Spine Robot Controller

v0.7

Generated by Doxygen 1.8.17

Chapter 1

Spine Surgery Robot Control Framework

1.1 Introduction

This is the introduction.

1.2 Installation

1.2.1 1: Opening the box

etc...

Chapter 2

Todo List

Member EthercatCommunication::EthercatNode::SetProfileVelocityParametersAll (ProfileVelocityParam &P)

Add error code to all functions.Instead of returning -1.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 EthercatCommunication Namespace Reference

ROS2 Headers.

Classes

- class EthercatNode

7.1.1 Detailed Description

ROS2 Headers.

7.2 EthercatLifeCycleNode Namespace Reference

Classes

- class EthercatLifeCycle

Chapter 8

Class Documentation

8.1 Controller Struct Reference

Controller Parameters.

```
#include <ecat_globals.hpp>
```

Public Attributes

- float left_x_axis_
- float left_y_axis_
- float right_x_axis_
- float right_y_axis_
- uint8_t blue_button_
- uint8_t green_button_
- uint8_t red_button_
- uint8_t yellow_button_
- uint8_t left_r_button_
- uint8_t left_l_button_
- uint8_t left_u_button_
- uint8_t left_d_button_
- uint8_t left_rb_button_
- uint8_t right_rb_button_
- uint8_t left_start_button_
- uint8_t right_start_button_
- uint8_t xbox_button_

8.1.1 Detailed Description

Controller Parameters.

8.1.2 Member Data Documentation

8.1.2.1 blue_button_

```
uint8_t Controller::blue_button_
```

8.1.2.2 green_button_

```
uint8_t Controller::green_button_
```

8.1.2.3 left_d_button_

```
uint8_t Controller::left_d_button_
```

8.1.2.4 left_l_button_

```
uint8_t Controller::left_l_button_
```

8.1.2.5 left_r_button_

```
uint8_t Controller::left_r_button_
```

8.1.2.6 left_rb_button_

```
uint8_t Controller::left_rb_button_
```

8.1.2.7 left_start_button_

```
uint8_t Controller::left_start_button_
```

8.1.2.8 left_u_button_

```
uint8_t Controller::left_u_button_
```

8.1.2.9 left_x_axis_

```
float Controller::left_x_axis_
```

8.1.2.10 left_y_axis_

```
float Controller::left_y_axis_
```

8.1.2.11 red_button_

```
uint8_t Controller::red_button_
```

8.1.2.12 right_rb_button_

```
uint8_t Controller::right_rb_button_
```

8.1.2.13 right_start_button_

```
uint8_t Controller::right_start_button_
```

8.1.2.14 right_x_axis_

```
float Controller::right_x_axis_
```

8.1.2.15 right_y_axis_

```
float Controller::right_y_axis_
```

8.1.2.16 xbox_button_

```
uint8_t Controller::xbox_button_
```

8.1.2.17 yellow_button_

```
uint8_t Controller::yellow_button_
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.2 CSPositionModeParam Struct Reference

Struct contains configuration parameters for cyclic sync. position mode.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t nominal_current
- uint16_t torque_constant
- uint32_t current_controller_gain
- uint32_t position_control_parameter_set
- uint32_t software_position_limit
- uint16_t motor_rated_torque
- uint32_t max_gear_input_speed
- uint32_t profile_vel
- uint32_t profile_acc
- uint32_t profile_dec
- uint32_t max_fol_err
- uint32_t max_profile_vel
- uint32_t quick_stop_dec
- uint32_t interpolation_time_period

8.2.1 Detailed Description

Struct contains configuration parameters for cyclic sync. position mode.

8.2.2 Member Data Documentation

8.2.2.1 current_controller_gain

```
uint32_t CSPositionModeParam::current_controller_gain
```

8.2.2.2 interpolation_time_period

```
uint32_t CSPositionModeParam::interpolation_time_period
```

8.2.2.3 max_fol_err

```
uint32_t CSPositionModeParam::max_fol_err
```

8.2.2.4 max_gear_input_speed

```
uint32_t CSPositionModeParam::max_gear_input_speed
```

8.2.2.5 max_profile_vel

```
uint32_t CSPositionModeParam::max_profile_vel
```

8.2.2.6 motor_rated_torque

```
uint16_t CSPositionModeParam::motor_rated_torque
```

8.2.2.7 nominal_current

```
uint32_t CSPositionModeParam::nominal_current
```

8.2.2.8 position_control_parameter_set

```
uint32_t CSPositionModeParam::position_control_parameter_set
```

8.2.2.9 profile_acc

```
uint32_t CSPositionModeParam::profile_acc
```

8.2.2.10 profile_dec

```
uint32_t CSPositionModeParam::profile_dec
```

8.2.2.11 profile_vel

```
uint32_t CSPositionModeParam::profile_vel
```

8.2.2.12 quick_stop_dec

```
uint32_t CSPositionModeParam::quick_stop_dec
```

8.2.2.13 software_position_limit

```
uint32_t CSPositionModeParam::software_position_limit
```

8.2.2.14 torque_constant

```
uint16_t CSPositionModeParam::torque_constant
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.3 CSTorqueModeParam Struct Reference

Struct contains configuration parameters for cyclic sync. torque mode.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t nominal_current
- uint16_t torque_constant
- uint32_t software_position_limit
- uint16_t motor_rated_torque
- uint32_t max_gear_input_speed
- uint32_t profile_vel
- uint32_t profile_acc
- uint32_t profile_dec
- uint32_t max_profile_vel
- uint32_t quick_stop_dec
- uint32_t interpolation_time_period

8.3.1 Detailed Description

Struct contains configuration parameters for cyclic sync. torque mode.

8.3.2 Member Data Documentation

8.3.2.1 interpolation_time_period

```
uint32_t CSTorqueModeParam::interpolation_time_period
```

8.3.2.2 max_gear_input_speed

```
uint32_t CSTorqueModeParam::max_gear_input_speed
```

8.3.2.3 max_profile_vel

```
uint32_t CSTorqueModeParam::max_profile_vel
```

8.3.2.4 motor_rated_torque

```
uint16_t CSTorqueModeParam::motor_rated_torque
```

8.3.2.5 nominal_current

```
uint32_t CSTorqueModeParam::nominal_current
```

8.3.2.6 profile_acc

```
uint32_t CSTorqueModeParam::profile_acc
```

8.3.2.7 profile_dec

```
uint32_t CSTorqueModeParam::profile_dec
```

8.3.2.8 profile_vel

```
uint32_t CSTorqueModeParam::profile_vel
```

8.3.2.9 quick_stop_dec

```
uint32_t CSTorqueModeParam::quick_stop_dec
```

8.3.2.10 software_position_limit

```
uint32_t CSTorqueModeParam::software_position_limit
```

8.3.2.11 torque_constant

```
uint16_t CSTorqueModeParam::torque_constant
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.4 CSVelocityModeParam Struct Reference

Struct contains configuration parameters for cyclic sync. velocity mode.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t nominal_current
- uint16_t torque_constant
- uint32_t current_controller_gain
- uint32_t velocity_control_parameter_set
- uint32_t software_position_limit
- uint16_t motor_rated_torque
- uint32_t max_gear_input_speed
- uint32_t profile_vel
- uint32_t profile_acc
- uint32_t profile_dec
- uint32_t max_fol_err
- uint32_t max_profile_vel
- uint32_t quick_stop_dec
- uint32_t interpolation_time_period

8.4.1 Detailed Description

Struct contains configuration parameters for cyclic sync. velocity mode.

8.4.2 Member Data Documentation

8.4.2.1 current_controller_gain

```
uint32_t CSVelocityModeParam::current_controller_gain
```

8.4.2.2 interpolation_time_period

```
uint32_t CSVelocityModeParam::interpolation_time_period
```

8.4.2.3 max_fol_err

```
uint32_t CSVelocityModeParam::max_fol_err
```

8.4.2.4 max_gear_input_speed

```
uint32_t CSVelocityModeParam::max_gear_input_speed
```

8.4.2.5 max_profile_vel

```
uint32_t CSVelocityModeParam::max_profile_vel
```

8.4.2.6 motor_rated_torque

```
uint16_t CSVelocityModeParam::motor_rated_torque
```

8.4.2.7 nominal_current

```
uint32_t CSVelocityModeParam::nominal_current
```

8.4.2.8 profile_acc

```
uint32_t CSVelocityModeParam::profile_acc
```

8.4.2.9 profile_dec

```
uint32_t CSVelocityModeParam::profile_dec
```

8.4.2.10 profile_vel

```
uint32_t CSVelocityModeParam::profile_vel
```

8.4.2.11 quick_stop_dec

```
uint32_t CSVelocityModeParam::quick_stop_dec
```

8.4.2.12 software_position_limit

```
uint32_t CSVelocityModeParam::software_position_limit
```

8.4.2.13 torque_constant

```
uint16_t CSVelocityModeParam::torque_constant
```

8.4.2.14 velocity_control_parameter_set

```
uint32_t CSVelocityModeParam::velocity_control_parameter_set
```

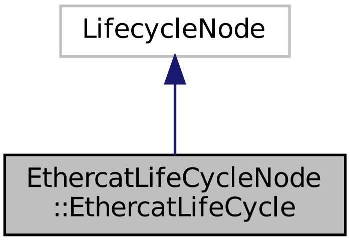
The documentation for this struct was generated from the following file:

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

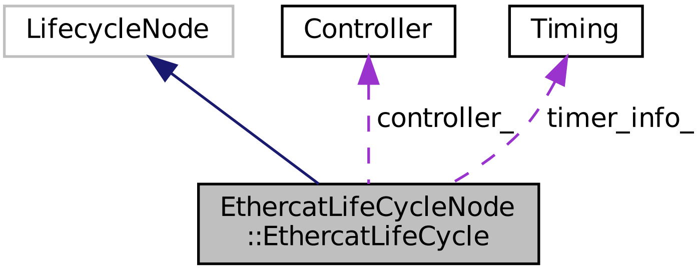
8.5 EthercatLifeCycleNode::EthercatLifeCycle Class Reference

```
#include <ecat_lifecycle.hpp>
```

Inheritance diagram for EthercatLifeCycleNode::EthercatLifeCycle:



Collaboration diagram for EthercatLifeCycleNode::EthercatLifeCycle:



Public Member Functions

- EthercatLifeCycle ()
- ∼EthercatLifeCycle ()

Private Types

- template<typename T = void>
  using TLSFAllocator = tlsf_heap_allocator< T >

Private Member Functions

- node_interfaces::LifecycleNodeInterface::CallbackReturn on_configure (const State &)

  Ethercat lifecycle node configuration function, node will start with this function For more information about Lifecyclenode and it's interfaces check below link : `https://design.ros2.org/articles/node_lifecycle.html`*.*
- node_interfaces::LifecycleNodeInterface::CallbackReturn on_activate (const State &)

  Activates Ethercat lifecycle node and starts real-time Ethercat communication. All publishing is done in real-time loop in this active state.
- node_interfaces::LifecycleNodeInterface::CallbackReturn on_deactivate (const State &)

  Deactivates Ethercat lifecycle node, turns of real-time communication.
- node_interfaces::LifecycleNodeInterface::CallbackReturn on_cleanup (const State &)

  Cleans up all variables and datas assigned by Ethercat lifecycle node.
- node_interfaces::LifecycleNodeInterface::CallbackReturn on_shutdown (const State &)

  Shuts down EtherCAT lifecycle node, releases Ethercat master.
- node_interfaces::LifecycleNodeInterface::CallbackReturn on_error (const State &)

  There isn't any error recovery functionality for this node, just resets nodes. Reconfiguration is needed for restarting communication.
- void HandleControlNodeCallbacks (const sensor_msgs::msg::Joy::SharedPtr msg)

  This function handles callbacks from control node. It will update received values from controller node.
- int SetComThreadPriorities ()

  Sets Ethercat communication thread's properties After this function called user must call StartEthercatCommunication() function].
- int InitEthercatCommunication ()

  Encapsulates all configuration steps for the EtherCAT communication with default slaves. And waits for connected slaves to become operational.
- int StartEthercatCommunication ()

  Starts EtherCAT communcation.
- void StartPdoExchange (void ∗instance)

  Realtime cyclic Pdo exchange function which will constantly read/write values from/to slaves.
- int GetComState ()

  Gets master's communication state.
- void ReadFromSlaves ()

  Reads data from slaves and updates received data structure to be published.
- int PublishAllData ()

  Publishes all data that master received and will be sent.
- void EnableMotors ()

  Enables connected motor drives based on CIA402.
- void WriteToSlavesVelocityMode ()

  Updates data that will be sent to slaves. This updated data will be published as well.
- void WriteToSlavesInPositionMode ()

Writes target position and control word to motor in profile position mode.

- void UpdateVelocityModeParameters ()

  Acquired data from subscribed controller topic will be assigned as motor speed parameter.
- void UpdatePositionModeParameters ()

  Acquired data from subscribed controller topic will be assigned as motor target position parameter.
- void UpdateCyclicPositionModeParameters ()

  Acquired data from subscribed controller topic will be assigned as motor cyclic target position parameter in configured interpolation time.
- void UpdateMotorStateVelocityMode ()

  Updates motor control world and motor state in velocity mode based on CIA402.
- void UpdateMotorStatePositionMode ()

  Updates motor control word and motor state in position mode based on CIA402 state machine,.
- void HandleGuiNodeCallbacks (const std_msgs::msg::UInt8::SharedPtr gui_sub)

  This function will handle values from GUI node. Updates parameters based on GUI node inputs.

Static Private Member Functions

- static void ∗ PassCyclicExchange (void ∗arg)

  Helper function to enter pthread_create, since pthread's are C function it doesn't accept class member function, to pass class member function this helper function is created.

Private Attributes

- rclcpp::TimerBase::SharedPtr timer_
- LifecyclePublisher< ecat_msgs::msg::DataReceived >::SharedPtr received_data_publisher_

  This lifecycle publisher will be used to publish received feedback data from slaves.
- LifecyclePublisher< ecat_msgs::msg::DataSent >::SharedPtr sent_data_publisher_

  This lifecycle publisher will be used to publish sent data from master to slaves.
- rclcpp::Subscription< sensor_msgs::msg::Joy >::SharedPtr joystick_subscriber_

  This subscriber will be used to receive data from controller node.
- rclcpp::Subscription< std_msgs::msg::UInt8 >::SharedPtr gui_subscriber_
- ecat_msgs::msg::DataReceived received_data_
- ecat_msgs::msg::DataSent sent_data_
- std::unique_ptr< EthercatNode > ecat_node_
- pthread_t ethercat_thread_

  pthread create required parameters.
- struct sched_param ethercat_sched_param_ = {}
- pthread_attr_t ethercat_thread_attr_
- int32_t err_
- uint8_t al_state_ = 0

  Application layer of slaves seen by master.(INIT/PREOP/SAFEOP/OP)
- uint32_t motor_state_ [g_kNumberOfServoDrivers]
- uint32_t command_ = 0x004F
- Controller controller_
- uint8_t gui_node_data_ = 1

  Values will be sent by controller node and will be assigned to variables below.
- uint8_t emergency_status_ = 1
- rclcpp::memory_strategy::MemoryStrategy::SharedPtr memory_strategy
- std::int32_t measurement_time = 0
- Timing timer_info_

8.5.1 Member Typedef Documentation

8.5.1.1 TLSFAllocator

```
template<typename T = void>
using EthercatLifeCycleNode::EthercatLifeCycle::TLSFAllocator = tlsf_heap_allocator<T>  [private]
```

8.5.2 Constructor & Destructor Documentation

8.5.2.1 EthercatLifeCycle()

```
EthercatLifeCycle::EthercatLifeCycle ( )
```

8.5.2.2 ∼EthercatLifeCycle()

```
EthercatLifeCycle::∼EthercatLifeCycle ( )
```

8.5.3 Member Function Documentation

8.5.3.1 EnableMotors()

```
void EthercatLifeCycle::EnableMotors ( )  [private]
```

Enables connected motor drives based on CIA402.

8.5.3.2 GetComState()

```
int EthercatLifeCycle::GetComState ( )  [private]
```

Gets master's communication state.

See also

ec_al_state_t

Returns

Application layer state for master.

8.5.3.3 HandleControlNodeCallbacks()

```
void EthercatLifeCycle::HandleControlNodeCallbacks (
             const sensor_msgs::msg::Joy::SharedPtr msg )  [private]
```

This function handles callbacks from control node. It will update received values from controller node.

8.5.3.4 HandleGuiNodeCallbacks()

```
void EthercatLifeCycle::HandleGuiNodeCallbacks (
             const std_msgs::msg::UInt8::SharedPtr gui_sub )  [private]
```

This function will handle values from GUI node. Updates parameters based on GUI node inputs.

8.5.3.5 InitEthercatCommunication()

```
int EthercatLifeCycle::InitEthercatCommunication ( )  [private]
```

Encapsulates all configuration steps for the EtherCAT communication with default slaves. And waits for connected slaves to become operational.

Returns

0 if succesful otherwise -1.

8.5.3.6 on_activate()

```
node_interfaces::LifecycleNodeInterface::CallbackReturn EthercatLifeCycle::on_activate (
             const State &  )  [private]
```

Activates Ethercat lifecycle node and starts real-time Ethercat communication. All publishing is done in real-time loop in this active state.

Returns

Success if activation succesfull,otherwise FAILURE

8.5.3.7 on_cleanup()

```
node_interfaces::LifecycleNodeInterface::CallbackReturn EthercatLifeCycle::on_cleanup (
             const State & ) [private]
```

Cleans up all variables and datas assigned by Ethercat lifecycle node.

Returns

Success if cleanup succesfull,otherwise FAILURE

8.5.3.8 on_configure()

```
node_interfaces::LifecycleNodeInterface::CallbackReturn EthercatLifeCycle::on_configure (
             const State & ) [private]
```

Ethercat lifecycle node configuration function, node will start with this function For more information about Lifecyclenode and it's interfaces check below link : `https://design.ros2.org/articles/node_lifecycle.html`.

See also

node_interfaces::LifecycleNodeInterface::CallbackReturn

Returns

Success if configuration succesfull,otherwise FAILURE

8.5.3.9 on_deactivate()

```
node_interfaces::LifecycleNodeInterface::CallbackReturn EthercatLifeCycle::on_deactivate (
             const State & ) [private]
```

Deactivates Ethercat lifecycle node, turns of real-time communication.

Returns

Success if deactivation succesfull,otherwise FAILURE

8.5.3.10 on_error()

```
node_interfaces::LifecycleNodeInterface::CallbackReturn EthercatLifeCycle::on_error (
            const State & )  [private]
```

There isn't any error recovery functionality for this node, just resets nodes. Reconfiguration is needed for restarting communication.

Returns

Success

8.5.3.11 on_shutdown()

```
node_interfaces::LifecycleNodeInterface::CallbackReturn EthercatLifeCycle::on_shutdown (
            const State & )  [private]
```

Shuts down EtherCAT lifecycle node, releases Ethercat master.

Returns

Success if shut down succesfull,otherwise FAILURE

8.5.3.12 PassCycylicExchange()

```
void * EthercatLifeCycle::PassCycylicExchange (
            void * arg )  [static], [private]
```

Helper function to enter pthread_create, since pthread's are C function it doesn't accept class member function, to pass class member function this helper function is created.

Parameters

arg	Pointer to current class instance.

Returns

void∗

8.5.3.13 PublishAllData()

```
int EthercatLifeCycle::PublishAllData ( )  [private]
```

Publishes all data that master received and will be sent.

Returns

0 if succesfull otherwise -1.

8.5.3.14 ReadFromSlaves()

```
void EthercatLifeCycle::ReadFromSlaves ( )  [private]
```

Reads data from slaves and updates received data structure to be published.

8.5.3.15 SetComThreadPriorities()

```
int EthercatLifeCycle::SetComThreadPriorities ( )  [private]
```

Sets Ethercat communication thread's properties After this function called user must call StartEthercatCommunication() function].

Returns

0 if succesfull, otherwise -1.

8.5.3.16 StartEthercatCommunication()

```
int EthercatLifeCycle::StartEthercatCommunication ( )  [private]
```

Starts EtherCAT communcation.

Returns

0 if succesfull, otherwise -1.

8.5.3.17 StartPdoExchange()

```
void EthercatLifeCycle::StartPdoExchange (
             void * instance )  [private]
```

Realtime cyclic Pdo exchange function which will constantly read/write values from/to slaves.

Parameters

arg	Used during pthread_create function to pass variables to realtime task.

Returns

NULL

8.5.3.18 UpdateCyclicPositionModeParameters()

```
void EthercatLifeCycle::UpdateCyclicPositionModeParameters ( )  [private]
```

Acquired data from subscribed controller topic will be assigned as motor cyclic target position parameter in configured interpolation time.

8.5.3.19 UpdateMotorStatePositionMode()

```
void EthercatLifeCycle::UpdateMotorStatePositionMode ( )  [private]
```

Updates motor control word and motor state in position mode based on CIA402 state machine,.

8.5.3.20 UpdateMotorStateVelocityMode()

```
void EthercatLifeCycle::UpdateMotorStateVelocityMode ( )  [private]
```

Updates motor control world and motor state in velocity mode based on CIA402.

8.5.3.21 UpdatePositionModeParameters()

```
void EthercatLifeCycle::UpdatePositionModeParameters ( )  [private]
```

Acquired data from subscribed controller topic will be assigned as motor target position parameter.

8.5.3.22 UpdateVelocityModeParameters()

```
void EthercatLifeCycle::UpdateVelocityModeParameters ( )  [private]
```

Acquired data from subscribed controller topic will be assigned as motor speed parameter.

8.5.3.23 WriteToSlavesInPositionMode()

```
void EthercatLifeCycle::WriteToSlavesInPositionMode ( )  [private]
```

Writes target position and control word to motor in profile position mode.

8.5.3.24 WriteToSlavesVelocityMode()

```
void EthercatLifeCycle::WriteToSlavesVelocityMode ( )  [private]
```

Updates data that will be sent to slaves. This updated data will be published as well.

8.5.4 Member Data Documentation

8.5.4.1 al_state_

```
uint8_t EthercatLifeCycleNode::EthercatLifeCycle::al_state_ = 0  [private]
```

Application layer of slaves seen by master.(INIT/PREOP/SAFEOP/OP)

8.5.4.2 command_

```
uint32_t EthercatLifeCycleNode::EthercatLifeCycle::command_ = 0x004F  [private]
```

8.5.4.3 controller_

```
Controller EthercatLifeCycleNode::EthercatLifeCycle::controller_  [private]
```

8.5.4.4 ecat_node_

```
std::unique_ptr<EthercatNode> EthercatLifeCycleNode::EthercatLifeCycle::ecat_node_  [private]
```

8.5.4.5 emergency_status_

```
uint8_t EthercatLifeCycleNode::EthercatLifeCycle::emergency_status_ = 1  [private]
```

8.5.4.6 err_

```
int32_t EthercatLifeCycleNode::EthercatLifeCycle::err_  [private]
```

8.5.4.7 ethercat_sched_param_

```
struct sched_param EthercatLifeCycleNode::EthercatLifeCycle::ethercat_sched_param_ = {}  [private]
```

8.5.4.8 ethercat_thread_

```
pthread_t EthercatLifeCycleNode::EthercatLifeCycle::ethercat_thread_  [private]
```

pthread create required parameters.

8.5.4.9 ethercat_thread_attr_

```
pthread_attr_t EthercatLifeCycleNode::EthercatLifeCycle::ethercat_thread_attr_  [private]
```

8.5.4.10 gui_node_data_

```
uint8_t EthercatLifeCycleNode::EthercatLifeCycle::gui_node_data_ = 1  [private]
```

Values will be sent by controller node and will be assigned to variables below.

8.5.4.11 gui_subscriber_

```
rclcpp::Subscription<std_msgs::msg::UInt8>::SharedPtr EthercatLifeCycleNode::EthercatLife←
Cycle::gui_subscriber_  [private]
```

8.5.4.12 joystick_subscriber_

```
rclcpp::Subscription<sensor_msgs::msg::Joy>::SharedPtr EthercatLifeCycleNode::EthercatLife↩
Cycle::joystick_subscriber_ [private]
```

This subscriber will be used to receive data from controller node.

8.5.4.13 measurement_time

```
std::int32_t EthercatLifeCycleNode::EthercatLifeCycle::measurement_time = 0 [private]
```

8.5.4.14 memory_strategy

```
rclcpp::memory_strategy::MemoryStrategy::SharedPtr EthercatLifeCycleNode::EthercatLifeCycle↩
::memory_strategy [private]
```

Initial value:

```
=
        std::make_shared<AllocatorMemoryStrategy<TLSFAllocator<void»>()
```

8.5.4.15 motor_state_

```
uint32_t EthercatLifeCycleNode::EthercatLifeCycle::motor_state_[g_kNumberOfServoDrivers] [private]
```

8.5.4.16 received_data_

```
ecat_msgs::msg::DataReceived EthercatLifeCycleNode::EthercatLifeCycle::received_data_ [private]
```

8.5.4.17 received_data_publisher_

```
LifecyclePublisher<ecat_msgs::msg::DataReceived>::SharedPtr EthercatLifeCycleNode::Ethercat↩
LifeCycle::received_data_publisher_ [private]
```

This lifecycle publisher will be used to publish received feedback data from slaves.

8.5.4.18 sent_data_

```
ecat_msgs::msg::DataSent EthercatLifeCycleNode::EthercatLifeCycle::sent_data_  [private]
```

8.5.4.19 sent_data_publisher_

```
LifecyclePublisher<ecat_msgs::msg::DataSent>::SharedPtr EthercatLifeCycleNode::EthercatLife↩
Cycle::sent_data_publisher_  [private]
```

This lifecycle publisher will be used to publish sent data from master to slaves.

8.5.4.20 timer_

```
rclcpp::TimerBase::SharedPtr EthercatLifeCycleNode::EthercatLifeCycle::timer_  [private]
```

8.5.4.21 timer_info_

```
Timing EthercatLifeCycleNode::EthercatLifeCycle::timer_info_  [private]
```

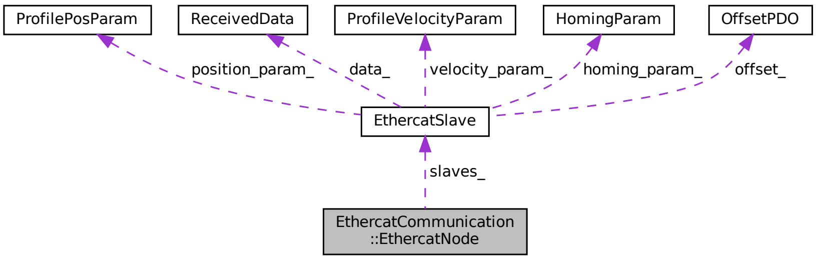
The documentation for this class was generated from the following files:

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
- /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp

8.6 EthercatCommunication::EthercatNode Class Reference

```
#include <ecat_node.hpp>
```

Collaboration diagram for EthercatCommunication::EthercatNode:



Public Member Functions

- EthercatNode ()
- ∼EthercatNode ()
- int ConfigureMaster ()

  Requests master instance and creates a domain for a master.
- void DefineDefaultSlaves ()

  Defines default connected slaves based on number of slaves. Specifies its position, vendor id , product code etc. Default connected slaves considered implementation specific. In our case it will be 3 motors and one EasyCAT slave.
- void SetCustomSlave (EthercatSlave c_slave, int position)

  Passes your defined slave to EthercatNode class.
- int ConfigureSlaves ()

  Obtains slave configuration for all slaves w.r.t master.
- int SetProfilePositionParameters (ProfilePosParam &P, int position)

  Set mode to ProfilePositionMode with specified parameters for servo drive on that position.
- int SetProfilePositionParametersAll (ProfilePosParam &P)

  Set the mode to ProfilePositionMode with specified Parameters for all servo drives on the bus.
- int SetProfileVelocityParameters (ProfileVelocityParam &P, int position)

  Set mode to ProfileVelocityMode with specified parameters for servo drive on that position.
- int SetProfileVelocityParametersAll (ProfileVelocityParam &P)

  Set mode to ProfileVelocityMode with specified parameters for all servo drives on the bus.
- int SetCyclicSyncPositionModeParameters (CSPositionModeParam &P, int position)

  Set the Cyclic Sync Position Mode Parameters for slave in specified physical position w.r.t. master.
- int SetCyclicSyncPositionModeParametersAll (CSPositionModeParam &P)

  Sets the Cyclic Synchronous Position Mode Parameters for all connected motor driver slaves.
- int MapDefaultPdos ()

  Maps default PDOs for our spine surgery robot implementation.
- int MapCustomPdos (ec_sync_info_t ∗syncs, ec_pdo_entry_reg_t ∗pdo_entry_reg, int position)

  Map Custom PDO based on your PDO mappings.
- void ConfigDcSyncDefault ()

  Configures DC sync for our default configuration.
- void ConfigDcSync (uint16_t assign_activate, int position)

  Configures DC synchronization for specified slave position.
- void CheckSlaveConfigurationState ()

  This function will check slave's application layer states. (INIT/PREOP/SAFEOP/OP)
- int CheckMasterState ()

  This function will check master's state, in terms of number of responding slaves and their application layer states.
- void CheckMasterDomainState ()

  Reads the state of a domain. Stores the domain state in the given state structure. Using this method, the process data exchange can be monitored in realtime.
- int ActivateMaster ()

  Activates master, after this function call realtime operation can start.
- int RegisterDomain ()

  Registers domain for each slave. This method has to be called after ecrt_master_activate() to get the mapped domain process data memory.
- int WaitForOperationalMode ()

  Puts all slave to operational mode.User must call this before entering real-time operation. Reason for this function is that, master and slave has to do several exchange before becoming operational. So this function does exchange between master and slaves for up to 10 sec, could finish earlier. If timeout occurs it will return -1.
- int OpenEthercatMaster ()

  Opens EtherCAT master via command line tool if it's not already on.
- int GetNumberOfConnectedSlaves ()

Get the Number Of physically Connected Slaves to the bus.And checks if specified NUM_OF_SLAVES is correct.

- void GetAllSlaveInformation ()

  Get the information of physically connected slaves to the master. This function will return connected slave's vendor id, product code.
- void DeactivateCommunication ()

  Deactivates slaves and can be called in real-time.
- void ReleaseMaster ()

  Deactivates and releases master shouldn't be called in real-time.
- int ShutDownEthercatMaster ()

  Shutdowns EtherCAT master via command line tool if it's not already off.

Public Attributes

- EthercatSlave slaves_ [NUM_OF_SLAVES]

Private Attributes

- int fd

  File descriptor to open and wake master from CLI.

8.6.1 Constructor & Destructor Documentation

8.6.1.1 EthercatNode()

```
EthercatNode::EthercatNode ( )
```

8.6.1.2 ∼EthercatNode()

```
EthercatNode::∼EthercatNode ( )
```

8.6.2 Member Function Documentation

8.6.2.1 ActivateMaster()

```
int EthercatNode::ActivateMaster ( )
```

Activates master, after this function call realtime operation can start.

Warning

Before activating master all configuration should be done

After calling this function you have to register domain(s) and start realtime task.

Returns

0 if succesful, otherwise -1.

8.6.2.2 CheckMasterDomainState()

```
void EthercatNode::CheckMasterDomainState ( )
```

Reads the state of a domain. Stores the domain state in the given state structure. Using this method, the process data exchange can be monitored in realtime.

8.6.2.3 CheckMasterState()

```
int EthercatNode::CheckMasterState ( )
```

This function will check master's state, in terms of number of responding slaves and their application layer states.

Returns

0 if succesful, otherwise -1

See also

ec_master_state_t structure.

8.6.2.4 CheckSlaveConfigurationState()

```
void EthercatNode::CheckSlaveConfigurationState ( )
```

This function will check slave's application layer states. (INIT/PREOP/SAFEOP/OP)

8.6.2.5 ConfigDcSync()

```
void EthercatNode::ConfigDcSync (
             uint16_t assign_activate,
             int position )
```

Configures DC synchronization for specified slave position.

Parameters

assign_activate	Activating DC synchronization for slave. 0x300 for Elmo | and same for EasyCAT

Note

Assign activate parameters specified in slaves ESI file

Parameters

position	

8.6.2.6 ConfigDcSyncDefault()

```
void EthercatNode::ConfigDcSyncDefault ( )
```

Configures DC sync for our default configuration.

8.6.2.7 ConfigureMaster()

```
int EthercatNode::ConfigureMaster ( )
```

Requests master instance and creates a domain for a master.

Note

Keep in mind that created master and domain are global variables.

Returns

0 if succesful otherwise -1.

8.6.2.8 ConfigureSlaves()

```
int EthercatNode::ConfigureSlaves ( )
```

Obtains slave configuration for all slaves w.r.t master.

Returns

0 if succesfull, otherwise -1.

8.6.2.9 DeactivateCommunication()

```
void EthercatNode::DeactivateCommunication ( )
```

Deactivates slaves and can be called in real-time.

8.6.2.10 DefineDefaultSlaves()

```
void EthercatCommunication::EthercatNode::DefineDefaultSlaves ( )
```

Defines default connected slaves based on number of slaves. Specifies its position, vendor id , product code etc. Default connected slaves considered implementation specific. In our case it will be 3 motors and one EasyCAT slave.

8.6.2.11 GetAllSlaveInformation()

```
void EthercatNode::GetAllSlaveInformation ( )
```

Get the information of physically connected slaves to the master. This function will return connected slave's vendor id, product code.

8.6.2.12 GetNumberOfConnectedSlaves()

```
int EthercatNode::GetNumberOfConnectedSlaves ( )
```

Get the Number Of physically Connected Slaves to the bus.And checks if specified NUM_OF_SLAVES is correct.

Returns

0 if NUM_OF_SLAVES setting is correct, otherwise -1.

8.6.2.13 MapCustomPdos()

```
int EthercatNode::MapCustomPdos (
            ec_sync_info_t * syncs,
            ec_pdo_entry_reg_t * pdo_entry_reg,
            int position )
```

Map Custom PDO based on your PDO mappings.

Note

You have to specify slave syncs and slave pdo registers before using function

Parameters

S	EthercatSlave instance
position	Physical position of your slave w.r.t master

Returns

0 if succesfull, -1 otherwise.

8.6.2.14 MapDefaultPdos()

```
int EthercatNode::MapDefaultPdos ( )
```

Maps default PDOs for our spine surgery robot implementation.

Note

This method is specific for our spinerobot implementation. If you have different topology or different servo drives use

See also

MapCustomPdos() function of modify this function based on your needs.

Returns

0 if succesfull, otherwise -1.

This part is specific for our Custom EASYCAT slave configuration To create your custom slave and variables you can add variables to

See also

OffsetPDO struct. Also you have add your variables to received data structure, you may have to create your custom msg files as well.

8.6.2.15 OpenEthercatMaster()

```
int EthercatNode::OpenEthercatMaster ( )
```

Opens EtherCAT master via command line tool if it's not already on.

Returns

0 if succesfull, otherwise -1.

8.6.2.16 RegisterDomain()

```
int EthercatNode::RegisterDomain ( )
```

Registers domain for each slave. This method has to be called after ecrt_master_activate() to get the mapped domain process data memory.

Returns

0 if succeful , otherwise -1

8.6.2.17 ReleaseMaster()

```
void EthercatNode::ReleaseMaster ( )
```

Deactivates and releases master shouldn't be called in real-time.

8.6.2.18 SetCustomSlave()

```
void EthercatNode::SetCustomSlave (
            EthercatSlave c_slave,
            int position )
```

Passes your defined slave to EthercatNode class.

Parameters

c_slave	first create your own EthercatSlave instance and modify it then pass it to configuration.
position	specify the physical connection position for your custom configured slave.

8.6.2.19 SetCyclicSyncPositionModeParameters()

```
int EthercatNode::SetCyclicSyncPositionModeParameters (
            CSPositionModeParam & P,
            int position )
```

Set the Cyclic Sync Position Mode Parameters for slave in specified physical position w.r.t. master.

Parameters

P	Cyclic Sync. Position Mode Parameters.
position	Physical position of slave to be configured

Returns

0 if sucessfull, otherwise -1.

8.6.2.20 SetCyclicSyncPositionModeParametersAll()

```
int EthercatNode::SetCyclicSyncPositionModeParametersAll (
             CSPositionModeParam & P )
```

Sets the Cyclic Synchronous Position Mode Parameters for all connected motor driver slaves.

Returns

0 if sucessful, otherwise -1.

8.6.2.21 SetProfilePositionParameters()

```
int EthercatNode::SetProfilePositionParameters (
             ProfilePosParam & P,
             int position )
```

Set mode to ProfilePositionMode with specified parameters for servo drive on that position.

Parameters

P	Profile position parameter structure specified by user.
position	Slave position

Returns

0 if succesfull, otherwise -1.

8.6.2.22 SetProfilePositionParametersAll()

```
int EthercatNode::SetProfilePositionParametersAll (
             ProfilePosParam & P )
```

Set the mode to ProfilePositionMode with specified Parameters for all servo drives on the bus.

Parameters

P	Profile position parameter structure specified by user.

Returns

0 if succesful, otherwise -1.

8.6.2.23 SetProfileVelocityParameters()

```
int EthercatNode::SetProfileVelocityParameters (
            ProfileVelocityParam & P,
            int position )
```

Set mode to ProfileVelocityMode with specified parameters for servo drive on that position.

Parameters

P	Profile velocity parameter structure specified by user.
position	Slave position

Returns

0 if succesful, -1 otherwise.

8.6.2.24 SetProfileVelocityParametersAll()

```
int EthercatNode::SetProfileVelocityParametersAll (
            ProfileVelocityParam & P )
```

Set mode to ProfileVelocityMode with specified parameters for all servo drives on the bus.

Parameters

P	Profile velocity parameter structure specified by user.

Returns

0 if succesfull, -1 otherwise.

Todo Add error code to all functions.Instead of returning -1.

8.6.2.25 ShutDownEthercatMaster()

```
int EthercatNode::ShutDownEthercatMaster ( )
```

Shutdowns EtherCAT master via command line tool if it's not already off.

Returns

0 if succesfull, otherwise -1.

8.6.2.26 WaitForOperationalMode()

```
int EthercatNode::WaitForOperationalMode ( )
```

Puts all slave to operational mode.User must call this before entering real-time operation. Reason for this function is that, master and slave has to do several exchange before becoming operational. So this function does exchange between master and slaves for up to 10 sec, could finish earlier. If timeout occurs it will return -1.

Returns

0 if succesfull, otherwise -1.

8.6.3 Member Data Documentation

8.6.3.1 fd

```
int EthercatCommunication::EthercatNode::fd [private]
```

File descriptor to open and wake master from CLI.

8.6.3.2 slaves_

```
EthercatSlave EthercatCommunication::EthercatNode::slaves_[NUM_OF_SLAVES]
```

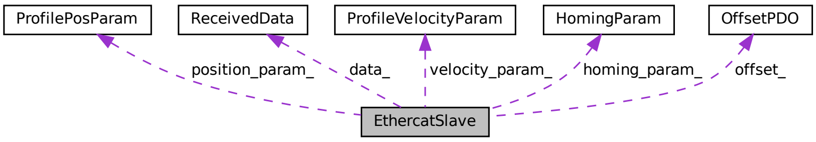
The documentation for this class was generated from the following files:

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_node.hpp
- /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_node.cpp

8.7 EthercatSlave Class Reference

```
#include <ecat_slave.hpp>
```

Collaboration diagram for EthercatSlave:



Public Member Functions

- EthercatSlave ()
- ∼EthercatSlave ()
- int CheckSlaveConfigState ()

  This function will check slave's application layer states. (INIT/PREOP/SAFEOP/OP)

Public Attributes

- ec_slave_config_t ∗ slave_config_

  Slave configuration parameters, assinged to each slave.
- ec_slave_config_state_t slave_config_state_

  Slave state handle to check if slave is online and slaves state machine status(INIT/PREOP/SAFEOP/0P)
- uint8_t ∗ slave_pdo_domain_

  PDO domain for data exchange.
- int32_t motor_state_

  Variable for checking motor state.
- ec_slave_info_t slave_info_

  Slave information data structure. This structure contains all information related to slave. It will be used to get slave's information from master.
- OffsetPDO offset_

  Offset for PDO entries to assign pdo registers.
- ReceivedData data_

  Received data from servo drivers.
- ProfileVelocityParam velocity_param_

  Slave velocity parameters.
- ProfilePosParam position_param_

  Slave position parameters.
- HomingParam homing_param_

Static Public Attributes

- const static uint32_t kSync0_shift_ = 0

  DC sync shift setting, zero will give best synchronization.

8.7.1 Constructor & Destructor Documentation

8.7.1.1 EthercatSlave()

```
EthercatSlave::EthercatSlave ( )
```

8.7.1.2 ∼EthercatSlave()

```
EthercatSlave::∼EthercatSlave ( )
```

8.7.2 Member Function Documentation

8.7.2.1 CheckSlaveConfigState()

```
int EthercatSlave::CheckSlaveConfigState ( )
```

This function will check slave's application layer states. (INIT/PREOP/SAFEOP/OP)

Note

This function shouldn't be called in real time context.For diagnosis you can use CheckDomainState() encapsulation in ecat_node.

Returns

0 if succesful.

8.7.3 Member Data Documentation

8.7.3.1 data_

```
ReceivedData EthercatSlave::data_
```

Received data from servo drivers.

8.7.3.2 homing_param_

```
HomingParam EthercatSlave::homing_param_
```

8.7.3.3 kSync0_shift_

```
const static uint32_t EthercatSlave::kSync0_shift_ = 0  [static]
```

DC sync shift setting, zero will give best synchronization.

8.7.3.4 motor_state_

```
int32_t EthercatSlave::motor_state_
```

Variable for checking motor state.

8.7.3.5 offset_

```
OffsetPDO EthercatSlave::offset_
```

Offset for PDO entries to assign pdo registers.

8.7.3.6 position_param_

```
ProfilePosParam EthercatSlave::position_param_
```

Slave position parameters.

8.7.3.7 slave_config_

```
ec_slave_config_t* EthercatSlave::slave_config_
```

Slave configuration parameters, assinged to each slave.

8.7.3.8 slave_config_state_

```
ec_slave_config_state_t EthercatSlave::slave_config_state_
```

Slave state handle to check if slave is online and slaves state machine status(INIT/PREOP/SAFEOP/0P)

8.7.3.9 slave_info_

```
ec_slave_info_t EthercatSlave::slave_info_
```

Slave information data structure. This structure contains all information related to slave. It will be used to get slave's information from master.

8.7.3.10 slave_pdo_domain_

```
uint8_t* EthercatSlave::slave_pdo_domain_
```

PDO domain for data exchange.

8.7.3.11 velocity_param_

```
ProfileVelocityParam EthercatSlave::velocity_param_
```

Slave velocity parameters.

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_slave.hpp
- /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_slave.cpp

8.8 HomingParam Struct Reference

Parameters that should be specified in homing mode.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t max_fol_err
- uint32_t max_profile_vel
- uint32_t quick_stop_dec
- uint32_t speed_for_switch_search
- uint32_t speed_for_zero_search
- uint32_t homing_acc
- uint16_t curr_threshold_homing

  Used when homing by touching mechanical limit and sensing current.
- int32_t home_offset

  Amount to move away from the sensed limit
- int8_t homing_method

8.8.1 Detailed Description

Parameters that should be specified in homing mode.

8.8.2 Member Data Documentation

8.8.2.1 curr_threshold_homing

```
uint16_t HomingParam::curr_threshold_homing
```

Used when homing by touching mechanical limit and sensing current.

8.8.2.2 home_offset

```
int32_t HomingParam::home_offset
```

Amount to move away from the sensed limit

8.8.2.3 homing_acc

```
uint32_t HomingParam::homing_acc
```

8.8.2.4 homing_method

```
int8_t HomingParam::homing_method
```

8.8.2.5 max_fol_err

```
uint32_t HomingParam::max_fol_err
```

8.8.2.6 max_profile_vel

```
uint32_t HomingParam::max_profile_vel
```

8.8.2.7 quick_stop_dec

```
uint32_t HomingParam::quick_stop_dec
```

8.8.2.8 speed_for_switch_search

```
uint32_t HomingParam::speed_for_switch_search
```

8.8.2.9 speed_for_zero_search

```
uint32_t HomingParam::speed_for_zero_search
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.9 OffsetPDO Struct Reference

offset for PDO entries to register PDOs.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t target_pos
- uint32_t target_vel
- uint32_t target_tor
- uint32_t max_tor
- uint32_t control_word
- uint32_t op_mode
- uint32_t profile_acc
- uint32_t profile_dec
- uint32_t quick_stop_dec
- uint32_t profile_vel
- uint32_t actual_pos
- uint32_t pos_fol_err
- uint32_t actual_vel
- uint32_t actual_cur
- uint32_t actual_tor
- uint32_t status_word
- uint32_t op_mode_display
- uint32_t error_code
- uint32_t extra_status_reg
- uint32_t r_limit_switch
- uint32_t l_limit_switch
- uint32_t emergency_switch

8.9.1 Detailed Description

offset for PDO entries to register PDOs.

8.9.2 Member Data Documentation

8.9.2.1 actual_cur

```
uint32_t OffsetPDO::actual_cur
```

8.9.2.2 actual_pos

```
uint32_t OffsetPDO::actual_pos
```

8.9.2.3 actual_tor

```
uint32_t OffsetPDO::actual_tor
```

8.9.2.4 actual_vel

```
uint32_t OffsetPDO::actual_vel
```

8.9.2.5 control_word

```
uint32_t OffsetPDO::control_word
```

8.9.2.6 emergency_switch

```
uint32_t OffsetPDO::emergency_switch
```

8.9.2.7 error_code

```
uint32_t OffsetPDO::error_code
```

8.9.2.8 extra_status_reg

```
uint32_t OffsetPDO::extra_status_reg
```

8.9.2.9 l_limit_switch

```
uint32_t OffsetPDO::l_limit_switch
```

8.9.2.10 max_tor

```
uint32_t OffsetPDO::max_tor
```

8.9.2.11 op_mode

```
uint32_t OffsetPDO::op_mode
```

8.9.2.12 op_mode_display

```
uint32_t OffsetPDO::op_mode_display
```

8.9.2.13 pos_fol_err

```
uint32_t OffsetPDO::pos_fol_err
```

8.9.2.14 profile_acc

```
uint32_t OffsetPDO::profile_acc
```

8.9.2.15 profile_dec

```
uint32_t OffsetPDO::profile_dec
```

8.9.2.16 profile_vel

```
uint32_t OffsetPDO::profile_vel
```

8.9.2.17 quick_stop_dec

```
uint32_t OffsetPDO::quick_stop_dec
```

8.9.2.18 r_limit_switch

```
uint32_t OffsetPDO::r_limit_switch
```

8.9.2.19 status_word

```
uint32_t OffsetPDO::status_word
```

8.9.2.20 target_pos

```
uint32_t OffsetPDO::target_pos
```

8.9.2.21 target_tor

```
uint32_t OffsetPDO::target_tor
```

8.9.2.22 target_vel

```
uint32_t OffsetPDO::target_vel
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.10 ProfilePosParam Struct Reference

Parameters that should be specified in position mode.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t profile_vel
- uint32_t profile_acc
- uint32_t profile_dec
- uint32_t max_fol_err
- uint32_t max_profile_vel
- uint32_t quick_stop_dec
- uint16_t motion_profile_type

8.10.1 Detailed Description

Parameters that should be specified in position mode.

8.10.2 Member Data Documentation

8.10.2.1 max_fol_err

```
uint32_t ProfilePosParam::max_fol_err
```

8.10.2.2 max_profile_vel

```
uint32_t ProfilePosParam::max_profile_vel
```

8.10.2.3 motion_profile_type

```
uint16_t ProfilePosParam::motion_profile_type
```

8.10.2.4 profile_acc

```
uint32_t ProfilePosParam::profile_acc
```

8.10.2.5 profile_dec

```
uint32_t ProfilePosParam::profile_dec
```

8.10.2.6 profile_vel

```
uint32_t ProfilePosParam::profile_vel
```

8.10.2.7 quick_stop_dec

```
uint32_t ProfilePosParam::quick_stop_dec
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.11 ProfileVelocityParam Struct Reference

Parameters that should be specified in velocity mode.

```
#include <ecat_globals.hpp>
```

Public Attributes

- uint32_t max_profile_vel
- uint32_t quick_stop_dec
- uint32_t profile_acc
- uint32_t profile_dec
- uint16_t motion_profile_type

8.11.1 Detailed Description

Parameters that should be specified in velocity mode.

8.11.2 Member Data Documentation

8.11.2.1 max_profile_vel

```
uint32_t ProfileVelocityParam::max_profile_vel
```

8.11.2.2 motion_profile_type

```
uint16_t ProfileVelocityParam::motion_profile_type
```

8.11.2.3 profile_acc

```
uint32_t ProfileVelocityParam::profile_acc
```

8.11.2.4 profile_dec

```
uint32_t ProfileVelocityParam::profile_dec
```

8.11.2.5 quick_stop_dec

```
uint32_t ProfileVelocityParam::quick_stop_dec
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.12 ReceivedData Struct Reference

Received feedback data from slaves.

```
#include <ecat_globals.hpp>
```

Public Attributes

- int32_t target_pos
- int32_t target_vel
- int16_t target_tor
- int16_t max_tor
- uint16_t control_word
- OpMode op_mode
- int32_t vel_offset
- int16_t tor_offset
- int32_t actual_pos
- int32_t actual_vel
- int16_t actual_cur
- int16_t actual_tor
- uint16_t status_word
- int8_t op_mode_display
- uint8_t left_limit_switch_val
- uint8_t right_limit_switch_val
- uint8_t s_emergency_switch_val

8.12.1 Detailed Description

Received feedback data from slaves.

8.12.2 Member Data Documentation

8.12.2.1 actual_cur

```
int16_t ReceivedData::actual_cur
```

8.12.2.2 actual_pos

```
int32_t ReceivedData::actual_pos
```

8.12.2.3 actual_tor

```
int16_t ReceivedData::actual_tor
```

8.12.2.4 actual_vel

```
int32_t ReceivedData::actual_vel
```

8.12.2.5 control_word

```
uint16_t ReceivedData::control_word
```

8.12.2.6 left_limit_switch_val

```
uint8_t ReceivedData::left_limit_switch_val
```

8.12.2.7 max_tor

```
int16_t ReceivedData::max_tor
```

8.12.2.8 op_mode

```
OpMode ReceivedData::op_mode
```

8.12.2.9 op_mode_display

```
int8_t ReceivedData::op_mode_display
```

8.12.2.10 right_limit_switch_val

```
uint8_t ReceivedData::right_limit_switch_val
```

8.12.2.11 s_emergency_switch_val

```
uint8_t ReceivedData::s_emergency_switch_val
```

8.12.2.12 status_word

```
uint16_t ReceivedData::status_word
```

8.12.2.13 target_pos

```
int32_t ReceivedData::target_pos
```

8.12.2.14 target_tor

```
int16_t ReceivedData::target_tor
```

8.12.2.15 target_vel

```
int32_t ReceivedData::target_vel
```

8.12.2.16 tor_offset

```
int16_t ReceivedData::tor_offset
```

8.12.2.17 vel_offset

```
int32_t ReceivedData::vel_offset
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.13 SdoRequest Struct Reference

EtherCAT SDO request structure for configuration phase.

```
#include <ecat_globals.hpp>
```

Public Attributes

- ec_sdo_request ∗ profile_acc
- ec_sdo_request ∗ profile_dec
- ec_sdo_request ∗ profile_vel
- ec_sdo_request ∗ quick_stop_dec
- ec_sdo_request ∗ motion_profile_type
- ec_sdo_request ∗ max_profile_vel
- ec_sdo_request ∗ max_fol_err
- ec_sdo_request ∗ speed_for_switch_search
- ec_sdo_request ∗ speed_for_zero_search
- ec_sdo_request ∗ homing_acc
- ec_sdo_request ∗ curr_threshold_homing
- ec_sdo_request ∗ home_offset
- ec_sdo_request ∗ homing_method

8.13.1 Detailed Description

EtherCAT SDO request structure for configuration phase.

8.13.2 Member Data Documentation

8.13.2.1 curr_threshold_homing

```
ec_sdo_request* SdoRequest::curr_threshold_homing
```

8.13.2.2 home_offset

```
ec_sdo_request* SdoRequest::home_offset
```

8.13.2.3 homing_acc

```
ec_sdo_request* SdoRequest::homing_acc
```

8.13.2.4 homing_method

```
ec_sdo_request* SdoRequest::homing_method
```

8.13.2.5 max_fol_err

```
ec_sdo_request* SdoRequest::max_fol_err
```

8.13.2.6 max_profile_vel

```
ec_sdo_request* SdoRequest::max_profile_vel
```

8.13.2.7 motion_profile_type

```
ec_sdo_request* SdoRequest::motion_profile_type
```

8.13.2.8 profile_acc

```
ec_sdo_request* SdoRequest::profile_acc
```

8.13.2.9 profile_dec

```
ec_sdo_request* SdoRequest::profile_dec
```

8.13.2.10 profile_vel

```
ec_sdo_request* SdoRequest::profile_vel
```

8.13.2.11 quick_stop_dec

```
ec_sdo_request* SdoRequest::quick_stop_dec
```

8.13.2.12 speed_for_switch_search

```
ec_sdo_request* SdoRequest::speed_for_switch_search
```

8.13.2.13 speed_for_zero_search

```
ec_sdo_request* SdoRequest::speed_for_zero_search
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp

8.14 Timing Class Reference

```
#include <timing.hpp>
```

Public Member Functions

- void GetTime ()
- void MeasureTimeDifference ()
- void OutInfoToFile ()

Public Attributes

- std::chrono::high_resolution_clock::time_point timer_start_
- std::chrono::high_resolution_clock::time_point last_start_time_
- std::chrono::duration< long, std::micro > time_span
- std::vector< long > timing_info_ = std::vector<long>(NUMBER_OF_SAMPLES)
- uint32_t counter_ = 0

8.14.1 Member Function Documentation

8.14.1.1 GetTime()

```
void Timing::GetTime ( )
```

8.14.1.2 MeasureTimeDifference()

```
void Timing::MeasureTimeDifference ( )
```

8.14.1.3 OutInfoToFile()

```
void Timing::OutInfoToFile ( )
```

8.14.2 Member Data Documentation

8.14.2.1 counter_

```
uint32_t Timing::counter_ = 0
```

8.14.2.2 last_start_time_

```
std::chrono::high_resolution_clock::time_point Timing::last_start_time_
```

8.14.2.3 time_span

```
std::chrono::duration<long,std::micro> Timing::time_span
```

8.14.2.4 timer_start_

```
std::chrono::high_resolution_clock::time_point Timing::timer_start_
```

8.14.2.5 timing_info_

```
std::vector<long> Timing::timing_info_ = std::vector<long>(NUMBER_OF_SAMPLES)
```

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

- /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/timing.hpp
- /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/timing.cpp

Chapter 9

File Documentation

9.1 /home/veysiadn/spinerobot_ws/src/ecat_pkg/CMakeLists.txt File Reference

Functions

- cmake_minimum_required (VERSION 3.5) project(ecat_pkg) if(NOT CMAKE_C_STANDARD) set(CMAKE_C_STANDARD 99) endif() if(NOT CMAKE_CXX_STANDARD) set(CMAKE_CXX_STANDARD 14) endif() add_compile_options(-g -w -Wall -Wextra -Wpedantic -I/opt/etherlab/include -L/opt/etherlab/lib -lethercat -lpthread -lrt -Wl
- rpath opt etherlab lib set (etherlab_include/opt/etherlab/include) set(etherlab_lib/opt/etherlab/lib/libethercat.so.1.1.0) set(ecat_node_include ∼/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/) set(node_name "ecat_node") find_package(ament_cmake REQUIRED) find_package(rclcpp REQUIRED) find_package(rclcpp_lifecyle REQUIRED) find_package(rttest) find_package(tlsf_cpp) find_package(ecat_msgs REQUIRED) find_package(sensor_msgs REQUIRED) add_executable(ecat_node src/main.cpp src/ecat_node.cpp src/ecat_slave.cpp src/ecat_lifecycle.cpp src/timing.cpp) target_include_directories(ecat_node PUBLIC $< BUILD_INTERFACE

Variables

- rpath Wl

9.1.1 Function Documentation

9.1.1.1 cmake_minimum_required()

```
cmake_minimum_required (
            VERSION 3.  5 )
```

9.1.1.2 set()

```
rpath opt etherlab lib set (
            etherlab_include/opt/etherlab/ include )
```

9.1.2 Variable Documentation

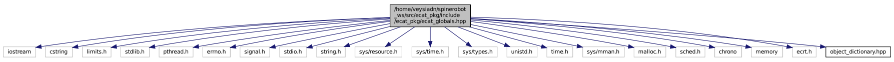
9.1.2.1 Wl

```
rpath Wl
```

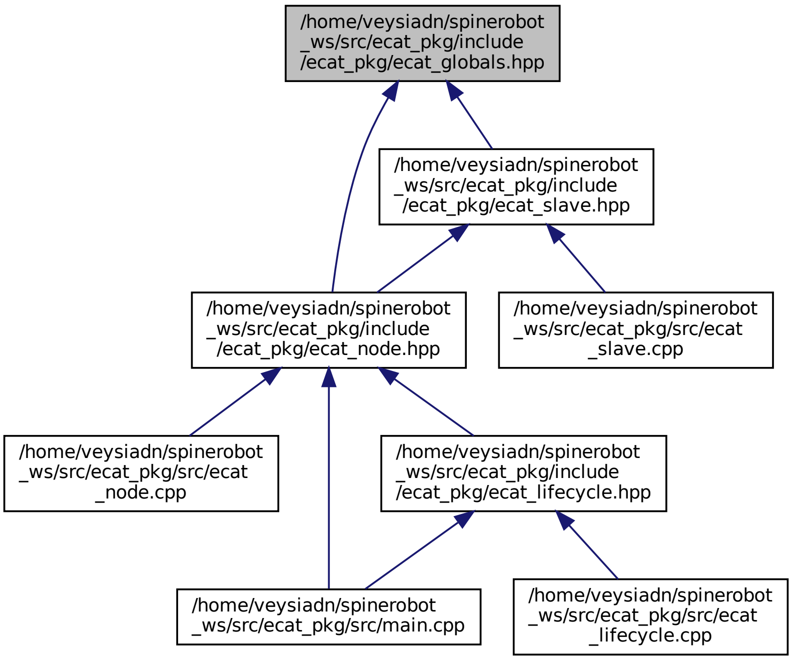
9.2 /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp File Reference

```
#include <iostream>
#include <cstring>
#include <limits.h>
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <sys/resource.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>
#include <sys/mman.h>
#include <malloc.h>
#include <sched.h>
#include <chrono>
#include <memory>
#include "ecrt.h"
#include "object_dictionary.hpp"
```

Include dependency graph for ecat_globals.hpp:



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


/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_slave.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_node.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_slave.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_node.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp

Classes

- struct Controller

  Controller Parameters.
- struct OffsetPDO

  offset for PDO entries to register PDOs.
- struct ReceivedData

  Received feedback data from slaves.
- struct SdoRequest

  EtherCAT SDO request structure for configuration phase.
- struct ProfilePosParam

  Parameters that should be specified in position mode.
- struct CSPositionModeParam

  Struct contains configuration parameters for cyclic sync. position mode.
- struct CSVelocityModeParam

  Struct contains configuration parameters for cyclic sync. velocity mode.
- struct CSTorqueModeParam

  Struct contains configuration parameters for cyclic sync. torque mode.
- struct HomingParam

  Parameters that should be specified in homing mode.
- struct ProfileVelocityParam

  Parameters that should be specified in velocity mode.

Macros

- #define NUM_OF_SLAVES 2

  Object dictionary paramaters PDO index and default values in here.
- #define FREQUENCY 1000

  Ethercat PDO exchange loop frequency in Hz.
- #define MEASURE_TIMING 1

  If you want to measure timings leave it as one, otherwise make it 0.
- #define VELOCITY_MODE 0

  set this to 1 if you want to use it in velocity mode (and set position mode 0)
- #define POSITION_MODE 0

  set this to 1 if you want to use it in position mode (and set velocity mode 0)
- #define CYCLIC_POSITION_MODE 1

  set this to 1 if you want to use it in cyclic synchronous position mode (and set velocity mode 0)
- #define GEAR_RATIO 103

  Define your own motor properties to configure increments in position mode.
- #define ENCODER_RESOLUTION 1024
- #define INC_PER_ROTATION GEAR_RATIO∗ENCODER_RESOLUTION∗4
- #define FIVE_DEGREE_CCW int(INC_PER_ROTATION/72)
- #define THIRTY_DEGREE_CCW int(INC_PER_ROTATION/12)
- #define PERIOD_NS (g_kNsPerSec/FREQUENCY)

  EtherCAT communication period in nanoseconds.
- #define PERIOD_US (PERIOD_NS / 1000)
- #define PERIOD_MS (PERIOD_US / 1000)
- #define FINAL_SLAVE (NUM_OF_SLAVES-1)
- #define TEST_BIT(NUM, N) ((NUM & (1 << N))>>N)

  To sync every cycle.
- #define SET_BIT(NUM, N) (NUM | (1 << N))
- #define RESET_BIT(NUM, N) (NUM & ∼(1 << N))
- #define TIMESPEC2NS(T) ((uint64_t) (T).tv_sec ∗ g_kNsPerSec + (T).tv_nsec)
- #define DIFF_NS(A, B) (((B).tv_sec - (A).tv_sec) ∗ g_kNsPerSec + (B).tv_nsec - (A).tv_nsec)
- #define CLOCK_TO_USE CLOCK_MONOTONIC

Enumerations

- enum OpMode {
  kProfilePosition = 1, kProfileVelocity = 3, kProfileTorque = 4, kHoming = 6,
  kInterpolatedPosition = 7, kCSPosition = 8, kCSVelocity = 9, kCSTorque = 10 }

  Motor operation modes.
- enum MotorStates {
  kReadyToSwitchOn = 1, kSwitchedOn, kOperationEnabled, kFault,
  kVoltageEnabled, kQuickStop, kSwitchOnDisabled, kWarning,
  kRemote, kTargetReached, kInternalLimitActivate }

  CIA 402 state machine motor states.
- enum ErrorRegisterBits {
  kGenericError = 0, kCurrentError, kVoltageError, kTemperatureError,
  kCommunicationError, kDeviceProfileSpecificError, kReserved, kMotionError }
- enum SensorConfig {
  kSensor1TypeNone =0, kSensor1TypeDigitalIncrementalEncoder1 =1, kSensor2TypeNone =0, kSensor2TypeDigitalIncrementa
  =256,
  kSensor2TypeAnalogIncrementalEncoderSinCos =512, kSensor2TypeSSIAbsoluteEncoder =768, kSensor3TypeNone
  =0, kSensor3TypeDigitalHallSensor =131072 }

  Sensor Configuration for motor for more information.

- enum ControlStructureBits {
kCurrentControlStructure = 0, kVelocityControlStructure = 4, kPositionControlStructure = 8, kGearLocation = 12,
kProcessValueReference = 14, kMainSensor = 16, kAuxiliarySensor = 20, kMountingPositionSensor1 = 24,
kMountingPositionSensor2 = 26, kMountingPositionSensor3 = 28 }

  Control structure configuration for control mechanism to select sensor structure specific to hardware.

Functions

- struct timespec timespec_add (struct timespec time1, struct timespec time2)

  Add two timespec struct.

Variables

- const uint32_t g_kNumberOfServoDrivers = 1

  Number of connected servo drives.
- const uint32_t g_kNsPerSec = 1000000000

  Nanoseconds per second.
- static volatile sig_atomic_t sig = 1
- ec_master_t ∗ g_master

  Extern global variable declaration.
- ec_master_state_t g_master_state

  EtherCAT master.
- ec_domain_t ∗ g_master_domain

  EtherCAT master state.
- ec_domain_state_t g_master_domain_state

  Ethercat data passing master domain.
- struct timespec g_sync_timer

  EtherCAT master domain state.
- const struct timespec g_cycle_time = {0, PERIOD_NS}

  timer for DC sync .
- uint32_t g_sync_ref_counter

  cycletime settings in ns.

9.2.1 Macro Definition Documentation

9.2.1.1 CLOCK_TO_USE

```
#define CLOCK_TO_USE CLOCK_MONOTONIC
```

9.2.1.2 CYCLIC_POSITION_MODE

```
#define CYCLIC_POSITION_MODE 1
```

set this to 1 if you want to use it in cyclic synchronous position mode (and set velocity mode 0)

9.2.1.3 DIFF_NS

```
#define DIFF_NS(
              A,
              B ) (((B).tv_sec - (A).tv_sec) * g_kNsPerSec + (B).tv_nsec - (A).tv_nsec)
```

9.2.1.4 ENCODER_RESOLUTION

```
#define ENCODER_RESOLUTION 1024
```

9.2.1.5 FINAL_SLAVE

```
#define FINAL_SLAVE (NUM_OF_SLAVES-1)
```

9.2.1.6 FIVE_DEGREE_CCW

```
#define FIVE_DEGREE_CCW int(INC_PER_ROTATION/72)
```

9.2.1.7 FREQUENCY

```
#define FREQUENCY 1000
```

Ethercat PDO exchange loop frequency in Hz.

9.2.1.8 GEAR_RATIO

```
#define GEAR_RATIO 103
```

Define your own motor properties to configure increments in position mode.

9.2.1.9 INC_PER_ROTATION

```
#define INC_PER_ROTATION GEAR_RATIO*ENCODER_RESOLUTION*4
```

9.2.1.10 MEASURE_TIMING

```
#define MEASURE_TIMING 1
```

If you want to measure timings leave it as one, otherwise make it 0.

9.2.1.11 NUM_OF_SLAVES

```
#define NUM_OF_SLAVES 2
```

Object dictionary paramaters PDO index and default values in here.

IgH EtherCAT library header file the user-space real-time interface library. IgH, EtherCAT related functions and data types. USER SHOULD DEFINE THIS AREAS Total number of connected slave to the bus.

9.2.1.12 PERIOD_MS

```
#define PERIOD_MS (PERIOD_US / 1000)
```

9.2.1.13 PERIOD_NS

```
#define PERIOD_NS (g_kNsPerSec/FREQUENCY)
```

EtherCAT communication period in nanoseconds.

9.2.1.14 PERIOD_US

```
#define PERIOD_US (PERIOD_NS / 1000)
```

9.2.1.15 POSITION_MODE

```
#define POSITION_MODE 0
```

set this to 1 if you want to use it in position mode (and set velocity mode 0)

9.2.1.16 RESET_BIT

```
#define RESET_BIT(
              NUM,
              N ) (NUM & ~(1 << N))
```

9.2.1.17 SET_BIT

```
#define SET_BIT(
              NUM,
              N ) (NUM | (1 << N))
```

9.2.1.18 TEST_BIT

```
#define TEST_BIT(
              NUM,
              N ) ((NUM & (1 << N))>>N)
```

To sync every cycle.

9.2.1.19 THIRTY_DEGREE_CCW

```
#define THIRTY_DEGREE_CCW int(INC_PER_ROTATION/12)
```

9.2.1.20 TIMESPEC2NS

```
#define TIMESPEC2NS(
              T ) ((uint64_t) (T).tv_sec * g_kNsPerSec + (T).tv_nsec)
```

9.2.1.21 VELOCITY_MODE

```
#define VELOCITY_MODE 0
```

set this to 1 if you want to use it in velocity mode (and set position mode 0)

9.2.2 Enumeration Type Documentation

9.2.2.1 ControlStructureBits

```
enum ControlStructureBits
```

Control structure configuration for control mechanism to select sensor structure specific to hardware.

See also

EPOS4-Firmware-Specification pg. 140

Enumerator

kCurrentControlStructure	These are bit locations not values for values. **See also** EPOS4-Firmware-Specification pg. 140 !!!
kVelocityControlStructure	0-3 , 4 bits. Val : 1 - PI current controller
kPositionControlStructure	4-7, 4bits. Val : 0 - None | 1 - PI Vecolity controller (low pass filter) | 2 - PI velocity controller (observer)
kGearLocation	8-11 , 4bits. Val : 0 - None | 1 - PID position controller
kProcessValueReference	1 bit Val : 0 - None | 1 - Gear Mounted on system
kMainSensor	14-15 2 bits. Val : 0 - On motor (or undefined) | 1 - On gear
kAuxiliarySensor	16-19 4 bits. Val : 0 - None | 1 - Sensor 1 | 2 - Sensor 2 | 3 - Sensor 3
kMountingPositionSensor1	20-23 4 bits. Val : 0 - None | 1 - Sensor 1 | 2 - Sensor 2 | 3 - Sensor 3
kMountingPositionSensor2	24-25 2 bits. Val : 0 - On motor (or undefined) | 1 - On gear
kMountingPositionSensor3	26-27 2 bits. Val : 0 - On motor (or undefined) | 1 - On gear

9.2.2.2 ErrorRegisterBits

```
enum ErrorRegisterBits
```

Enumerator

kGenericError	
kCurrentError	
kVoltageError	
kTemperatureError	
kCommunicationError	
kDeviceProfileSpecificError	
kReserved	
kMotionError	

9.2.2.3 MotorStates

```
enum MotorStates
```

CIA 402 state machine motor states.

Enumerator

kReadyToSwitchOn	
kSwitchedOn	

Enumerator

kOperationEnabled	
kFault	
kVoltageEnabled	
kQuickStop	
kSwitchOnDisabled	
kWarning	
kRemote	
kTargetReached	
kInternalLimitActivate	

9.2.2.4 OpMode

```
enum OpMode
```

Motor operation modes.

Enumerator

kProfilePosition	
kProfileVelocity	
kProfileTorque	
kHoming	
kInterpolatedPosition	
kCSPosition	
kCSVelocity	
kCSTorque	

9.2.2.5 SensorConfig

```
enum SensorConfig
```

Sensor Configuration for motor for more information.

See also

EPOS4-Firmware-Specification Pg.138

Enumerator

kSensor1TypeNone	
kSensor1TypeDigitalIncrementalEncoder1	
kSensor2TypeNone	

Enumerator

kSensor2TypeDigitalIncrementalEncoder2	
kSensor2TypeAnalogIncrementalEncoderSinCos	
kSensor2TypeSSIAbsoluteEncoder	
kSensor3TypeNone	
kSensor3TypeDigitalHallSensor	

9.2.3 Function Documentation

9.2.3.1 timespec_add()

```
struct timespec timespec_add (
            struct timespec time1,
            struct timespec time2 )  [inline]
```

Add two timespec struct.

Parameters

time1	Timespec struct 1
time2	Timespec struct 2

Returns

Addition result

9.2.4 Variable Documentation

9.2.4.1 g_cycle_time

```
const struct timespec g_cycle_time = {0, PERIOD_NS}
```

timer for DC sync .

9.2.4.2 g_kNsPerSec

```
const uint32_t g_kNsPerSec = 1000000000
```

Nanoseconds per second.

9.2.4.3 g_kNumberOfServoDrivers

```
const uint32_t g_kNumberOfServoDrivers = 1
```

Number of connected servo drives.

9.2.4.4 g_master

```
ec_master_t* g_master
```

Extern global variable declaration.

9.2.4.5 g_master_domain

```
ec_domain_t* g_master_domain
```

EtherCAT master state.

9.2.4.6 g_master_domain_state

```
ec_domain_state_t g_master_domain_state
```

Ethercat data passing master domain.

9.2.4.7 g_master_state

```
ec_master_state_t g_master_state
```

EtherCAT master.

9.2.4.8 g_sync_ref_counter

```
uint32_t g_sync_ref_counter
```

cycletime settings in ns.

9.2.4.9 g_sync_timer

```
struct timespec g_sync_timer
```

EtherCAT master domain state.

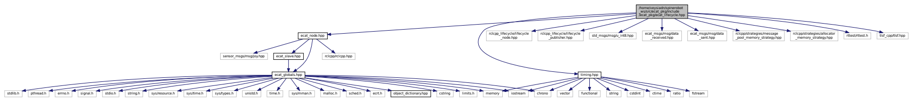
9.2.4.10 sig

```
volatile sig_atomic_t sig = 1  [static]
```

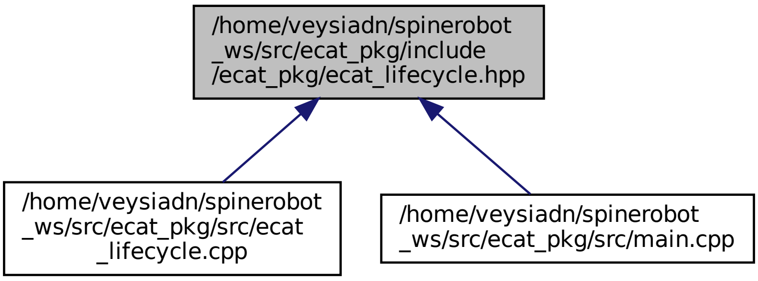
9.3 /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat↩ _lifecycle.hpp File Reference

```
#include "ecat_node.hpp"
#include "timing.hpp"
#include <rclcpp_lifecycle/lifecycle_node.hpp>
#include <rclcpp_lifecycle/lifecycle_publisher.hpp>
#include "std_msgs/msg/u_int8.hpp"
#include "ecat_msgs/msg/data_received.hpp"
#include "ecat_msgs/msg/data_sent.hpp"
#include <rclcpp/strategies/message_pool_memory_strategy.hpp>
#include <rclcpp/strategies/allocator_memory_strategy.hpp>
#include <rttest/rttest.h>
#include <tlsf_cpp/tlsf.hpp>
```

Include dependency graph for ecat_lifecycle.hpp:



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


/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp

Classes

- class EthercatLifeCycleNode::EthercatLifeCycle

Namespaces

- EthercatLifeCycleNode

Typedefs

- template<typename T = void>
  using TLSFAllocator = tlsf_heap_allocator< T >

9.3.1 Typedef Documentation

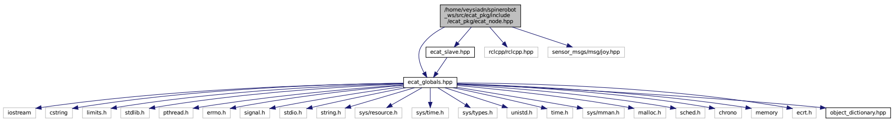
9.3.1.1 TLSFAllocator

```
template<typename T = void>
using TLSFAllocator = tlsf_heap_allocator<T>
```

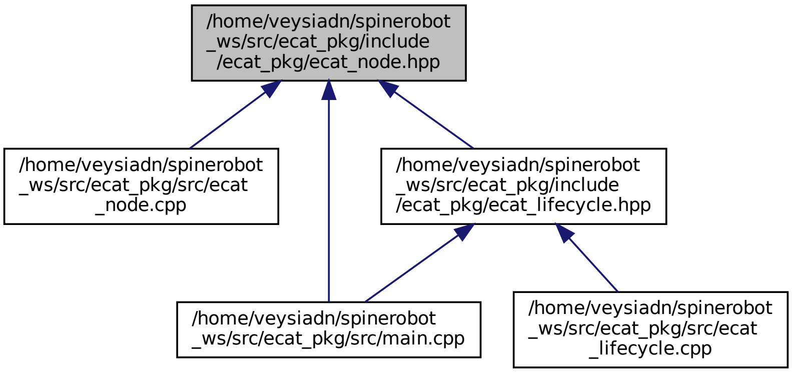
9.4 /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_node.hpp File Reference

```
#include "ecat_globals.hpp"
#include "ecat_slave.hpp"
#include <rclcpp/rclcpp.hpp>
#include "sensor_msgs/msg/joy.hpp"
```

Include dependency graph for ecat_node.hpp:



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


/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_node.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_node.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp

Classes

- class EthercatCommunication::EthercatNode

Namespaces

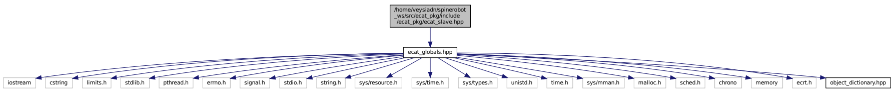
- EthercatCommunication

  ROS2 Headers.

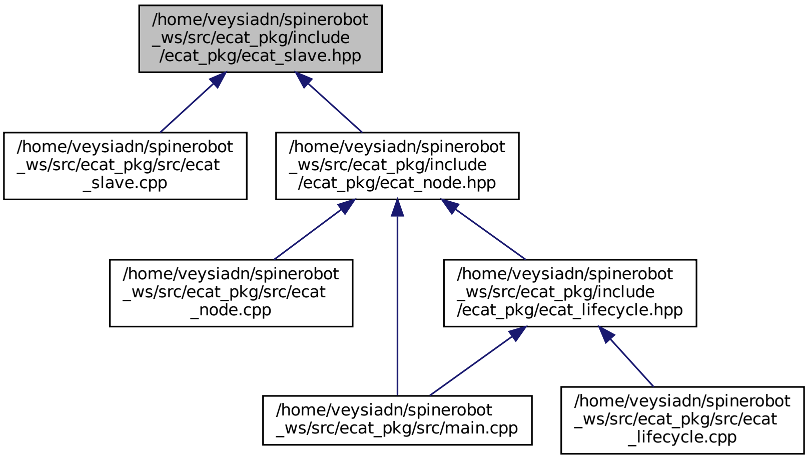
9.5 /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat↩_slave.hpp File Reference

```
#include "ecat_globals.hpp"
```

Include dependency graph for ecat_slave.hpp:



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

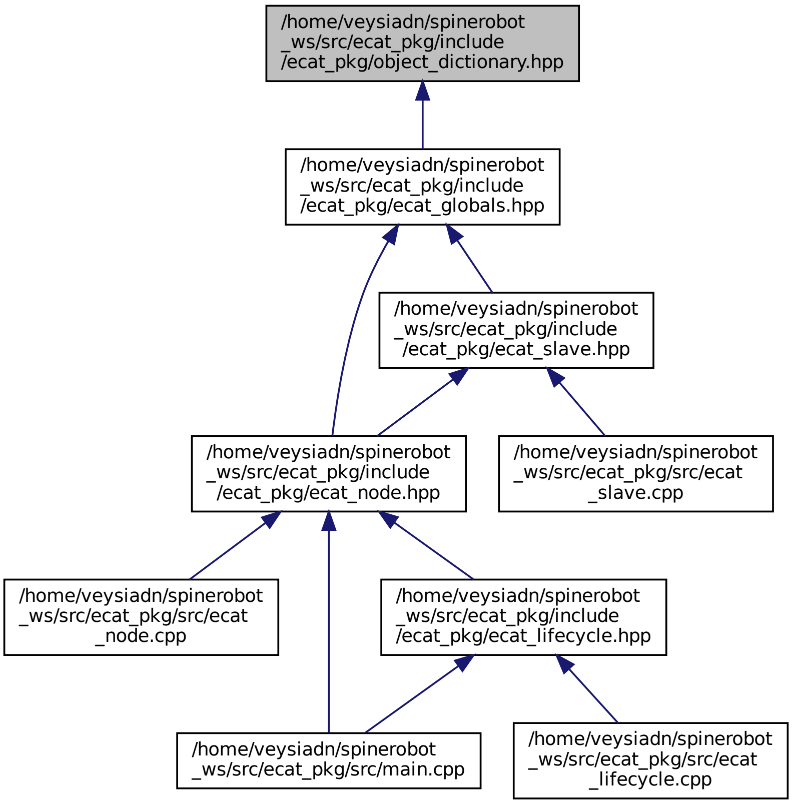

/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_slave.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_slave.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_node.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_node.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp

Classes

- class EthercatSlave

9.6 /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_↩ pkg/object_dictionary.hpp File Reference

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


/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/object_dictionary.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_globals.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_slave.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_node.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_slave.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_node.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp

Macros

- #define OD_CONTROL_WORD 0x6040,0x00
- #define OD_STATUS_WORD 0x6041,0x00
- #define OD_OPERATION_MODE 0x6060,0x00
- #define OD_OPERATION_MODE_DISPLAY 0x6061,0x00
- #define OD_TARGET_POSITION 0x607A,0x00
- #define OD_POSITION_ACTUAL_VAL 0x6064,0x00
- #define OD_POSITION_DEMAND 0x6062,0x00
- #define OD_POSITON_FOLLOWING_ERROR 0x60F4,0X00
- #define OD_POSITONCOUNTS 0x6063,0x00
- #define OD_MAX_FOLLOWING_ERROR 0x6065,0x00
- #define OD_VELOCITY_ACTUAL_VALUE 0x606C,0x00
- #define OD_VELOCITY_ADDRESS 0x6069,0x00
- #define OD_TARGET_VELOCITY 0x60FF,0x00
- #define OD_VELOCITY_OFFSET 0x60B1,0x00
- #define OD_INTERPOLATION_TIME_PERIOD 0X60C2,0X01

- #define OD_INTERPOLATION_TIME_UNIT 0X60C2,0X02
- #define OD_MOTOR_DATA_NOMINAL_CURRENT 0X3001,0X01
- #define OD_MOTOR_DATA_OUTPUT_CURRENT_LIMIT 0X3001,0X02
- #define OD_MOTOR_DATA_NUMBER_OF_POLE_PAIRS 0X3001,0X03
- #define OD_MOTOR_DATA_THERMAL_TIME_CONSTANT_WINDINGS 0X3001,0X04
- #define OD_MOTOR_DATA_TORQUE_CONSTANT 0X3001,0X05
- #define OD_GEAR_REDUCTION_NUMERATOR 0X3003,0X01
- #define OD_GEAR_REDUCTION_DENOMINATOR 0X3003,0X02
- #define OD_GEAR_MAX_INPUT_SPEED 0X3003,0X03
- #define OD_GEAR_MISC_CONFIGURATION 0X3003,0X04
- #define OD_DIGITAL_INCREMENTAL_ENCODER_1_TYPE 0X3010,0X02
- #define OD_ERROR_REGISTER 0X1001,0X00
- #define OD_ACTIVE_FIELDBUS 0X2010,0X00
- #define OD_CUSTOM_PERSISTENT_MEMORY_1 0X210C,0X01
- #define OD_CUSTOM_PERSISTENT_MEMORY_2 0X210C,0X02
- #define OD_CUSTOM_PERSISTENT_MEMORY_3 0X210C,0X03
- #define OD_CUSTOM_PERSISTENT_MEMORY_4 0X210C,0X04
- #define OD_POWER_SUPPLY_VOLTAGE 0X2200,0X01
- #define OD_SENSOR_CONFIGURATION 0X3000,0X01

  Object Dictionary for Axis configurations.
- #define OD_CONTROL_STRUCTURE 0X3000,0X02
- #define OD_PROFILE_VELOCITY 0x6081,0x00
- #define OD_MAX_PROFILE_VELOCITY 0x6080,0x00
- #define OD_MAX_MOTOR_SPEED 0X607F,0X00
- #define OD_PROFILE_ACCELERATION 0x6083,0x00
- #define OD_PROFILE_DECELERATION 0x6084,0x00
- #define OD_QUICK_STOP_DECELERATION 0x6085,0x00
- #define OD_MOTION_PROFILE_TYPE 0x6086,0x00
- #define OD_LINEAR_RAMP_TRAPEZOIDAL 0x00,0x00
- #define OD_VELOCITY_ENCODER_RESOLUTION_NUM 0x6094,0x01
- #define OD_VELOCITY_ENCODER_RESOLUTION_DEN 0x6094,0x02
- #define OD_DIGITAL_INPUTS 0x60FD,0x00
- #define OD_DIGITAL_OUTPUTS 0x60FE,0x01
- #define OD_DC_CIRCUIT_LINK_VOLTAGE 0x6079,0x00
- #define OD_TARGET_TORQUE 0x6071,0x00
- #define OD_TORQUE_MAX 0x6072,0x00
- #define OD_TORQUE_ACTUAL_VALUE 0x6077,0x00
- #define OD_TORQUE_OFFSET 0x60b2,0x00
- #define OD_MAX_CURRENT 0x6073, 0x00
- #define OD_CURRENT_ACTUAL_VALUE 0x6078, 0x00
- #define OD_ERROR_CODE 0x603F, 0x00
- #define OD_EXTRA_STATUS_REGISTER 0x2085, 0x00
- #define OD_CHECK_ERROR 0x1002,0x00
- #define OD_QUICK_STOP_MODE 0x605A,0x00
- #define OD_STOP_OPTION_CODE 0x605D,0x00
- #define SM_COMM_RESET 0x81
- #define SM_FULL_RESET 0x82
- #define SM_START 0x01
- #define SM_GO_READY_TO_SWITCH_ON 0x06
- #define SM_GO_SWITCH_ON 0x07
- #define SM_GO_ENABLE 0X0F
- #define SM_GO_SWITCH_ON_DISABLE 0x00
- #define SM_RUN 0x1F
- #define SM_EXPEDITE 0x3F

- #define SM_QUICKSTOP 0x02
- #define SM_RELATIVE_POS 0X7F
- #define SM_FSAFROMSTATUSWORD(SW) (SW & 0x006f)
- #define SM_NOT_READY_TO_SWITCH_ON 0b00000000
- #define SM_NOT_READY_TO_SWITCH_ON_2 0b00100000
- #define SM_SWITCH_ON_DISABLED 0b01000000
- #define SM_SWITCH_ON_DISABLED_2 0b01100000
- #define SM_READY_TO_SWITCH_ON 0b00100001
- #define SM_SWITCHED_ON 0b00100011
- #define SM_OPERATION_ENABLED 0b00100111
- #define SM_QUICK_STOP_ACTIVE 0b00000111
- #define SM_FAULT_REACTION_ACTIVE 0b00001111
- #define SM_FAULTREACTIONACTIVE2 0b00101111
- #define SM_FAULT 0b00001000
- #define SM_FAULT2 0b00101000
- #define SM_SW_READY_TO_SWITCH_ON 0x0001
- #define SM_SW_SWITCHED_ON 0x0002
- #define SM_SW_OPERATION_ENABLED 0x0004
- #define SM_SW_FAULT 0x0008
- #define SM_SW_VOLTAGE_ENABLED 0x0010
- #define SM_SW_QUICK_STOP 0x0020
- #define SM_SW_SWITCH_ON_DISABLED 0x0040
- #define SM_SW_WARNING 0x0080
- #define SM_SW_REMOTE 0x0200
- #define SM_SW_TARGET_REACHED 0x0400
- #define SM_SW_INTERNAL_LIMIT_ACTIVE 0x0800
- #define SM_CW_SWITCH_ON 0x0001
- #define SM_CW_ENABLE_VOLTAGE 0x0002
- #define SM_CW_QUICK_STOP 0x0004
- #define SM_CW_ENABLE_OPERATION 0x0008
- #define SM_CW_FAULT_RESET 0x0080
- #define SM_CW_OD_HALT 0x0100

9.6.1 Macro Definition Documentation

9.6.1.1 OD_ACTIVE_FIELDBUS

```
#define OD_ACTIVE_FIELDBUS 0X2010,0X00
```

9.6.1.2 OD_CHECK_ERROR

```
#define OD_CHECK_ERROR 0x1002,0x00
```

9.6.1.3 OD_CONTROL_STRUCTURE

```
#define OD_CONTROL_STRUCTURE 0X3000,0X02
```

9.6.1.4 OD_CONTROL_WORD

```
#define OD_CONTROL_WORD 0x6040,0x00
```

9.6.1.5 OD_CURRENT_ACTUAL_VALUE

```
#define OD_CURRENT_ACTUAL_VALUE 0x6078, 0x00
```

9.6.1.6 OD_CUSTOM_PERSISTENT_MEMORY_1

```
#define OD_CUSTOM_PERSISTENT_MEMORY_1 0X210C,0X01
```

9.6.1.7 OD_CUSTOM_PERSISTENT_MEMORY_2

```
#define OD_CUSTOM_PERSISTENT_MEMORY_2 0X210C,0X02
```

9.6.1.8 OD_CUSTOM_PERSISTENT_MEMORY_3

```
#define OD_CUSTOM_PERSISTENT_MEMORY_3 0X210C,0X03
```

9.6.1.9 OD_CUSTOM_PERSISTENT_MEMORY_4

```
#define OD_CUSTOM_PERSISTENT_MEMORY_4 0X210C,0X04
```

9.6.1.10 OD_DC_CIRCUIT_LINK_VOLTAGE

```
#define OD_DC_CIRCUIT_LINK_VOLTAGE 0x6079,0x00
```

9.6.1.11 OD_DIGITAL_INCREMENTAL_ENCODER_1_TYPE

```
#define OD_DIGITAL_INCREMENTAL_ENCODER_1_TYPE 0X3010,0X02
```

9.6.1.12 OD_DIGITAL_INPUTS

```
#define OD_DIGITAL_INPUTS 0x60FD,0x00
```

9.6.1.13 OD_DIGITAL_OUTPUTS

```
#define OD_DIGITAL_OUTPUTS 0x60FE,0x01
```

9.6.1.14 OD_ERROR_CODE

```
#define OD_ERROR_CODE 0x603F, 0x00
```

9.6.1.15 OD_ERROR_REGISTER

```
#define OD_ERROR_REGISTER 0X1001,0X00
```

9.6.1.16 OD_EXTRA_STATUS_REGISTER

```
#define OD_EXTRA_STATUS_REGISTER 0x2085, 0x00
```

9.6.1.17 OD_GEAR_MAX_INPUT_SPEED

```
#define OD_GEAR_MAX_INPUT_SPEED 0X3003,0X03
```

9.6.1.18 OD_GEAR_MISC_CONFIGURATION

```
#define OD_GEAR_MISC_CONFIGURATION 0X3003,0X04
```

9.6.1.19 OD_GEAR_REDUCTION_DENOMINATOR

```
#define OD_GEAR_REDUCTION_DENOMINATOR 0X3003,0X02
```

9.6.1.20 OD_GEAR_REDUCTION_NUMERATOR

```
#define OD_GEAR_REDUCTION_NUMERATOR 0X3003,0X01
```

9.6.1.21 OD_INTERPOLATION_TIME_PERIOD

```
#define OD_INTERPOLATION_TIME_PERIOD 0X60C2,0X01
```

9.6.1.22 OD_INTERPOLATION_TIME_UNIT

```
#define OD_INTERPOLATION_TIME_UNIT 0X60C2,0X02
```

9.6.1.23 OD_LINEAR_RAMP_TRAPEZOIDAL

```
#define OD_LINEAR_RAMP_TRAPEZOIDAL 0x00,0x00
```

9.6.1.24 OD_MAX_CURRENT

```
#define OD_MAX_CURRENT 0x6073, 0x00
```

9.6.1.25 OD_MAX_FOLLOWING_ERROR

```
#define OD_MAX_FOLLOWING_ERROR 0x6065,0x00
```

9.6.1.26 OD_MAX_MOTOR_SPEED

```
#define OD_MAX_MOTOR_SPEED 0X607F,0X00
```

9.6.1.27 OD_MAX_PROFILE_VELOCITY

```
#define OD_MAX_PROFILE_VELOCITY 0x6080,0x00
```

9.6.1.28 OD_MOTION_PROFILE_TYPE

```
#define OD_MOTION_PROFILE_TYPE 0x6086,0x00
```

9.6.1.29 OD_MOTOR_DATA_NOMINAL_CURRENT

```
#define OD_MOTOR_DATA_NOMINAL_CURRENT 0X3001,0X01
```

9.6.1.30 OD_MOTOR_DATA_NUMBER_OF_POLE_PAIRS

```
#define OD_MOTOR_DATA_NUMBER_OF_POLE_PAIRS 0X3001,0X03
```

9.6.1.31 OD_MOTOR_DATA_OUTPUT_CURRENT_LIMIT

```
#define OD_MOTOR_DATA_OUTPUT_CURRENT_LIMIT 0X3001,0X02
```

9.6.1.32 OD_MOTOR_DATA_THERMAL_TIME_CONSTANT_WINDINGS

```
#define OD_MOTOR_DATA_THERMAL_TIME_CONSTANT_WINDINGS 0X3001,0X04
```

9.6.1.33 OD_MOTOR_DATA_TORQUE_CONSTANT

```
#define OD_MOTOR_DATA_TORQUE_CONSTANT 0X3001,0X05
```

9.6.1.34 OD_OPERATION_MODE

```
#define OD_OPERATION_MODE 0x6060,0x00
```

9.6.1.35 OD_OPERATION_MODE_DISPLAY

```
#define OD_OPERATION_MODE_DISPLAY 0x6061,0x00
```

9.6.1.36 OD_POSITION_ACTUAL_VAL

```
#define OD_POSITION_ACTUAL_VAL 0x6064,0x00
```

9.6.1.37 OD_POSITION_DEMAND

```
#define OD_POSITION_DEMAND 0x6062,0x00
```

9.6.1.38 OD_POSITON_FOLLOWING_ERROR

```
#define OD_POSITON_FOLLOWING_ERROR 0x60F4,0X00
```

9.6.1.39 OD_POSITONCOUNTS

```
#define OD_POSITONCOUNTS 0x6063,0x00
```

9.6.1.40 OD_POWER_SUPPLY_VOLTAGE

```
#define OD_POWER_SUPPLY_VOLTAGE 0X2200,0X01
```

9.6.1.41 OD_PROFILE_ACCELERATION

```
#define OD_PROFILE_ACCELERATION 0x6083,0x00
```

9.6.1.42 OD_PROFILE_DECELERATION

```
#define OD_PROFILE_DECELERATION 0x6084,0x00
```

9.6.1.43 OD_PROFILE_VELOCITY

```
#define OD_PROFILE_VELOCITY 0x6081,0x00
```

9.6.1.44 OD_QUICK_STOP_DECELERATION

```
#define OD_QUICK_STOP_DECELERATION 0x6085,0x00
```

9.6.1.45 OD_QUICK_STOP_MODE

```
#define OD_QUICK_STOP_MODE 0x605A,0x00
```

9.6.1.46 OD_SENSOR_CONFIGURATION

```
#define OD_SENSOR_CONFIGURATION 0X3000,0X01
```

Object Dictionary for Axis configurations.

9.6.1.47 OD_STATUS_WORD

```
#define OD_STATUS_WORD 0x6041,0x00
```

9.6.1.48 OD_STOP_OPTION_CODE

```
#define OD_STOP_OPTION_CODE 0x605D,0x00
```

9.6.1.49 OD_TARGET_POSITION

```
#define OD_TARGET_POSITION 0x607A,0x00
```

9.6.1.50 OD_TARGET_TORQUE

```
#define OD_TARGET_TORQUE 0x6071,0x00
```

9.6.1.51 OD_TARGET_VELOCITY

```
#define OD_TARGET_VELOCITY 0x60FF,0x00
```

9.6.1.52 OD_TORQUE_ACTUAL_VALUE

```
#define OD_TORQUE_ACTUAL_VALUE 0x6077,0x00
```

9.6.1.53 OD_TORQUE_MAX

```
#define OD_TORQUE_MAX 0x6072,0x00
```

9.6.1.54 OD_TORQUE_OFFSET

```
#define OD_TORQUE_OFFSET 0x60b2,0x00
```

9.6.1.55 OD_VELOCITY_ACTUAL_VALUE

```
#define OD_VELOCITY_ACTUAL_VALUE 0x606C,0x00
```

9.6.1.56 OD_VELOCITY_ADDRESS

```
#define OD_VELOCITY_ADDRESS 0x6069,0x00
```

9.6.1.57 OD_VELOCITY_ENCODER_RESOLUTION_DEN

```
#define OD_VELOCITY_ENCODER_RESOLUTION_DEN 0x6094,0x02
```

9.6.1.58 OD_VELOCITY_ENCODER_RESOLUTION_NUM

```
#define OD_VELOCITY_ENCODER_RESOLUTION_NUM 0x6094,0x01
```

9.6.1.59 OD_VELOCITY_OFFSET

```
#define OD_VELOCITY_OFFSET 0x60B1,0x00
```

9.6.1.60 SM_COMM_RESET

```
#define SM_COMM_RESET 0x81
```

9.6.1.61 SM_CW_ENABLE_OPERATION

```
#define SM_CW_ENABLE_OPERATION 0x0008
```

9.6.1.62 SM_CW_ENABLE_VOLTAGE

```
#define SM_CW_ENABLE_VOLTAGE 0x0002
```

9.6.1.63 SM_CW_FAULT_RESET

```
#define SM_CW_FAULT_RESET 0x0080
```

9.6.1.64 SM_CW_OD_HALT

```
#define SM_CW_OD_HALT 0x0100
```

9.6.1.65 SM_CW_QUICK_STOP

```
#define SM_CW_QUICK_STOP 0x0004
```

9.6.1.66 SM_CW_SWITCH_ON

```
#define SM_CW_SWITCH_ON 0x0001
```

9.6.1.67 SM_EXPEDITE

```
#define SM_EXPEDITE 0x3F
```

9.6.1.68 SM_FAULT

```
#define SM_FAULT 0b00001000
```

9.6.1.69 SM_FAULT2

```
#define SM_FAULT2 0b00101000
```

9.6.1.70 SM_FAULT_REACTION_ACTIVE

```
#define SM_FAULT_REACTION_ACTIVE 0b00001111
```

9.6.1.71 SM_FAULTREACTIONACTIVE2

```
#define SM_FAULTREACTIONACTIVE2 0b00101111
```

9.6.1.72 SM_FSAFROMSTATUSWORD

```
#define SM_FSAFROMSTATUSWORD(
              SW ) (SW & 0x006f)
```

9.6.1.73 SM_FULL_RESET

```
#define SM_FULL_RESET 0x82
```

9.6.1.74 SM_GO_ENABLE

```
#define SM_GO_ENABLE 0X0F
```

9.6.1.75 SM_GO_READY_TO_SWITCH_ON

```
#define SM_GO_READY_TO_SWITCH_ON 0x06
```

9.6.1.76 SM_GO_SWITCH_ON

```
#define SM_GO_SWITCH_ON 0x07
```

9.6.1.77 SM_GO_SWITCH_ON_DISABLE

```
#define SM_GO_SWITCH_ON_DISABLE 0x00
```

9.6.1.78 SM_NOT_READY_TO_SWITCH_ON

```
#define SM_NOT_READY_TO_SWITCH_ON 0b00000000
```

9.6.1.79 SM_NOT_READY_TO_SWITCH_ON_2

```
#define SM_NOT_READY_TO_SWITCH_ON_2 0b00100000
```

9.6.1.80 SM_OPERATION_ENABLED

```
#define SM_OPERATION_ENABLED 0b00100111
```

9.6.1.81 SM_QUICK_STOP_ACTIVE

```
#define SM_QUICK_STOP_ACTIVE 0b00000111
```

9.6.1.82 SM_QUICKSTOP

```
#define SM_QUICKSTOP 0x02
```

9.6.1.83 SM_READY_TO_SWITCH_ON

```
#define SM_READY_TO_SWITCH_ON 0b00100001
```

9.6.1.84 SM_RELATIVE_POS

```
#define SM_RELATIVE_POS 0X7F
```

9.6.1.85 SM_RUN

```
#define SM_RUN 0x1F
```

9.6.1.86 SM_START

```
#define SM_START 0x01
```

9.6.1.87 SM_SW_FAULT

```
#define SM_SW_FAULT 0x0008
```

9.6.1.88 SM_SW_INTERNAL_LIMIT_ACTIVE

```
#define SM_SW_INTERNAL_LIMIT_ACTIVE 0x0800
```

9.6.1.89 SM_SW_OPERATION_ENABLED

```
#define SM_SW_OPERATION_ENABLED 0x0004
```

9.6.1.90 SM_SW_QUICK_STOP

```
#define SM_SW_QUICK_STOP 0x0020
```

9.6.1.91 SM_SW_READY_TO_SWITCH_ON

```
#define SM_SW_READY_TO_SWITCH_ON 0x0001
```

9.6.1.92 SM_SW_REMOTE

```
#define SM_SW_REMOTE 0x0200
```

9.6.1.93 SM_SW_SWITCH_ON_DISABLED

```
#define SM_SW_SWITCH_ON_DISABLED 0x0040
```

9.6.1.94 SM_SW_SWITCHED_ON

```
#define SM_SW_SWITCHED_ON 0x0002
```

9.6.1.95 SM_SW_TARGET_REACHED

```
#define SM_SW_TARGET_REACHED 0x0400
```

9.6.1.96 SM_SW_VOLTAGE_ENABLED

```
#define SM_SW_VOLTAGE_ENABLED 0x0010
```

9.6.1.97 SM_SW_WARNING

```
#define SM_SW_WARNING 0x0080
```

9.6.1.98 SM_SWITCH_ON_DISABLED

```
#define SM_SWITCH_ON_DISABLED 0b01000000
```

9.6.1.99 SM_SWITCH_ON_DISABLED_2

```
#define SM_SWITCH_ON_DISABLED_2 0b01100000
```

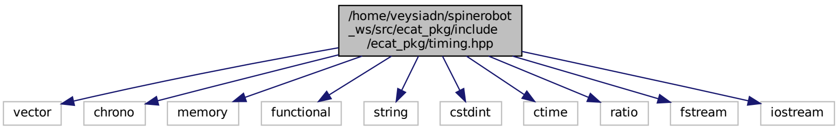
9.6.1.100 SM_SWITCHED_ON

```
#define SM_SWITCHED_ON 0b00100011
```

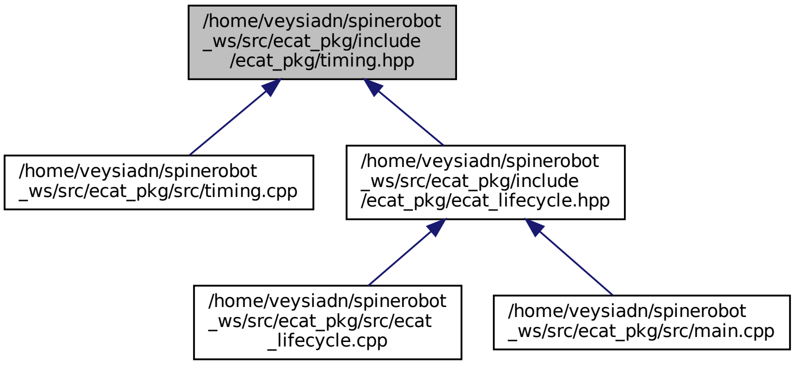
9.7 /home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/timing.hpp File Reference

```
#include <vector>
#include <chrono>
#include <memory>
#include <functional>
#include <string>
#include <cstdint>
#include <ctime>
#include <ratio>
#include <fstream>
#include <iostream>
```

Include dependency graph for timing.hpp:



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


/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/timing.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/timing.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/include/ecat_pkg/ecat_lifecycle.hpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp
/home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp

Classes

- class Timing

Macros

- #define NUMBER_OF_SAMPLES 11E4

9.7.1 Macro Definition Documentation

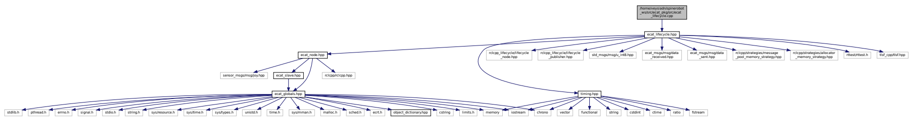
9.7.1.1 NUMBER_OF_SAMPLES

```
#define NUMBER_OF_SAMPLES 11E4
```

9.8 /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_lifecycle.cpp File Reference

```
#include <ecat_lifecycle.hpp>
```

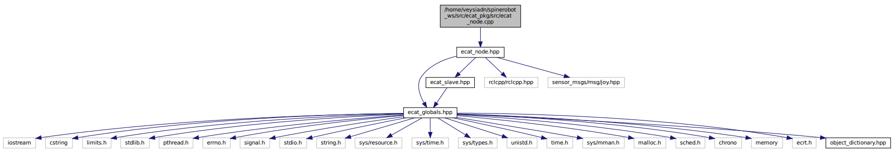
Include dependency graph for ecat_lifecycle.cpp:



9.9 /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_node.cpp File Reference

```
#include "ecat_node.hpp"
```

Include dependency graph for ecat_node.cpp:



Variables

- ec_master_t ∗ g_master = NULL

  Extern global variable declaration.
- ec_master_state_t g_master_state = {}

  EtherCAT master.
- ec_domain_t ∗ g_master_domain = NULL

  EtherCAT master state.
- ec_domain_state_t g_master_domain_state = {}

  Ethercat data passing master domain.
- struct timespec g_sync_timer

  EtherCAT master domain state.
- uint32_t g_sync_ref_counter = 0

  cycletime settings in ns.

9.9.1 Variable Documentation

9.9.1.1 g_master

```
ec_master_t* g_master = NULL
```

Extern global variable declaration.

9.9.1.2 g_master_domain

```
ec_domain_t* g_master_domain = NULL
```

EtherCAT master state.

9.9.1.3 g_master_domain_state

```
ec_domain_state_t g_master_domain_state = {}
```

Ethercat data passing master domain.

9.9.1.4 g_master_state

```
ec_master_state_t g_master_state = {}
```

EtherCAT master.

9.9.1.5 g_sync_ref_counter

```
uint32_t g_sync_ref_counter = 0
```

cycletime settings in ns.

9.9.1.6 g_sync_timer

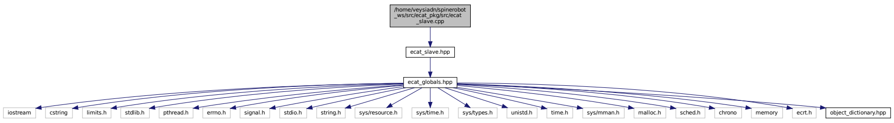
```
struct timespec g_sync_timer
```

EtherCAT master domain state.

9.10 /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/ecat_slave.cpp File Reference

```
#include "ecat_slave.hpp"
```

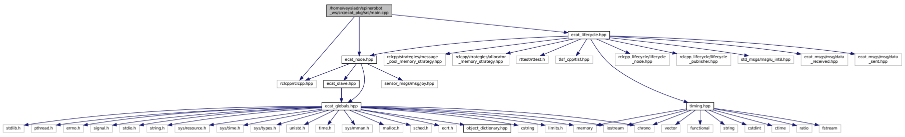
Include dependency graph for ecat_slave.cpp:



9.11 /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/main.cpp File Reference

```
#include "rclcpp/rclcpp.hpp"
#include "ecat_node.hpp"
#include "ecat_lifecycle.hpp"
```

Include dependency graph for main.cpp:



Functions

- void signalHandler (int)
- int main (int argc, char ∗∗argv)

Variables

- std::unique_ptr< EthercatLifeCycleNode::EthercatLifeCycle > ecat_lifecycle_node

9.11.1 Function Documentation

9.11.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

9.11.1.2 signalHandler()

```
void signalHandler (
            int )
```

9.11.2 Variable Documentation

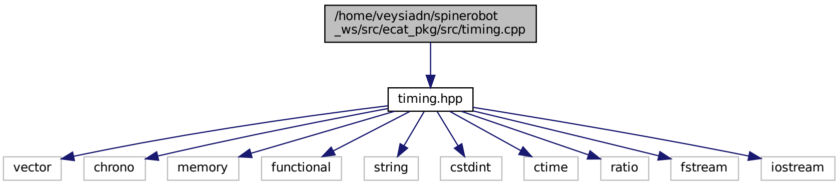
9.11.2.1 ecat_lifecycle_node

```
std::unique_ptr<EthercatLifeCycleNode::EthercatLifeCycle> ecat_lifecycle_node
```

9.12 /home/veysiadn/spinerobot_ws/src/ecat_pkg/src/timing.cpp File Reference

```
#include "timing.hpp"
```

Include dependency graph for timing.cpp:



Index