ZebOS-XP®
Network Platform

Version 1.4
Extended Performance

Synchronous Ethernet
Developer Guide

December 2015

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900
http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:
support@ipinfusion.com

Contents

Preface

This guide describes the ZebOS-XP application programming interface (API) for Synchronous Ethernet (SyncE).

Audience

This guide is intended for developers who write code to customize and extend Synchronous Ethernet.

Conventions

Table P-1 shows the conventions used in this guide.

Table P-1: Conventions

Convention	Description
Italics	Emphasized terms; titles of books
Note:	Special instructions, suggestions, or warnings
`monospaced type`	Code elements such as commands, functions, parameters, files, and directories

Contents

This guide contains these chapters:

- Chapter 1, *Synchronous Ethernet Overview*
- Chapter 2, *Synchronous Ethernet Command API*

Related Documents

The following guides are related to this document:

- *Synchronous Ethernet Command Reference*
- *Synchronous Ethernet Configuration Guide*

For more about Provider Backbone Bridging, see the following:

- *Carrier Ethernet Command Reference*
- *Carrier Ethernet Developer Guide*
- *Carrier Ethernet Configuration Guide*

Note: All ZebOS-XP technical manuals are available to licensed customers at http://www.ipinfusion.com/support/document_list.

Support

For support-related questions, contact support@ipinfusion.com.

Comments

If you have comments, or need to report a problem with the content, contact techpubs@ipinfusion.com.

CHAPTER 1 Synchronous Ethernet Overview

This chapter is an overview of the ZebOS-XP implementation of Synchronous Ethernet which distributes frequency over Ethernet physical links.

Frequency Synchronization in TDM Networks

Telecommunications systems are based on time-division multiplexing (TDM) technologies such as T1/E1, Synchronous Optical Networking (SONET), Synchronous Digital Hierarchy (SDH), and Plesiochronous Digital Hierarchy (PDH) which transmit constant bit rate (CBR) traffic such as digitized voice and video. TDM technologies support small transmission delay and small transmission delay variation—two key parameters for voice and video transmission. Small transmission delay can only be achieved if data buffering at each node is minimal. This implies that all nodes in a TDM-based network must be tightly synchronized to a common clock (frequency) to prevent data loss.

Isolated physical clocks cannot be expected to agree with each other over time. If different frequencies are used on opposite sides of a TDM link, frequency discrepancies can lead to valid bits being lost (overflow) and invalid bits being inserted (underflow), in a phenomena known as “bit slips”.

To avoid bit slips, every TDM network has a clock hierarchy:

- At the top is the most accurate clock which is called is the Primary Reference Clock (PRC) or Primary Reference Source (PRS).
- At the next level of hierarchy is the Synchronization Supply Unit (SSU) or Building Integrated Timing Supply (BITS).
- Below that, each “network element” (switch) contains an internal clock called the SDH Equipment Clock (SEC) or SONET Minimum Clock (SMC). Normally the SEC/SMC is locked in frequency to one of its incoming traffic line signals (from a PRS/PRC, an SSU/BITS, or another SEC/SMC), while all outgoing line signals are timed by the SEC/SMC (this is called the locked mode).

In this way, a master-slave tree is built where synchronization travels from the PRC/PRS down all the clock chains formed by the branches of SECs/SMCs. Clocks derived in this manner are said to be “traceable” to a PRC/PRS.

Frequency Synchronization in Ethernet

Packet -witching networks such as Ethernet were designed to transport asynchronous data where strict synchronization is not needed because traffic is highly buffered. However, with current Carrier Ethernet evolution, Ethernet is replacing the SONET/SDH/PDH infrastructure and carrying TDM traffic.This means that packet networks need frequency synchronization functions, especially at their edge. The adaptation of TDM signals into a packet network is called circuit emulation services (CES).

Synchronous Ethernet, as defined in ITU-T G.8261, is an evolution of conventional Ethernet with SONET/SDH/PDH-based synchronization. Synchronous Ethernet is used to synchronize and send frequency information to nodes on the network. Synchronous Ethernet provides only frequency synchronization, not time or phase synchronization.

Synchronous Ethernet uses the physical layer to synchronize all links to the same frequency. Because Carrier Ethernet deployments are commonly high-speed point-to-point connections, the physical layer (ETY) works well in asynchronous mode as packets are buffered in each node.

In Synchronous Ethernet, each switch contains an internal clock called the Ethernet Equipment Clock (EEC) defined in ITU-T G.8262. An EEC is which is equivalent to an SEC/SMC. A standard IEEE 802.3 free-running clock has an accuracy of +/-100 ppm (parts per million). An EEC traceable to a PRC/PRS has an accuracy of +/-4.6 ppm.

Figure 1-1 shows a clock hierarchy in a network running Synchronous Ethernet. As in a TDM network, a PRC/PRS and SSU/BITS are at the top of the hierarchy with Ethernet switches running EECs below. At lower levels, the EECs in other switches synchronize their frequency to the EECs in switches above them in the hierarchy. Figure 1-1 also shows that a switch can have more than one frequency source from which it can choose.

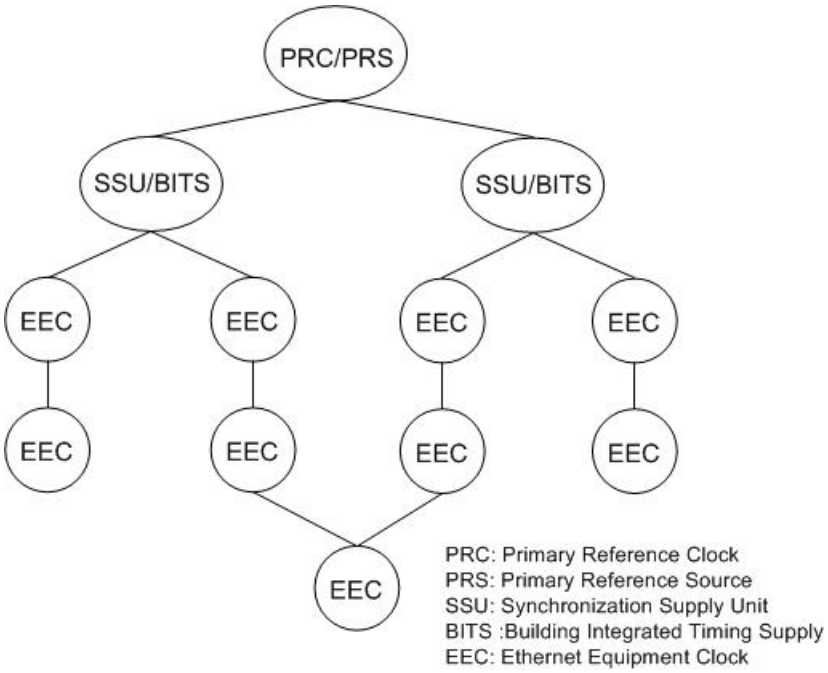

Figure 1-1: Synchronous Ethernet Clock Hierarchy

The nodes in a network running Synchronous Ethernet follow these steps:

1. The physical layer extracts the frequency of a timing signal from a high quality frequency reference traceable to a primary reference clock and passes it to a system clock.
2. The system clock locks to this frequency which becomes the candidate frequency reference.
3. This reference frequency is distributed to downstream Synchronous Ethernet nodes.
4. The receivers at the downstream automatically lock onto the physical layer clock of the received signal and get an accurate and stable frequency reference.
5. The receivers then distribute the frequency to downstream Synchronous Ethernet nodes.

Using this method, frequency originating from a PRC/PRS is distributed via the intermediate EECs.

Figure 1-2 shows frequency distribution in a network with both TDM and Ethernet devices. TDM A synchronizes its frequency to a PRC/PRS. TDM B, TDB C, TDM D, and TDM E are locked to the frequency of TDM A. Through Synchronous Ethernet, Ether B is also locked to the frequency of TDM A. Ether A and Ether D are locked to the frequency of Ether B. Ether C uses its own free-running clock and its frequency is not locked to that of the other Ethernet switches.

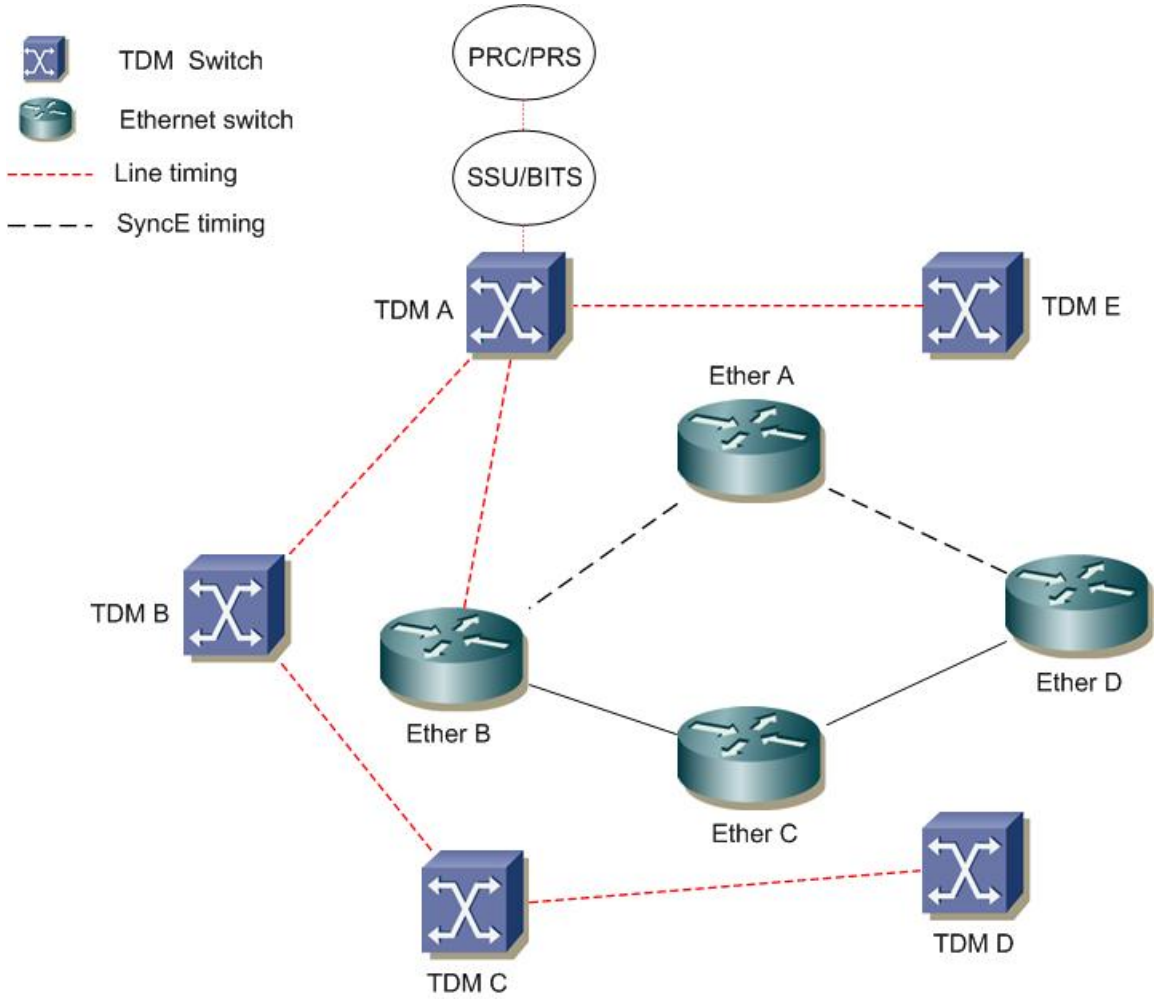

Figure 1-2: Frequency Distribution in a Hybrid Network

All devices supporting Synchronous Ethernet must send and receive data in cycles of fixed size and duration. The data size depends on the Ethernet speed. The rate of transmission is 8000 cycles per second. Each device must be able to support a system timing master, which is the synchronization source. A synchronous port is the port on which synchronization information is received. All Synchronous Ethernet frames coming from the synchronous port are the source of synchronization for all other ports on the device.

Synchronous Ethernet requires hardware support, as the packet processor must be able to select an incoming interface for clock recovery and then use this to synchronize all output interfaces to the same frequency. The hardware must also tightly monitor the frequency and phase of the data rate on incoming links.

Ethernet Synchronization Message Channel

ITU-T G.8264 defines the Ethernet Synchronization Message Channel (ESMC) which is the protocol that carries the Synchronization Status Message (SSM) between nodes to synchronize frequency. ESMC is based on IEEE 802.3 Organization Specific Slow Protocol (OSSP).

There are two types of ESMC messages:

- The Information Message is sent once per second; it acts like a heartbeat.
- The Event Message is sent immediately after an event which causes an SSM value to change.

Both message types contain a SSM.

Quality Level

The SSM code (along with the Timing Marker) represents the quality level of the Ethernet Equipment Clock (EEC) in the physical layer. The quality level defines the accuracy of that clock that is driving the synchronization chain. The higher the stratum value, the higher the accuracy of the clock. The PRC/PRS is the most accurate. When there are multiple clock sources, the quality level helps a node select the best source with which to synchronize. The quality level is transported in SSM messages between adjacent nodes.

Figure 1-3 shows the relation between layers 1 and 2, with layer 1 carrying the frequency and layer 2 carrying a ESMC packet that indicates the quality level.

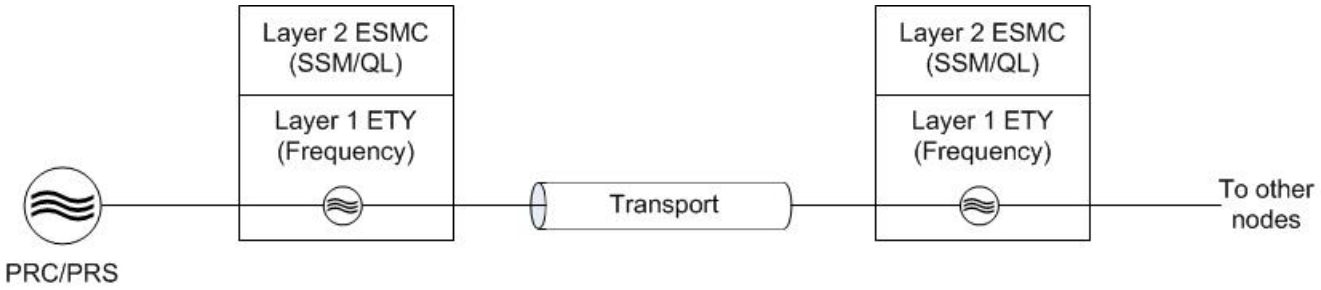

Figure 1-3: Layer 1 and Layer 2

ITU-T G.8262 classifies the quality levels for EECs as follows:

- EEC-Option I: Networks optimized for the 2,048 kbit/s hierarchy
- EEC-Option II: Networks optimized for the 1,544 kbit/s hierarchy that includes the rates 1,544 kbit/s, 6,312 kbit/s, and 44,736 kbit/s
- EEC-Option III: Networks optimized for the 1,544 kbit/s hierarchy that includes the rates 1,544 kbit/s, 6,312 kbit/s, 33,064 kbit/s, 44,736 kbit/s, and 97,728 kbit/s

Specific quality levels are defined in ITU-T G.781 for each EEC option above.

An administrator can force the quality level on an interface enabled for Synchronous Ethernet or on the input clock source itself. Quality level checking can also be disabled. If quality level checking is disabled, the priority attribute is used to select the best network clock source. The priority is set by a network administrator.

Signal-Fail Condition

Signal fail for a synchronization source is activated in case of defects detected in the server layers. In addition, an unconnected synchronization signal has also signal fail active to allow correct processing in the QL-disabled mode.

To avoid reactions on short pulses or intermittent signal fail information, the signal fail information is passed through hold-off and wait-to-restore timers before it is considered by the clock selection algorithm. The signal-fail condition of an interface is handled asynchronously and notification from the hardware will trigger the control-plane actions.

Timing Loops

This is a network condition where a slave clock providing synchronization becomes locked to its own timing signal, either directly or via other network equipment. Timing loops should be prevented in networks by careful network design. The control plane does not validate configurations to prevent timing loops. Timing loop prevention must be handled by network administrators.

Summary of ITU-T Recommendations

This chapter cites these recommendations from the International Telecommunication Union (ITU-T):

- ITU-T Rec. G.781: Synchronization layer functions
- ITU-T Rec. G.8261: Timing and synchronization aspects in packet networks
- ITU-T Rec. G.8262: Timing characteristics of synchronous Ethernet equipment slave clocks
- ITU-T Rec. G.8264: Distribution of timing through packet networks

CHAPTER 2 Synchronous Ethernet Command API

This chapter contains the Application Programming Interface (API) for the synchronous Ethernet (SyncE) commands.

Include Files

To call the functions in this chapter, you must include one or both of these files:

- `synced/synce_api.h`
- `synced/synce_cli.h`

Data Structures

The data structures to which this chapter refers are defined in `synced/synce_types.h`.

synce_config

This data structure stores global configure mode information about Synchronous Ethernet.

Member	Description
`synce_master`	Backpointer reference to master information
`is_primary_node`	Synchronization source for this network element; determines if it is the primary node connected to a primary source or an intermediate node
`synce_ifp_tree`	The `synce_ifp` tree
`ext_cmd`	External command information
`sync_option`	Synchronization option
`sync_gen`	Synchronization option generation
`clock_selection_mode`	QL-enabled or QL-disabled mode
`revertive_mode`	Revertive or non-revertive mode

Definition

```
struct synce_config
{
  /* backpointer reference to master info */
  struct synce_master *synce_master;

  /* sync source for this Network element, determines if its the primary
```

```
   * node connected to primary source or intermediate synce node */
  bool_t is_primary_node;

  /* This is the synce_ifp tree (element is struct synce_ifp) */
  struct avl_tree *synce_ifp_tree;

  /* external command information*/
  struct ext_cmd_info *ext_cmd;
  /* Synchronization option */
#define SYNCE_SYNCHRONIZATION_OPTION_ONE     1
#define SYNCE_SYNCHRONIZATION_OPTION_TWO     2
#define SYNCE_SYNCHRONIZATION_OPTION_THREE   3
  u_int8_t sync_option;

  /* Synchronization option GEN */
#define SYNCE_SYNCHRONIZATION_OPTION_GEN1 1
#define SYNCE_SYNCHRONIZATION_OPTION_GEN2 2
  u_int8_t sync_gen;

  /* QL-enabled or QL-disabled mode */
#define SYNCE_CLOCK_SELECTION_QL_ENABLE   1
#define SYNCE_CLOCK_SELECTION_QL_DISABLE   2
  u_int8_t clock_selection_mode;

  /* Revertive or Non-Revertive mode of operation */
#define SYNCE_REVERTIVE 1
#define SYNCE_NON_REVERTIVE 2
  u_int8_t revertive_mode;
};
```

synce_ifp

This data structure stores interface mode information about Synchronous Ethernet.

Member	Description
ifp	Back pointer reference to interface structure
ifindex	Interface index
syncem	Backpointer reference to master information
skaddr	Socket address
synce_if_type	Whether the interface is an input source or output source
is_selected_source	Whether this interface is the selected source for locking the internal clock
is_esmc_recd	Quick check for ESMC reception status
stateChgTime	Timestamp for ESMC state changes

Member	Description
u_int8_t recd_ql	The received (decoded) ESMC message when interface is the input source
recd_event_flag	Event flag in the received ESMC PDU that indicates if the QL has changed.
esmc_curr_ts	Validation counter for ESMC received per second (maximum 10 per second); enforcing is difficult as it requires time manipulation in seconds
counter_esmc_recd	Validation counter for ESMC received per second (maximum 10 per second)
counter_esmc_sent	Validation counter for ESMC sent per second (maximum 10 per second)
esmc_state	ESMC state of the interface
event_flag	If QL changes for this interface via external commands, notify other nodes about the change
esmc_timeout_while	ESMC timeout handler thread reference
synce_mode	Whether the interface is Synchronous Ethernet enabled
operation_mode	Whether the interface is synchronous or non-synchronous
ifp_conf_ql	Configured QL on this interface. If this is an input-source interface, then this QL value is considered for clock selection irrespective of QL received in ESMC. If this is an output-source, this QL would be sent out in ESMC through this interface, irrespective of QL chosen by clock selection. Initialized to -1.
ifp_conf_priority	Priority of the input source interface (default = 0)
signal_fail	Whether signal-fail is active on this interface
wait_to_restore_time	Wait-to-restore time for handling signal-fail condition
wtr_while	The signal fail true handler thread reference
holdoff_time	Hold-off time for handling signal-fail condition
hold_off_while	The signal fail false handler thread reference
trigger_csa	This flag is set when QL or priority is changed and for other scenarios that demand CSA to be run again
ext_cmd_flags	External commands active on this interface

Definition

```
struct synce_ifp
{
  struct interface *ifp;
  s_int32_t ifindex;
  struct synce_master *syncem;
  struct sockaddr skaddr;

  /* inteface being the input or output source */
#define SYNCE_IF_TYPE_INPUT_SOURCE   1
#define SYNCE_IF_TYPE_OUTPUT_SOURCE  2
```

```
  u_int8_t synce_if_type;

  /* if this is the selected source for locking internal clock */
  bool_t is_selected_source;

  /* boolean value for quick check for esmc reception status */
  bool_t is_esmc_recd;

  /* Timestamp for esmc state changes */
  pal_time_t stateChgTime;

  /* the received(decoded) esmc msg incase interface is input-source */
  s_int8_t recd_ql;
  u_int8_t recd_event_flag;

  /* validation counter for esmc recd per second (max 10 per second)
   * enforcing is difficult as it requies time manipulation for seconds */
  struct pal_timeval esmc_curr_ts;
  u_int8_t counter_esmc_recd;

  /* validation counter for esmc sent per second (max 10 per second) */
  u_int8_t counter_esmc_sent;

  /* ESMC state on the interface */
#define SYNCE_ESMC_IDLE     0
#define SYNCE_ESMC_START    1
#define SYNCE_ESMC_OK       2
#define SYNCE_ESMC_FAILED   3
  u_int8_t esmc_state;

  /* If QL is changed for this interface via external commands,indicate to
   * other node about the QL change */
  bool_t event_flag;

  /* The esmc timeout handler thread reference */
  struct thread *esmc_timeout_while;

  /* configurations parameters */
  /* boolean to indicate if the interface is synchronous ethernet enabled */
#define SYNCE_INTF_ENABLED  1
#define SYNCE_INTF_DISABLED 2
  u_int8_t synce_mode;

  /* boolean to indicate if the interface is synchrons mode or non-sync */
#define SYNCE_INTF_SYNCHRONOUS     1
#define SYNCE_INTF_NON_SYNCHRONOUS 2
  u_int8_t operation_mode;

  /* configured QL on this interface. If this is input-source interface, then
   * this is the QL value considred for Clock-selection irrespective of QL
```



```
   * received in ESMC. If its output-source,  this QL would be sent out in
   * ESMC through this interface, irrespective of QL chosen by Clock-selection.
   * Initialized to -1 */
  s_int8_t ifp_conf_ql;

  /* priority of the input-source interface (default = 0) */
  u_int8_t ifp_conf_priority;

  /* boolean to indicate the signal-Fail state at interface */
  bool_t signal_fail;

  /* WTR timer for handling signal-Fail condition */
  u_int32_t wait_to_restore_time;

  /* The signal fail true handler thread reference */
  struct thread *wtr_while;

  /* Hold-Off timer for handling signal-Fail condition */
  u_int32_t holdoff_time;

  /* The signal fail false handler thread reference */
  struct thread *hold_off_while;

  /* This flag is set when QL or priority is changed
   * and also for other scenarios that demand CSA to be run again */
  bool_t trigger_csa;

  /* flag to show the external commands active on this interface */
#define SYNCE_LOCKOUT_ENABLED           (1 << 0)
#define SYNCE_FORCE_SWITCH_ENABLED      (1 << 1)
#define SYNCE_MANUAL_SWITCH_ENABLED     (1 << 2)
  u_int8_t ext_cmd_flags;

};
```

Functions

The functions in this section implement the commands in the *Synchronous Ethernet Command Reference*.

Function	Description
synce_api_add_if	Enables Synchronous Ethernet on a interface
synce_api_clear_ext_cmd	Clears the switch-source (manual or forced) setting
synce_api_config_init	Initializes the global `synce_config` struct with default values
synce_api_delete_if	Disables Synchronous Ethernet on an interface

Function	Description
synce_api_force_source	Forcefully selects a synchronization source
synce_api_manual_source	Manually selects a synchronization source
synce_api_no_source	Deletes a input or output source
synce_api_ql_init	Initializes the quality level when the synchronization option has changed
synce_api_set_clear_lockout	Sets or clears the lockout for a source
synce_api_set_if_type_priority	Sets an interface as an input source or output source
synce_api_set_ql	Sets the quality level (QL) for the for the timing source on an interface
synce_api_set_sync_mode	Configures an interface as synchronous or non-synchronous
synce_api_set_timers	Sets the wait-to-restore and hold-off times
synce_display_clk_details	Displays clock information for the node
synce_display_input_src	Displays information about all input sources
synce_display_prc	Displays primary reference clock information

synce_api_add_if

This function enables Synchronous Ethernet on a interface.

This function implements the `synce` command in interface mode.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_add_if (struct interface *ifp)
```

Input Parameters

`ifp` Interface

Output Parameters

None

Return Value

RESULT_OK when the function succeeds or the interface is already enabled for SyncE

RESULT_ERROR when the `synce_ifp` struct cannot be created

SYNCE_ERROR_AVL_INSERT when the `synce_config->synce_ifp_tree` member cannot be created

SYNCE_ERR_SYNCE_NOT_ENABLED when SyncE is not enabled in configure mode

synce_api_clear_ext_cmd

This function clears the switch-source (manual or forced) setting.

This function implements the `clear switch-source` command.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_clear_ext_cmd (struct interface *ifp)
```

Input Parameters

`ifp`	Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

SYNCE_ERR_IF_NOT_INPUT_SOURCE when the interface is not an input source

SYNCE_ERR_IF_EXT_CMD_INACTIVE when switch-source is not set

SYNCE_SUCCESS when the function succeeds

synce_api_config_init

This function initializes the global `synce_config` struct with default values.

This function implements the `synce` command in configure mode.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_config_init(struct synce_config *sconfig)
```

Input Parameters

None

Output Parameters

`sconfig`	Global configure mode information

Return Value

RESULT_OK when the function succeeds

RESULT_ERROR when:

- The SyncE master structure cannot be found
- The `sconfig->synce_ifp_tree` member cannot be created
- Memory for clock source cannot be allocated
- Memory for the external command cannot be allocated

synce_api_delete_if

This function disables Synchronous Ethernet on an interface.

This function implements the `no synce` command.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_delete_if (struct interface *ifp)
```

Input Parameters

`ifp`	Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_SYNCE_NOT_ENABLED when SyncE is not enabled in configure mode

SYNCE_ERR_IFP_NOT_FOUND when the interface is not found in `synce_ifp_tree`

SYNCE_ERROR_AVL_DELETE when there was an error disabling SyncE on the interface

RESULT_ERROR when there is an error running the clock selection algorithm after disabling SyncE on the interface

RESULT_OK when the function succeeds

synce_api_force_source

This function forcefully selects a synchronization source.

This function implements the `switch-source force` command.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_force_source (struct interface *ifp)
```

Input Parameters

`ifp`	Interface

Output Parameters

None

Return Value

SYNCE_FAILURE when the SyncE master structure cannot be found

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

SYNCE_ERR_IF_NOT_INPUT_SOURCE when the interface is not an input source

SYNCE_ERR_IFP_LOCKOUT_ENABLED when lockout is set for the timing source

SYNCE_ERR_ESMC_NOT_ACTIVE_ON_IF when ESMC is not active on the interface

SYNCE_SUCCESS when the function succeeds

SYNCE_ERR_IFP_PRIORITY_DIS when priority is set to 0

SYNCE_ERR_QL_NOT_VALID when the quality level is not valid on this interface

synce_api_manual_source

This function manually selects a synchronization source.

This function implements the `switch-source manually` command.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_manual_source (struct interface *ifp)
```

Input Parameters

`ifp`	Interface

Output Parameters

None

Return Value

SYNCE_FAILURE when the SyncE master structure cannot be found

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

SYNCE_ERR_IF_NOT_INPUT_SOURCE when the interface is not an input source

SYNCE_ERR_IFP_LOCKOUT_ENABLED when lockout is set for the interface

SYNCE_ERR_IFP_SIGNAL_FAIL_ACTIVE when signal fail is active for the interface

SYNCE_ERR_IFP_QL_DNU when the quality level is QL-DNU or QL-DUS for the interface

SYNCE_ERR_ESMC_NOT_ACTIVE_ON_IF when ESMC is not active on the interface

SYNCE_ERR_FORCE_ACTIVE_ON_IF when forceful selection is active on the interface

SYNCE_ERR_QL_NOT_GREATEST when the quality level of the interface is not greater the quality level of the other SyncE interfaces

SYNCE_SUCCESS when the function succeeds

SYNCE_ERR_IFP_PRIORITY_DIS when priority is set to 0

SYNCE_ERR_QL_NOT_VALID when the quality level is not valid on this interface

synce_api_no_source

This function deletes a input or output source.

This function implements the `no input-source` and `no output-source` commands.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_no_source (u_int8_t synce_if_type, struct synce_ifp *synce_if)
```

Input Parameters

`synce_if_type` Interface type; one of these constants from `synced/synce_types.h`:

`SYNCE_IF_TYPE_INPUT_SOURCE`

`SYNCE_IF_TYPE_OUTPUT_SOURCE`

`synce_if` SynchE interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `synce_if` is NULL

SYNCE_SUCCESS when the function succeeds

SYNCE_ERR_INCORRECT_IF_TYPE when `synce_if_type` type does not matches the configured interface type

synce_api_ql_init

This function initializes the quality level when the synchronization option has changed.

This function implements the `synchronization option` command.

Syntax

```
#include "synced/synce_api.h"
void
synce_api_ql_init (struct synce_config *synce_confp)
```

Input Parameters

`synce_confp` Global configure mode information

Output Parameters

None

Return Value

None

synce_api_set_clear_lockout

This function sets or clears the lockout for a source.

This function implements the `set lockout` and `clear lockout` commands.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_set_clear_lockout (bool_t lockout_status, struct interface *ifp)
```

Input Parameters

`lockout_status`	Whether to set or clear the lockout:
`PAL_TRUE`	Set the lockout
`PAL_FALSE`	Clear the lockout
`ifp`	Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

SYNCE_FAILURE when the SyncE master structure cannot be found

SYNCE_SUCCESS when the function succeeds

SYNCE_ERR_IF_LOCKOUT_DISABLED when lockout is not enabled on this interface

synce_api_set_if_type_priority

This function sets an interface as an input source or output source.

This function implements the `input-source` and `output-source` commands in interface Synchronous Ethernet mode.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_set_if_type_priority (u_int8_t priority,
                                u_int8_t synce_if_type,
                                struct interface *ifp)
```

Input Parameters

`priority`	Priority <0-255>: 1 is the highest, 255 is the lowest; 0 means the source will not be considered by the clock selection algorithm. This value is ignored when `synce_if_type` is `SYNCE_IF_TYPE_OUTPUT_SOURCE`.
`synce_if_type`	Interface type; one of these constants from `synced/synce_types.h`**:**

SYNCE_IF_TYPE_INPUT_SOURCE	
SYNCE_IF_TYPE_OUTPUT_SOURCE	
ifp	Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

SYNCE_ERR_PRIORITY_SET when the interface type and priority are already set to the given values

RESULT_OK when the function succeeds

synce_api_set_ql

This function sets the quality level (QL) for the timing source on an interface.

This function implements the `quality-level QL_VAL` command.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_set_ql (char *ql_str,
                  struct interface *ifp)
```

Input Parameters

ql_val	Quality level; one of the constants below defined in `synced/synce_esmc.h`. The quality level you can specify depends on the values of the `sync_option` and `sync_gen` members in the synce_config structure. See ITU-T Rec. G.781 for details.
PRC	QL_PRC: Primary Reference Clock
SSU_A	QL_SSU_A: Types I or V slave clock
SSU_B	QL_SSU_B: Type VI slave clock
SEC	QL_SEC: SDH Equipment Clock
DNU	QL_DNU: Do not use this signal for synchronization
STU	QL_STU: Synchronized – traceability unknown
ST2	QL_ST3: Traceable to stratum 2
ST3	QL_ST3: Traceable to stratum 3
ST3E	QL_ST3E: Traceable to stratum 3E
SMC	QL_SMC: Traceable to SONET clock self timed
TNC	QL_TNC: Traceable to Transit Node Clock
PROV	QL_PROV: Provisionable by the network operator
DUS	QL_DUS: Do not use this signal for synchronization

`ifp` Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

SYNCE_ERR_QL_NOT_VALID when `ql_val` is not valid for the synchronization option setting

SYNCE_ERR_QL_SET when the quality level is already set to `ql_val`

SYNCE_SUCCESS when the function succeeds

SYNCE_ERR_MANUAL_SWITCH_ACTIVE when trying to set quality level when manual switch is enabled

synce_api_set_sync_mode

This function configures an interface as synchronous or non-synchronous:

- A synchronous interface extracts the frequency of its input signal from synchronization packets and passes it to the internal clock
- A non-synchronous interface does not participate in the synchronization process

This function implements the `mode(synchronous | non-synchronous)` command.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_set_sync_mode (u_int8_t sync_mode, struct interface *ifp)
```

Input Parameters

`sync_mode` Mode; one of these constants from `synced/synce_types.h`:

`SYNCE_INTF_SYNCHRONOUS`

`SYNCE_INTF_NON_SYNCHRONOUS`

`ifp` Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_OPERATION_MODE_SET when the interface is already set to the given `sync_mode`

RESULT_OK when the function succeeds

synce_api_set_timers

This function sets the wait-to-restore and hold-off times:

- The wait-to-restore time ensures that a synchronization source that previously failed is considered by the selection process again only if it is fault free for a certain time
- The hold-off time ensures that short activation of signal fail is not passed to the selection process

This function implements the `wait-to-restore <0-12>` and `hold-off HOLDOFFVAL` commands in interface Synchronous Ethernet mode.

Syntax

```
#include "synced/synce_api.h"
s_int16_t
synce_api_set_timers (u_int16_t wtr_val, u_int16_t hold_val, struct interface *ifp)
```

Input Parameters

`wtr_val`	Wait-to-restore time in minutes <0-12>
`hold_val`	Hold-off time in milliseconds <300-1800>
`ifp`	Interface

Output Parameters

None

Return Value

SYNCE_ERR_IFP_NOT_FOUND when `ifp` or `ifp->info` is NULL

SYNCE_ERR_IFP_SYNCE_DISABLED when SyncE is disabled on the interface

SYNCE_ERR_IFP_ASYNC_MODE when the SyncE mode is non-synchronous

RESULT_OK when the function succeeds

synce_display_clk_details

This function displays clock information for the node.

This function implements the `show synce details` command.

Syntax

```
#include "synced/synce_cli.h"
s_int16_t
synce_display_clk_details (struct cli *cli)
```

Input Parameters

None

Output Parameters

`cli`	CLI structure

Return Value

SYNCE_FAILURE when the SyncE master structure cannot be found

SYNCE_SUCCESS when the function succeeds

synce_display_input_src

This function displays information about all input sources.

This function implements the `show synce input-sources` command.

Syntax

```
#include "synced/synce_cli.h"
s_int16_t
synce_display_input_src (struct cli *cli)
```

Input Parameters

None

Output Parameters

`cli`	CLI structure

Return Value

SYNCE_FAILURE when the SyncE master structure cannot be found

SYNCE_SUCCESS when the function succeeds

synce_display_prc

This function displays primary reference clock information.

This function implements the `show synce input-sources` and `show synce details` commands.

Syntax

```
#include "synced/synce_cli.h"
s_int16_t
synce_display_prc (struct cli *cli)
```

Input Parameters

None

Output Parameters

`cli`	CLI structure

Return Value

SYNCE_FAILURE when the SyncE master structure cannot be found

SYNCE_SUCCESS when the function succeeds

Index

s