WR_FILTER2, 76
WR_FSYNC_TIM, 76
WR_KPV_BRD, 68
WR_KPV_LINKA, 68
WR_KPV_LINKB, 68
WR_KPV_LINKC, 68
WR_KPV_LINKD, 68
WR_LSYNC_TIM, 76
WR_OFF1_2, 75
WR_OFF3_4, 75
WR_RAM_SEQ, 72
WR_RAM_SEQA, 72
WR_RAM_SEQB, 72
WR_RAM_SEQC, 73

WR_RAM_SEQD, 73
WR_RESCLK, 76
WR_RST_BRD, 70
WR_RST_FIFOA, 69
WR_RST_FIFOB, 70
WR_RST_FIFOC, 70
WR_RST_FIFOD, 70
WR_RST_LOG, 74
WR_SENSOFF, 75
WR_SENSON, 75
WR_SEQABORT, 74
WR_SEQSTART, 75
WR_SEQSTOP, 75
WR_STOP_CI, 76
WR_TST_BRD, 69
WR_TST_FIFOA, 69
WR_TST_FIFOB, 69
WR_TST_FIFOC, 69
WR_TST_FIFOD, 69
WR_VBIAS, 74
WR_VRESET, 74
define.h(2.12), 55
delay_transfer
acq104.c, 43
delta_time
_s_FIFO, 7
disable_sensor
mem104.c, 151
DISCONNECTED
socket.h, 202
DISK_FLUSH
spooler.c, 205
DIT
command.def, 47
Dit
server104.h, 178
dlanalog
win104.c, 229
dlascii
win104.c, 227
dlgetc
win104.c, 227
dlhex
win104.c, 228
dlint
win104.c, 228
dlkey
win104.c, 226
dlmsg
win104.c, 227
dlnomsg
win104.c, 227
DOUBLE
command.def, 48
dprint_slist
list104.c, 122
DRYRUN
command.def, 50
dummy_acquisition
acq104.c, 41
dummy_image
acq104.c, 43
DUMMYACQ
command.def, 51
dump_acquisition_status
util104.c, 221
dump_data
acq104.c, 39
dump_init_file
util104.c, 222
dump_packet
util104.c, 221
dump_seq_mem
mem104.c, 144
dump_sock_status
util104.c, 221
DUMPMEM
command.def, 47
duration
multiple, 22

elapsed_time
socket.c, 195
EMBED_CMD
socket.h, 202
EMBED_DATA
socket.h, 202
EMBED_TS
socket.h, 202
EMBEDDED_CMD_PORT
define.h, 62
EMBEDDED_DATA_PORT
define.h, 62
EMBEDSERVER
define.h, 66
embedsys_keepalive
exec104.c, 95
embedsys_keepalive_old
exec104.c, 96
empty_events_h
spooler.c, 206
empty_fifo
spooler.c, 206
empty_socket
sock104.c, 190
EN1
lpt104.h, 134
EN2

lpt104.h, 134
EN3
lpt104.h, 134
enable_sensor
mem104.c, 151
ENBI
lpt104.h, 133
ENDFRAME
define.h, 84
ENIRQ
lpt104.h, 133
Err104
server104.h, 174
ERR_ADCONV
define.h, 80
ERR_ADDR
define.h, 80
ERR_AMPL
define.h, 81
ERR_AMPL_
define.h, 81
err_max
errTIM, 16
err_min
errTIM, 16
ERR_OFFSET
define.h, 80
err_time
errTIM, 16
ERR_VLEVEL
define.h, 80
ErrADConv
AnalogBoard, 13
ErrAddr
AnalogBoard, 12
ErrAmplifier
AnalogBoard, 12
ErrAmplifier_
AnalogBoard, 12
errfunc
sock_t, 27
ERRLOG
spooler.c, 205
errlog
win104.c, 230
ErrOffset
AnalogBoard, 12
ERROR
define.h, 66
error_code
util104.c, 217
errTIM, 16
err_max, 16
err_min, 16
err_time, 16
last_clr, 16
errtim
spooler.c, 208
ErrVlevel
AnalogBoard, 12
ETH_TRANS_TIME
define.h, 62
event_index
_s_FIFO, 8
event_struct, 17
arg_1, 17
arg_2, 17
max_duration, 18
min_duration, 18
msg, 17
operation, 17
priority, 17
time_submitted, 17
event_void
spooler.c, 208
events_h
spooler.c, 208
exec104.c
canjump, 96
embedsys_keepalive, 95
embedsys_keepalive_old, 96
LOGFILETIMEOUT, 88
main_loop, 96
MainLoop, 96
MAX, 88
MenuLoop, 89
nop_timeout, 95
packetack, 91
packetack_gbridge, 92
packetack_guilab, 93
packetdecode, 93
packetread_gbridge, 91
packetsend, 91
packetsend_gbridge, 92
packetsend_guilab, 93
pfds, 96
POLLMS, 88
read_pk_param, 90
remove_disconnected_sources, 95
remove_embed_sources, 95
remove_event_source, 94
remove_event_sourcetype, 94
remove_nop_timeout, 95
send_image, 93
update_event_sources, 94
write_pk_param, 90
write_seqfile, 89
exec104.c(2.27), 87

execute_cmd
mem104.c, 154
exercise_seq_mem
mem104.c, 145
EXPERT
command.def, 49
ExpertMode
server104.h, 181
Extra_sub
server104.h, 174

f_sched
spooler.c, 207
facility
util104.c, 224
FAULT
lpt104.h, 133
fd
mem104.c, 156
fifo
s_FIFO, 24
FIFO_LSW_DAT
define.h, 77
FIFO_MASK
spooler.c, 205
FIFO_MSW_DAT
define.h, 77
FIFO_ORDIGIT
define.h, 77
FIFO_OVERFLOW
define.h, 77
fifo_overflow
acq104.c, 33
FIFO_READY
define.h, 77
fifo_ready
acq104.c, 34
FIFO_WIDTH
spooler.c, 205
FIFOTST
command.def, 49
file_n
server104.h, 180
FILLMEM0
command.def, 47
FILTER_ACTIVE
define.h, 78
Filtro
AnalogBoard, 11
finish
util104.c, 217
finish_win
win104.c, 231
finisw_win
win104.c, 226
flag_ovf
acq104.c, 45
FLIPSVB
command.def, 50
flush_serial
serial.c, 163
FRAME_ABORT
define.h, 86
FRAME_FINISHED
define.h, 86
FRAME_INTEG
define.h, 86
FRAME_MULTI
define.h, 86
FRAME_READY
define.h, 86
FRAME_STARTED
define.h, 86
FRAME_STOP
define.h, 86
frame_ready
acq104.c, 33
FREERUN
command.def, 50
FSYNC_LSYNC
define.h, 83

G_REVISION
gversion.h, 114
G_SOURCES
gversion.h, 114
GB_ACQ_PROT_ERR
gerrno.h, 107
GB_ACQ_TIMEOUT
gerrno.h, 108
GB_BADADDR
gerrno.h, 102
GB_CFTISIO_ERR
gerrno.h, 108
GB_CHKSUM_ERR
gerrno.h, 109, 110
GB_CMD_NOTACK
gerrno.h, 109, 110
GB_DS9_NOXPA
gerrno.h, 109
GB_EADCONV
gerrno.h, 105
GB_EBADARG
gerrno.h, 102
GB_EBUFFID
gerrno.h, 104
GB_EBUFFVERIFY
gerrno.h, 104

GB_ECOMMEMBED
gerrno.h, 112
GB_EDECODER
gerrno.h, 105
GB_EFDATA
gerrno.h, 107
GB_EFHEADER
gerrno.h, 107
GB_EFILEOPEN
gerrno.h, 108
GB_EFOVERRUN
gerrno.h, 107
GB_EHAMEG
gerrno.h, 106
GB_EHEADDESYNC
gerrno.h, 106
GB_EIDLERUN
gerrno.h, 106
GB_EIMGSYNC
gerrno.h, 106
GB_EINTERNAL
gerrno.h, 108
GB_EINVALMASK
gerrno.h, 110
GB_EINVALMEM
gerrno.h, 102
GB_ELAMP
gerrno.h, 103
GB_ELINK
gerrno.h, 103
GB_ELINKVB
gerrno.h, 104
GB_EMEMALLOC
gerrno.h, 102
GB_EMMEMIO
gerrno.h, 101
GB_ENOLINK
gerrno.h, 104
GB_ENOSENSOR
gerrno.h, 105
GB_ENOVCC
gerrno.h, 105
GB_ENVALBOARD
gerrno.h, 102
GB_ENVALOPT
gerrno.h, 108
GB_EPCKPROTOCOL
gerrno.h, 110
GB_EPIXLESS
gerrno.h, 106
GB_EPIXMORE
gerrno.h, 106
GB_ERRMASK
define.h, 81
gb_errno
gerrno.h, 113
server104.h, 176
GB_ERROR
gerrno.h, 100
GB_ESAVEIMG
gerrno.h, 109
GB_ESENSNOTOP
gerrno.h, 105
GB_ESEQNOTOP
gerrno.h, 104
GB_ESEQPROG
gerrno.h, 103
GB_ESEQRUNNING
gerrno.h, 105
GB_ESEQVERIFY
gerrno.h, 103
GB_ESLIST
gerrno.h, 109
GB_ESPROGFORMAT
gerrno.h, 103
GB_ESPROGLONG
gerrno.h, 103
GB_ESPROGSHORT
gerrno.h, 103
GB_ESYSNOTOP
gerrno.h, 104
GB_ETIMEOUT
gerrno.h, 107
GB_ETRLINKVB
gerrno.h, 104
GB_EVLEVEL
gerrno.h, 105
GB_FITGB_EPCK0
gerrno.h, 110, 111
GB_FITGB_EPCK1
gerrno.h, 110, 111
GB_FITGB_EPCK2
gerrno.h, 110, 111
GB_FITS_KEY_EIO
gerrno.h, 108
GB_GBRIDGE_KILL
gerrno.h, 108
GB_IO_CLOSED
gerrno.h, 112
GB_IO_CONNECT_ERR
gerrno.h, 112
GB_IO_EBADFD
gerrno.h, 111
GB_IO_EOF
gerrno.h, 111
GB_IO_NONBLOCK_ERR
gerrno.h, 112
GB_IO_POLLERR

gerrno.h, 112
GB_IO_POLLHUP
gerrno.h, 112
GB_IO_POLLNVAL
gerrno.h, 112
GB_IO_TIME_CONNECT
gerrno.h, 111
GB_IO_TIME_READ
gerrno.h, 111
GB_IO_TIME_WRITE
gerrno.h, 111
GB_MATCH_ROW
gerrno.h, 107
GB_NOMMAPMEM
gerrno.h, 102
GB_NOP_TIMEOUT
gerrno.h, 112
GB_NSPACE_INVAL
gerrno.h, 108
GB_PROGERR
gerrno.h, 106
GB_PROT_EFORMAT
gerrno.h, 109, 110
GB_PROT_ERR
gerrno.h, 109, 110
GB_RANGE_ROW
gerrno.h, 107
gb_stpcpy
string104.c, 210
gb_strconcat
string104.c, 210
gb_strndup
string104.c, 210
gb_strrstr
string104.c, 211
GB_WARNING
gerrno.h, 100
GBSERVER
define.h, 66
GENERIC
define.h, 85
gerrno.h
ACQ_TASK, 101
GB_ACQ_PROT_ERR, 107
GB_ACQ_TIMEOUT, 108
GB_BADADDR, 102
GB_CFTISIO_ERR, 108
GB_CHKSUM_ERR, 109, 110
GB_CMD_NOTACK, 109, 110
GB_DS9_NOXPA, 109
GB_EADCONV, 105
GB_EBADARG, 102
GB_EBUFFID, 104
GB_EBUFFVERIFY, 104
GB_ECOMMEMBED, 112
GB_EDECODER, 105
GB_EFDATA, 107
GB_EFHEADER, 107
GB_EFILEOPEN, 108
GB_EFOVERRUN, 107
GB_EHAMEG, 106
GB_EHEADDESYNC, 106
GB_EIDLERUN, 106
GB_EIMGSYNC, 106
GB_EINTERNAL, 108
GB_EINVALMASK, 110
GB_EINVALMEM, 102
GB_ELAMP, 103
GB_ELINK, 103
GB_ELINKVB, 104
GB_EMEMALLOC, 102
GB_EMMEMIO, 101
GB_ENOLINK, 104
GB_ENOSENSOR, 105
GB_ENOVCC, 105
GB_ENVALBOARD, 102
GB_ENVALOPT, 108
GB_EPCKPROTOCOL, 110
GB_EPIXLESS, 106
GB_EPIXMORE, 106
gb_errno, 113
GB_ERROR, 100
GB_ESAVEIMG, 109
GB_ESENSNOTOP, 105
GB_ESEQNOTOP, 104
GB_ESEQPROG, 103
GB_ESEQRUNNING, 105
GB_ESEQVERIFY, 103
GB_ESLIST, 109
GB_ESPROGFORMAT, 103
GB_ESPROGLONG, 103
GB_ESPROGSHORT, 103
GB_ESYSNOTOP, 104
GB_ETIMEOUT, 107
GB_ETRLINKVB, 104
GB_EVLEVEL, 105
GB_FITGB_EPCK0, 110, 111
GB_FITGB_EPCK1, 110, 111
GB_FITGB_EPCK2, 110, 111
GB_FITS_KEY_EIO, 108
GB_GBRIDGE_KILL, 108
GB_IO_CLOSED, 112
GB_IO_CONNECT_ERR, 112
GB_IO_EBADFD, 111
GB_IO_EOF, 111
GB_IO_NONBLOCK_ERR, 112
GB_IO_POLLERR, 112
GB_IO_POLLHUP, 112

GB_IO_POLLNVAL, 112
GB_IO_TIME_CONNECT, 111
GB_IO_TIME_READ, 111
GB_IO_TIME_WRITE, 111
GB_MATCH_ROW, 107
GB_NOMMAPMEM, 102
GB_NOP_TIMEOUT, 112
GB_NSPACE_INVAL, 108
GB_PROGERR, 106
GB_PROT_EFORMAT, 109, 110
GB_PROT_ERR, 109, 110
GB_RANGE_ROW, 107
GB_WARNING, 100
INITDEV_TASK, 100
INTERNAL_TASK, 101
PROGRAM_TASK, 101
PROTOCOL_TASK, 101
SOCKETIO_TASK, 101
TEST_TASK, 101
gerrno.h(2.7), 98
get_current_time
socket.c, 195
get_socket_descriptor
sock104.c, 188
gid
server104.c, 170
server104.h, 183
giolamp
server104.h, 183
sock104.c, 193
GPacket, 19
ack_sn, 19
header, 19
payload, 19
pck_sn, 19
GROUP
command.def, 48
GUICLIENT
define.h, 66
GVERSION
gversion.h, 114
gversion.h, 114
G_REVISION, 114
G_SOURCES, 114
GVERSION, 114

H_COL
define.h, 60
H_ROW
define.h, 60
Hameg_fd
serial.c, 166
HAMEG_SERIAL
define.h, 62
handle_image_acquisition
acq104.c, 43
handle_multi_acquisition
acq104.c, 45
handle_oneboard_acquisition
acq104.c, 41
hbar
lpt104.h, 135
HDR_NUM
socket.h, 201
HDR_SIZE
socket.h, 201
header
GPacket, 19
hex
cmd_t, 15
hi_prio
spooler.c, 208
HIGH_PRI
spooler.c, 205

Id
AnalogBoard, 10
IDLE_ACQ
define.h, 86
IDLE_RUN
define.h, 86
IDLE_STOP
define.h, 86
IdleState
define.h, 85
IdleStatus
server104.h, 181
ignore_signals
util104.c, 220
IMG_HEADER
socket.h, 201
index
sock_t, 26
INFO
define.h, 65
infolog
win104.c, 230
INIT
lpt104.h, 132
init_hameg
serial.c, 164
init_power
serial.c, 165
initdev
util104.c, 216
INITDEV_TASK
gerrno.h, 100
initmenu

win104.c, 226
initvar104
server104.c, 168
initwin
win104.c, 226
INMASK
lpt104.h, 134
integ_number
multi_status, 20
integ_total
multi_status, 20
INTEGRA
command.def, 50
INTEGRATING
define.h, 78
integriamo
acq104.c, 43
INTERNAL_TASK
gerrno.h, 101
ioevents
sock_t, 27
iofunc
sock_t, 27
iolog
win104.c, 229
IRQ
lpt104.h, 134
ISA_BASE
define.h, 67
isa_ram
mem104.c, 156
ISA_START
define.h, 67
ISA_WINSIZE
define.h, 67
isfull_fifo
spooler.c, 206
ismultiinteg
util104.c, 223
isvoid_fifo
spooler.c, 206

KEEP_ALIVE
spooler.c, 205
KEEPALIVE_INTVL
define.h, 62
Keepgoing
server104.h, 173
keyList
slist.h, 186

l_clear
lpt104.c, 126
l_error
lpt104.c, 128
l_function
lpt104.c, 128
l_get_port
lpt104.c, 124
l_header
lpt104.c, 127
l_height
lpt104.c, 129
l_home
lpt104.c, 127
l_init_port
lpt104.c, 125
l_opera
lpt104.c, 128
l_operation
lpt104.c, 128
l_output
lpt104.c, 125
l_port
lpt104.c, 129
l_position
lpt104.c, 125
l_senddata
lpt104.c, 124
l_stdmessage
lpt104.c, 127
l_w_ch_pos
lpt104.c, 126
l_w_string
lpt104.c, 126
l_width
lpt104.c, 129
l_x
lpt104.c, 129
l_y
lpt104.c, 129
last_clr
errTIM, 16
LCD_CLR
spooler.c, 205
LCD_ERR
spooler.c, 205
LCD_DEFAULT_CELLHEIGHT
lpt104.h, 132
LCD_DEFAULT_CELLWIDTH
lpt104.h, 131
LCD_DEFAULT_HEIGHT
lpt104.h, 131
LCD_DEFAULT_WIDTH
lpt104.h, 131
lcd_init
lpt104.c, 127
Lcd_present

server104.h, 174
LE
lpt104.h, 135
LETTURA_PIX
define.h, 83
LF
lpt104.h, 132
Link
AnalogBoard, 11
LINK_ERROR
define.h, 79
link_error
mem104.c, 155
LINK_ERROR_RATE
define.h, 79
LINK_STATUS
define.h, 78
link_status
mem104.c, 153
LINK_VB_ERR
define.h, 79
LINK_VB_TRERR
define.h, 79
list104.c
compare_cmd, 117
compare_socket, 117
compare_socktype, 117
dprint_slist, 122
mempool_alloc, 121
mempool_index, 120
slist_append, 119
slist_data_from_command, 122
slist_data_from_descriptor, 121
slist_data_from_socktype, 121
slist_data_from_value, 120
slist_delete_node, 118
slist_find_custom, 119
slist_free, 118
slist_length, 119
slist_new, 118
slist_nth_data, 120
slist_remove, 118
slist_remove_disconnected, 121
list104.c(2.2), 116
LISTENING
socket.h, 202
lo_prio
spooler.c, 208
LOADOBJ
command.def, 47
LOADWAVE
command.def, 47
local_server_accept
socket.c, 198
local_server_new
socket.c, 198
Logfile
server104.h, 174
Logfile_age
server104.h, 180
LOGFILETIMEOUT
exec104.c, 88
logmsg
server104.h, 176
logname
server104.h, 176
LOW_PRI
spooler.c, 205
lpt104.c
_XOPEN_SOURCE, 124
__USE_XOPEN, 124
l_clear, 126
l_error, 128
l_function, 128
l_get_port, 124
l_header, 127
l_height, 129
l_home, 127
l_init_port, 125
l_opera, 128
l_operation, 128
l_output, 125
l_port, 129
l_position, 125
l_senddata, 124
l_stdmessage, 127
l_w_ch_pos, 126
l_w_string, 126
l_width, 129
l_x, 129
l_y, 129
lcd_init, 127
lpt104.c(2.5), 123
lpt104.h
ACK, 134
BACKLIGHT_OFF, 132
BACKLIGHT_ON, 132
BASE_LPT0, 132
bigchar, 135
bignum, 135
BL, 135
BUSY, 134
CGmode, 135
EN1, 134
EN2, 134
EN3, 134
ENBI, 133
ENIRQ, 133

FAULT, 133
hbar, 135
INIT, 132
INMASK, 134
IRQ, 134
LCD_DEFAULT_CELLHEIGHT, 132
LCD_DEFAULT_CELLWIDTH, 131
LCD_DEFAULT_HEIGHT, 131
LCD_DEFAULT_WIDTH, 131
LE, 135
LF, 132
nACK, 134
nFAULT, 133
nLF, 132
nSEL, 133
nSTRB, 132
NUM_CCs, 132
OUTMASK, 133
PAPEREND, 133
port_access, 136
port_access_multiple, 136
port_in, 135
port_out, 135
RS, 135
RW, 134
SEL, 133
SELIN, 133
standard, 135
STRB, 132
vbar, 135
lpt104.h(2.4), 130
LSYNC_VCLK
define.h, 83

MAGICMASK
define.h, 66
main
server104.c, 169
main_loop
exec104.c, 96
MainLoop
exec104.c, 96
mainloop_iterate
sock104.c, 191
map_isa_ram
mem104.c, 138
MAX
acq104.c, 31
exec104.c, 88
max_duration
event_struct, 18
MAX_ERR
acq104.c, 31
mem104.c
_rmem104_byte, 140
_rmem104_word, 139
_wmem104_byte, 140
_wmem104_word, 139
abort_broadcast, 150
abort_sequencer, 148
analog_boardId, 142
analog_boardStatus, 142
disable_sensor, 151
dump_seq_mem, 144
enable_sensor, 151
execute_cmd, 154
exercise_seq_mem, 145
fd, 156
isa_ram, 156
link_error, 155
link_status, 153
map_isa_ram, 138
program_DAC_filter, 152
program_DAC_Vlevel, 154
read_converters, 146
read_log, 145
reset_logfifo, 145
rmem104, 140
sawtooth_dac_test, 153
sensor_onoff, 152
sequencer_counter, 147
sequencer_running, 147
sequencer_status, 146
start_broadcast, 149
start_sequencer, 147
stop_broadcast, 149
stop_CI_broadcast, 149
stop_CI_sequencer, 148
stop_idle, 150
stop_kindly, 150
stop_sequencer, 148
test_board_id, 144
test_read, 143
test_seq_mem, 144
test_write, 143
treset_program, 155
unmap_isa_ram, 139
whichfilter, 152
wmem104, 141
zero_seq_mem, 141
mem104.c(2.11), 137
MEM_CHUNK_SIZE
slist.h, 185
MEM_POOL_ITEM
slist.h, 185
mempool
server104.c, 170
server104.h, 177

mempool_alloc
list104.c, 121
mempool_index
list104.c, 120
Menu
server104.h, 173
MenuLoop
exec104.c, 89
MESSAGGIO
define.h, 65
MIN
acq104.c, 31
MIN_ARRAY_SCANTIME
define.h, 65
min_duration
event_struct, 18
msg
event_struct, 17
msg1
win104.c, 231
MSGLEVEL
command.def, 51
MULTI
command.def, 50
multi_status, 20
integ_number, 20
integ_total, 20
multic, 20
multic_item, 20
multiprogram, 20
MULTI_VALID
define.h, 65
multic
multi_status, 20
multic_item
multi_status, 20
multical
util104.c, 223
multidummy
util104.c, 223
multifile
util104.c, 224
multiple, 22
duration, 22
operation, 22
original_row, 22
seq_row, 22
multiprogram
multi_status, 20
multistatus
server104.h, 182
multiswitch
sock104.c, 193
multivalid
util104.c, 224
my_bell
util104.c, 223
MYERR
define.h, 64
MYFALSE
define.h, 64
MYNOTREADY
define.h, 65
MYTRUE
define.h, 64

N_COL
define.h, 60
N_ROW
define.h, 60
nACK
lpt104.h, 134
name
cmd_t, 15
NBOARD
define.h, 60
Ncol
server104.h, 177
next
_SList, 9
NEXTRA
define.h, 61
nFAULT
lpt104.h, 133
Ngroup
server104.h, 179
NGROUP_FREERUN
define.h, 63
NGROUP_MAX
define.h, 63
NHEADER
define.h, 61
nLF
lpt104.h, 132
NLOOSE
define.h, 61
NOERR
define.h, 85
NOISE
command.def, 48
NoiseMode
server104.h, 175
NOMEMORYMAPPED
define.h, 85
NON_BLOCK
define.h, 63
NOP
command.def, 51

NOP_IDLE
define.h, 85
NOP_SENT
define.h, 85
nop_timeout
exec104.c, 95
NopStatus
define.h, 85
nopStatus
server104.h, 181
Nrow
server104.h, 177
nSEL
lpt104.h, 133
nSTRB
lpt104.h, 132
NUM_CCs
lpt104.h, 132
NUM_DRYRES
define.h, 82
NUM_LETTURE
define.h, 83
NUM_SHORTRES
define.h, 82

off_power
serial.c, 165
Offset12
AnalogBoard, 11
Offset34
AnalogBoard, 12
oldtio
serial.c, 166
open_giolamp
sock104.c, 192
open_log_file
util104.c, 219
open_serial
serial.c, 163
OPER_TYPE
spooler.c, 205
Opera
server104.h, 175
operation
event_struct, 17
multiple, 22
optarg
server104.c, 169
opterr
server104.c, 170
optind
server104.c, 169
optopt
server104.c, 170

original_row
multiple, 22
OUTMASK
lpt104.h, 133

packetack
exec104.c, 91
packetack_gbridge
exec104.c, 92
packetack_guilab
exec104.c, 93
packetdecode
exec104.c, 93
packetio.c
_packetsend, 159
calculate_checksum, 158
checksum_error, 158
packetread, 158
packetsend_ack, 159
packetio.c(2.3), 157
packetread
packetio.c, 158
packetread_gbridge
exec104.c, 91
packetsend
exec104.c, 91
packetsend_ack
packetio.c, 159
packetsend_gbridge
exec104.c, 92
packetsend_guilab
exec104.c, 93
PAPEREND
lpt104.h, 133
payload
GPacket, 19
Pck_seqnum
server104.h, 180
PCK_SIZE
socket.h, 201
pck_sn
GPacket, 19
pfds
exec104.c, 96
PIX_LETTURA
define.h, 83
PIXEL_WIDTH
define.h, 83
POLLFDN
define.h, 63
POLLMS
exec104.c, 88
port_access
lpt104.h, 136

port_access_multiple
lpt104.h, 136
port_in
lpt104.h, 135
port_out
lpt104.h, 135
print_alimentation
util104.c, 215
print_fits_error
data104.c, 52
print_help
server104.c, 168
print_named_ascii
serial.c, 162
printanalog
util104.c, 214
printfilter
util104.c, 215
printlink
util104.c, 215
printsensors
util104.c, 214
printstatus
util104.c, 214
priority
_s_FIFO, 8
event_struct, 17
PRIORITY_LEVEL
spooler.c, 205
PRIVEMBED
define.h, 66
procname
util104.c, 214
Prog_age
server104.h, 180
PROG_DIT
define.h, 82
Prog_dit
AnalogBoard, 14
Prog_dryres
AnalogBoard, 13
Prog_fsync
AnalogBoard, 14
Prog_lsync
AnalogBoard, 14
Prog_readclk
AnalogBoard, 13
Prog_readdel
AnalogBoard, 13
Prog_resclk
AnalogBoard, 13
Prog_resnum
AnalogBoard, 13
program_DAC_filter
mem104.c, 152
program_DAC_Vlevel
mem104.c, 154
program_sequencer
acq104.c, 38
PROGRAM_TASK
gerrno.h, 101
PROTOCOL_TASK
gerrno.h, 101
PW_amp
server104.h, 182
PW_stat
server104.h, 182
PW_volt
server104.h, 182

quadrant_ready
acq104.c, 34
QUADRANTS
command.def, 48

R_endframe
server104.h, 178
R_fsync_lsync
server104.h, 178
R_lsync_vclk
server104.h, 179
random_action
acq104.c, 36
RD_ANALOGID
define.h, 76
RD_BOARD
define.h, 73
RD_DAT_FIFOA
define.h, 72
RD_DAT_FIFOB
define.h, 72
RD_DAT_FIFOC
define.h, 72
RD_DAT_FIFOD
define.h, 72
RD_DELAY_0
define.h, 69
RD_FIFO_STAT
define.h, 67
RD_FRM_CNTA
define.h, 71
RD_FRM_CNTB
define.h, 71
RD_FRM_CNTC
define.h, 71
RD_FRM_CNTD
define.h, 71
RD_IDENT

define.h, 67
RD_KPV_LINKA
define.h, 68
RD_KPV_LINKB
define.h, 68
RD_KPV_LINKC
define.h, 68
RD_KPV_LINKD
define.h, 69
RD_LOG_FIFO
define.h, 74
RD_LOG_STAT
define.h, 74
RD_LSW_FIFOA
define.h, 70
RD_LSW_FIFOB
define.h, 70
RD_LSW_FIFOC
define.h, 70
RD_LSW_FIFOD
define.h, 70
RD_MSW_FIFOA
define.h, 71
RD_MSW_FIFOB
define.h, 71
RD_MSW_FIFOC
define.h, 71
RD_MSW_FIFOD
define.h, 71
RD_NBCONV
define.h, 75
RD_PCOUNTER
define.h, 74
RD_RAM_SEQ
define.h, 73
read_converters
mem104.c, 146
READ_FROM_104
define.h, 62
read_image
acq104.c, 42
Read_image_timeout
server104.h, 179
read_init_file
util104.c, 222
read_log
mem104.c, 145
Read_Pix
acq104.c, 31
read_pk_param
exec104.c, 90
read_power
serial.c, 165
read_quadrant
acq104.c, 39
read_testimage
acq104.c, 39
READING_PIXEL
define.h, 78
READLOG
command.def, 51
READPARM
command.def, 47
REG_OFFSET
define.h, 78
REINIT
command.def, 50
reinitdev
util104.c, 216
REMOTE_HOST
define.h, 61
REMOTE_USER
define.h, 61
remove_disconnected_sources
exec104.c, 95
remove_embed_sources
exec104.c, 95
remove_event_source
exec104.c, 94
remove_event_sourcetype
exec104.c, 94
remove_nop_timeout
exec104.c, 95
reprogram_vlevel
acq104.c, 40
resched
spooler.c, 207
reset_fifo
acq104.c, 32
reset_fpga
acq104.c, 32
reset_logfifo
mem104.c, 145
RESET_SCAN
define.h, 84
restart_daemon
util104.c, 220
rmem104
mem104.c, 140
RS
lpt104.h, 135
RUN
command.def, 49
Run
server104.h, 177
RW
lpt104.h, 134

s
sock_addr, 25
S104AState
define.h, 85
S104Info
define.h, 86
S104MemErr
define.h, 85
S104SockStatus
socket.h, 202
S104SockType
socket.h, 202
s_FIFO, 24
bot_fifo, 24
fifo, 24
top_fifo, 24
s_in
sock_addr, 25
s_sched
spooler.c, 207
s_un
sock_addr, 25
s_wrapper
spooler.c, 207
sa
sock_t, 27
SAS_ABORT
define.h, 85
SAS_BUSY
define.h, 85
SAS_DUMMY
define.h, 85
SAS_ERR
define.h, 85
SAS_FREERUN
define.h, 85
SAS_IDLE
define.h, 85
SAS_MULTI
define.h, 85
SAS_RUNNING
define.h, 85
SAS_STOP
define.h, 85
SAS_TEST
define.h, 85
save_data
data104.c, 53
sawtooth_dac_test
mem104.c, 153
SCAN_LSYNC
define.h, 84
Scan_time
server104.h, 179
SEL
lpt104.h, 133
SELIN
lpt104.h, 133
send_image
exec104.c, 93
Sendmsg
server104.h, 174
sensor_onoff
mem104.c, 152
SENSOR_STATUS
define.h, 77, 78
Sensore
AnalogBoard, 11
seq_row
multiple, 22
seqfile
AnalogBoard, 13
SEQMEM
command.def, 49
Sequencer
AnalogBoard, 11
sequencer_counter
mem104.c, 147
SEQUENCER_RUN
define.h, 78
sequencer_running
mem104.c, 147
sequencer_status
mem104.c, 146
serial.c
BAUDRATE, 162
charname, 166
close_serial, 162
CR, 162
flush_serial, 163
Hameg_fd, 166
init_hameg, 164
init_power, 165
off_power, 165
oldtio, 166
open_serial, 163
print_named_ascii, 162
read_power, 165
serial_command, 164
serial_decode, 164
serial_rdline, 163
serial_wrline, 163
serial.c(2.10), 161
serial_command
serial.c, 164
serial_decode
serial.c, 164
serial_fd

server104.h, 181
serial_rdline
serial.c, 163
serial_wrline
serial.c, 163
server104.c
gid, 170
initvar104, 168
main, 169
mempool, 170
optarg, 169
opterr, 170
optind, 169
optopt, 170
print_help, 168
setowner, 168
uid, 170
server104.c(2.8), 167
server104.h
Acq_type, 174
Acquired, 177
Analog, 182
array, 177
array_size, 177
Base_scan, 178
Channel, 176
Curframe, 179
Curgroup, 180
current_group, 180
Cycle_status, 181
Cycle_time, 181
D_lsync, 179
Dit, 178
Err104, 174
ExpertMode, 181
Extra_sub, 174
file_n, 180
gb_errno, 176
gid, 183
giolamp, 183
IdleStatus, 181
Keepgoing, 173
Lcd_present, 174
Logfile, 174
Logfile_age, 180
logmsg, 176
logname, 176
mempool, 177
Menu, 173
multistatus, 182
Ncol, 177
Ngroup, 179
NoiseMode, 175
nopStatus, 181
Nrow, 177
Opera, 175
Pck_seqnum, 180
Prog_age, 180
PW_amp, 182
PW_stat, 182
PW_volt, 182
R_endframe, 178
R_fsync_lsync, 178
R_lsync_vclk, 179
Read_image_timeout, 179
Run, 177
Scan_time, 179
Sendmsg, 174
serial_fd, 181
SVBCheck, 175
SVBTest, 175
SynchroTest, 175
Testmode, 180
timestamp, 181
Tint, 178
Transfer_time, 182
uid, 183
VBuff_id, 174
Verbose, 173
Windowed, 173
server104.h(2.18), 171
set_close_on_exec
socket.c, 195
set_descriptor_flags
socket.c, 196
setowner
server104.c, 168
SList
slist.h, 185
slist
slist.h, 185
slist.h
cmdList, 186
CompareFunc, 185
keyList, 186
MEM_CHUNK_SIZE, 185
MEM_POOL_ITEM, 185
SList, 185
slist, 185
slist.h(2.3), 184
slist_append
list104.c, 119
slist_data_from_command
list104.c, 122
slist_data_from_descriptor
list104.c, 121
slist_data_from_socktype
list104.c, 121

slist_data_from_value
list104.c, 120
slist_delete_node
list104.c, 118
slist_find_custom
list104.c, 119
slist_free
list104.c, 118
slist_length
list104.c, 119
slist_new
list104.c, 118
slist_nth_data
list104.c, 120
slist_remove
list104.c, 118
slist_remove_disconnected
list104.c, 121
sock
union_t, 28
sock104.c
accept_connection, 192
check_socket_status, 189
check_sockettype_status, 189
close_all, 191
close_embed_sockets, 191
close_giolamp, 192
close_listen_socket, 190
close_socket, 190
close_sockettype, 191
command_sock, 189
empty_socket, 190
get_socket_descriptor, 188
giolamp, 193
mainloop_iterate, 191
multiswitch, 193
open_giolamp, 192
sockerr_func, 189
update_event_sources, 188
sock104.c(2.8), 187
sock_addr, 25
s, 25
s_in, 25
s_un, 25
sock_read
socket.c, 197
sock_t, 26
errfunc, 27
index, 26
ioevents, 27
iofunc, 27
sa, 27
sockfd, 26
sockstatus, 26
socktype, 26
SOCK_TIMEOUT
define.h, 63
sock_write
socket.c, 197
SOCKDS9
command.def, 50
sockerr_func
sock104.c, 189
socket.c
elapsed_time, 195
get_current_time, 195
local_server_accept, 198
local_server_new, 198
set_close_on_exec, 195
set_descriptor_flags, 196
sock_read, 197
sock_write, 197
tcp_server_accept, 198
tcp_server_new, 197
tcp_socket_connect, 196
socket.c(2.5), 194
socket.h
Callback_t, 202
CLOSED, 202
CONNECTED, 202
DISCONNECTED, 202
EMBED_CMD, 202
EMBED_DATA, 202
EMBED_TS, 202
HDR_NUM, 201
HDR_SIZE, 201
IMG_HEADER, 201
LISTENING, 202
PCK_SIZE, 201
S104SockStatus, 202
S104SockType, 202
socket.h(2.5), 200
SOCKETIO_TASK
gerrno.h, 101
sockfd
sock_t, 26
sockstatus
sock_t, 26
socktype
sock_t, 26
spooler.c
add_f_entry, 206
add_s_entry, 206
DISK_FLUSH, 205
empty_events_h, 206
empty_fifo, 206
ERRLOG, 205
errtim, 208

event_void, 208
events_h, 208
f_sched, 207
FIFO_MASK, 205
FIFO_WIDTH, 205
hi_prio, 208
HIGH_PRI, 205
isfull_fifo, 206
isvoid_fifo, 206
KEEP_ALIVE, 205
LCD_CLR, 205
LCD_ERR, 205
lo_prio, 208
LOW_PRI, 205
OPER_TYPE, 205
PRIORITY_LEVEL, 205
resched, 207
s_sched, 207
s_wrapper, 207
Verbose, 207
spooler.c(0.12), 203
ST_DUMMY
define.h, 86
ST_IDLE
define.h, 86
ST_INTEG
define.h, 86
standard
lpt104.h, 135
START
command.def, 49
start_broadcast
mem104.c, 149
start_sequencer
mem104.c, 147
STATUS
command.def, 51
Status
AnalogBoard, 10
STD_INI_FILE
util104.c, 213
STOP
command.def, 49
stop_broadcast
mem104.c, 149
stop_CI_broadcast
acq104.c, 32
mem104.c, 149
stop_CI_sequencer
mem104.c, 148
stop_daemon
util104.c, 219
stop_idle
mem104.c, 150
stop_kindly
mem104.c, 150
stop_sequencer
mem104.c, 148
STRB
lpt104.h, 132
strdup_printf
string104.c, 209
strdup_vprintf
string104.c, 211
strerror_104
util104.c, 218
string104.c
gb_stpcpy, 210
gb_strconcat, 210
gb_strndup, 210
gb_strrstr, 211
strdup_printf, 209
strdup_vprintf, 211
string104.c(1.6), 209
SVBCHECK
command.def, 48
SVBCheck
server104.h, 175
SVBTEST
command.def, 48
SVBTest
server104.h, 175
SYNCHRO
command.def, 48
SynchroTest
server104.h, 175
syntetize_multi
acq104.c, 44
syntetize_program
acq104.c, 36
sys_running_status
acq104.c, 40
SYS_TIMEOUT
define.h, 63
syslog_message
util104.c, 218

tcp_server_accept
socket.c, 198
tcp_server_new
socket.c, 197
tcp_socket_connect
socket.c, 196
test_board_id
mem104.c, 144
test_read
mem104.c, 143
test_seq_mem

mem104.c, 144
TEST_TASK
gerrno.h, 101
test_write
mem104.c, 143
Testmode
server104.h, 180
Time_cycle
AnalogBoard, 14
time_submitted
_s_FIFO, 7
event_struct, 17
time_to_submit
_s_FIFO, 7
TIMEOUT_READ
define.h, 63
timestamp
server104.h, 181
Tint
server104.h, 178
top_fifo
s_FIFO, 24
Transfer_time
server104.h, 182
TRESET
define.h, 82
treset_program
mem104.c, 155

uid
server104.c, 170
server104.h, 183
union_t, 28
cmd, 28
data, 28
sock, 28
unmap_isa_ram
mem104.c, 139
update_event_sources
exec104.c, 94
sock104.c, 188
updatemenu
win104.c, 226
util104.c
ALT_INI_FILE, 213
catch_signals, 220
create_filepath, 218
daemon_is_already_running, 219
dump_acquisition_status, 221
dump_init_file, 222
dump_packet, 221
dump_sock_status, 221
error_code, 217
facility, 224
finish, 217
ignore_signals, 220
initdev, 216
ismultiinteg, 223
multical, 223
multidummy, 223
multifile, 224
multivalid, 224
my_bell, 223
open_log_file, 219
print_alimentation, 215
printanalog, 214
printfilter, 215
printlink, 215
printsensors, 214
printstatus, 214
procname, 214
read_init_file, 222
reinitdev, 216
restart_daemon, 220
STD_INI_FILE, 213
stop_daemon, 219
strerror_104, 218
syslog_message, 218
util104.c(2.22), 212

V_bias
AnalogBoard, 11
V_reset
AnalogBoard, 11
vbar
lpt104.h, 135
VBuff_id
server104.h, 174
VCC_3V
define.h, 80
VCC_5V
define.h, 79
VCC_9V
define.h, 79
VCC_ON
define.h, 80
VCC_OPTO
define.h, 79
VCC_SENSOR
define.h, 80
VccOn
AnalogBoard, 12
VCLK_CLK1
define.h, 84
VCLK_RESET
define.h, 84
VCLK_SCAN
define.h, 84

VERBOSE
command.def, 51
Verbose
server104.h, 173
spooler.c, 207
verify_sequencer
acq104.c, 37

whichfilter
mem104.c, 152
win104.c
bmsg1, 231
dlanalog, 229
dlascii, 227
dlgetc, 227
dlhex, 228
dlint, 228
dlkey, 226
dlmsg, 227
dlnomsg, 227
errlog, 230
finish_win, 231
finisw_win, 226
infolog, 230
initmenu, 226
initwin, 226
iolog, 229
msg1, 231
updatemenu, 226
win104.c(2.8), 225
Windowed
server104.h, 173
wmem104
mem104.c, 141
WR_BROADCAST
define.h, 73
WR_FILTER1
define.h, 76
WR_FILTER2
define.h, 76
WR_FSYNC_TIM
define.h, 76
WR_KPV_BRD
define.h, 68
WR_KPV_LINKA
define.h, 68
WR_KPV_LINKB
define.h, 68
WR_KPV_LINKC
define.h, 68
WR_KPV_LINKD
define.h, 68
WR_LSYNC_TIM
define.h, 76
WR_OFF1_2
define.h, 75
WR_OFF3_4
define.h, 75
WR_RAM_SEQ
define.h, 72
WR_RAM_SEQA
define.h, 72
WR_RAM_SEQB
define.h, 72
WR_RAM_SEQC
define.h, 73
WR_RAM_SEQD
define.h, 73
WR_RESCLK
define.h, 76
WR_RST_BRD
define.h, 70
WR_RST_FIFOA
define.h, 69
WR_RST_FIFOB
define.h, 70
WR_RST_FIFOC
define.h, 70
WR_RST_FIFOD
define.h, 70
WR_RST_LOG
define.h, 74
WR_SENSOFF
define.h, 75
WR_SENSON
define.h, 75
WR_SEQABORT
define.h, 74
WR_SEQSTART
define.h, 75
WR_SEQSTOP
define.h, 75
WR_STOP_CI
define.h, 76
WR_TST_BRD
define.h, 69
WR_TST_FIFOA
define.h, 69
WR_TST_FIFOB
define.h, 69
WR_TST_FIFOC
define.h, 69
WR_TST_FIFOD
define.h, 69
WR_VBIAS
define.h, 74
WR_VRESET
define.h, 74

write_pk_param
exec104.c, 90
write_primary_DU
data104.c, 53
write_primary_HU
data104.c, 53
write_seqfile
exec104.c, 89
WRITEPARM
command.def, 47

zero_seq_mem
mem104.c, 141