MT90502 API
User Guide

Part Number: MT90502

Revision Number: **2.8**

Issue Date: **August 2004**

ZARLINK™
SEMICONDUCTOR

Table of Contents

1.0 Overview . . . 7
1.1 Definitions . . . 7
1.2 Documentation and Coding Conventions . . . 7
1.2.1 Code Header Files . . . 8
1.2.2 Compilation Options . . . 8
1.2.2.1 MT90502_COMPILE_OPTION_NON_SHARED_MEMORY . . . 8
1.3 API Function Summary: . . . 8
1.4 User Supplied Function Summary . . . 12
Write Functions . . . 12
1.5 System Architecture . . . 13
2.0 API Function Descriptions . . . 19
2.1 Initialization Functions: . . . 21
2.1.1 mt90502_open . . . 21
2.1.2 mt90502_init_sil_suppr_profile . . . 21
2.1.2.1 Structure MT90502_INIT_SIL_SUPPR_PROFILE . . . 22
2.1.3 mt90502_open_instance_size . . . 23
2.1.3.1 Structure MT90502_INSTANCE_SIZE . . . 23
2.1.4 mt90502_close . . . 24
2.1.4.1 Structure MT90502_CLOSE_CHIP . . . 24
2.1.5 mt90502_get_hw_revision . . . 24
2.1.5.1 Structure MT90502_REVISION . . . 25
2.2 ATM Functions: . . . 25
2.2.1 mt90502_open_aal2_vc . . . 25
2.2.1.1 Structure MT90502_AAL2_VC . . . 26
2.2.2 mt90502_open_data_vc . . . 28
2.2.2.1 Structure MT90502_DATA_VC . . . 28
2.2.3 mt90502_close_vc . . . 30
2.2.3.1 Structure MT90502_CLOSE_VC . . . 31
2.2.4 mt90502_open_cid . . . 31
2.2.4.1 Structure MT90502_OPEN_CID . . . 32
2.2.5 mt90502_map_cid . . . 33
2.2.5.1 Structure MT90502_MAP_CID . . . 34
2.2.6 mt90502_close_cid . . . 36
2.2.6.1 Structure MT90502_CLOSE_CID . . . 36
2.3 TDM Functions . . . 36
2.3.1 mt90502_open_rx_xxpcm_channel . . . 36
2.3.1.1 Structure MT90502_RX_XXPCM_CHAN . . . 37
2.3.2 mt90502_open_tx_xxpcm_channel . . . 40
2.3.2.1 Structure MT90502_TX_XXPCM_CHAN . . . 41
2.3.3 mt90502_open_rx_hdlc_stream . . . 44
2.3.3.1 Structure MT90502_RX_HDLC_STREAM . . . 45
2.3.4 mt90502_open_tx_hdlc_stream . . . 46
2.3.4.1 Structure MT90502_TX_HDLC_STREAM . . . 47
2.3.5 mt90502_close_hdlc_stream . . . 48
2.3.5.1 Structure MT90502_CLOSE_STREAM . . . 49
2.3.6 mt90502_open_rx_hdlc_channel . . . 49
2.3.6.1 Structure MT90502_RX_HDLC_CHAN . . . 50
2.3.7 mt90502_open_tx_hdlc_channel . . . 51
2.3.7.1 Structure MT90502_TX_HDLC_CHAN . . . 52
2.3.8 mt90502_open_channel_in_loopback, . . . 53
2.3.8.1 Structure MT90502_LLL_CHAN . . . 54
2.3.9 mt90502_close_channel . . . 54

Table of Contents

2.3.9.1 Structure MT90502_CLOSE_CHAN . . . 55
2.3.10 mt90502_tx_change_compression . . . 55
2.3.10.1 Structure MT90502_CH_COMP . . . 56
2.3.11 mt90502_open_phasing_tsst . . . 56
2.3.11.1 Structure MT90502_OPEN_PHASING_TSST . . . 57
2.3.12 mt90502_query_phasing_tsst . . . 58
2.3.12.1 Structure MT90502_QUERY_PHASING_TSST . . . 58
2.3.13 mt90502_close_phasing_tsst . . . 59
2.3.13.1 Structure MT90502_CLOSE_PHASING_TSST . . . 59
2.3.14 mt90502_tx_change_silence_suppr . . . 59
2.3.14.1 Structure MT90502_CH_SIL_SUP . . . 60
2.4 Statistics Functions . . . 60
2.4.1 mt90502_get_chip_statistics . . . 61
2.4.1.1 Statistics Structure MT90502_CHIP_STATS . . . 61
2.4.2 mt90502_convert_chip_statistics_to_text . . . 71
2.4.2.1 Structure MT90502_CONVERT_CHIP_STATS . . . 72
2.4.3 mt90502_get_vc_statistics . . . 72
2.4.3.1 Structure MT90502_VC_STATS . . . 73
2.4.4 mt90502_convert_vc_statistics_to_text . . . 75
2.4.4.1 Structure MT90502_CONVERT_VC_STATS . . . 76
2.4.5 mt90502_get_cid_statistics . . . 76
2.4.5.1 Structure MT90502_CID_STATS . . . 77
2.4.6 mt90502_convert_cid_statistics_to_text . . . 79
2.4.6.1 Structure MT90502_CONVERT_CID_STATS . . . 79
2.4.7 mt90502_get_rx_xxpcm_chan_statistics . . . 79
2.4.7.1 Structure MT90502_RX_XXPCM_CHAN_STATS . . . 80
2.4.8 mt90502_get_tx_xxpcm_chan_statistics . . . 83
2.4.8.1 Structure MT90502_TX_XXPCM_CHAN_STATS . . . 84
2.4.9 mt90502_get_rx_hdlc_chan_statistics . . . 87
2.4.9.1 Structure MT90502_RX_HDLC_CHAN_STATS . . . 88
2.4.10 mt90502_get_tx_hdlc_chan_statistics . . . 90
2.4.10.1 Structure MT90502_TX_HDLC_CHAN_STATS . . . 91
2.4.11 mt90502_convert_<xxxx>_chan_statistics_to_text . . . 93
2.4.11.1 Structure MT90502_CONVERT_<XXXX>_CHAN_STATS . . . 93
2.4.12 mt90502_update_counters . . . 94
2.4.12.1 Structure MT90502_UPDATE_COUNTERS . . . 94
2.5 Utility Functions . . . 95
2.5.1 mt90502_get_handle_list . . . 95
2.5.1.1 Structure MT90502_HANDLE_REQUEST . . . 96
2.5.2 mt90502_query_handle_type . . . 96
2.5.2.1 Structure MT90502_QUERY_HANDLE . . . 97
2.5.3 mt90502_get_li_uui_change_event . . . 98
2.5.3.1 Structure MT90502_LI_EVENT . . . 98
2.6 Diagnostics Functions . . . 100
2.6.1 mt90502_get_h100_diagnostics . . . 100
2.6.1.1 Structure MT90502_H100_DIAG . . . 100
2.6.2 mt90502_convert_h100_diagnostics_to_text . . . 102
2.6.2.1 Structure MT90502_CONVERT_H100_DIAG . . . 103
2.6.3 mt90502_get_console_msgs . . . 103
2.6.3.1 Structure MT90502_CONSOLE_MSG . . . 103
2.7 H.100 Functions . . . 104
2.7.1 mt90502_set_h100_master_mode . . . 104

Table of Contents

2.7.1.1 Structure MT90502_H100_MASTER_PARMS . . . 104
2.7.2 mt90502_set_h100_slave_mode . . . 105
2.7.2.1 Structure MT90502_H100_SLAVE_PARMS . . . 105
2.8 Data Cell Functions . . . 106
2.8.1 mt90502_send_data_cell . . . 106
2.8.1.1 Structure MT90502_TX_DATA_CELL . . . 107
2.8.2 mt90502_send_test_cell . . . 107
2.8.2.1 Structure MT90502_TX_TEST_CELL . . . 108
2.8.3 mt90502_receive_data_cell . . . 109
2.8.3.1 Structure MT90502_RX_DATA_CELL . . . 110
2.9 AAL2 Mini-Packet Functions . . . 111
2.9.1 mt90502_send_aal2_mini_pkt . . . 111
2.9.1.1 Structure MT90502_AAL2_SEND_MINI_PKT . . . 112
2.9.2 mt90502_receive_aal2_mini_pkt . . . 112
2.9.2.1 Structure MT90502_AAL2_RCV_MINI_PKT . . . 113
2.9.3 mt90502_send_cas_pkt . . . 115
2.9.3.1 Structure MT90502_SEND_CAS_PKT . . . 115
2.9.4 mt90502_cas_refresh . . . 116
2.9.4.1 Structure MT90502_CAS_REFRESH . . . 117
2.9.5 mt90502_cancel_cas . . . 117
2.9.5.1 Structure MT90502_CANCEL_CAS . . . 118
2.9.6 mt90502_receive_cid_event . . . 118
2.9.6.1 Structure MT90502_RCV_CID_EVENT . . . 119
2.10 GPIO Functions . . . 121
2.10.1 mt90502_set_gpio_value . . . 121
2.10.1.1 Structure MT90502_SET_GPIO_PARMS . . . 121
2.10.2 mt90502_get_gpio_value . . . 122
2.10.2.1 Structure MT90502_GET_GPIO_PARMS . . . 122
2.11 Clock Recovery Functions . . . 123
2.11.1 mt90502_get_adaptive_clk_recov_pnt . . . 123
2.11.1.1 Structure MT90502_ADAP_PNT . . . 124
2.12 Interrupt Functions . . . 125
2.12.1 mt90502_interrupt_service_routine . . . 126
2.12.1.1 Structure MT90502_INT_STRUCT . . . 127
2.12.1.2 Structure MT90502_INT_FLAGS . . . 127
2.12.2 mt90502_mask_interrupt . . . 132
2.12.2.1 Structure MT90502_MASK_INT_PARMS . . . 132
2.12.3 mt90502_configure_interrupts . . . 133
3.0 User Supplied Function Descriptions . . . 134
3.1 Write Functions . . . 134
3.1.1 mt90502_driver_write_api, _apiisr, _osisr . . . 134
3.1.1.1 Structure MT90502_WRITE_PARMS . . . 135
3.1.2 mt90502_driver_write_smear_api, _apiisr, osisr . . . 135
3.1.2.1 Structure MT90502_WRITE_SMEAR_PARMS . . . 136
3.1.3 mt90502_driver_write_burst_api, _apiisr, osisr . . . 137
3.1.3.1 Structure MT90502_WRITE_BURST_PARMS . . . 137
3.2 Read Functions . . . 138
3.2.1 mt90502_driver_read_api, _apiisr, _osisr . . . 138
3.2.1.1 Structure MT90502_READ_PARMS . . . 138
3.2.2 mt90502_driver_read_burst_api, _apiisr, _osisr . . . 139
3.2.2.1 Structure MT90502_READ_BURST_PARMS . . . 139
3.2.3 mt90502_driver_read_debug_api, _apiisr, _osisr . . . 140

Table of Contents

3.2.3.1 Structure MT90502_READ_DEBUG_PARMS . . . 140
3.3 Interrupt Service Routine Called From API . . . 141
3.3.1 mt90502_access_apiisr . . . 141
3.3.1.1 Structure MT90502_PIPE_STRUCT . . . 143
4.0 Return Codes . . . 143
5.0 Configuration Structures . . . 144
5.1 Structure MT90502_CONF . . . 144
5.1.1 General Parameters . . . 144
5.1.2 Interrupt Configuration Parameters . . . 152
5.1.3 Memory Configuration Parameters . . . 152
5.1.4 UTOPIA Port Physical Configuration Parameters . . . 153
5.1.4.1 Utopia Clock Divider Configuration Parameters . . . 155
5.1.5 UTOPIA Operational Characteristics Parameters . . . 157
5.1.5.1 General . . . 157
5.1.5.2 Cell Routing . . . 157
5.1.5.3 Flow Control . . . 162
5.1.6 CID Configuration Parameters . . . 164
5.1.7 TDM Configuration Parameters . . . 165
5.1.8 HDLC Configuration Parameters . . . 169
5.1.9 Tone Buffer Configuration Parameters . . . 169
5.1.10 Silence Suppression Configuration Parameters . . . 170
5.1.10.1 Structure MT90502_SIL_SUPPR_PROFILE . . . 173
5.2 Structure MT90502_CONF_INTERRUPTS . . . 177

1.0 Overview

This document defines the C-language application programming interface functions.[1]

The library was compiled using Microsoft Visual C++ 6.0.

1.1 Definitions

Channel A 64 Kb TDM stream.

CID An AAL2 channel identifier.

LUT Look Up Table.

Mini Packet An AAL2 CPS – Packet

RX The receive direction with respect to the UTOPIA bus. Thus, RX means out of the chip when referring to the TDM bus and into the chip when referring to the UTOPIA bus.

SID Silence Insertion Descriptor.

TX The transmit direction with respect to the UTOPIA bus. Thus, TX means into the chip when referring to the TDM bus and out of the chip when referring to the UTOPIA bus.

TSST A TDM time-slot stream.

VC An ATM virtual circuit.

1.2 Documentation and Coding Conventions

In this document:

- all addresses are byte addresses.
- numbers are decimal unless otherwise specified.
- a word is 16 bits, and a byte 8 bits.
- all memory locations are laid-out in the little endian format.
- when a parameter value is greater than 32 bits it is stored in an array where the lowest indexed element contains the LSB.

All function parameters are passed in C structures to allow for compatibility of code upgrades. Each parameter is documented here with 3 fields:

Direction – indicates if the parameter is an input (IN), output (OUT), or input and output (IO) of the function. When a parameter is a pointer the direction is indicated as direction/direction, where the first direction refers to the pointer itself (typically IN) and the second direction (after the slash) refers to the memory pointed to by the pointer. Thus, a pointer direction of IN/OUT indicates that the pointer is an input to the function (i.e., the value of the pointer will not be modified), and the memory pointed to by the pointer is used for output.

Type – indicates the C type of the parameter. A ULONG is an unsigned 32-bit value. Parameters may also be declared as arrays and are documented here as ULONG[x] where x indicates the number of elements. Also used in the API are unsigned characters (8-bit values) indicated as BYTEs. As with ULONGs, parameters may also be declared as arrays and are documented here as BYTE[x] where x indicates the number of elements.

1. The MT90502 API software was developed with the assistance of OCTASIC Inc.

Default – indicates the default value the parameter is initialized to by an associated function for initializing the structure. All values of MT90502_INVALID_xxx means that the _def function will initialize the parameter to a value which indicates invalid for that parameter. The API will return an error if the parameter remains invalid when the structure is passed to a function that uses the parameter as an input.

Every function has an associated "_def" or default version that initializes the parameter structure. Even if the function requires no inputs there is a _def version. If the _def function is always used to initialize parameter structures, future versions of the API can be backward compatible with older user code as any new feature parameters can be initialized properly.

1.2.1 Code Header Files

The code of the API is split into three compilable entities: API, APIISR, and APIMI (these blocks are described later in System Architecture Section). Because the code is in separate entities, each entity has its respective. H file for the functions exported by that entity. These files are needed by the user application to call the functions. The files are listed below, as well as their relation to the code entities:

- API => mt90502_api.h
- APIISR => mt90502_apiisr.h
- APIMI => mt90502_apimi.h

Also, as explained later in this document, the user must supply C code to the API. The user code provides the link between the API and APIISR entities, and allows the three entities to perform read and write accesses to the chip. These functions are described in Section 3.0 "User Supplied Function Descriptions" on page 134. The definitions of the structures needed by all user-supplied functions are contained in mt90502_apiud.h. The file is needed by the user-supplied functions for the definitions of the structures used.

1.2.2 Compilation Options

Compile time options that can be defined.

1.2.2.1 MT90502_COMPILE_OPTION_NON_SHARED_MEMORY

This option disables all internal functions used to maintain the coherence of the API and APIISR instance memories when shared amongst more than one application.

If you are not sharing the API memory to more than one application you can define this option to disable internal functions and increase performance slightly.

1.3 API Function Summary:

Initialization Functions

mt90502_open -

Performs all the necessary operations to initialize the chip.

mt90502_init_suppr_profile -

Loads default values into a silence suppression profile.

mt90502_open_instance_size -

Returns the required size of the instance structure.

mt90502_close -

Performs all necessary clean-up to cease using the chip.

mt90502_get_hw_revision -

Returns the device revision number.

ATM Functions

mt90502_open_aal2_vc -

Opens a bi-directional AAL2 VC from TDM bus to a UTOPIA bus.

mt90502_open_data_vc -

Opens a uni-directional data from a UTOPIA bus to the Data cell FIFO or back to a UTOPIA bus.

mt90502_close_vc -

Closes the specified VC.

mt90502_open_cid -

Configures a mini packet stream with a given CID on a VC.

mt90502_map_cid -

Associates transmit and receive channels and/or CPU buffer to UUIs of a mini packet stream.

mt90502_close_cid -

Closes a CID and releases all resources reserved by it. All channels mapped to the CID are closed as well.

TDM Functions

mt90502_open_rx_xxpcm_channel -

Configures a receive XXPCM channel.

mt90502_open_tx_xxpcm_channel -

Configures a transmit XXPCM channel.

mt90502_open_rx_hdlc_stream -

Configures an HDLC stream of 1 or more TSSTs to be used for reception and returns a handle for it.

mt90502_open_tx_hdlc_stream -

Configures an HDLC stream of 1 or more TSSTs to be used for transmission and returns a handle for it.

mt90502_close_hdlc_stream -

Closes an HDLC stream and releases any associated HDLC channels and TSSTs.

mt90502_open_rx_hdlc_channel -

Configures a receive HDLC channel.

mt90502_open_tx_hdlc_channel -

Configures a transmit HDLC channel.

mt90502_open_channel_in_loopback -

Opens a unidirectional loopback channel from TDM bus to TDM bus.

mt90502_close_channel -

Closes any channel and releases associated resources.

mt90502_tx_change_compression -

Changes the compression type of a channel.

mt90502_open_phasing_tsst -

Configures a phasing TSST for communicating packetization boundaries to other devices.

mt90502_querry_phasing_tsst -

Returns the current state of a phasing TSST.

mt90502_close_phasing_tsst -

Closes a phasing TSST.

mt90502_tx_change_silence_suppr -

Changes the configuration of TX silence suppression dynamically.

Statistics Functions

mt90502_get_chip_statistics -

Returns a general chip statistics structure.

mt90502_convert_chip_statistics_to_text -

Converts the chip statistics structure to text.

mt90502_get_vc_statistics -

Gets statistics structure for an open AAL2 VC.

mt90502_convert_vc_statistics_to_text -

Converts a VC statistics structure to text.

mt90502_get_cid_statistics -

Gets statistics structure for an open CID.

mt90502_convert_cid_statistics_to_text -

Converts a CID statistics structure to text.

mt90502_get_rx_xxpcm_channel_statistics -

Gets statistics structure for an open receive XXPCM channel.

mt90502_get_tx_xxpcm_channel_statistics -

Gets statistics structure for an open transmit XXPCM channel.

mt90502_get_rx_hdlc_channel_statistics -

Gets statistics structure for an open receive HDLC channel.

mt90502_get_tx_hdlc_channel_statistics -

Gets statistics structure for an open transmit HDLC channel.

mt90502_convert_X_chan_statistics_to_text -

4 functions that convert any channel statistics structure to text.

mt90502_update_counters -

Updates statistics counters.

Utility Functions

mt90502_get_handle_list -

Retrieves a list of channel handles of a user specified type.

mt90502_query_handle_type -

Returns the type of a handle.

mt90502_get_li_uui_change_event -

Retrieves detected LI changes.

Diagnostic Functions

mt90502_get_h100_diagnostics -

Gets diagnostics structure for H100 bus.

mt90502_convert_h100_diagnostics_to_text -

Converts an H100 bus diagnostics structure to text.

mt90502_get_console_messages -

Gets diagnostic API console messages.

H100 Functions

mt90502_set_h100_master_mode -

Sets the chip's role as master on the H100 bus.

mt90502_set_h100_slave_mode -

Sets which master the chip will obey.

Data Cell Functions

mt90502_send_data_cell -

Inserts a raw ATM cell in the TX data cell FIFO.

mt90502_send_test_cell -

Inserts a raw ATM cell in the TX data cell FIFO that will be treated as received on a specified UTOPIA RX port.

mt90502_receive_data_cell -

Retrieves a raw ATM cell from the RX data cell FIFO if one is available.

AAL2 Mini-Packet Functions

mt90502_send_aal2_mini_pkt -

Inserts an AAL2 mini-packet onto any open channel.

mt90502_receive_aal2_mini_pkt -

Retrieves an AAL2 mini-packet from an open CPU channel.

mt90502_send_cas_pkt -

Inserts a CAS packet onto any open channel.

mt90502_cas_refresh -

Initiates CAS refresh on open channels.

mt90502_cancel_cas -

Cancels the automatic transmission of CAS refresh packets.

mt90502_receive_cid_event -

Retrieves a CID event from the API.

Clock Recovery Functions

mt90502_get_adaptive_clock_recovery_point -

Used to fetch adaptive clock recovery information from the device.

Interrupt Functions

mt90502_interrupt_service_routine -

Function to be called when the chip asserts its interrupt.

mt90502_mask_interrupt -

Function is called to temporarily disable the chip's interrupt pins.

mt90502_configure_interrupts -

Function is called to change the configuration of interrupts

1.4 User Supplied Function Summary

In order to allow implementation independence the API functions make all accesses to the device through user supplied read and write functions. The requirements and considerations for these routines can be found in Section 3.0 "User Supplied Function Descriptions" on page 134.

Write Functions

mt90502_driver_write_ api, apiisr, osisr -

Performs a single word write to the chip.

mt90502_driver_write_smear_ api, apiisr, osisr -

Performs a smear of a word to a block of addresses.

mt90502_driver_write_burst_ api, apiisr, osisr -

Performs a write of a block of words to a contiguous address block.

Read Functions

mt90502_driver_read_ api, apiisr, osisr -

Performs a single word read from the chip.

mt90502_driver_read_burst_ api, apiisr, osisr -

Performs a burst of reads from the chip.

mt90502_driver_read_debug_ api, apiisr, osisr -

Performs a burst of reads from the chip with parity.

API ISR Interface

mt90502_access_apiisr -

API ISR entry point for API code block.

1.5 System Architecture

The API is structured such that the code is stateless. All state of the API is contained in user allocated memory. This memory is referred to as the instance structures of the chip. For every API function called by the user, one of the chip's instance structure pointers is provided as a parameter. This allows the API code to service multiple chips. The instance structure pointers may be stored by the user in an array, and indexed by chip number. When an API function is to be called, the appropriate pointer can then be retrieved from the list, via the chip's index, and passed to the function.

The system architecture of the API is described below for an embedded system in two different interrupt-handling methods: with and without deferred procedure calls. In the first case a deferred procedure call is not used, and the API's ISR is called by the OS's ISR directly at the interrupt priority level. This architecture is depicted below. All blocks shaded in dark gray in the two figures are API code. All other blocks represent code provided by the user.

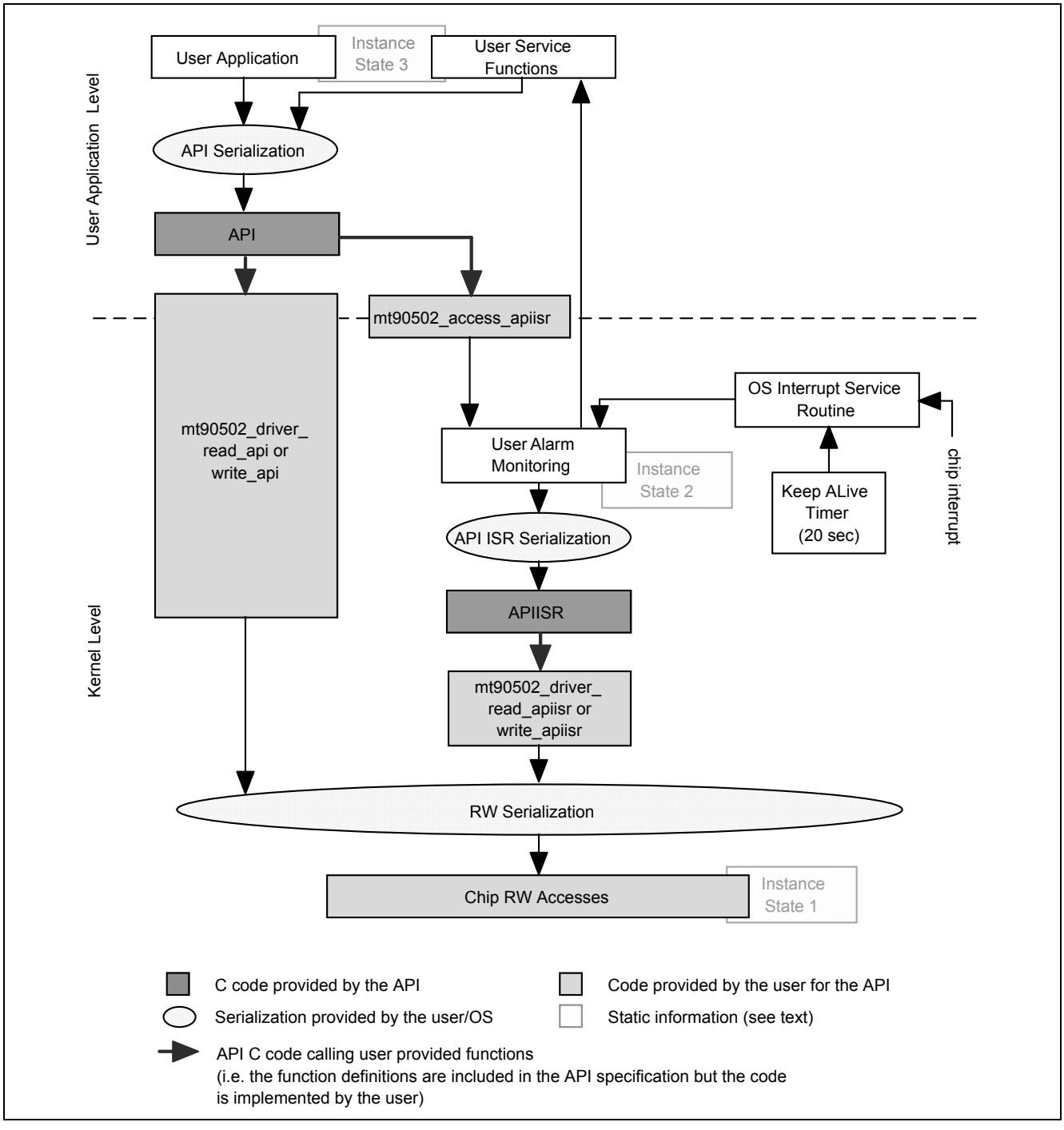


Figure 1 - System Architecture without Deferred Interrupt Procedure Call

The next figure depicts an architecture which uses deferred procedure calls. The OS's ISR simply defers the calling of the API's ISR to a later time, and at a lower interrupt priority level.

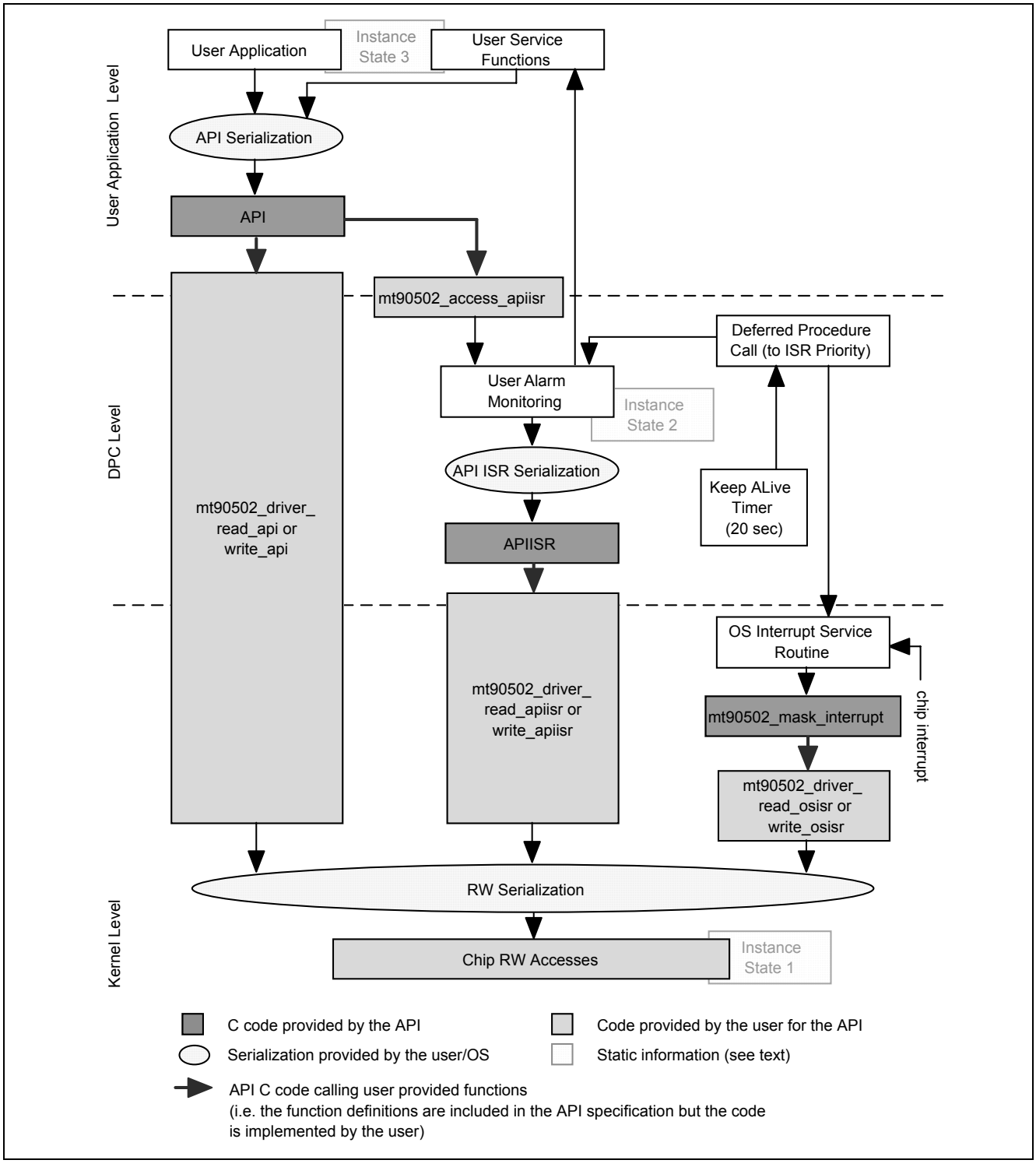


Figure 2 - System Architecture with Deferred Interrupt Procedure Call

In both architectures an API serialization layer is needed to avoid a race condition between two threads, utilizing the same instance structure pointers, and attempting to call an API function. The serialization may be implemented in

the form of a semaphore or mutex, for example. The serialization layer lies between the API and the user application (and is managed by the user).

Another serialization layer is needed for the APIISR code entity. The code entities are described below. Because the user is responsible for calling the APIISR within this entity when an interrupt is received, and because the API can itself call the APIISR, a race condition exists. The serialization performed at this level could be implemented via interrupt priority levels, for example.

Finally, the chip-level read and write accesses must be serialized as well. The serialization is necessary because a read/write access is split into several accesses to the CPU indirection registers of the chip. Thus, to insure that an access is completed correctly, these accesses to the indirection registers must be an atomic operation.

The API is contained in three different code entities, each of which may run at a different software/OS layer. The three sections correspond to the boxes in the figures labeled:

- API
- APIISR
- mt90502_mask_interrupt

The API code entity contains the majority of the functions that are called by the user, at a user priority level. These functions are not as fast as the APIISR function, and thus should not be serialized with interrupt execution. Because of this, and because the APIISR often runs at a higher priority level, the APIISR must be separated from the main API code.

The APIISR code entity contains the API's interrupt handling function. The function in this code entity is called by the OS's ISR upon receiving an interrupt from the hardware. It can also be called from the API code entity to access resources that the API and the ISR share. Thus, serialization between the OS's ISR's calls and the API's calls to this entity must be implemented by the user.

The API's ISR does not have to run at the same priority level as the OS's ISR. To do this, the interrupt signal line must be masked out in the MT90502 to be able to execute at a lower priority level (temporarily disabling the interrupt). The smallest code entity, mt90502_mask_interrupt, performs this task. It does so with only two accesses to the chip: one read and one write. The read is performed to query the state of the chip's interrupt register. This is necessary for systems which have multiple devices sharing an interrupt line. If the chip has flagged an interrupt, a write is performed to the chip to disable the interrupt pin. Needless to say it executes very quickly, thus allowing other high-priority interrupts to be serviced immediately. Because this function has no access to the instance structures of the chip, it need not be serialized with any other part of code. This function masks out all interrupts for a period of 16ms. If the API's ISR has not completed within 16 ms of masking out the interrupts, another interrupt will be generated. If no interrupt is present, the function will return a status code that allows the user to avoid an unnecessary call to the APIISR.

Because the API and APIISR entities may lie in different OS priority levels, and because some OSs protect and separate kernel space memory from user space memory, the two code entities cannot access the same memory. Thus, each code entity needs a pointer to its own distinct block of memory. The API entity needs a pointer to a block of user space memory, and the APIISR entity needs a block of kernel space memory. Each portion can only access its memory block.

As stated earlier, the API and APIISR entities share some information. Some API functions need to return information which is gathered by the APIISR and stored in its instance structure. Because the API does not have direct access to the APIISR's instance structure, the API is given access to the APIISR's information through a user supplied function, mt90502_access_apiisr. The function serves as a "messaging pipe" between the two entities. See mt90502_access_apiisr.

As stated earlier, the API is structured to support multiple chips. Each chip instance requires its own pair of pointers to user allocated memory: the API instance structure and the APIISR instance structure. These pointers can be stored in an array, and indexed by chip number. When the OS enforces independent memory spaces two arrays must be kept: one in the user application's memory space for the API instance structures, and the other in the ISR's

memory space for the APIISR structures. The two arrays are depicted in the figures above as “Instance State 3” and “Instance State 2”, respectively.

The size of these memory blocks is determined by the API function mt90502_open_instance_size, described later in this document. The function is called for each chip, before initially configuring it. The function takes a chip configuration structure as a parameter and uses it to return the memory size required for the API and APIISR instance structures. See mt90502_open_instance_size.

The read and write routines supplied by the user are used by the API functions to access all chips which the API code is servicing. The chip and its associated instance structure are configured via a call to the function mt90502_open. The function receives a chip configuration structure as a parameter. In this structure is the user_chip_number parameter which is intended to be the index of the chip being opened. Because every API and APIISR function receives a pointer to an instance structure as the first parameter, the chip number is available to all API functions. The only use of the user_chip_number by the API is to provide it as a parameter to the read/write functions. Thus, by associating a chip number to a particular chip, the correct device can be accessed in the user provided read/write routines. For example, chip number could be associated to a base address in the system. The user can then offset the provided address of a read/write routine and perform an access to the correct device. As illustrated in the two figures above, this information is easily stored as an array of chip specific information (e.g. base addresses) and can be indexed by the chip number. Note that the same chip number can be used to access system arrays kept by the user in different memory regions (e.g., user vs. kernel):

- API instance structure pointer array (Instance State 3),
- APIISR instance structure pointer array (Instance State 2),
- Read/Write function chip info (Instance State 1).

The two figures above indicate that two or three versions of the same read and write functions must be supplied. These functions differ only in the layer of their entry point. The functions in the group mt90502_driver_read_api or write_api are accessible only from the user application space, the group mt90502_driver_read_apiisr or write_apiisr from the DPC priority level in kernel space, and the group mt90502_driver_read_osisr or write_osisr from the interrupt priority level in the kernel space. In the case where deferred procedure calls are not used, the third group is not needed.

The mt90502_interrupt_service_routine, located in the APIISR code entity, returns a vector of the interrupts that were serviced; it is the responsibility of the User Alarm Monitoring function to call any required user functions to continue the servicing in the user application. For example if the user wanted to service data cells (AAL0) as soon as they are received, the alarm_data_cell_fifo_int_conf parameter would be set to MT90502_INT_TIMEOUT and the alarm_data_cell_fifo_int_timeout parameter would be set to the minimum desired delay value (e.g. 1ms) (the two parameters are part of the chip configuration structure MCA2_CONF). When a data cell arrives, the chip would assert an interrupt at most 1ms later and no more frequently than every ms. In response to the interrupt the OS ISR calls the API ISR which services the interrupt and returns the vector indicating a data cell interrupt. The User Alarm Monitoring function then calls the user routine (in the user space) for data cells which calls the API routine mt90502_receive_data_cell to obtain the cell.

The next figure depicts the system architecture used to perform the debugging of the API. The architecture is implemented on a Windows NT platform. Note that the API’s ISR is located in the user space to facilitate debugging. Also important is the presence of a separate thread. This thread is dedicated to handling interrupts only. It waits for a flag from the OS’s ISR indicating that an interrupt has been generated. Upon receiving the flag, the interrupt thread calls the API’s ISR. The thread then performs appropriate actions based on the value of the event vector returned by the API’s ISR.

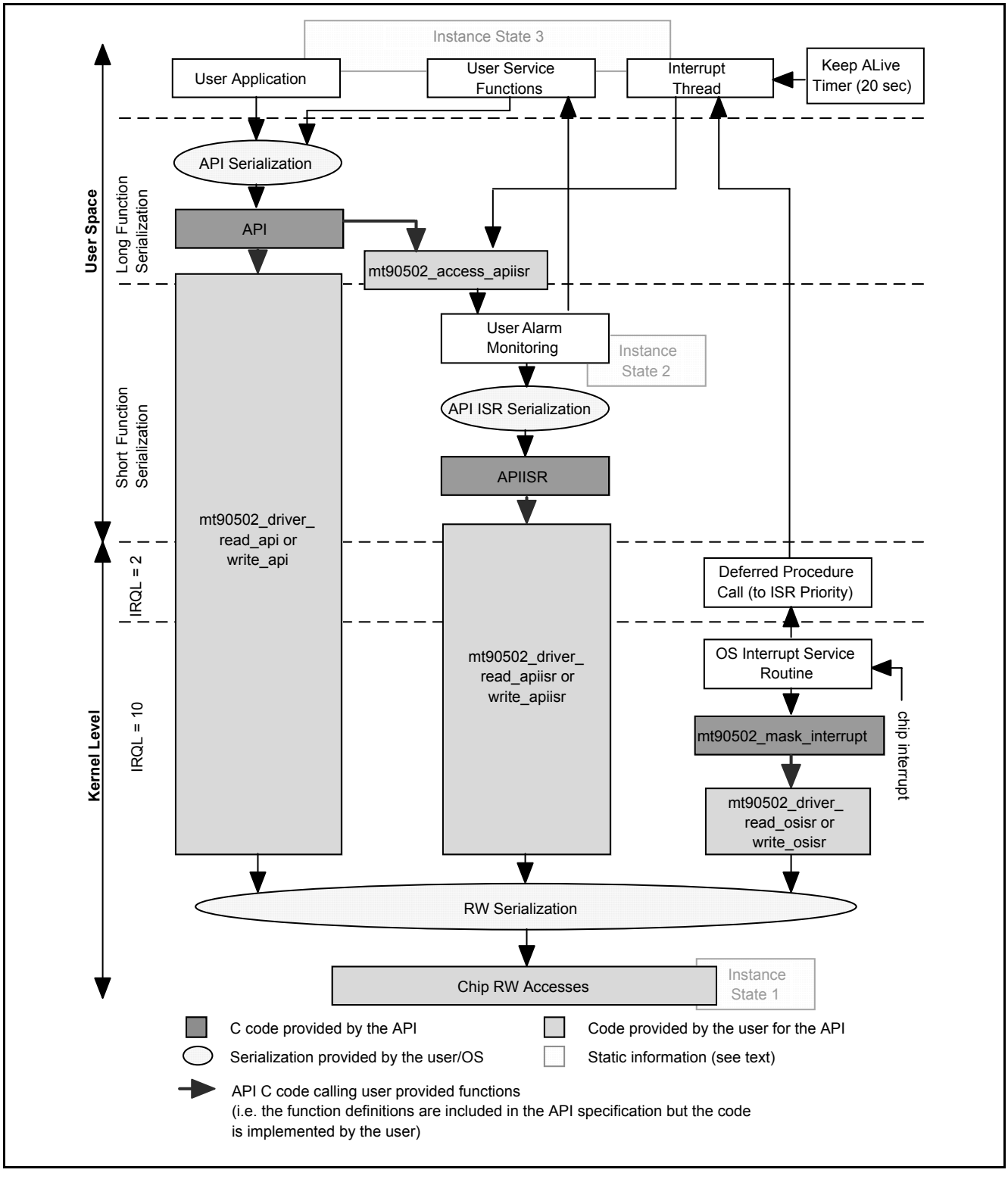


Figure 3 - NT System Architecture

The APIISR must be called at least every 20 seconds. If it is not, counters within the chip will not be updated correctly in the API, causing the API to fall out of synchronization with the chip, which can lead to system to failure. In a system where the interrupt line of the chip is routed to a CPU, code in the APIISR insures that it will be called at least at the required frequency. In the case where the interrupt line is not physically routed, a keep-alive timer is needed by the system, as illustrated in the system architecture figures above. The timer insures the APIISR is called at least every 20 seconds. Although calling the APIISR every 20 seconds is enough to keep the chip running correctly, VC and chip statistics counters have to be kept up to date more frequently via calls to the mt90502_update_counters function. If such calls are not done frequently enough incorrect statistics may be present in the API structures.

2.0 API Function Descriptions

Each function's use as well as its parameters are described here in detail. The typical usage of the above functions is as follows:

- A parameter structure is allocated.
- The appropriate open default configuration function is called. These functions are identified by the "_def" suffix at the end of the function name.
- The user changes the default configuration structure to suit his needs.
- The actual function is called.

An example of this sequence is the initialization of the chip. Note that in the following example the system architecture is assumed to have all code (API and APIISR) in the same memory space:

```
#include "mt90502_api.h"

void main( )

{

            MT90502_INSTANCE_API*              pmt90502_api;

            MT90502_INSTANCE_APIISR*           pmt90502_apiisr;

            MT90502_CONF                       mt90502_conf;

            MT90502_INSTANCE_SIZE              mt90502_inst_size;

            ULONG                              result;


            // Inserting default values into structure configuration parameters.

            mt90502_open_def(&mt90502_conf);


            // Change default parameters as needed (e.g. changing the clock
            frequencies).

            mt90502_conf.upclk_freq = 30000000;

            mt90502_conf.mclk_freq = 60000000;


            // Inserting default values into MT90502_INSTANCE_SIZE structure
            parameters.
```

```
            mt90502_open_instance_size_def(&mt90502_conf, &mt90502_inst_size);


            // Get the size of the MT90502_INSTANCE structures.
            result = mt90502_open_instance_size(&mt90502_conf,
            &mt90502_inst_size);
            if (result != MT90502ER_GENERIC_OK)
{
                        // Error handling.
}


            // Allocate memory for the mt90502_instance structure
            pmt90502_api = (MT90502_INSTANCE_API*)malloc(
            mt90502_inst_size.instance_api_size);
            if (pmt90502_api == NULL)
{
                        // Error handling.
}
            pmt90502_apiisr = (MT90502_INSTANCE_APIISR*)malloc(
            mt90502_inst_size.instance_apiisr_size);
            if (pmt90502_apiisr == NULL)
{
                        // Error handling.
}


            // Perform the actual configuration of the chip.
            result = mt90502_open(pmt90502_api, &mt90502_conf);
            if (result != MT90502ER_GENERIC_OK)
{
                        // Error handling.
}
}
```

Every function has a pointer to the chip's API instance structure as the first parameter. This instance structure is created by the user before the call to mt90502_open and is unique to each chip being managed by the software.

The structure keeps the state of an instance of a chip and is required to perform any operations on the chip. The APIISR instance structure is kept by the system, and passed as a parameter to the APIISR code entity when the interrupt service routine is to be called (by the user or the API entity).

2.1 Initialization Functions:

2.1.1 mt90502_open

Using the provided configuration structure MT90502_CONF, mt90502_open performs all the necessary operations to configure the chip and initialize the instance structure. Note that the functions mt90502_open_def and mt90502_open_instance_size are typically called, in their respective order, before this function.

The mt90502_open_def function inserts default values into the MT90502_CONF structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_def( MT90502_CONF* pmt90502_conf );

ULONG mt90502_open( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONF* pmt90502_conf );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to the chip's API instance structure. This structure will be filled in by this function call. It contains information of the current state and configuration of the chip. After initialization by mt90502_open this structure is supplied to all subsequent function calls. The structure must be created and kept by the application software until mt90502_close is called.

pmt90502_conf a pointer to an initial configuration structure MT90502_CONF. The definition of the structure is provided in Section 5.0 "Configuration Structures", as are the default values inserted by mt90502_open_def.

2.1.2 mt90502_init_sil_suppr_profile

This function inserts a default silence suppression profile into the provided MT90502_SIL_SUPPR_PROFILE structure, if a default silence suppression profile is needed. If required, this function should be called after the mt90502_open_def, and before the mt90502_open.

The mt90502_init_sil_suppr_profile_def function inserts default values into the MT90502_INIT_SIL_SUPPR_PROFILE structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_init_sil_suppr_profile_def(
            MT90502_INIT_SIL_SUPPR_PROFILE* pinit_profile );
```

```
ULONG mt90502_init_sil_suppr_profile(
            MT90502_INIT_SIL_SUPPR_PROFILE* pinit_profile );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pinit_profile a pointer to an initial configuration structure MT90502_INIT_SIL_SUPPR_PROFILE. The definitions of the structure’s parameters are provided below.

2.1.2.1 Structure MT90502_INIT_SIL_SUPPR_PROFILE

psil_suppr_profile MT90502_SIL_SUPPR_PROFILE structure

This parameter is a pointer to a silence suppression profile. The structure is filled with the default profile indicated in MT90502_SIL_SUPPR_PROFILE Structure. The memory for the structure, and for the structure’s state table pointer (sil_suppr_state_table) must be allocated by the user before this function is called.

Direction: IN/OUT Type: POINTER

Default: NULL

psil_suppr_state_table pointer to table

A pointer to a silence suppression state table that will be initialized to default values by this function and inserted into the initialized MT90502_SIL_SUPPR_PROFILE. See structure MT90502_SIL_SUPPR_PROFILE for a description of silence suppression state table. The values for the table are diagramed below. If this pointer is NULL then no table will be created.

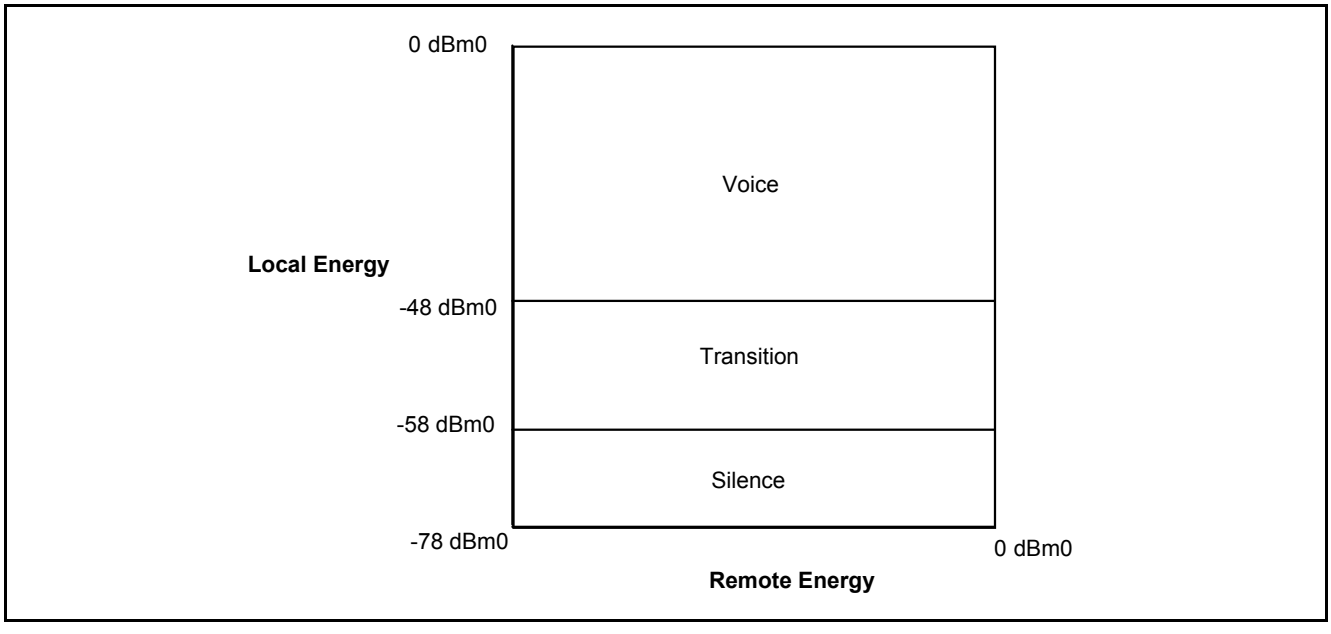


Figure 4 - Default Silence Suppression State Table

Direction: IN/OUT Type: POINTER

Default: NULL

2.1.3 mt90502_open_instance_size

Using the provided configuration structure MT90502_CONF, mt90502_open_instance_size calculates the amount of memory required for the MT90502_INSTANCE_API and MT90502_INSTANCE_APIISR structures of the chip. An MT90502_INSTANCE_API structure and an MT90502_INSTANCE_APIISR structure must be allocated and pointers created by the user before calling the mt90502_open function; both pointers must point to blocks of contiguous memory whose sizes are determined by this function.

The mt90502_open_instance_size_def function inserts default values into the MT90502_INSTANCE_SIZE structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_instance_size_def(MT90502_INSTANCE_SIZE* pinstance_size );

ULONG mt90502_open_instance_size( MT90502_CONF* pconf,
            MT90502_INSTANCE_SIZE* pinstance_size );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pconf a pointer to an initial configuration structure MT90502_CONF. The definition of the structure is provided in Section 5.0 "Configuration Structures". See mt90502_open_def for a default configuration of the chip. The user allocates this structure.

pinstance_size pointer to an MT90502_INSTANCE_SIZE structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.1.3.1 Structure MT90502_INSTANCE_SIZE

instance_api_size ?? – ??

This value is returned by the function and indicates the size, in bytes, of the MT90502_INSTANCE_API memory block that must be allocated to support the supplied configuration.

Direction: Out Type: ULONG

Default: NOT MODIFIED

instance_apiisr_size ?? – ??

This value is returned by the function and indicates the size, in bytes, of the MT90502_INSTANCE_APIISR memory block that must be allocated to support the supplied configuration.

Direction: Out Type: ULONG

Default: NOT MODIFIED

2.1.4 mt90502_close

This function closes any VCs or channels that may still be open and then puts the chip in reset.

The mt90502_close_def function inserts default values into the MT90502_CLOSE_CHIP structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_close_def( MT90502_INSTANCE_API* pmt90502_api,
                         MT90502_CLOSE_CHIP* pclose_chip );

ULONG mt90502_close( MT90502_INSTANCE_API* pmt90502_api,
                     MT90502_CLOSE_CHIP* pclose_chip );
```

Return Values

MT90502ER_GENERIC_OK Indicate success.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api pointer to the instance structure of the chip.

pclose_chip pointer to an MT90502_CLOSE_CHIP structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.1.4.1 Structure MT90502_CLOSE_CHIP

Currently there are no parameters for this structure.

2.1.5 mt90502_get_hw_revision

This routine returns the hardware revision number of the MT90502. The revision number is contained in a register of the device. This function may be called before the device is open and only requires upclk to be present on the device.

The mt90502_get_hw_revision_def function inserts default values into the MT90502_REVISION structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_hw_revision_def(MT90502_REVISION* prevision );

ULONG mt90502_get_hw_revision(MT90502_REVISION* prevision );
```

Return Values

MT90502ER_GENERIC_OK Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

prevision pointer to an MT90502_REVISION structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.1.5.1 Structure MT90502_REVISION

user_chip_number identifier

This number is carried down to the user-supplied read/write routines to distinguish which chip the API is servicing. This can be used as an array index of the chip to be serviced to retrieve the correct instance pointer. If only one chip is being serviced by the API, then this parameter can be ignored. See System Architecture Section.

Direction IN Type: ULONG

Default: UNDEFINED

revision_number 0 – ??

This value is returned by the function and indicates the revision of the device.

Direction: Out Type: ULONG

Default: UNDEFINED

2.2 ATM Functions:

2.2.1 mt90502_open_aal2_vc

This function opens an AAL2 VC on the UTOPIA bus.

If the call to this function is successful, cells can be sent by the TXSAR and received by the RXSAR immediately following the call. However, no source of cells is associated to the VC. Thus no cells will be assembled by the TXSAR until active channels are mapped to the VC. The handling of received AAL2 packets depends default_rx_uui_mapping parameter of the MT90502_CONF structure. The routing of received AAL2 cells can be modified by opening and mapping CIDs on this VC. CIDs are opened by the function mt90502_open_cid. Channels are mapped to a VC via the function mt90502_map_cid. Channels are opened by the functions mt90502_open_rx_xxpcm_channel, mt90502_open_tx_xxpcm_channel, mt90502_open_rx_hdlc_channel, and mt90502_open_tx_hdlc_channel.

When the UTOPIA module receives cells for this VC, they will be routed according to the rx_normal_cell_routing and rx_oam_cell_routing fields. This routing must not conflict with other VCs that are received on a common UTOPIA port. Two VCs conflict when, after the application of the u_txp_network_mask (where p is the port), they have the same header on the same UTOPIA port. Each UTOPIA port is given a u_txp_network_mask during the call to mt90502_open.

In some applications it may be necessary to map the transmit and receive mini packet streams of a given CID onto separate VCs. In this situation the returned vc_hndl describes both VCs and any mini packet streams mapped to this vc_hndl will use separate VCs for transmitted and received mini packets. (See split_vc parameter)

This function returns a handle by which the API identifies this VC.

The mt90502_open_aal2_vc_def function inserts default values into the AAL2 VC configuration structure, MT90502_AAL2_VC. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_aal2_vc_def ( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_AAL2_VC* paal2_vc );

ULONG mt90502_open_aal2_vc ( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_AAL2_VC* paal2_vc );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

paal2_vc a pointer to an MT90502_AAL2_VC structure. The definitions of the structure’s elements are listed below. The user allocates this structure.

2.2.1.1 Structure MT90502_AAL2_VC

pvc_hndl handle

a pointer to a single ULONG which returns the handle for the created AAL2 VC(s). This handle is a unique value that identifies the VC(s) in all future function calls affecting the streams carried on the VC(s). The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

header 32 bit field

Header of the VC. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: IN Type: ULONG

Default: MT90502_NULL_HEADER

split_streams TRUE / FALSE

If TRUE the mini packet streams associated with the returned vc_hndl will use separate VCs for transmit and receive streams. In this mode the header parameter represents the header fields to be used for the transmitted VC. The rx_header parameter represents the expected header for the received streams VC.

Direction: IN Type: ULONG

Default: FALSE

rx_header 32 bit field

When split_streams is TRUE, this parameter represents the expected header for the received streams VC. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP. When split_streams is FALSE this parameter is ignored.

Direction: IN Type: ULONG

Default: MT90502_NULL_HEADER

rx_tx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

Sets the UTOPIA TX port to which the VC’s cells are destined once they exit the TXSAR, and from which UTOPIA RX port cells enter the chip. The VC(s) can only be associated to one port. This field works in conjunction with loopback.

Direction: IN Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

loopback TRUE / FALSE

In the TX direction, the cells produced by the TXSAR can either exit the chip from the determined UTOPIA port (no loopback, FALSE) or can be treated as if they had entered the chip from that same port (loopback, TRUE). See rx_tx_utopia_port above. A typical application is to perform self tests by feeding data on the TDM bus and routing the cells produced by the TXSAR back to the RXSAR, which disassembles the cells, and puts the data on the TDM bus.

Direction: IN Type: ULONG

Default: FALSE

rx_normal_cell_routing 0, or the OR of any or all of:
MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_DATA_CELL_FIFO

This field routes the VC's non-OAM cells entering the chip via rx_tx_utopia_port. The values can be ORed together to broadcast the cell. Note that the value selected for rx_tx_utopia port cannot be selected for rx_normal_cell_routing. Since this function opens an AAL2 VC, which must be disassembled by the RX SAR, normal cells are automatically routed to the RXSAR. If the field is set to 0, the normal cells on this VC will only go to the RXSAR.

Direction: IN Type: ULONG

Default: 0

rx_oam_cell_routing 0, or the OR of any or all of:
MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_DATA_CELL_FIFO

This field routes the VC's OAM cells entering the chip via rx_tx_utopia_port. The values can be ORed together to broadcast the cell. Note that the value selected for rx_tx_utopia port cannot be selected for rx_oam_cell_routing. If the field is set to 0, the OAM cells on this VC will be discarded.

Direction: IN Type: ULONG

Default: 0

tx_max_sar_delay 125 – 2048000 us

Max time, in us, the TX SAR will wait to send a partially filled cell on the network. This value is interpreted in units of 125 us (TDM frames).

Direction: IN Type: ULONG

Default: 1000

adap_src_a TRUE / FALSE

Whether the VC is to be used in the RX direction to generate clock recovery points for buffer A.

Direction: IN Type: ULONG

Default: FALSE

adap_src_b see **adap_src_a**

Default: FALSE

force_single_packet_cell TRUE/FALSE

If TRUE all AAL2 mini-packets sent on this VC will be sent in a cell with no other AAL2 mini-packets.

Direction: IN Type: ULONG

Default: FALSE

2.2.2 mt90502_open_data_vc

This function opens a VC from a UTOPIA bus to the data cell FIFO or back to a UTOPIA bus. The VC can be opened for many uses, such as terminating the VC in the data cell FIFO, routing the VC to a secondary SAR, or performing a header change on the VC.

When the UTOPIA module receives cells for this VC, they will be routed according to the rx_normal_cell_routing and rx_oam_cell_routing fields. This routing must not conflict with other VCs that are received on a common UTOPIA port. Two VCs conflict when, after the application of the u_txp_network_mask (where p is the port), the received cells have the same header on the same UTOPIA port. Each UTOPIA port is given a u_txp_network_mask during the call to mt90502_open.

This function returns a handle by which the API identifies this VC.

The mt90502_open_data_vc_def function inserts default values into the DATA VC configuration structure, MT90502_DATA_VC. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_data_vc_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_DATA_VC* pdata_vc);

ULONG mt90502_open_data_vc( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_DATA_VC* pdata_vc);
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

pdata_vc a pointer to an MT90502_DATA_VC structure. The definitions of the structure's elements are listed below.

2.2.2.1 Structure MT90502_DATA_VC

pvc_hndl handle

a pointer to a single ULONG which returns the handle for the created data VC. This handle is a unique value that identifies the data VC in all future function calls affecting this VC. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

header 32 bit field

header of the VC. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: IN Type: ULONG

Default: MT90502_NULL_HEADER

rx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

The UTOPIA port on which cells for this VC will enter the chip.

Direction: IN Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

rx_normal_cell_routing 0, or the OR of any or all of:
MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_DATA_CELL_FIFO

The routing of non-OAM cells once they have entered the chip via the UTOPIA port specified by rx_utopia_port. Since this is a data VC, the cells cannot be routed to the RXSAR. A value of 0 will cause non-OAM cells to be discarded.

Direction: IN Type: ULONG

Default: MT90502_DATA_CELL_FIFO

rx_oam_cell_routing 0, or the OR of any or all of:
MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_DATA_CELL_FIFO

The routing of OAM cells once they have entered the chip via the UTOPIA port specified in rx_utopia_port. A value of 0 will cause OAM cells to be discarded.

Direction: IN Type: ULONG

Default: MT90502_DATA_CELL_FIFO

replace_gfc TRUE / FALSE

Whether the GFC bits of the cell's header are to be changed, or not, by the VC's LUT entry. The new value of the GFC bits is determined by the value of new_gfc.

Direction: IN Type: ULONG

Default: FALSE

replace_vpi TRUE / FALSE

Whether the VPI bits of the cell's header are to be changed, or not, by the VC's LUT entry. The new value of the VPI bits is determined by the value of new_vpi.

Direction: IN Type: ULONG

Default: FALSE

replace_vci TRUE / FALSE

Whether the VCI bits of the cell's header are to be changed, or not, by the VC's LUT entry. The new value of the VCI bits is determined by the value of new_vci.

Direction: IN Type: ULONG

Default: FALSE

new_gfc 4 bit field

The new GFC bits of the cell's header if GFC replacement is requested (replace_gfc = TRUE). The GFC bits will be replaced by the LUT entry corresponding to the VC, as the VC is routed.

Direction: IN Type: ULONG

Default: 0x0

new_vpi 8 bit field

The new VPI bits of the cell's header if VPI replacement is requested (replace_vpi = TRUE). The VPI bits will be replaced by the LUT entry corresponding to the VC, as the VC is routed.

Direction: IN Type: ULONG

Default: 0x00

new_vci 16 bit field

The new VCI bits of the cell's header if VCI replacement is requested (replace_vci = TRUE). The VCI bits will be replaced by the LUT entry corresponding to the VC, as the VC is routed.

Direction: IN Type: ULONG

Default: 0x0000

2.2.3 mt90502_close_vc

This function closes the VC indicated by the handle pvc_hndl, regardless of the payload type of the VC (AAL2 or data). All resources that were reserved by a call to mt90502_open_aal2_vc or mt90502_open_data_vc are released. If the VC is an AAL2 VC, all channels allocated to the VC will also be closed by this function.

The mt90502_close_vc_def function inserts default values into the MT90502_CLOSE_VC structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_close_vc_def(

            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_VC* pclose_vc);
```

```
ULONG mt90502_close_vc(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_VC* pclose_vc );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success.

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_ap a pointer to an API instance structure of the chip.

pclose_vc pointer to an MT90502_CLOSE_VC structure. The definitions of the structure’s elements are listed below.

2.2.3.1 Structure MT90502_CLOSE_VC

pvc_hndl handle

a pointer to a single ULONG, containing the handle of the VC created by a call to mt90502_open_xxxx_vc. The value of the handle will be modified to a unique value for closed handles as a code check.

Direction: IN/IO Type: POINTER

Default: NULL

2.2.4 mt90502_open_cid

This function opens a mini packet stream identified by a CID, associates that stream to an open AAL2 VC indicated by vc_hndl, and allocates any required resources to support a mini packet stream.

This function returns a handle by which the API identifies this channel.

The mt90502_open_cid_def function inserts default values into the MT90502_OPEN_CID structure. The default value of a structure field is indicated following the field’s description.

Usage

```
#include “mt90502_api.h”

ULONG mt90502_open_cid_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_OPEN_CID* popen_cid );

ULONG mt90502_open_cid(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_OPEN_CID* popen_cid );
```

Return Values

MT90502ER_GENERIC_OK	on success.

Also, see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

popen_cid a pointer to an MT90502_OPEN_CID structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.2.4.1 Structure MT90502_OPEN_CID

pcid_hndl handle

a pointer to a single ULONG which returns the handle for the created mini packet stream. This handle is a unique value that identifies the stream in all future function calls affecting this stream. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

vc_hndl handle

The handle returned from the call to mt90502_open_aal2_vc. This handle specifies the VC to which the new mini packet stream will be associated.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

cid_num 1 – 255

The CID value which distinguishes this mini packet stream within the VC indicated by vc_hndl.

Direction: IN Type: ULONG

Default: MT90502_INVALID_CID

rx_voice_one_only TRUE / FALSE

If TRUE, when the rx_uui_mapping for UUIs 0–15 is MT90502_RX_MINI_PKT_BUFFER, the first mini-packet received on UUIs 0–15 will generate an entry in the CID event buffer and the mapping will be modified by the API to MT90502_DELETE_MINI_PKT. Voice packets will be discarded until the CID is mapped or the CID is closed and re-opened with a new mapping.

By configuring traffic for an unmapped CID to the RX CPU mini packet buffer the processor can receive packets even though the CID has no configured TDM channel. If traffic is detected that indicates a voice connection is required the user software can map the CID to the appropriate TDM channel. This allows oversubscription of CIDs.

Direction: IN Type: ULONG

Default: FALSE

rx_uui_mapping[17] MT90502_DELETE_MINI_PKT
MT90502_RX_MINI_PKT_BUFFER
MT90502_PRESERVE_MAPPING
MT90502_RX_CHANNEL

Each element of the array indicates the mapping of a received UUI value(s) on the mini packet stream identified by cid_hndl. The indices correspond to UUI values in the following way:

index = 0 => UUIs = 0 – 15
index = 1 => UUI = 16
index = 2 => UUI = 17
… …
index = 16 => UUI = 31

Each mini-packet received on this stream can be either: deleted (MT90502_DELETE_MINI_PKT); sent to an open receive channel identified by a receive ch_hndl, (returned by an mt90502_open_rx_xxpcm_channel or mt90502_open_rx_hdlc_channel function call); sent to the CPU RX mini-packet buffer (MT90502_RX_MINI_PKT_BUFFER); or the current mapping in the device for the UUI value can be preserved. If enable_cas in the MT90502_CONF structure is set to TRUE, UUI 24 should be set to MT90502_RX_MINI_PKT_BUFFER to allow CAS reception for API processing per ITU standard I366.2.

Direction: IN Type: ULONG[17]

Default: MT90502_PRESERVE_MAPPING

tx_chan_mapping

MT90502_NO_TX_CHANNEL
MT90502_PRESERVE_MAPPING
MT90502_TX_CHANNEL

This parameter is used to either: map no transmit channel (MT90502_NO_TX_CHANNEL); preserve the current channel mapping for this mini packet stream (MT90502_PRESERVE_MAPPING); or map an open transmit channel (opened via a call to mt90502_open_tx_xxpcm_channel or mt90502_open_tx_hdlc_channel), to the mini packet stream. When no channel is mapped to the mini packet stream, CPU transmitted mini packets may still be sent.

The UUI used for transmitting mini packets on a CID is controlled by other structures. In the case of CPU sent mini packets it is controlled by the packet the CPU forms, for HDLC streams it can be embedded in the frame, and for xxPCM channels it is always 0-15.

Direction: IN Type: ULONG

Default: MT90502_PRESERVE_MAPPING

rx_ch_hndl

The handle of the RX channel to which the CID is mapped in the RX direction. If one or more nodes of the rx_uui_mapping array is set to MT90502_RX_CHANNEL this must contain a valid handle. The channel can be either xxPCM or HDLC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

tx_ch_hndl

The handle of the TX channel to which the CID is mapped in the TX direction. If the tx_chan_mapping is set to MT90502_TX_CHANNEL this must contain a valid handle. The channel can be either xxPCM or HDLC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

2.2.5 mt90502_map_cid

This function maps how the device should direct received mini packets based on the mini packet's UUI field and what open TDM channel should be mapped on this CID in the transmit direction.

This function may be called multiple times for an open CID in order to map or un map available TDM channels.

The mt90502_map_cid_def function inserts default values into the MT90502_MAP_CID structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_map_cid_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_MAP_CID* pmap_cid );

ULONG mt90502_map_cid( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_MAP_CID* pmap_cid );
```

Return Values

MT90502ER_GENERIC_OK	on success.

Also, see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

pmap_cid a pointer to an MT90502_MAP_CID structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.2.5.1 Structure MT90502_MAP_CID

cid_hndl handle

The handle returned from the call to mt90502_open_cid. This handle specifies the mini packet stream for which UUIs are being mapped.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

rx_voice_one_only TRUE / FALSE

If TRUE, when the rx_uui_mapping for UUIs 0–15 is MT90502_RX_MINI_PKT_BUFFER, the first mini-packet received on UUIs 0–15 will generate an entry in the CID event buffer and the mapping will be modified by the API to MT90502_DELETE_MINI_PKT. Voice packets will be discarded until the CID is re-mapped, or the CID or VC is closed and re-opened with a new mapping.

Direction: IN Type: ULONG

Default: FALSE

rx_uui_mapping[17] MT90502_DELETE_MINI_PKT
MT90502_RX_MINI_PKT_BUFFER
MT90502_PRESERVE_MAPPING
MT90502_RX_CHANNEL

Each element of the array indicates the mapping of a received UUI value(s) on the mini packet stream identified by cid_hndl. The indices correspond to UUI values in the following way:

index = 0 => UUIs = 0 – 15
index = 1 => UUI = 16
index = 2 => UUI = 17
… …
index = 16 => UUI = 31

Each mini-packet received on this stream can be either: deleted (MT90502_DELETE_MINI_PKT); sent to an open receive channel identified by a receive

ch_hndl (returned by an mt90502_open_rx_xxpcm_channel or mt90502_open_rx_hdlc_channel function call); sent to the CPU RX mini-packet buffer (MT90502_RX_MINI_PKT_BUFFER); or the current mapping in the device for the UUI value can be preserved. If enable_cas in the MT90502_CONF structure is set to TRUE, UUI 24 should be set to MT90502_RX_MINI_PKT_BUFFER to allow CAS reception for API processing per ITU standard I366.2.

Direction: IN Type: ULONG[17]

Default: MT90502_DELETE_MINI_PKT

tx_chan_mapping MT90502_NO_TX_CHANNEL
MT90502_PRESERVE_MAPPING
MT90502_TX_CHANNEL

This parameter is used to either: map no transmit channel (MT90502_NO_TX_CHANNEL); preserve the current channel mapping for this mini packet stream (MT90502_PRESERVE_MAPPING); or map an open transmit channel (opened via a call to mt90502_open_tx_xxpcm_channel or mt90502_open_tx_hdlc_channel), to the mini packet stream. When no channel is mapped to the mini packet stream, CPU transmitted mini packets may still be sent.

The UUI used for transmitting mini packets on a CID is controlled by other structures. In the case of CPU sent mini packets it is controlled by the packet the CPU forms, for HDLC streams it can be embedded in the frame, and for xxPCM channels it is always 0-15.

Direction: IN Type: ULONG

Default: MT90502_NO_TX_CHANNEL

rx_ch_hndl

The handle to the RX channel to which the CID is mapped to in the RX direction if one or more nodes of the rx_uui_mapping array is set to MT90502_RX_CHANNEL. The channel can be either xxPCM or HDLC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

tx_ch_hndl

The handle to the TX channel to which the CID is mapped to in the TX direction if the tx_chan_mapping is set to MT90502_TX_CHANNEL. The channel can be either xxPCM or HDLC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

cancel_cas_refresh TRUE / FALSE

If set to TRUE the CAS refresh packet transmissions by the API will be canceled if they are currently enabled. See cas_refresh parameter of the mt90502_send_cas_pkt function.

Direction: IN Type: ULONG

Default: TRUE

2.2.6 mt90502_close_cid

This function closes the mini packet stream indicated by the handle cid_hndl. All resources that were reserved by the call to mt90502_open_cid are released. All channels allocated to the CID will also be closed by this function.

The mt90502_close_cid_def function inserts default values into the MT90502_CLOSE_CID structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_close_cid_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_CID* pclose_cid );

ULONG mt90502_close_cid( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_CID* pclose_cid );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to an API instance structure of the chip.

pclose_cid pointer to an MT90502_CLOSE_CID structure. The definitions of the structure's elements are listed below.

2.2.6.1 Structure MT90502_CLOSE_CID

pcid_hndl handle

a pointer to a single ULONG, containing the handle of the CID created by a call to mt90502_open_cid. The value of the handle will be modified to a unique value for closed handles as a code check.

Direction: IN/IO Type: POINTER

Default: NULL

2.3 TDM Functions

2.3.1 mt90502_open_rx_xxpcm_channel

This function opens a received XXPCM channel allocating all required resources. Once opened it may be mapped to a mini packet stream using the function mt90502_map_cid.

The directions TX and RX are with respect to the UTOPIA bus. Thus, a TX TSST enters the chip on the TDM bus, and an RX TSST exits the chip on the TDM bus.

A TSST can only be used once, whether it is in an XXPCM channel, an HDLC stream, or a low-latency-loopback channel.

This function returns a handle by which the API identifies this channel.

The mt90502_open_rx_xxpcm_channel_def function inserts default values into the xxPCM channel configuration structure, MT90502_RX_XXPCM_CHAN. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_rx_xxpcm_channel_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_RX_XXPCM_CHAN* prx_xxpcm_chan );

ULONG mt90502_open_rx_xxpcm_channel(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_RX_XXPCM_CHAN* prx_xxpcm_chan );
```

Return Values

MT90502ER_GENERIC_OK	on success.

Also, see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

prx_xxpcm_chan a pointer to an MT90502_RX_XXPCM_CHAN structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.3.1.1 Structure MT90502_RX_XXPCM_CHAN

pch_hndl handle

a pointer to a single ULONG which returns the handle for the created PCM / ADPCM channel. This handle is a unique value that identifies the channel in all future function calls affecting this channel. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

timeslot

0 – 127 for stream frequency of 8 MHz
0 – 63 for stream frequency of 4 MHz
0 – 31 for stream frequency of 2 MHz

The timeslot of the TSST of the channel. Note that the allowed values are affected by the format of the stream and frequency of the clock controlling the stream (see TDM Configuration section of the MT90502_CONF structure). This value may also reserve additional timeslots if the channel has silence suppression enabled. See Silence Suppression Configuration Parameters.

Direction: IN Type: ULONG

Default: MT90502_INVALID_TIMESLOT

stream

0 – 31 for max_stream of 32
0 – 15 for max_stream of 16
0 – 7 for max_stream of 8
0 – 3 for max_stream of 4

The stream of the TSST of the channel. Note that the allowed values are also affected by: the value of max_stream, which was specified during the call to mt90502_open; and whether this channel is configured with silence suppression See Silence Suppression Configuration Parameters.

Direction: IN Type: ULONG

Default: MT90502_INVALID_STREAM

pcm_law_translation

MT90502_ULAW_TO_ALAW
MT90502_ULAW_TO_ULAW
MT90502_ALAW_TO_ULAW
MT90502_ALAW_TO_ALAW

Received PCM samples can have their PCM Law translated before being placed on the on the H.1x0 bus. MT90502_ULAW_TO_ALAW will translate received u-Law samples to A-Law before passing them to the H.1x0 bus. The behavior of MT90502_ULAW_TO_ULAW is dependent on the setting of u_law_no_zero in the MT90502_CONF structure. MT90502_ALAW_TO_ULAW. will translate received A-Law samples to u-Law. Direction: IN Type: ULONG

Default: MT90502_ULAW_TO_ULAW

compression_rate

MT90502_COMP_PCM_64KBPS
MT90502_COMP_ADPCM_40KBPS
MT90502_COMP_ADPCM_32KBPS
MT90502_COMP_ADPCM_24KBPS
MT90502_COMP_ADPCM_16KBPS
MT90502_COMP_AUTO_DETECT

The compression rate of the channel. The channel can be compressed with PCM (64 kbps) or ADPCM (40, 32, 24, 16 kbps). If the compression is set to MT90502_COMP_AUTO_DETECT compression changes will be determined from the value of the received LI field and placed on the H.100 bus according to the format specified by the adpcm_bit_positions parameter and dstream format of the selected TSST. If the detect_li_change and, for 40-byte or 44-byte mode, detect_uui_change are set to TRUE then compression changes will also be reported.

Direction: IN Type: ULONG

Default: MT90502_COMP_PCM_64KBPS

number_of_edus

MT90502_1_EDU
MT90502_2_EDU
MT90502_3_EDU
MT90502_4_EDU
MT90502_5_EDU
MT90502_44_BYTE
MT90502_40_BYTE
MT90502_8_EDU

The number of EDUs (Elementary Data Units) the device will use to disassemble the mini-packet. For MT90502_44_BYTE and MT90502_40_BYTE the number of EDUs is dependant on the compression_rate and is adjusted by the device to maintain 44 or 40 Byte mini-packets. In 44 and 40 byte EDU modes MT90502_COMP_PCM_64KBPS, MT90502_COMP_ADPCM_32KBPS or MT90502_COMP_AUTO_DETECT are the only valid compression_rate configurations. If the length indicated by the LI field and/or UUI field of the received mini-packet is incompatible with this parameter the packet will be discarded.

Direction: IN Type: ULONG

Default: MT90502_5_EDU

num_uui_counter_bits 0 – 4

This field determines the number of least-significant bits of the UUI that are used as a counter for sequencing voice packets. For 44 and 40 byte EDU mode the num_uui_counter_bits must be 3 or less.

Direction: IN Type: ULONG

Default: 4

underrun_padding MT90502_UR_PAD_TONE_BUFFER_X (0 - 63)
MT90502_UR_PAD_SILENT_NOISE_X (0 - 31)
MT90502_UR_PAD_NULL_PATTERN_X (0 - 31)
MT90502_UR_PAD_WITH_SIDS
MT90502_UR_PAD_NO_PADDING

The selection of the byte that will be sent in the case of an underrun. An X in a value is to be replaced by a number in the corresponding range. If MT90502_UR_PAD_NO_PADDING is selected then the XXPCM channel statistic underrun_byte_cnt is not valid. MT90502_UR_PAD_WITH_SIDS will pad with the configured silence noise source and the energy level of the last received SID.

Direction: IN Type: ULONG

Default: MT90502_UR_PAD_NULL_PATTERN_31

detect_li_change TRUE / FALSE

If TRUE then changes in non-zero values of LI of received voice mini-packets will be reported (e.g. because generic SID packets have an LI of 0 they are ignored for this evaluation). This event can be used to communicate compression rate changes to user software. See User Service Routines.

Direction: IN Type: ULONG

Default: TRUE

detect_uui_match 0-4

The number UUI bits to match for detecting changes. The bits used are the most significant bits of the UUI field. For example if one bit is specified only UUI[4] will be compared.

Direction: IN Type: ULONG

Default: 0

detect_uui_change TRUE / FALSE

If TRUE then changes values of the UUI of received voice mini-packets will be reported (generic SID packets have an LI of 0 and they are ignored for this evaluation). This event can be used to communicate compression rate changes to user software. See User Service Routines.

Direction: IN Type: ULONG

Default: FALSE

maximum_pdv 0 – (see below)

The amount of PDV the channel must absorb in microseconds. A larger value for this field means the amount of data buffered for disassembling the cells will be larger. A value of 1 ms means that +/- 1 ms of delay variation will be absorbed; thus a worst case variation of 2 ms between two packets could potentially be absorbed.

The maximum value for this parameter is determined by the size of the circular buffers selected in mt90502_open (rx_circular_buffer_size) and the number of EDUs in a mini packet selected above (number_of_edus). The following equation gives maximum value:

max value = 125 * (rx_circular_buffer_size – 4 – (number_of_edus * 8))/2

Direction: IN Type: ULONG

Default: 8000

mini_pkt_loss_compensation TRUE / FALSE

When set to TRUE the mini-packet loss compensation circuit will be enabled. This mechanism uses the UUI sequence number to place the mini-packet correctly in the RX circular buffer. This parameter should be set to FALSE if the number of UUI counter bits used is insufficient to perform PDV compensation.

Direction: IN Type: ULONG

Default: TRUE

adap_src_a TRUE / FALSE

Whether the mini packets of the channel are to be used in the RX direction to generate clock recovery points for buffer A.

Direction: IN Type: ULONG

Default: FALSE

adap_src_b see **adap_src_a**

Default: FALSE

2.3.2 mt90502_open_tx_xxpcm_channel

This function opens an XXPCM channel in the transmit direction allocating all required resources. Once opened it may be mapped to a mini packet stream using the function mt90502_map_cid

The directions TX and RX are with respect to the UTOPIA bus. Thus, a TX TSST enters the chip on the TDM bus, and an RX TSST exits the chip on the TDM bus.

A TSST can only be used once, whether it is in an XXPCM channel, an HDLC stream, or a low-latency-loopback channel.

This function returns a handle by which the API identifies this channel.

The mt90502_open_tx_xxpcm_channel_def function inserts default values into the xxPCM channel configuration structure, MT90502_TX_XXPCM_CHAN. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"
```

```
ULONG mt90502_open_tx_xxpcm_channel_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_XXPCM_CHAN* ptx_xxpcm_chan );

ULONG mt90502_open_tx_xxpcm_channel(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_XXPCM_CHAN* ptx_xxpcm_chan );
```

Return Values

MT90502ER_GENERIC_OK	on success.

Also, see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

ptx_xxpcm_chan a pointer to an MT90502_TX_XXPCM_CHAN structure. The definitions of the structure’s elements are listed below. The user allocates this structure.

2.3.2.1 Structure MT90502_TX_XXPCM_CHAN

pch_hndl handle

a pointer to a single ULONG which returns the handle for the created PCM / ADPCM channel. This handle is a unique value that identifies the channel in all future function calls affecting this channel. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

timeslot

0 – 127 for stream frequency of 8 MHz
0 – 63 for stream frequency of 4 MHz
0 – 31 for stream frequency of 2 MHz

The timeslot of the TSST of the channel. Note that the allowed values are affected by the format of the stream and frequency of the clock controlling the stream (see TDM Configuration section of the MT90502_CONF structure). This value may also reserve additional timeslots if the channel has silence suppression enabled. See Silence Suppression Configuration Parameters.

Direction: IN Type: ULONG

Default: MT90502_INVALID_TIMESLOT

stream

0 – 31 for max_stream of 32
0 – 15 for max_stream of 16
0 – 7 for max_stream of 8
0 – 3 for max_stream of 4

The stream of the TSST of the channel. Note that the allowed values are also affected by the value of max_stream, which was specified during the call to mt90502_open; and whether this channel is configured with silence suppression See Silence Suppression Configuration Parameters.

Direction: IN Type: ULONG

Default: MT90502_INVALID_STREAM

pcm_law_translation

MT90502_ULAW_TO_ALAW
MT90502_ULAW_TO_ULAW
MT90502_ALAW_TO_ULAW
MT90502_ALAW_TO_ALAW

PCM samples from the H.1x0 bus can have their PCM Law translated before being placed in a TX mini-packet. MT90502_ULAW_TO_ALAW will translate H.1x0 u-Law samples to A-Law before placing them in a TX mini-packet. The behavior of MT90502_ULAW_TO_ULAW is dependent on the setting of u_law_no_zero in the MT90502_CONF structure. MT90502_ALAW_TO_ULAW. will translate received A-Law samples to u-Law.

Direction: IN Type: ULONG

Default: MT90502_ULAW_TO_ULAW

compression_rate

MT90502_COMP_PCM_64KBPS
MT90502_COMP_ADPCM_40KBPS
MT90502_COMP_ADPCM_32KBPS
MT90502_COMP_ADPCM_24KBPS
MT90502_COMP_ADPCM_16KBPS
MT90502_COMP_AUTO_DETECT

The compression rate of the channel. The channel can be compressed with PCM (64 kbps) or ADPCM (40, 32, 24, 16 kbps). If ADPCM is used and the bytes from the H100 bus include ADPCM compression indication then the compression rate in the TX direction can be auto-detected. Also if the streams are properly formatted auto-detection can include PCM.

Direction: IN Type: ULONG

Default: MT90502_COMP_PCM_64KBPS

number_of_edus

MT90502_1_EDU
MT90502_2_EDU
MT90502_3_EDU
MT90502_4_EDU
MT90502_5_EDU
MT90502_44_BYTE
MT90502_40_BYTE
MT90502_8_EDU

The number of EDUs (Elementary Data Units) of the channel transmitted in one mini-packet. For MT90502_44_BYTE and MT90502_40_BYTE the number of EDUs is dependant on the compression_rate and is adjusted by the device to maintain 44 or 40 Byte mini-packets. In 44 and 40 byte EDU modes MT90502_COMP_PCM_64KBPS, MT90502_COMP_ADPCM_32KBPS or MT90502_COMP_AUTO_DETECT are the only valid compression_rate configurations.

Direction: IN Type: ULONG

Default: 5

phase

0 – (number_of_edus – 1)

The relationship in time between packet assembly of all connections of a particular EDU size can be controlled by specifying a phase and sub-phase in which assembly begins. The device arbitrarily selects a frame to be considered phase 0, sub-phase 0 for a given packet EDU size. Phase specifies the EDU relative to phase 0 and sub-phase specifies a sample offset within the EDU. For example connections opened with a phase of 1 and sub-phase of 2 will begin assembly 10 frames after a connection of phase 0, sub-phase 0.

For 44 and 40 Byte EDU modes the phasing is specified relative to the largest EDU value for the connection which is 11 for 44 Byte mode and 10 for 40 Byte mode. When the channel is transporting PCM (5.5 and 5 EDUs) packets will additionally be assembled 44 frames before/after the specified phase. For example in 44 byte mode if the phase is specified as 9 and subphase 2 then, when the channel is carrying PCM, a packet will be assembled in both phase 9 subphase 2 and phase 3 subphase 6

Devices attached to the H.1x0 bus can determine the frame that establishes a given phase using phasing TSSTs (see mt90502_open_phasing_tsst function).

Direction: IN Type: ULONG

Default: 0

sub_phase 0 – 7

See phase parameter.

Direction: IN Type: ULONG

Default: 0

uui_value 4 bit field

This field determines the value of lower 4 UUI bits not used as a counter (see num_uui_counter_bits) for transmitted voice mini-packets. For example if the value of num_uui_counter_bits is 3, only the most significant bit of this field will be used to form the UUI. Note the MSB of the 5 bit UUI field is 0 for voice mini-packets.

Direction: IN Type: ULONG

Default: 0x0

uui_increment 1 – 4

This field is the increment of the sequence number per UUI. Note that this value will affect the mini-packet counter.

Direction: IN Type: ULONG

Default: 1

num_uui_counter_bits 0 – 4

This field determines the number of least-significant bits of the UUI that are used as a counter for sequencing voice packets. For 44 and 40 byte EDU mode the num_uui_counter_bits must be 3 or less.

Direction: IN Type: ULONG

Default: 4

sil_suppr_profile 0 – 255, MT90502_SIL_SUPPR_DISABLED

Determines the silence suppression profile to be used for this channel. The profiles are initialized by mt90502_open. See Silence Suppression Configuration Parameters.

Direction: IN Type: ULONG

Default: MT90502_SIL_SUPPR_DISABLED

auto_dc_offset_corr TRUE / FALSE

If TRUE the API will calibrate the DC offset correction for the TX and RX (if used) channel. The DC offset correction is an 8 bit twos-complement value used to remove DC offset from the Linear values before being used to calculate the energy of the Local and Remote signals for silence suppression. To calibrate the API will disable TX silence suppression for approximately 5 seconds to allow the hardware to sample the signals. If sil_suppr_profile is set to MT90502_SIL_SUPPR_DISABLED this parameter is ignored.

Direction: IN Type: ULONG

Default: FALSE

tx_dc_offset_corr 0 – 255
MT90502_PRESERVE_DC_OFFSET

If auto_dc_offset_corr is set to TRUE or sil_suppr_profile is set to MT90502_SIL_SUPPR_DISABLED this parameter is ignored. Otherwise, this parameter determines the value of DC offset correction for the TX channel.

Direction: IN Type: ULONG

Default: MT90502_PRESERVE_DC_OFFSET

rx_dc_offset_corr 0 – 255
MT90502_PRESERVE_DC_OFFSET

If auto_dc_offset_corr is set to TRUE or sil_suppr_profile is set to MT90502_SIL_SUPPR_DISABLED this parameter is ignored. Otherwise, this parameter determines the value of DC offset correction for the RX channel.

Direction: IN Type: ULONG

Default: MT90502_PRESERVE_DC_OFFSET

2.3.3 mt90502_open_rx_hdlc_stream

This function opens an HDLC stream for receiving mini packets. The RX stream starts with the timeslot timeslot on the stream and is num_tssts long.

Also specified is the format of the HDLC stream. See HDLC Format, Including Zero-insertion And Extraction.

A TSST can only be used once, whether it is in an XXPCM channel, an HDLC stream, or a low-latency-loopback channel.

This function returns a handle by which the API identifies this stream.

The mt90502_open_rx_hdlc_stream_def function inserts default values into the HDLC stream configuration structure, MT90502_RX_HDLC_STREAM. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_rx_hdlc_stream_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_RX_HDLC_STREAM* prx_hdlc_stream );

ULONG mt90502_open_rx_hdlc_stream( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_RX_HDLC_STREAM* prx_hdlc_stream );
```

Return Values

`MT90502ER_GENERIC_OK`	indicates success.

Also, see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

prx_hdlc_stream a pointer to an MT90502_RX_HDLC_STREAM structure. The definitions of the structure’s elements are listed below. The user allocates this structure.

2.3.3.1 Structure MT90502_RX_HDLC_STREAM

pstream_hndl handle

a pointer to a single ULONG which returns the handle for the created HDLC stream. This handle is a unique value that identifies the stream in all future function calls affecting this stream. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

max_channels 1 – 256

The maximum number of voice channels that can be opened on this HDLC stream. This value allocates channel resources within the chip out of the total capacity of 1023 channels. The minimum resources required for a stream is determined by the number of TSSTs used in the stream (num_tssts). Setting this field to a lower value will not conserve any resources.

Direction: IN Type: ULONG

Default: 1

timeslot

0 – 127 for stream frequency of 8 MHz
0 – 63 for stream frequency of 4 MHz
0 – 31 for stream frequency of 2 MHz

The beginning timeslot of the series of consecutive TSSTs of the HDLC stream. Note that this value is also affected by the frequency of the clock controlling the stream.

Direction: IN Type: ULONG

Default: MT90502_INVALID_TIMESLOT

stream

0 – 31 for max_stream of 32
0 – 15 for max_stream of 16
0 – 7 for max_stream of 8
0 – 3 for max_stream of 4

The TDM stream containing the consecutive TSSTs of the HDLC stream. Note that this value is also affected by the value of max_stream, which was specified during the call to mt90502_open.

Direction: IN Type: ULONG

Default: MT90502_INVALID_STREAM

num_tssts 1 – 128 for stream frequency of 8 MHz
1 – 64 for stream frequency of 4 MHz
1 – 32 for stream frequency of 2 MHz

Indicates the number of consecutive TSSTs used in the HDLC stream. The RX stream will start from the TSST described by timeslot and stream, and will use the next num_tssts – 1 TSSTs of the same stream. Note that the stream must not cross a frame boundary.

For example, if:

rx_timeslot = 2
rx_stream = 3
rx_num_tssts = 4

then the RX bit/byte stream will consist of the timeslots 2, 3, 4, and 5 on stream 3.

Direction: IN Type: ULONG

Default: 1

hdlc_header_type MT90502_HDLC_NO_HEADER
MT90502_HDLC_ADD1
MT90502_HDLC_ADD2
MT90502_HDLC_ADD1_CTRL
MT90502_HDLC_ADD2_CTRL

Determines the bytes of the HDLC header. There can be 0, 1, or 2 address bytes, as well as a possible control byte. See HDLC Format, Including Zero-insertion And Extraction

Direction: IN Type: ULONG

Default: MT90502_HDLC_NO_HEADER

hdlc_crc_present TRUE / FALSE

Determines whether or not there is a CRC word present at the end of each HDLC packet. See HDLC Format, Including Zero-insertion And Extraction

Direction: IN Type: ULONG

Default: FALSE

2.3.4 mt90502_open_tx_hdlc_stream

This function opens an HDLC stream for transmitting mini packets. The stream starts with the timeslot timeslot on the stream stream and is num_tssts long.

Also specified is the format of the HDLC stream. See HDLC Format, Including Zero-insertion And Extraction.

A TSST can only be used once, whether it is in an XXPCM channel, an HDLC stream, or a low-latency-loopback channel.

This function returns a handle by which the API identifies this stream.

The mt90502_open_tx_hdlc_stream_def function inserts default values into the HDLC stream configuration structure, MT90502_TX_HDLC_STREAM. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"
```

```
ULONG mt90502_open_tx_hdlc_stream_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_HDLC_STREAM* ptx_hdlc_stream );

ULONG mt90502_open_hdlc_stream( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_HDLC_STREAM* ptx_hdlc_stream );
```

Return Values

`MT90502ER_GENERIC_OK` indicates success.

Also, see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

ptx_hdlc_stream a pointer to an MT90502_TX_HDLC_STREAM structure. The definitions of the structure’s elements are listed below. The user allocates this structure.

2.3.4.1 Structure MT90502_TX_HDLC_STREAM

pstream_hndl handle

a pointer to a single ULONG which returns the handle for the created HDLC stream. This handle is a unique value that identifies the stream in all future function calls affecting this stream. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

max_channels 1 – 256

The maximum number of voice channels that can be opened on this HDLC stream. This value allocates channel resources within the chip out of the total capacity of 1023 channels. The minimum resources required for a stream is determined by the number of TSSTs used in the stream (num_tssts). Setting this field to a lower value will not conserve any resources.

Direction: IN Type: ULONG

Default: 1

timeslot
0 – 127 for stream frequency of 8 MHz
0 – 63 for stream frequency of 4 MHz
0 – 31 for stream frequency of 2 MHz

The beginning timeslot of the series of consecutive TSSTs of the HDLC stream. Note that this value is also affected by the frequency of the clock controlling the stream.

Direction: IN Type: ULONG

Default: MT90502_INVALID_TIMESLOT

stream
0 – 31 for max_stream of 32
0 – 15 for max_stream of 16
0 – 7 for max_stream of 8
0 – 3 for max_stream of 4

The TDM stream containing the consecutive TSSTs of the HDLC stream. Note that this value is also affected by the value of max_stream, which was specified during the call to mt90502_open.

Direction: IN Type: ULONG

Default: MT90502_INVALID_STREAM

num_tssts

1 – 128 for stream frequency of 8 MHz
1 – 64 for stream frequency of 4 MHz
1 – 32 for stream frequency of 2 MHz

Indicates the number of consecutive TSSTs used in the HDLC stream. The TX stream will start from the TSST described by timeslot and stream, and will use the next num_tssts – 1 TSSTs of the same stream. Note that the stream must not cross a frame boundary.

For example, if:

tx_timeslot = 2
tx_stream = 3
rx_tx_num_tssts = 4

then the TX bit/byte stream will consist of the timeslots 2, 3, 4, and 5 on stream 3.

Direction: IN Type: ULONG

Default: 1

hdlc_header_type

MT90502_HDLC_NO_HEADER
MT90502_HDLC_ADD1
MT90502_HDLC_ADD2
MT90502_HDLC_ADD1_CTRL
MT90502_HDLC_ADD2_CTRL

Determines the bytes of the HDLC header. There can be 0, 1, or 2 address bytes, as well as a possible control byte. See HDLC Format, Including Zero-insertion And Extraction.

Direction: IN Type: ULONG

Default: MT90502_HDLC_NO_HEADER

hdlc_ignore_address

TRUE / FALSE

If TRUE the address field in the HDLC stream will be ignored. This limits the stream to carrying only 1 channel. See HDLC Format, Including Zero-insertion And Extraction.

Direction: IN Type: ULONG

Default: TRUE

hdlc_crc_present

TRUE / FALSE

Determines whether or not there is a CRC word present at the end of each HDLC packet. See HDLC Format, Including Zero-insertion And Extraction

Direction: IN Type: ULONG

Default: FALSE

2.3.5 mt90502_close_hdlc_stream

This function releases the TSSTs reserved to form the HDLC stream indicated by stream_hndl. If this stream has associated HDLC channels then those channels will be closed also.

The mt90502_close_hdlc_stream_def function inserts default values into the MT90502_CLOSE_STREAM structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_close_hdlc_stream_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_STREAM* pclose_stream );

ULONG mt90502_close_hdlc_stream( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_STREAM* pclose_stream );
```

Return Values

MT90502ER_GENERIC_OK indicates success.

Also, see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

pclose_stream pointer to an MT90502_CLOSE_STREAM structure. The definitions of the structure's elements are listed below.

2.3.5.1 Structure MT90502_CLOSE_STREAM

pstream_hndl handle

a pointer to the handle indicating the HDLC stream to be closed. The value of the handle will be modified to a unique value for closed handles as a code check.

Direction: IN/IO Type: POINTER

Default: NULL

2.3.6 mt90502_open_rx_hdlc_channel

This function opens a receive HDLC channel identified by hdlc_address on the receive HDLC stream indicated by stream_hndl. The new HDLC channel can then be associated to a mini packet stream by the function mt90502_map_cid. The handle identifying the new HDLC channel is returned in pch_hndl.

This function returns a handle by which the API identifies this channel.

The mt90502_open_rx_hdlc_channel_def function inserts default values into the HDLC channel configuration structure, MT90502_RX_HDLC_CHAN. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_rx_hdlc_channel_def(
            MT90502_INSTANCE_API*       pmt90502_api,       MT90502_RX_HDLC_CHAN*
            prx_hdlc_chan );

ULONG mt90502_open_rx_hdlc_channel (
            MT90502_INSTANCE_API*       pmt90502_api,       MT90502_RX_HDLC_CHAN*
            prx_hdlc_chan );
```

Return Values

MT90502ER_GENERIC_OK indicates success.

Also, see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_ap a pointer to an instance structure of the chip.

prx_hdlc_chan a pointer to an MT90502_RX_HDLC_CHAN structure. The definitions of the structure’s elements are listed below. The user allocates this structure.

2.3.6.1 Structure MT90502_RX_HDLC_CHAN

pch_hndl handle

a pointer to a single ULONG which returns the handle for the created HDLC channel. This handle is a unique value that identifies the channel in all future function calls affecting this channel. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

stream_hndl handle

The handle returned from a call to mt90502_open_rx_hdlc_stream. This handle specifies the HDLC stream on which the channel is to be created.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

hdlc_address 0 – 127

The address of the HDLC header. This field is ignored if the hdlc_header_type of the specified HDLC stream is set to MT90502_HDLC_NO_HEADER.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HDLC_ADDRESS

hdlc_control_byte MT90502_CTRL_BYTE_MT90502_GENERATED
MT90502_CTRL_BYTE_UUI_IN_HIGH_BITS

Determines the value of the HDLC control byte in the RX direction. The control byte can either contain the UUI value of the packet in the 5 most significant bits, or it can contain the value of hdlc_control_byte_val. In the case where the UUI value is to be inserted, the remaining 3 least significant bits of the control byte will be filled with the 3 least significant bits of hdlc_control_byte_val. This value will be ignored if hdlc_header_type of the specified HDLC stream indicates that there is no control byte present.

Direction: IN Type: ULONG

Default: MT90502_CTRL_BYTE_MT90502_GENERATED

hdlc_control_byte_val 8 bit field

The value given to the control byte in the RX direction. This value is ignored if hdlc_header_type of the indicated HDLC stream indicates that there is no control byte present. See **rx_hdlc_control_byte**.

Direction: IN Type: ULONG

Default: 0x00

num_uui_counter_bits 0 – 4

This field determines the number of least-significant bits of the UUI that are used as a counter for sequencing voice packets.

Direction: IN Type: ULONG

Default: 4

adap_src_a TRUE / FALSE

Whether the mini packets of the channel are to be used in the RX direction to generate clock recovery points for buffer A.

Direction: IN Type: ULONG

Default: FALSE

adap_src_b see **adap_src_a**

Default: FALSE

2.3.7 mt90502_open_tx_hdlc_channel

This function opens a transmit HDLC channel identified by hdlc_address on the transmit HDLC stream indicated by stream_hndl. The new HDLC channel can then be associated to a mini packet stream by the function mt90502_map_cid. The handle identifying the new HDLC channel is returned in pch_hndl.

This function returns a handle by which the API identifies this channel.

The mt90502_open_tx_hdlc_channel_def function inserts default values into the HDLC channel configuration structure, MT90502_TX_HDLC_CHAN. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_tx_hdlc_channel_def(
            MT90502_INSTANCE_API*       pmt90502_api,       MT90502_TX_HDLC_CHAN*
            ptx_hdlc_chan );

ULONG mt90502_open_tx_hdlc_channel (
            MT90502_INSTANCE_API*       pmt90502_api,       MT90502_TX_HDLC_CHAN*
            ptx_hdlc_chan );
```

Return Values

`MT90502ER_GENERIC_OK` indicates success.

Also, see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api a pointer to an instance structure of the chip.

ptx_hdlc_chan a pointer to an MT90502_TX_HDLC_CHAN structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.3.7.1 Structure MT90502_TX_HDLC_CHAN

pch_hndl handle

a pointer to a single ULONG which returns the handle for the created HDLC channel. This handle is a unique value that identifies the channel in all future function calls affecting this channel. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

stream_hndl handle

The handle returned from a call to mt90502_open_tx_hdlc_stream. This handle specifies the HDLC stream on which the channel is to be created.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

hdlc_address 0 – 127

The address of the HDLC header. This field is ignored if the hdlc_header_type is set to MT90502_HDLC_NO_HEADER, or hdlc_ignore_address is set to TRUE for the specified stream.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HDLC_ADDRESS

uui_source MT90502_UUI_MT90502_GENERATED
MT90502_UUI_IN_HDLC_CTRL_BYTE
MT90502_UUI_IN_HDLC_AAL2_HEADER

Determines the source of the UUI value used in the TX direction. The UUI value can be generated by the MT90502 (uui_value), taken from the HDLC control byte (if a control byte is present), or taken from the AAL2 header provided in the HDLC stream (if the chip is configured for AAL2 headers in the HDLC streams, see HDLC Configuration Parameters section of the MT90502_CONF structure).

Direction: IN Type: ULONG

Default: MT90502_UUI_MT90502_GENERATED

uui_value 4 bit field

This field determines the value of lower 4 UUI bits not used as a counter (see num_uui_counter_bits) for transmitted voice mini-packets, and if MT90502_UUI_MT90502_GENERATED is selected for uui_source. For example if the value of num_uui_counter_bits is 3, only the most significant bit of this field will be used to form the UUI. Note the MSB of the 5 bit UUI field is 0 for voice mini-packets.

Direction: IN Type: ULONG

Default: 0x0

uui_increment 1 – 4

This field is the increment of the sequence number per UUI. Note that this value will affect the mini-packet counter.

Direction: IN Type: ULONG

Default: 1

num_uui_counter_bits 0 – 4

This field determines the number of least-significant bits of the UUI that are used as a counter for sequencing voice packets, and if MT90502_UUI_MT90502_GENERATED is selected for uui_source.

Direction: IN Type: ULONG

Default: 4

2.3.8 mt90502_open_channel_in_loopback,

This function opens a channel in low-latency-loopback from TDM bus to TDM bus. Low-latency-loopback implies that the data coming in on the TX TSST of the channel will be placed on the RX TSST of the channel with 2 frames of delay.

A handle to this channel is returned. This handle is necessary to reference the newly created channel in the future.

The directions TX and RX of the TSSTs are with respect to the UTOPIA ports. Thus, a TX TSST enters the chip, via the TDM bus.

A TSST can only be used once, whether it is in an XXPCM channel, an HDLC stream, or a low-latency-loopback channel.

There is a maximum of 128 low-latency-loopback channels that can be open per chip.

This function returns a handle by which the API identifies this channel.

The mt90502_open_channel_in_loopback_def function inserts default values into the Low Latency Loopback channel configuration structure, MT90502_LLL_CH. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_open_channel_in_loopback_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_LLL_CHAN* plll_ch );

ULONG mt90502_open_channel_in_loopback(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_LLL_CHAN* plll_ch );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

plll_ch pointer to an MT90502_LLL_CHAN structure. The definitions of the structure's elements are listed below.

2.3.8.1 Structure MT90502_LLL_CHAN

pch_hndl handle

pointer to a single ULONG which returns the handle for the created low-latency-loopback channel. This handle is a unique value that identifies the channel in all future function calls affecting this channel. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

tx_timeslot

0 – 127 for stream frequency of 8 MHz
0 – 63 for stream frequency of 4 MHz
0 – 31 for stream frequency of 2 MHz

The timeslot of the TX TSST. Note that the directions TX and RX are with respect to the UTOPIA ports. Thus, a TX TSST enters the chip, and RX TSST exits the chip.

Direction: IN Type: ULONG

Default: MT90502_INVALID_TIMESLOT

tx_stream

0 – 31 for max_stream of 32
0 – 15 for max_stream of 16
0 – 7 for max_stream of 8
0 – 3 for max_stream of 4

The stream of the TX TSST. Note that this value is also affected by the value of max_stream, which was specified during the call to mt90502_open.

Direction: IN Type: ULONG

Default: MT90502_INVALID_STREAM

rx_timeslot see **tx_timeslot**

Default: MT90502_INVALID_TIMESLOT

rx_stream see **tx_stream**

Default: MT90502_INVALID_STREAM

2.3.9 mt90502_close_channel

This function closes the channel indicated by pch_hndl, regardless of the type (TX or RX XXPCM or HDLC, or low-latency-loopback) of the channel.

This function releases all resources that were reserved by the call to the function that opened the channel.

The mt90502_close_channel_def function inserts default values into the MT90502_CLOSE_CHAN structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_close_channel_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_CHAN* pclose_chan );

ULONG mt90502_close_channel( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_CHAN* pclose_chan );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pclose_chan pointer to an MT90502_CLOSE_CHAN structure. The definitions of the structure’s elements are listed below.

2.3.9.1 Structure MT90502_CLOSE_CHAN

pch_hndl handle

pointer to the handle which was created by the call to the function which opened the channel. This handle is modified to a unique value for closed handles as a code check.

Direction: IN/IO Type: POINTER

Default: NULL

2.3.10 mt90502_tx_change_compression

This function changes the compression being used for an XXPCM channel. While the chip automatically adjusts for changes in compression in the RX direction by using the LI field and can adjust to TX changes within ADPCM types if configured for MT90502_COMP_AUTO_DETECT, the chip needs to be informed by software about switches between PCM and ADPCM. This function can also be used to manage all compression changes.

The mt90502_tx_change_compression_def function inserts default values into the MT90502_CH_COMP structure. The default value of a structure field is indicated following the field’s description.

Usage

```
#include “mt90502_api.h”

ULONG mt90502_tx_change_compression_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CH_COMP* pch_comp );

ULONG mt90502_tx_change_compression(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CH_COMP* pch_comp );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pch_comp pointer to an MT90502_CH_COMP structure. The definitions of the structure’s elements are listed below.

2.3.10.1 Structure MT90502_CH_COMP

ch_hndl 32 bits value

Handle which was created by the call to the function which opened the channel. This handle must represent the address of an XXPCM channel.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

compression_rate

MT90502_COMP_PCM_64KBPS
MT90502_COMP_ADPCM_40KBPS
MT90502_COMP_ADPCM_32KBPS
MT90502_COMP_ADPCM_24KBPS
MT90502_COMP_ADPCM_16KBPS
MT90502_COMP_AUTO_DETECT

The new compression rate of the channel. The channel can be compressed with PCM (64 kbps) or ADPCM (40, 32, 24, 16 kbps). If ADPCM is used and the bytes from the H100 bus include ADPCM compression indication then the compression rate in the TX direction can be auto-detected.

Direction: IN Type: ULONG

Default: MT90502_COMP_PCM_64KBPS

2.3.11 mt90502_open_phasing_tsst

This function drives the specified TSST with a counting pattern in order for external devices to be aware of what frame sample begins the formation of a CID packet in the chip's SAR process. Any given frame can begin a mini-packet and once established packets will always be formed every (8 x EDU #) of frames. In order to control the trade-off between delay and bandwidth efficiency the MT90502 supports phasing of mini-packet construction. For any given EDU value the MT90502 selects a frame as phase 0 and sub-phase 0. At the time a TSST is assigned to a CID stream (channel open) it is placed in a specific relationship to all other CID streams by specifying the phase and sub-phase it should start in. The phase determines which EDU relative to 0 and the sub-phase determines which sample of the 8 samples of the selected EDU the packet formation starts on.

Each value driven represents a phase/sub-phase of a given EDU size. For example if the # of EDUs were 5 then the counting pattern would cycle from 0 to 39 (i.e. 5*8 values) where 0 would be transmitted in the frame that was considered phase 0 sub-phase 0 for all CID streams configured for 5 EDUs. The value 1 would represent phase 0 sub-phase 1 and the value 9 would be phase 1, sub-phase 1. The general translation is 8*phase+sub-phase=count value.

Note that for each phasing type opened, one connection of the 1023 connection capacity is lost. It does not matter how many TSSTs the phasing type (counting pattern) is assigned to only one connection is lost.

The H.100 control signals must be present and valid for proper operation of the H.100 bus. The establishment of the phasing TSST may take up to 100 milliseconds after the H.100 signals are valid. The mt90502_query_phasing_tsst may be used to determine if the phasing TSST is valid.

The mt90502_open_phasing_tsst_def function inserts default values into the MT90502_OPEN_PHASING_TSST structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"
```

```
ULONG mt90502_open_phasing_tsst_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_OPEN_PHASING_TSST* pphase );

ULONG mt90502_open_phasing_tsst (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_OPEN_PHASING_TSST* pphase );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pphase pointer to an MT90502_OPEN_PHASING_TSST structure. The definitions of the structure’s elements are listed below.

2.3.11.1 Structure MT90502_OPEN_PHASING_TSST

pphase_hndl handle

a pointer to a single ULONG which returns the handle for the created phasing connection. This handle is a unique value that identifies the VC in all future function calls affecting this VC. The user allocates the ULONG for the handle.

Direction: IN/OUT Type: POINTER

Default: NULL

timeslot

0 – 127 for stream frequency of 8 MHz
0 – 63 for stream frequency of 4 MHz
0 – 31 for stream frequency of 2 MHz

The timeslot of the TSST to be driven. Note that this value is also affected by the frequency of the clock controlling the stream.

Direction: IN Type: ULONG

Default: MT90502_INVALID_TIMESLOT

stream

0 – 31 for max_stream of 32
0 – 15 for max_stream of 16
0 – 7 for max_stream of 8
0 – 3 for max_stream of 4

The TDM stream containing the TSST to be driven. Note that this value is also affected by the value of max_stream, which was specified during the call to mt90502_open.

Direction: IN Type: ULONG

Default: MT90502_INVALID_STREAM

phasing_type

MT90502_PHASING_8_COUNT
MT90502_PHASING_16_COUNT
MT90502_PHASING_24_COUNT
MT90502_PHASING_32_COUNT

MT90502_PHASING_40_COUNT
MT90502_PHASING_64_COUNT

The count specified is determined by the number of EDUs used for packets of a connection. For example, 5 EDU packets would require a count of 40 to determine packet assembly phasing.

Direction: IN Type: ULONG

Default: MT90502_INVALID_PHASING_TYPE

2.3.12 mt90502_query_phasing_tsst

This function is used to determine the validity of the phasing TSST indicated by the handle pphase_hndl.

The mt90502_query_phasing_tsst_def function inserts default values into the MT90502_QUERY_PHASING_TSST structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_query_phasing_tsst_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_QUERY_PHASING_TSST* pphase );

ULONG mt90502_query_phasing_tsst (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_QUERY_PHASING_TSST* pphase );
```

Return Values

`MT90502ER_GENERIC_OK` indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pphase pointer to an MT90502_QUERY_PHASING_TSST structure. The definitions of the structure's elements are listed below.

2.3.12.1 Structure MT90502_QUERY_PHASING_TSST

pphase_hndl handle

a pointer to a single ULONG which was returned by the call to mt90502_open_phasing_tsst.

Direction: IN Type: POINTER

Default: NULL

phasing_tsst_valid TRUE / FALSE

If TRUE the phasing TSST indicated by pphase_hndl is valid on the H.100 bus, the H.100 control signals are present and the API has established the correct pattern. If FALSE the phasing TSST is not established. This could be due to the H.100 control signals not being present or the API has not completed the establishment of the proper pattern.

Direction: OUT Type: ULONG

Default: FALSE

2.3.13 mt90502_close_phasing_tsst

This function closes the phasing TSST indicated by the handle pphase_hndl. The TSST will no longer be driven with the counting pattern and may be used for other purposes.

The mt90502_close_phasing_tsst_def function inserts default values into the MT90502_CLOSE_PHASING_TSST structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_close_phasing_tsst_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_PHASING_TSST* pphase );

ULONG mt90502_close_phasing_tsst (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CLOSE_PHASING_TSST* pphase );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pphase pointer to an MT90502_CLOSE_PHASING_TSST structure. The definitions of the structure's elements are listed below.

2.3.13.1 Structure MT90502_CLOSE_PHASING_TSST

pphase_hndl handle

a pointer to a single ULONG which was returned by the call to mt90502_open_phasing_tsst. This handle is modified to a unique value for closed handles as a code check.

Direction: IN/OUT Type: POINTER

Default: NULL

2.3.14 mt90502_tx_change_silence_suppr

This function changes the current state of TX silence suppression being used for an XXPCM channel.

The mt90502_tx_change_silence_suppr_def function inserts default values into the MT90502_CH_SIL_SUP structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_tx_change_silence_suppr_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CH_SIL_SUP* pch_sil_sup );
```

```
ULONG mt90502_tx_change_silence_suppr (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CH_SIL_SUP* pch_sil_sup );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pch_sil_sup pointer to an MT90502_CH_SIL_SUP structure. The definitions of the structure’s elements are listed below.

2.3.14.1 Structure MT90502_CH_SIL_SUP

pch_hndl handle

pointer to the handle which was created by the call to the function which opened the channel. This handle must point to an XXPCM channel.

Direction: IN Type: POINTER

Default: MT90502_INVALID_HANDLE

sil_suppr_profile see MT90502_TX_XXPCM_CHAN Structure

Direction: IN Type: ULONG

Default: MT90502_SIL_SUPPR_DISABLED

auto_dc_offset_corr see MT90502_TX_XXPCM_CHAN Structure

Direction: IN Type: ULONG

Default: FALSE

tx_dc_offset_corr see MT90502_TX_XXPCM_CHAN Structure

Direction: IN Type: ULONG

Default: MT90502_PRESERVE_DC_OFFSET

rx_dc_offset_corr see MT90502_TX_XXPCM_CHAN Structure

Direction: IN Type: ULONG

Default: MT90502_PRESERVE_DC_OFFSET

2.4 Statistics Functions

Unless otherwise noted, detected conditions indicate events that have occurred since the previous read of the same set of statistics. (i.e. values are reset when read by a statistics function.) All counts (identified by the name ending in ‘_cnt’ are only reset when the underlying entity is opened. (e.g. all counters returned by mt90502_get_vc_statistics were reset when the VC for which statistics are returned was opened.) These counters are free running for the existence of the underlying entity and may wrap.

2.4.1 mt90502_get_chip_statistics

This function fills an MT90502_CHIP_STATS structure with the current statistics for the chip. All statistics returned by this function are initialized (e.g. counters set to 0) by the function mt90502_open.

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_chip_statistics_def function inserts default values into the MT90502_CHIP_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_chip_statistics_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CHIP_STATS* pchip_stats );

ULONG mt90502_get_chip_statistics( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CHIP_STATS* pchip_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api pointer to an instance structure of the chip

pchip_stats pointer to an MT90502_CHIP_STATS statistics structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.1.1 Statistics Structure MT90502_CHIP_STATS

reset_statistics TRUE / FALSE

Resets the statistics counters for the retuned by this function after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

chip_open_time 0 – ??

The amount of time in seconds since the chip was opened by the mt90502_open_chip function.

Direction: OUT Type: ULONG

Default: 0

num_aal2_vcs_open 0 – ??

The number of AAL2 VCs currently open.

Direction: OUT Type: ULONG

Default: 0

num_data_vcs_open 0 – ??

The number of data VCs currently open.

Direction: OUT Type: ULONG

Default: 0

num_cids_open 0 – ??

The number of mini packet streams currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_hdlc_streams_open 0 – ??

The number of receive HDLC streams currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_hdlc_streams_open 0 – ??

The number of transmit HDLC streams currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_hdlc_channels_open 0 – ??

The number of receive HDLC channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_hdlc_channels_open 0 – ??

The number of transmit HDLC channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_pcm_channels_open 0 – ??

The number of fixed receive 64kbps PCM channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_adpcm16_channels_open 0 – ??

The number of fixed receive ADPCM 16 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_adpcm24_channels_open 0 – ??

The number of fixed receive ADPCM 24 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_adpcm32_channels_open 0 – ??

The number of fixed receive ADPCM 32 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_adpcm40_channels_open 0 – ??

The number of fixed receive ADPCM 40 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_rx_auto_detect_channels_open 0 – ??

The number of auto detect receive ADPCM channels currently open. The device determines the actual PCM type of the channel dynamically.

Direction: OUT Type: ULONG

Default: 0

num_tx_pcm_channels_open 0 – ??

The number of fixed transmit 64kbps PCM channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_adpcm16_channels_open 0 – ??

The number of fixed transmit ADPCM 16 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_adpcm24_channels_open 0 – ??

The number of fixed transmit ADPCM 24 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_adpcm32_channels_open 0 – ??

The number of fixed transmit ADPCM 32 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_adpcm40_channels_open 0 – ??

The number of fixed transmit ADPCM 40 kbps channels currently open.

Direction: OUT Type: ULONG

Default: 0

num_tx_auto_detect_channels_open 0 – ??

The number of auto detect transmit ADPCM channels currently open. The device determines the actual PCM type of the channel dynamically.

Direction: OUT Type: ULONG

Default: 0

num_lll_channels_open 0 – ??

The number of low-latency-loopback channels currently open.

Direction: OUT Type: ULONG

Default: 0

ssrama_parity_error0_cnt 0 – ??

A count of the number of parity errors detected in bits 7:0 of bank A of the external SSRAM. The count is an approximation based on the number of transitions from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

ssrama_parity_error1_cnt 0 – ??

A count of the number of parity errors detected in bits 15:8 of bank A of the external SSRAM. The count is an approximation based on the number of transitions from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

ssramb_parity_error0_cnt see **ssrama_parity_error0_cnt**

Default: 0

ssramb_parity_error1_cnt see **ssrama_parity_error1_cnt**

Default: 0

sdrama_parity_error0_cnt 0 – ??

A count of the number of parity errors detected in bits 7:0 of bank A of the external SDRAM. The count is an approximation based on the number of transitions from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

sdrama_parity_error1_cnt 0 – ??

A count of the number of parity errors detected in bits 15:8 of bank A of the external SDRAM. The count is an approximation based on the number of transitions from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

sdramb_parity_error0_cnt see **sdrama_parity_error0_cnt**

Default: 0

sdramb_parity_error1_cnt see **sdrama_parity_error1_cnt**

Default: 0

phy_alarm_a_cnt 0 – ??

A count of the number of PHY alarms generated by PHY A. The count is an approximation based on the number of transitions from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

phy_alarm_b_cnt see **phy_alarm_a_cnt**

Default: 0

rxa_parity_error_cnt 0 – ??

A count of the number of parity errors detected in cells entering the chip via port A. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

rxb_parity_error_cnt 0 – ??

A count of the number of parity errors detected in cells entering the chip via port B. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

rxc_parity_error_cnt 0 – ??

A count of the number of parity errors detected in cells entering the chip via port C. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

h100_out_of_sync_cnt 0 – ??

A count of the number of times the H.100 slave of the chip lost its framing on the H.100 bus. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

h100_clk_a_bad_cnt 0 – ??

A count of the number of times the H.100 clock CT_C8_A was deemed bad. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

h100_clk_b_bad_cnt 0 – ??

A count of the number of times the H.100 clock CT_C8_B was deemed bad. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

h100_frame_a_bad_cnt 0 – ??

A count of the number of times the H.100 frame CT_FRAME_A was deemed bad. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

h100_frame_b_bad_cnt 0 – ??

A count of the number of times the H.100 frame CT_FRAME_B was deemed bad. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

hdlc_misaligned_flag_cnt 0 – ??

A count of the number of misaligned flags (i.e. not byte aligned) received in TX HDLC packets. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

hdlc_bad_idle_code_cnt 0 – ??

A count of the number of idle codes received in the middle of TX HDLC packets. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

hdlc_long_packet_cnt 0 – ??

A count of the number of TX HDLC packets received with a packet length greater than 64 bytes. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

hdlc_short_packet_cnt 0 – ??

A count of the number of TX HDLC packets received with 0 data bytes. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

rxsar_cell_loss_cnt 0 – ??

A count of the number of cells lost due to an overflow of the RX SAR cell FIFO. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register. Direction: OUT Type: ULONG

Default: 0

txa_cell_loss_cnt 0 – ??

A count of the number of cells lost due to an overflow of the TX A cell FIFO. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

txb_cell_loss_cnt 0 – ??

A count of the number of cells lost due to an overflow of the TX B cell FIFO. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

txc_cell_loss_cnt 0 – ??

A count of the number of cells lost due to an overflow of the TX C cell FIFO. The count is an approximation based on the changes from inactive to active of the corresponding interrupt register.

Direction: OUT Type: ULONG

Default: 0

total_cell_loss_cnt[2] 64 bit unsigned counter

A count of the total number of cells lost in the RX SAR, TX A, TX B, and TX C cell buffers. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

txsar_cell_cnt[2] 64 bit unsigned counter

A count of the total number of cells assembled by the TX SAR. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

rxsar_cell_cnt[2] 64 bit unsigned counter

A count of the total number of cells disassembled by the RX SAR. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

txa_cell_cnt[2] 64 bit unsigned counter

A count of the total number of cells transmitted on UTOPIA port A. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

rxa_cell_cnt[2] 64 bit unsigned counter

A count of the total number of cells received on UTOPIA port A. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

txb_cell_cnt[2] see **txa_cell_cnt[2]**

Default: 0

rxb_cell_cnt[2] see **rxa_cell_cnt[2]**

Default: 0

txc_cell_cnt[2] see **txa_cell_cnt[2]**

Default: 0

rxc_cell_cnt[2] see **rxa_cell_cnt[2]**

Default: 0

txdata_cell_cnt[2] 64 bit unsigned counter

A count of the total number of cells inserted into the TX data cell FIFO by the CPU. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

rxdata_cell_cnt[2] 64 bit unsigned counter

A count of the total number of received cells routed to the RX data cell FIFO. Element 0 of the array contains the 32 least significant bits of the counter, and element 1 contains the 32 most significant bits.

Direction: OUT Type: ULONG[2]

Default: 0

cell_assembly_event_buf_overflow TRUE / FALSE

If TRUE an overflow has occurred in the cell event assembly queue since the last time this function was called. This event occurs when the TX SAR is out of bandwidth to process received HDLC packets. The bandwidth of the TX SAR is dependent on mclk.

Direction: OUT Type: ULONG

Default: FALSE

rx_mini_pkt_buf_overflow TRUE / FALSE

If TRUE an overflow has occurred in the CPU destined mini-packet buffer since the last time this function was called. See the rx_mini_pkt_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

rx_data_buf_overflow TRUE / FALSE

If TRUE the RX data buffer has overflowed. See the rx_data_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

rx_event_buf_overflow TRUE / FALSE

If TRUE the RX event buffer has overflowed. See the rx_event_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

adap_a_buf_overflow TRUE / FALSE

If TRUE the adaptive clock recovery A buffer has overflowed. See the adap_a_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

adap_b_buf_overflow TRUE / FALSE

If TRUE the adaptive clock recovery B buffer has overflowed. See the adap_b_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_rx_mini_pkt_buf_overflow TRUE / FALSE

If TRUE the soft CPU mini packet buffer has overflowed. See the soft_rx_mini_pkt_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_cid_event_buf_overflow TRUE / FALSE

If TRUE the soft CID event buffer has overflowed. See the soft_cid_event_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_rx_data_buf_overflow TRUE / FALSE

If TRUE the soft RX data buffer has overflowed. See the soft_rx_data_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_li_uui_change_buf_overflow TRUE / FALSE

If TRUE the soft LI change buffer has overflowed. See the soft_li_uui_change_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_adap_a_buf_overflow TRUE / FALSE

If TRUE the adaptive clock recovery A buffer has overflowed. See the soft_adap_a_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_adap_b_buf_overflow TRUE / FALSE

If TRUE the adaptive clock recovery B buffer has overflowed. See the soft_adap_b_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

soft_console_buf_overflow TRUE / FALSE

If TRUE the soft console buffer has overflowed. See the soft_console_buffer_size configuration parameter.

Direction: OUT Type: ULONG

Default: FALSE

ct_netref1_value 0 / 1

The current value on pin ct_netref1.

Direction: OUT Type: ULONG

Default: 0

ct_netref2_value see **ct_netref1_value**

Default: 0

2.4.2 mt90502_convert_chip_statistics_to_text

This function converts an MT90502_CHIP_STATS statistics structure to a text string. The MT90502_CHIP_STATS statistics structure is returned by the mt90502_get_chip_statistics function.

The mt90502_convert_chip_statistics_to_text_def function inserts default values into the MT90502_CONVERT_CHIP_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_convert_chip_statistics_to_text_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONVERT_CHIP_STATS* pconvert_chip_stats );

ULONG mt90502_convert_chip_statistics_to_text(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONVERT_CHIP_STATS* pconvert_chip_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always

Parameters

pmt90502_api

pointer to an instance structure of the chip

pconvert_chip_stats

pointer to an MT90502_CONVERT_CHIP_STATS structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.2.1 Structure MT90502_CONVERT_CHIP_STATS

pchip_stats

pointer to an MT90502_CHIP_STATS statistics structure to be converted to text. The definitions of the structure's elements are listed in the mt90502_get_chip_statistics function description.

Direction: IN/IN Type: POINTER

Default: NULL

pstring

pointer to the returned text string. The required length of the string is defined by MT90502_CHIP_STATS_STRING_LENGTH (in bytes). The user allocates the string.

Direction: IN/OUT Type: POINTER

Default: NULL

2.4.3 mt90502_get_vc_statistics

This function fills an MT90502_VC_STATS structure with the current statistics for a VC. All statistics returned by this function are initialized (e.g. counters set to 0) by the function mt90502_open_aal2_vc, or mt90502_open_data_vc

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_vc_statistics_def function inserts default values into the MT90502_VC_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_vc_statistics_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_VC_STATS* pvc_stats );

ULONG mt90502_get_vc_statistics(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_VC_STATS* pvc_stats );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success always

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

pvc_stats

pointer to an MT90502_VC_STATS statistics structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.3.1 Structure MT90502_VC_STATS

vc_hndl handle

handle returned from the call to mt90502_open_aal2_vc, or mt90502_open_data_vc.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

reset_statistics TRUE / FALSE

Resets the statistics counters for the identified VC after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

header see **MT90502_AAL2_VC** structure

Direction: OUT Type: same

Default: MT90502_NULL_HEADER

split_streams see **MT90502_AAL2_VC** structure

Direction: IN Type: same

Default: FALSE

rx_header see **MT90502_AAL2_VC** structure

Direction: IN Type: same

Default: MT90502_NULL_HEADER

rx_tx_utopia_port see **MT90502_AAL2_VC** structure

Direction: OUT Type: same

Default: MT90502_INVALID_UTOPIA_PORT

loopback see **MT90502_AAL2_VC** structure

Direction: OUT Type: same

Default: FALSE

rx_normal_cell_routing see **MT90502_AAL2_VC** structure

Direction: OUT Type: same

Default: 0

rx_oam_cell_routing see **MT90502_AAL2_VC** structure

Direction: OUT Type: same

Default: 0

modified_header 32 bit field

The header of the data VC once its corresponding LUT entry has performed header replacement. This parameter will be 0 for an AAL2 VC.

Direction: OUT Type: ULONG

Default: MT90502_NULL_HEADER

num_open_cid 0 – 255

The number of mini packet streams that are open on this VC. This parameter will be 0 for a data VC.

Direction: OUT Type: ULONG

Default: 0

cid_hndls[256] array of handles

An array of 256 receive channel handles. Entry X of the array corresponds to the channel with CID number X. If the value of an entry is MT90502_INVALID_HANDLE then there is no channel associated to the AAL2 VC with that CID number. This field does not apply for data VCs.

Direction: OUT Type: ULONG[256]

Default: MT90502_INVALID_HANDLE

vc_open_time 0 – ??

The amount of time in seconds since this VC was opened by the mt90502_open_aal2_vc or mt90502_open_aal2_vc functions.

Direction: OUT Type: ULONG

Default: 0

tx_cell_cnt[2] 64 bit counter

The number of cells transmitted on this AAL2 VC's UTOPIA port. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant. This field is set to 0 for data VCs.

Direction: OUT Type: ULONG[2]

Default: 0

rx_cell_cnt[2] 64 bit counter

The number of cells received on this AAL2 VC's UTOPIA port. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant. This field is set to 0 for data VCs.

Direction: OUT Type: ULONG[2]

Default: 0

rx_aal2_parity_error_cnt 32 bit counter

The number of received cells that had parity errors detected on the AAL2 Start Field (STF). The cell is discarded. This parameter will be 0 for a data VC.

Direction: OUT Type: ULONG

Default: 0

rx_aal2_offset_error_cnt 32 bit counter

The number of received cells where the offset value in the AAL2 STF did not correspond to the expected offset. The ATM cell is processed using the received STF value. Cells were probably lost. This parameter will be 0 for a data VC.

Direction: OUT Type: ULONG

Default: 0

rx_aal2_hec_error_cnt 32 bit counter

The number of received ATM cells that contained a mini packet with a HEC error detected in a mini packet header. Once this error is encountered in an ATM cell this mini packet and the remainder of the cell are discarded. This parameter will be 0 for a data VC.

Direction: OUT Type: ULONG

Default: 0

rx_aal2_seq_num_error_cnt 32 bit counter

The number of times the sequence number bit of the AAL2 STF of a received ATM cell did not correspond to the expected value. The ATM cell is processed using the received STF. Cells were probably lost. This parameter will be 0 for a data VC.

Direction: OUT Type: ULONG

Default: 0

2.4.4 mt90502_convert_vc_statistics_to_text

This function converts an MT90502_VC_STATS statistics structure to a text string. The MT90502_VC_STATS statistics structure is returned by the mt90502_get_vc_statistics function.

The mt90502_convert_vc_statistics_to_text_def function inserts default values into the MT90502_CONVERT_VC_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_convert_vc_statistics_to_text_def (
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_CONVERT_VC_STATS * pconvert_vc_stats );

ULONG mt90502_convert_vc_statistics_to_text (
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_CONVERT_VC_STATS * pconvert_vc_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always

Parameters

pmt90502_api

pointer to an instance structure of the chip

pconvert_vc_stats

pointer to an MT90502_CONVERT_VC_STATS structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.4.1 Structure MT90502_CONVERT_VC_STATS

pvc_stats

pointer to an MT90502_VC_STATS statistics structure to be converted to text. The definitions of the structure's elements are listed in the mt90502_get_vc_statistics function description.

Direction: IN/IN Type: POINTER

Default: NULL

pstring

pointer to the returned string. The required length of the string is defined by MT90502_VC_STATS_STRING_LENGTH (in bytes). The user allocates the string.

Direction: IN/OUT Type: POINTER

Default: NULL

2.4.5 mt90502_get_cid_statistics

This function fills an MT90502_CID_STATS structure with the current statistics for a mini packet stream. All statistics returned by this function are initialized (e.g. counters set to 0) by the function mt90502_open_cid.

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_cid_statistics_def function inserts default values into the MT90502_CID_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_cid_statistics_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CID_STATS* pcid_stats );

ULONG mt90502_get_cid_statistics(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CID_STATS* pcid_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

pcid_stats

pointer to an MT90502_CID_STATS statistics structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.5.1 Structure MT90502_CID_STATS

cid_hndl handle

handle returned from the call to mt90502_open_cid.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

reset_statistics TRUE / FALSE

Resets the statistics counters for the identified CID after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

vc_hndl handle

handle returned from the call to mt90502_open_aal2_vc for the VC containing this mini packet stream.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

cid 8 bit field

CID field within AAL2 VC indicated by vc_hndl.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

rx_uui_mapping[17] see **MT90502_MAP_CID** structure

Direction: OUT Type: ULONG[17]

Default: MT90502_DELETE_MINI_PKT

tx_chan_mapping see **MT90502_MAP_CID** structure

Direction: OUT Type: ULONG

Default: MT90502_NO_TX_CHANNEL

rx_ch_hndl handle

The handle to the RX channel to which the CID is mapped to in the RX direction if one or more nodes of the rx_uui_mapping array is set to MT90502_RX_CHANNEL. The channel can be either xxPCM or HDLC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

tx_ch_hndl handle

The handle to the TX channel to which the CID is mapped to in the TX direction if the tx_chan_mapping is set to MT90502_TX_CHANNEL. The channel can be either xxPCM or HDLC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

cid_open_time 0 – ??

The amount of time in seconds since this mini packet stream was opened by the mt90502_open_cid function.

Direction: OUT Type: ULONG

rx_cas_cnt 0 – ??

The number of CAS events received on this CID. This count does not include any redundant packets.

Direction: OUT Type: ULONG

Default: 0

tx_cas_cnt 0 – ??

The number of CAS packets sent on this CID. This count does not include any redundant packets.

Direction: OUT Type: ULONG

Default: 0

rx_cas_crc_error_cnt 0 – ??

The number of CAS packets received on this CID that had bad CRCs.

Direction: OUT Type: ULONG

Default: 0

rx_cas_redundant_error_cnt 0 – ??

The number of missing redundant CAS packets that should have been received per Type 3 protocol.

Direction: OUT Type: ULONG

Default: 0

rx_cpu_mini_pkt_cnt 0 – ??

The count of received mini packets that were directed to the CPU buffer.

Direction: OUT Type: ULONG

Default: 0

tx_cpu_mini_pkt_cnt 0 – ??

The count of transmitted mini packets that were sent by the CPU.

Direction: OUT Type: ULONG

Default: 0

2.4.6 mt90502_convert_cid_statistics_to_text

This function converts an MT90502_CID_STATS statistics structure to a text string. The MT90502_CID_STATS statistics structure is returned by the mt90502_get_cid_statistics function.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_convert_cid_statistics_to_text_def (
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_CONVERT_CID_STATS * pconvert_cid_stats );

ULONG mt90502_convert_cid_statistics_to_text (
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_CONVERT_CID_STATS * pconvert_cid_stats );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success always

Parameters

pmt90502_api

pointer to an instance structure of the chip

pconvert_cid_stats

pointer to an MT90502_CONVERT_CID_STATS structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.6.1 Structure MT90502_CONVERT_CID_STATS

pcid_stats

pointer to an MT90502_CID_STATS statistics structure to be converted to text. The definitions of the structure's elements are listed in the mt90502_get_cid_statistics function description.

Direction: IN/IN Type: POINTER

Default: NULL

pstring

pointer to the returned string. The required length of the string is defined by MT90502_CID_STATS_STRING_LENGTH (in bytes). The user allocates the string.

Direction: IN/OUT Type: POINTER

Default: NULL

2.4.7 mt90502_get_rx_xxpcm_chan_statistics

This function fills an MT90502_RX_XXPCM_CHAN_STATS structure with the current statistics of the channel indicated by ch_hndl. All statistics returned by this function are initialized (e.g. counters set to 0) by the call to mt90502_open_rx_xxpcm_channel.

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_rx_xxpcm_chan_statistics_def function inserts default values into the MT90502_RX_XXPCM_CHAN_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "MT90502_api.h"

ULONG MT90502_get_rx_xxpcm_chan_statistics_def(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_RX_XXPCM_CHAN_STATS * prx_ch_stats );

ULONG MT90502_get_rx_xxpcm_chan_statistics(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_RX_XXPCM_CHAN_STATS * prx_ch_stats );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success always.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

prx_ch_stats

pointer to an MT90502_RX_XXPCM_CHAN_STATS statistics structure. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.7.1 Structure MT90502_RX_XXPCM_CHAN_STATS

ch_hndl handle

handle indicating the receive channel for which the statistics are requested. This handle is created from the call to mt90502_open_rx_xxpcm_channel.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

reset_statistics TRUE / FALSE

Resets the statistics counters for the identified channel after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

channel_initialized TRUE / FALSE

Indicates whether the first mini-packet of the channel has been received yet.

Direction: OUT Type: ULONG

Default: FALSE

cid_hndl handle

MT90502_INVALID_HANDLE

The handle of the mini packet stream to which the channel is associated. This handle was returned from the call to mt90502_open_cid. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

cid_number 1 – 255
MT90502_INVALID_CID

The CID used for this channel within the AAL2 VC.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

vc_hndl handle
MT90502_INVALID_HANDLE

The handle of the VC to which the channel is associated. This handle was returned from the call to mt90502_open_aal2_vc. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header see **MT90502_AAL2_VC** structure

If the channel has not been mapped to a mini packet stream the returned header will be all zeros.

Direction: OUT Type: same

Default: 0

rx_tx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_INVALID_UTOPIA_PORT

The UTOPIA port associated with the VC carrying this xxPCM channel. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_UTOPIA_PORT

Direction: OUT Type: same

Default: MT90502_INVALID_UTOPIA_PORT

compression_rate See **MT90502_RX_CHAN** structure

Direction: OUT Type: ULONG

Default: MT90502_COMP_PCM_64KBPS

number_of_edus See **MT90502_RX_CHAN** structure

Direction: OUT Type: ULONG

Default: 5

current_li 1-63
MT90502_INVALID_LI

This is the current LI of mini packets being received on the channel. The value of MT90502_INVALID_LI indicates invalid and is used before the first mini packet is received.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_LI

chan_open_time 0 – ??

The amount of time in seconds since this receive xxPCM channel was opened by the mt90502_open_rx_xxpcm_channel function.

Direction: OUT Type: ULONG

Default: 0

mini_pkt_cnt[2] 64 bit counter

The number of AAL2 mini-packets of this channel received on UTOPIA. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

mini_pkt_loss_cnt[2] 64 bit counter

The number of AAL2 mini-packets not received. These are expected events when silence suppression is active. The number of packets lost is determined through the use of the UUI sequence values, if configured for the channel, and the associated received timestamp of the packet.

The received timestamp can determine to within PDV time the number of cells lost. The UUI field then indicates precisely how many cells within the PDV window were lost as long as the UUI sequence range can span the PDV. For example, if the length of a packet is 5 EDUs (5 ms between packets) and the UUI sequence counter is 3 bits (8 values), then the maximum amount of PDV that will allow an accurate count is 5 ms * 8 = 40 ms. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

underrun_event_cnt 32 bit counter

The number of AAL2 mini-packet underrun events detected on this channel.

Direction: OUT Type: ULONG

Default: 0

overrun_event_cnt 32 bit counter

The number of AAL2 mini-packet overrun events detected on this channel.

Direction: OUT Type: ULONG

Default: 0

underrun_byte_cnt 32 bit counter

The number of padding bytes that were sent on the H100 bus due to underruns on the channel. If the rx_underrun_padding field of the channel is set to MT90502_UR_PAD_NO_PADDING when the channel is opened the value of this counter is undefined.

Direction: OUT Type: ULONG

Default: 0

pdv_monitored_cnt 32 bit unsigned counter

The number of times the packet delay variation (PDV) of the channel has been monitored for excess delay.

Direction: OUT Type: ULONG

Default: 0

pdv_absorbtion_buffer_min_fill -2047 – 2047

The minimum fill of the PDV absorption buffer observed on the channel (in frames). Values greater than the circular buffer size or smaller than zero indicate that slips occurred.

Direction: OUT Type: ULONG

Default: 0

pdv_absorbtion_buffer_max_fill -2047 – 2047

The maximum fill of the PDV absorption buffer observed on the channel (in frames). Values greater than the circular buffer size or smaller than zero indicate that slips occurred.

Direction: OUT Type: ULONG

Default: 0

monitored_pdv 0 – 4095

The PDV (in frames) present on the channel during the last monitoring period.

Direction: OUT Type: ULONG

Default: 0

2.4.8 mt90502_get_tx_xxpcm_chan_statistics

This function fills an MT90502_TX_XXPCM_CHAN_STATS structure with the current statistics of the channel indicated by ch_hndl. All statistics returned by this function are initialized (e.g. counters set to 0) by the call to mt90502_open_tx_xxpcm_channel.

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_tx_xxpcm_chan_statistics_def function inserts default values into the MT90502_TX_XXPCM_CHAN_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_tx_xxpcm_chan_statistics_def(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_TX_XXPCM_CHAN_STATS * ptx_ch_stats );

ULONG mt90502_get_tx_xxpcm_chan_statistics(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_TX_XXPCM_CHAN_STATS * ptx_ch_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always.

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

ptx_ch_stats

pointer to an MT90502_TX_XXPCM_CHAN_STATS statistics structure. The definitions of the structure’s elements are listed below. The user allocates this structure.

2.4.8.1 Structure MT90502_TX_XXPCM_CHAN_STATS

ch_hndl handle

handle indicating the transmit channel for which the statistics are requested. This handle is created from the call to mt90502_open_tx_xxpcm_channel.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

reset_statistics TRUE / FALSE

Resets the statistics counters for the identified channel after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

cid_hndl handle
MT90502_INVALID_HANDLE

The handle of the mini packet stream to which the channel is associated. This handle was returned from the call to mt90502_open_cid. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

cid_number 1 – 255
MT90502_INVALID_CID

The CID used for this channel within the AAL2 VC.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

vc_hndl handle
MT90502_INVALID_HANDLE

The handle of the VC to which the channel is associated. This handle was returned from the call to mt90502_open_aal2_vc. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header see **MT90502_AAL2_VC** structure

If the channel has not been mapped to a mini packet stream the returned header will be all zeros.

Direction: OUT Type: same

Default: 0

rx_tx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_INVALID_UTOPIA_PORT

The UTOPIA port associated with the VC carrying this xxPCM channel. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_UTOPIA_PORT

Direction: OUT Type: same

Default: MT90502_INVALID_UTOPIA_PORT

compression_rate See **MT90502_TX_CHAN** structure

Direction: OUT Type: ULONG

Default: MT90502_INVALID_COMPRESSION

number_of_edus See **MT90502_TX_CHAN** structure

Direction: OUT Type: ULONG

Default: 0

chan_open_time 0 – ??

The amount of time in seconds since this transmit xxPCM channel was opened by the mt90502_open_tx_xxpcm_channel function.

Direction: OUT Type: ULONG

Default: 0

mini_pkt_cnt[2] 64 bit counter

The number of AAL2 mini-packets queued for transmission on this channel. This includes mini packets that are eventually suppressed by silence suppression and any SID packets sent. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

byte_cnt[2] 64 bit counter

This counter sums the LI value plus 1 of each mini-packet transmitted, which represents the number of xxPCM bytes and SID bytes this channel has transmitted on this VC. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

sid_count[2] 64 bit counter

The number of SID packets generated for this channel. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

sil_profile_number 0 – 255
MT90502_SIL_SUPPR_DISABLED

This field is the index into the profile table. The index was supplied to the call to mt90502_open_tx_xxpcm_channel. If this field is disabled then the remaining fields of this structure are to be ignored. See Silence Suppression Configuration Parameters.

Direction: OUT Type: ULONG

Default: MT90502_SIL_SUPPR_DISABLED

additional_delay 0 – 25 (ms)

The delay inserted in the TX direction of the channel. See Silence Suppression Configuration Parameters.

Direction: OUT Type: ULONG

Default: 0

voice_to_silence_time 0 – 1000 (ms)

The period of continuous silence that must be present on the channel to switch the silence suppression state from voice to silence. See Silence Suppression Configuration Parameters.

Direction: OUT Type: ULONG

Default: 0

silence_to_voice_time 0 – 25 (ms)

The period of continuous voice that must be present on the channel to switch the silence suppression state from silent to voice. See Silence Suppression Configuration Parameters.

Direction: OUT Type: ULONG

Default: 0

sid_energy_calculation_period 1 – 64 (ms)

The period of data used to calculate the SID values. See Silence Suppression Configuration Parameters.

Direction: OUT Type: ULONG

Default: 0

silence_suppression_state MT90502_SIL_SUPPR_VOICE
MT90502_SIL_SUPPR_SILENT

The current silence suppression state of the channel. This indicates if the mini packets are being sent or suppressed.

Direction: OUT Type: ULONG

Default: MT90502_SIL_SUPPR_VOICE

local_dc_offset_correction -127– 127

The current offset applied to the local energy calculations to compensate for a DC offset. The value is in linear steps and thus is added to the uncompressed data. This value is recalculated every second.

Direction: OUT Type: ULONG

Default: 0

remote_dc_offset_correction -127 – 127

The current offset applied to the remote energy calculations to compensate for a DC offset. The value is in linear steps and thus is added to the uncompressed data. This value is recalculated every second.

Direction: OUT Type: ULONG

Default: 0

2.4.9 mt90502_get_rx_hdlc_chan_statistics

This function fills an MT90502_RX_HDLC_CHAN_STATS structure with the current statistics of the channel indicated by ch_hndl. All statistics returned by this function are initialized (e.g. counters set to 0) by the call to mt90502_open_rx_hdlc_channel.

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_rx_hdlc_chan_statistics_def function inserts default values into the MT90502_RX_HDLC_CHAN_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_rx_hdlc_chan_statistics_def(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_RX_HDLC_CHAN_STATS* prx_ch_stats );

ULONG mt90502_get_rx_hdlc_chan_statistics(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_RX_HDLC_CHAN_STATS* prx_ch_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

prx_ch_stats

pointer to an MT90502_RX_HDLC_CHAN_STATS statistics structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.4.9.1 Structure MT90502_RX_HDLC_CHAN_STATS

ch_hndl handle

handle indicating the receive channel for which the statistics are requested. This handle is created from the call to mt90502_open_rx_hdlc_channel.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

reset_statistics TRUE / FALSE

Resets the statistics counters for the identified channel after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

chan_initialized TRUE / FALSE

Indicates whether the first mini-packet of the channel has been received yet.

Direction: IN Type: ULONG

Default: FALSE

stream_hndl handle

The handle of the HDLC stream to which the channel is associated. This handle was returned from the call to mt90502_open_rx_hdlc_stream. This channel was associated to this stream by the function mt90502_open_hdlc_channel.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

hdlc_address 0 – 127
MT90502_INVALID_HDLC_ADDRESS

The address of the HDLC header. This field is set to MT90502_INVALID_HDLC_ADDRESS if the hdlc_header_type of the associated stream is set to MT90502_HDLC_NO_HEADER.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HDLC_ADDRESS

cid_hndl handle
MT90502_INVALID_HANDLE

The handle of the mini packet stream to which the channel is associated. This handle was returned from the call to mt90502_open_cid. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

cid_number 1 – 255
MT90502_INVALID_CID

The CID used for this channel within the AAL2 VC. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_CID

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

vc_hndl handle
MT90502_INVALID_HANDLE

The handle of the VC to which the channel is associated. This handle was returned from the call to mt90502_open_aal2_vc. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header see **MT90502_AAL2_VC** structure

If the channel has not been mapped to a mini packet stream the returned header will be all zeros.

Direction: OUT Type: same

Default: 0

rx_tx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_INVALID_UTOPIA_PORT

The UTOPIA port associated with the VC carrying this xxPCM channel. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_UTOPIA_PORT

Direction: OUT Type: same

Default: MT90502_INVALID_UTOPIA_PORT

chan_open_time 0 – ??

The amount of time in seconds since this receive HDLC channel was opened by the mt90502_open_rx_hdlc_channel function.

Direction: OUT Type: ULONG

Default: 0

mini_pkt_cnt[2] 64 bit counter

The number of AAL2 mini-packets of this channel received on UTOPIA. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

buffer_packet_loss_cnt 32 bit counter

The number of HDLC packets lost due to an overflow of the channel's circular buffer. This can indicate that the received bandwidth has exceeded the bandwidth of the TDM bus for the channel. This may be due to a higher than expected PDV of the network, a misbehaving source of this channel or other channels of this stream, the rx_circular_buffer_size was set too small in mt90502_open, or the HDLC stream carrying this channel was configured with too few TSSTs for the channel(s) it is carrying.

Direction: OUT Type: ULONG

Default: 0

descriptor_packet_loss_cnt 32 bit counter

The number of HDLC packets lost due to an overflow of the HDLC descriptor queue. The HDLC descriptor queue is a shared resource of all channels carried on an HDLC stream. Its size is determined when the stream is opened as 16*max_channels parameter. The causes of this event are similar to rx_buffer_packet_loss_cnt except the cause is the number of received packets, regardless of the packet size, relative to the ability to transmit them on the TDM bus.

Direction: OUT Type: ULONG

Default: 0

2.4.10 mt90502_get_tx_hdlc_chan_statistics

This function fills an MT90502_TX_HDLC_CHAN_STATS structure with the current statistics of the channel indicated by ch_hndl. All statistics returned by this function are initialized (e.g. counters set to 0) by the call to mt90502_open_tx_hdlc_channel.

The statistics returned by this function can be reset via the reset_statistics parameter.

The mt90502_get_tx_hdlc_chan_statistics_def function inserts default values into the MT90502_TX_HDLC_CHAN_STATS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_tx_hdlc_chan_statistics_def(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_TX_HDLC_CHAN_STATS* ptx_ch_stats );

ULONG mt90502_get_tx_hdlc_chan_statistics(
            MT90502_INSTANCE_API * pmt90502_api,
            MT90502_TX_HDLC_CHAN_STATS* ptx_ch_stats );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

ptx_ch_stats

pointer to an MT90502_TX_HDLC_CHAN_STATS statistics structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.4.10.1 Structure MT90502_TX_HDLC_CHAN_STATS

ch_hndl handle

handle indicating the transmit channel for which the statistics are requested. This handle is created from the call to mt90502_open_tx_hdlc_channel.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

reset_statistics TRUE / FALSE

Resets the statistics counters for the identified channel after returning their current values.

Direction: IN Type: ULONG

Default: FALSE

stream_hndl handle

The handle of the HDLC stream to which the channel is associated. This handle was returned from the call to mt90502_open_tx_hdlc_stream. This channel was associated to this stream by the function mt90502_open_hdlc_channel.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

hdlc_address 0 – 127
MT90502_INVALID_HDLC_ADDRESS

The address of the HDLC header. This field is set to MT90502_INVALID_HDLC_ADDRESS if the hdlc_header_type is set to MT90502_HDLC_NO_HEADER.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HDLC_ADDRESS

cid_hndl handle MT90502_INVALID_HANDLE

The handle of the mini packet stream to which the channel is associated. This handle was returned from the call to mt90502_open_cid. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

cid_number 1 – 255
MT90502_INVALID_CID

The CID used for this channel within the AAL2 VC. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_CID

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

vc_hndl handle
MT90502_INVALID_HANDLE

The handle of the VC to which the channel is associated. This handle was returned from the call to mt90502_open_aal2_vc. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_HANDLE

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header see **MT90502_AAL2_VC** structure

If the channel has not been mapped to a mini packet stream the returned header will be all zeros.

Direction: OUT Type: same

Default: 0

rx_tx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC
MT90502_INVALID_UTOPIA_PORT

The UTOPIA port associated with the VC carrying this xxPCM channel. If the channel has not been mapped to a mini packet stream the returned value will be MT90502_INVALID_UTOPIA_PORT

Direction: OUT Type: same

Default: MT90502_INVALID_UTOPIA_PORT

chan_open_time 0 – ??

The amount of time in seconds since this transmit HDLC channel was opened by the mt90502_open_tx_hdlc_channel function.

Direction: OUT Type: ULONG

Default: 0

mini_pkt_cnt[2] 64 bit counter

The number of AAL2 mini-packets of this channel transmitted on UTOPIA. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

byte_cnt[2] 64 bit counter

This counter sums the transmitted LI value plus 1 of each mini-packet, which represents the number of payload bytes this channel transmitted on the VC including SID packets. Element 0 of this field contains the 32 least significant bits of the counter, and element 1 the 32 most significant.

Direction: OUT Type: ULONG[2]

Default: 0

2.4.11 mt90502_convert_<xxxx>_chan_statistics_to_text

This section describes 4 functions for converting channel statistics structures to a text string. The 4 functions are:

mt90502_convert_rx_xxpcm_chan_statistics_to_text -
converts the MT90502_RX_XXPCM_CHAN_STATS structure to text

mt90502_convert_tx_xxpcm_chan_statistics_to_text -
converts the MT90502_TX_XXPCM_CHAN_STATS structure to text

mt90502_convert_rx_hdlc_chan_statistics_to_text -
converts the MT90502_RX_HDLC_CHAN_STATS structure to text

mt90502_convert_tx_hdlc_chan_statistics_to_text -
converts the MT90502_TX_HDLC_CHAN_STATS structure to text

Each of the statistics structures are returned by the associated get statistics function.

Each function has an associated mt90502_convert_<xxxx>_chan_statistics_to_text_def function which inserts default values into the MT90502_CONVERT_<XXXX>_CHAN_STATS structure. The default value of a structure field is indicated following the field's description.

Usage (rx_xxpcm shown)

```
#include "mt90502_api.h"

ULONG mt90502_convert_rx_xxpcm_chan_statistics_to_text_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONVERT_RX_XXPCM_CHAN_STATS* pconvert_ch_stats );

ULONG mt90502_convert_rx_xxpcm_chan_statistics_to_text (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONVERT_RX_XXPCM_CHAN_STATS* pconvert_ch_stats );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success always

Parameters

pmt90502_api

pointer to an instance structure of the chip

pconvert_ch_stats

pointer to a MT90502_CONVERT_<XXXX>_CHAN_STATS structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.4.11.1 Structure MT90502_CONVERT_<XXXX>_CHAN_STATS

pch_stats

pointer to an MT90502_RX_XXPCM_CHAN_STATS, MT90502_TX_XXPCM_CHAN_STATS, MT90502_RX_HDLC_CHAN_STATS, or MT90502_TX_HDLC_CHAN_STATS statistics structure to be converted to text. The definitions of the structure's elements are listed in the associated get statistics function description.

Direction: IN/IN Type: POINTER

Default: NULL

pstring

pointer to the returned string. The required length of the string is defined by MT90502_CHAN_STATS_STRING_LENGTH (the largest of the 4 text strings in bytes). The user allocates the string.

Direction: IN/OUT Type: POINTER

Default: NULL

2.4.12 mt90502_update_counters

This function is called periodically by the user application. It is used to update and extend statistics counters in the chip. An update cycle must be completed regularly to prevent the counters from wrapping and causing bad extended statistics counter values. The function will indicate if statistics have been corrupted. Corrupted statistics have no side effects on the operation of the chip.

The maximum time between updates without corrupting is dependent on the number of channels open and rate at which the channels operate and the frequency of the chip mclk among other things. The function services as many channels as possible within the max_update_time supplied by the user. The maximum period between servicing of the same channel can be as large as 30 seconds. (See update_complete parameter.)

The mt90502_update_counters_def function inserts default values into the fields of the MT90502_UPDATE_COUNTERS structure. The default value of a structure field is indicated below the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_update_counters_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_UPDATE_COUNTERS * pupdate_counters );

ULONG mt90502_update_counters (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_UPDATE_COUNTERS * pupdate_counters );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

a pointer to the MT90502_INSTANCE_API structure of the chip for which the statistics counters are to be updated.

pupdate_counters

a pointer to an MT90502_UPDATE_COUNTERS structure. The definitions of the elements of the structure are provided below.

2.4.12.1 Structure MT90502_UPDATE_COUNTERS

reset_statistics TRUE / FALSE

Resets all statistics counters of the chip. Resetting the counters of the chip does not respect the max_update_time parameter and may exceed the specified time.

Direction: IN Type: ULONG

Default: FALSE

max_update_time 1 – 1048575 ms

The maximum time in ms this function will run. The update function will return when one cycle is complete (i.e. update_complete=TRUE) or this time is expired. The function will begin updating where it left off in the cycle on subsequent calls.

Direction: IN Type: ULONG

Default: 500

update_complete TRUE / FALSE

If TRUE a cycle of updates (i.e. all channels have been serviced once) has been completed during this function execution. If FALSE not all updates were completed in the allowed max_update_time. A subsequent call will return TRUE when a cycle has been completed. Once a cycle has been completed a new cycle will be started on the next call of mt90502_update_counters. A cycle should be completed every 30 seconds to avoid corrupted statistics.

Direction: OUT Type: ULONG

Default: FALSE

statistics_corrupted TRUE / FALSE

If TRUE statistics counters have been corrupted due to wrapping. The counters should be reset.

Direction: OUT Type: ULONG

Default: FALSE

2.5 Utility Functions

2.5.1 mt90502_get_handle_list

This function returns a list of handles of a certain type.

The mt90502_get_handle_list_def function inserts default values into the MT90502_HANDLE_REQUEST structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_handle_list_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_HANDLE_REQUEST* phandle_request );

ULONG mt90502_get_handle_list( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_HANDLE_REQUEST* phandle_request );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip.

phandle_request

pointer to an MT90502_HANDLE_REQUEST structure that defines the list being requested. The user allocates this structure. The definitions of the structure's elements are listed below.

2.5.1.1 Structure MT90502_HANDLE_REQUEST

max_hndl 1 – 2097152

Maximum number of handles to be returned in the MT90502_HANDLE_LIST structure.

Direction: IN Type: ULONG

Default: 0

hndl_type

MT90502_HNDL_AAL2_VC
MT90502_HNDL_DATA_VC
MT90502_HNDL_CID
MT90502_HNDL_RX_XXPCM_CHAN
MT90502_HNDL_TX_XXPCM_CHAN
MT90502_HNDL_RX_HDLC_CHAN
MT90502_HNDL_TX_HDLC_CHAN
MT90502_HNDL_RX_HDLC_STREAM
MT90502_HNDL_TX_HDLC_STREAM
MT90502_HNDL_LLL_CHAN

Defines the type of handle that is being requested.

Direction: IN Type: ULONG

Default: MT90502_HNDL_AAL2_VC

num_valid_hndl 0 – max_hndl

This value is the number of valid handles returned. Note if the returned list is max_hndl handles long there may be more handles of the requested type.

Direction: OUT Type: ULONG

Default: 0

phndl_list pointer

Pointer to a list of ULONGs. The length of the list is max_hndl. This list will be filled by the function with all handles of the requested handle type. The user allocates this structure.

Direction: IN/OUT Type: POINTER

Default: NULL

2.5.2 mt90502_query_handle_type

This function returns the type of the handle indicated by the value of the parameter q_hndl.

The mt90502_query_handle_type_def function inserts default values into the MT90502_QUERY_HANDLE structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_query_handle_type_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_QUERY_HANDLE* pquery_handle );

ULONG mt90502_query_handle_type( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_QUERY_HANDLE* pquery_handle );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success
MT90502_HNDL_ERR	The handle is not valid.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip.

pquery_handle

pointer to an MT90502_QUERY_HANDLE structure that defines the handle query. The user allocates this structure. The definitions of the structure's elements are listed below.

2.5.2.1 Structure MT90502_QUERY_HANDLE

q_hndl handle

the handle of the VC or channel for which the type is being queried. This handle is returned by the call to the function that opened the channel or VC.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

hndl_type

MT90502_HNDL_AAL2_VC
MT90502_HNDL_DATA_VC
MT90502_HNDL_CID
MT90502_HNDL_RX_XXPCM_CHAN
MT90502_HNDL_TX_XXPCM_CHAN
MT90502_HNDL_RX_HDLC_CHAN
MT90502_HNDL_TX_HDLC_CHAN
MT90502_HNDL_RX_HDLC_STREAM
MT90502_HNDL_TX_HDLC_STREAM
MT90502_HNDL_LLL_CHAN

identifies the type of the handle.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE_TYPE

2.5.3 mt90502_get_li_uui_change_event

This function returns a list of channels with LI or UUI change events and new LI or UUI values. The events are buffered in an API maintained soft buffer. See the soft_li_uui_change_buffer_size in the MT90502_CONF structure.

With the addition of the ATM Forum AF-VMOA-0145 profiles for PCM/ADPCM transport UUI changes indicate changes in compression in addition to LI changes for ITU 366.2 profiles. The concept of LI changes is extended to cover UUI changes.

The mt90502_get_li_uui_change_event_def function inserts default values into the MT90502_LI_EVENT structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_li_uui_change_event_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_LI_EVENT* pli_event );

ULONG mt90502_get_li_uui_change_event(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_LI_EVENT* pli_event );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success
MT90502ER_NO_LI_EVENTS	when there are no LI events to report.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

pli_event

pointer to an MT90502_LI_EVENT structure containing the LI or UUI change events. The user allocates this structure. The definitions of the structure's elements are listed below.

2.5.3.1 Structure MT90502_LI_EVENT

reset_buffers TRUE / FALSE

If set to TRUE, the hardware and software buffers for the LI and UUI events will be emptied. When set to TRUE the function will not return an event, and more_events will be set to FALSE.

Direction: IN Type: ULONG

Default: FALSE

ch_hndl handle

The handle indicating the channel on which the LI or UUI change event occurred. The handle was returned by the channel open function and can be either XXPCM, or HDLC.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header 32 bit field

The ATM cell header defined in mt90502_open_aal2_vc for the VC on which the LI or UUI change event occurred. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: OUT Type: ULONG

Default: MT90502_NULL_HEADER

utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

The UTOPIA port that the mini-packet was received on.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

cid 8 bit field

The CID value of the received mini-packet.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

uui 5 bit field

The new UUI value of the received mini-packet.

Direction: OUT Type: ULONG

Default: 0

li 5 bit field

The new LI value of the received mini-packet.

Direction: OUT Type: ULONG

Default: 0

more_events TRUE / FALSE

True if there are more LI events to be retrieved.

Direction: OUT Type: ULONG

Default: FALSE

2.6 Diagnostics Functions

2.6.1 mt90502_get_h100_diagnostics

This function fills an MT90502_H100_DIAG structure with the current diagnostic information of the H100 bus of the chip.

The mt90502_get_h100_diagnostics_def function inserts default values into the MT90502_H100_DIAG structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_h100_diagnostics_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_H100_DIAG* ph100_diag );

ULONG mt90502_get_h100_diagnostics(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_H100_DIAG* ph100_diag );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

ph100_diag

pointer to an MT90502_H100_DIAG structure to be filled in by this routine. The user allocates this structure.

2.6.1.1 Structure MT90502_H100_DIAG

h100_clk_a_bad TRUE / FALSE

If TRUE, the H.100 signal ct_c8_a has failed to comply with the H.100 specification. This monitors the clock edges and will detect period violations of ± 35 ns from the 122 ns nominal specification to within the resolution of the mclk frequency.

Direction: OUT Type: ULONG

Default: FALSE

h100_clk_b_bad TRUE / FALSE

If TRUE, the H.100 signal ct_c8_b has failed to comply with the H.100 specification.

Direction: OUT Type: ULONG

Default: FALSE

h100_frame_a_bad TRUE / FALSE

If TRUE, the H.100 signal ct_frame_a has failed to comply with the H.100 specification. This monitors that the H.100 frame signal and will detect a violation if it is not asserted once, and only once every 1024 H.100 bus clock cycles.

Direction: OUT Type: ULONG

Default: FALSE

h100_frame_b_bad TRUE / FALSE

If TRUE, the H.100 signal ct_frame_b has failed to comply with the H.100 specification.

Direction: OUT Type: ULONG

Default: FALSE

bus_master MT90502_H100_MASTER_A
MT90502_H100_MASTER_B

Which clock is currently the master clock of the bus.

Direction: OUT Type: ULONG

Default: MT90502_H100_MASTER_A

bus_master_bad TRUE / FALSE

If TRUE, the bus master clock has failed to comply with the H.100 specification.

Direction: OUT Type: ULONG

Default: FALSE

bus_backup MT90502_H100_BACKUP_A
MT90502_H100_BACKUP_B

Which clock is the current backup clock.

Direction: OUT Type: ULONG

Default: MT90502_H100_BACKUP_A

bus_backup_bad TRUE / FALSE

If TRUE, the backup clock has failed to comply with the H.100 specification.

Direction: OUT Type: ULONG

Default: FALSE

master_mode see MT90502_H100_MASTER_PARMS structure of mt90502_set_h100_master_mode.
The master mode of the chip.

Direction: OUT Type: same

Default: MT90502_H100_MASTERA

slave_mode see MT90502_H100_SLAVE_PARMS structure of mt90502_set_h100_slave_mode.
The slave mode of the chip.

Direction: OUT Type: same

Default: MT90502_H100_TRACKA

h100_clk_a_bad_cnt
A count of the number of times the signal CT_C8_A was deemed bad. The count is an approximation based on the transitions of the interrupt register bit corresponding to the signal.

Direction: OUT Type: ULONG

Default: 0

h100_clk_b_bad_cnt

A count of the number of times the signal CT_C8_B was deemed bad. The count is an approximation based on the transitions of the interrupt register bit corresponding to the signal.

Direction: OUT Type: ULONG

Default: 0

h100_frame_a_bad_cnt

A count of the number of times the signal CT_FRAME_A was deemed bad. The count is an approximation based on the transitions of the interrupt register bit corresponding to the signal.

Direction: OUT Type: ULONG

Default: 0

h100_frame_b_bad_cnt

A count of the number of times the signal CT_FRAME_B was deemed bad. The count is an approximation based on the transitions of the interrupt register bit corresponding to the signal.

Direction: OUT Type: ULONG

Default: 0

2.6.2 mt90502_convert_h100_diagnostics_to_text

This function converts an MT90502_H100_DIAG structure to a text string. The MT90502_H100_DIAG structure is returned by the mt90502_get_h100_diagnostics function.

The mt90502_convert_h100_diagnostics_to_text_def function inserts default values into the MT90502_CONVERT_H100_DIAG structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_convert_h100_diagnostics_to_text_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONVERT_H100_DIAG* pconvert_h100_diag );

ULONG mt90502_convert_h100_diagnostics_to_text(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONVERT_H100_DIAG* pconvert_h100_diag );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success always

Parameters

pmt90502_api

pointer to an instance structure of the chip

pconvert_h100_diag

pointer to an MT90502_CONVERT_H100_DIAG structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.6.2.1 Structure MT90502_CONVERT_H100_DIAG

ph100_diag

pointer to an MT90502_H100_DIAG structure to be converted to text. The definition of the structure elements is provided in the mt90502_get_h100_diagnostics function description.

Direction: IN/IN Type: POINTER

Default: NULL

pstring

pointer to the returned string. The required length of the string is defined by MT90502_H100_DIAG_STRING_LENGTH (in bytes). The user allocates the string.

Direction: IN/OUT Type: POINTER

Default: NULL

2.6.3 mt90502_get_console_msgs

This function returns debug messages from the API in a text string. These messages include detail of errors and warnings.

The mt90502_get_console_msgs_def function inserts default values into the MT90502_CONSOLE_MSG structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_console_msgs_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONSOLE_MSG* pconsole_msg );

ULONG mt90502_get_console_msgs( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONSOLE_MSG* pconsole_msg );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip.

pconsole_msg

pointer to an MT90502_CONSOLE_MSG structure to be filled in by this routine. The definitions of the structure's elements are listed below. The user allocates this structure.

2.6.3.1 Structure MT90502_CONSOLE_MSG

pstring

pointer to the returned string. The required length of the string is mt90502_console_buffer_size bytes that was configured by mt90502_open. The user allocates the string.

Direction: IN/OUT Type: POINTER

Default: NULL

2.7 H.100 Functions

2.7.1 mt90502_set_h100_master_mode

This function sets the role of the chip as bus master on the H.100 bus.

The mt90502_set_h100_master_mode_def function inserts default values into the MT90502_H100_MASTER_PARMS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_set_h100_master_mode_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_H100_MASTER_PARMS* ph100_master_parms );

ULONG mt90502_set_h100_master_mode(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_H100_MASTER_PARMS* ph100_master_parms );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

ph100_master_parms

pointer to an MT90502_H100_MASTER_PARMS structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.7.1.1 Structure MT90502_H100_MASTER_PARMS

master_mode

MT90502_H100_MASTERA
MT90502_H100_MASTERB
MT90502_H100_MASTERAB
MT90502_H100_BACKUPA
MT90502_H100_BACKUPB
MT90502_H100_DISABLED

Determines which H.100 clocks the chip is to drive.

The "_MASTER" modes drive the corresponding ct_c8 and ct_frame signal(s) as well as the compatibility signals. The "_BACKUP" modes drive the corresponding ct_c8 and ct_frame signals; the ct_c8/ct_frame signals will be generated in phase with the master ct_c8/ct_frame signals in backup mode. The "_DISABLED" mode does not drive any clock or frame signals. The initial setting is "_DISABLED" when the mt90502_open function returns.

Direction: IN Type: ULONG

Default: MT90502_H100_MASTERA

2.7.2 mt90502_set_h100_slave_mode

This function sets the slave mode of the chip and determines the clock used by the chip to synchronize all data transfers on the H.100 bus.

If the chip does a fallback onto another clock then data transfers will continue to be synchronized on the fallback clock until slave mode is set once again, regardless of the state of the chosen clock.

The mt90502_set_h100_slave_mode_def function inserts default values into the MT90502_H100_SLAVE_PARMS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_set_h100_slave_mode_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_H100_SLAVE_PARMS*
            ph100_slave_parms );

ULONG mt90502_set_h100_slave_mode( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_H100_SLAVE_PARMS*
            ph100_slave_parms );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

ph100_slave_parms

pointer to an MT90502_H100_SLAVE_PARMS structure. The definitions of the structure's elements are listed below.

2.7.2.1 Structure MT90502_H100_SLAVE_PARMS

slave_mode

MT90502_H100_TRACKA
MT90502_H100_TRACKB
MT90502_H100_TRACKA_FALLBACKB
MT90502_H100_TRACKB_FALLBACKA
MT90502_H100_DISABLED

Determines how the chip is to synchronize its data transfers on the H.100 bus.

The "_TRACK" modes with no "_FALLBACK" perform data transfers synchronized to the "_TRACKx" clock no matter the condition of that clock or associated frame signal. The "_FALLBACK" modes synchronize data transfers to the "_FALLBACKx" clock and associated frame signal if the "_TRACKx_" clock or associated frame signal is not behaving according to the H.100 specification. The "_DISABLED" mode disables all transfers on the H.100 bus. Thus, no data can be placed on the bus, and the received data is ignored. The initial setting is "_DISABLED" when the mt90502_open function returns.

Direction: IN Type: ULONG

Default: MT90502_H100_TRACKA_FALLBACKB

2.8 Data Cell Functions

2.8.1 mt90502_send_data_cell

This function transmits a CPU generated ATM cell. The cell is placed at the tail of the data cell FIFO of the TXSAR. Once the cell reaches the head of the FIFO the cell will be transmitted on the specified UTOPIA port.

The mt90502_send_data_cell_def function inserts default values into the MT90502_TX_DATA_CELL structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_send_data_cell_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_DATA_CELL* ptx_data_cell );

ULONG mt90502_send_data_cell( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_DATA_CELL* ptx_data_cell );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success.

`MT90502ER_SEND_DATA_CELL_BUFFER_FULL`
there is no room in the send data cell buffer. The cell was not queued to be sent.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
pointer to an instance structure of the chip

ptx_data_cell
pointer to an MT90502_TX_DATA_CELL structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.8.1.1 Structure MT90502_TX_DATA_CELL

header 32 bit field

The header of the cell. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: IN Type: ULONG

Default: MT90502_NULL_HEADER

payload[12] 12 element array of 32 bit fields

An array of the 48 payload bytes of the cell. The payload bytes of the cell are arranged in the array as follows:

	b31 - b24	b23 - b16	b15 -b8	b7 - b0
payload[0]	payload byte 0	payload byte 1	payload byte 2	payload byte 3
payload[1]	payload byte 4	payload byte 5	payload byte 6	payload byte 7
…	…	…	…	…
payload[11]	payload byte 44	payload byte 45	payload byte 46	payload byte 47

Direction: IN Type: ULONG[12]

Default: 0

tx_utopia_port 0 or the OR of any or all of:
MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

Indicates how the data cell is to be routed by the UTOPIA module. The cell can be broadcast, so the values can be ORed together. If set to 0, the cell will be discarded.

Direction: IN Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

2.8.2 mt90502_send_test_cell

This function transmits a CPU generated ATM cell. The cell is placed at the tail of the data cell FIFO of the TXSAR. Once the cell reaches the head of the FIFO the cell will be treated as if it were received on the specified UTOPIA port (i.e. it will use the LUT for the specified port and the entry identified by the header to rout the cell). This function can be used to test RX hardware or software functions of the system.

The MT90502_send_test_cell_def function inserts default values into the MT90502_TX_TEST_CELL structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_send_test_cell_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_TEST_CELL* ptx_test_cell );

ULONG mt90502_send_test_cell( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_TX_TEST_CELL* ptx_test_cell );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success.

`MT90502ER_SEND_DATA_CELL_BUFFER_FULL`
there is no room in the send data cell buffer. The cell was not queued to be sent.
Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
pointer to an instance structure of the chip

ptx_test_cell
pointer to an MT90502_TX_TEST_CELL structure. The definitions of the structure's elements are listed below.

2.8.2.1 Structure MT90502_TX_TEST_CELL

header 32 bit field

The header of the cell. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: IN Type: ULONG

Default: MT90502_NULL_HEADER

payload[12] 12 element array of 32 bit fields

An array of the 48 payload bytes of the cell. The payload bytes of the cell are arranged in the array as follows:

	b31 - b24	b23 - b16	b15 -b8	b7 - b0
payload[0]	payload byte 0	payload byte 1	payload byte 2	payload byte 3
payload[1]	payload byte 4	payload byte 5	payload byte 6	payload byte 7
…	…	…	…	…
payload[11]	payload byte 44	payload byte 45	payload byte 46	payload byte 47

Direction: IN Type: ULONG[12]

Default: 0

rx_utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

Indicates which port's LUT will be used to treat the cell.

Direction: IN Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

2.8.3 mt90502_receive_data_cell

This function retrieves the oldest received data cell. The cells are buffered in the SSRAM and/or an API maintained soft buffer in received order. See the soft_rx_data_buffer_size in the MT90502_CONF structure.

The mt90502_receive_data_cell_def function inserts default values into the MT90502_RX_DATA_CELL structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_receive_data_cell_def( MT90502_INSTANCE_API * pmt90502_api,
            MT90502_RX_DATA_CELL * prx_data_cell );

ULONG mt90502_receive_data_cell( MT90502_INSTANCE_API * pmt90502_api,
            MT90502_RX_DATA_CELL * prx_data_cell );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_RECEIVE_DATA_CELL_BUFFER_EMPTY`
when there are no data cells in the received data cell buffer. The returned structure is invalid.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
pointer to an instance structure of the chip

prx_data_cell

pointer to an MT90502_RX_DATA_CELL structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.8.3.1 Structure MT90502_RX_DATA_CELL

reset_buffers TRUE / FALSE

If set to TRUE, the hardware and software buffers for the data cells will be emptied. When set to TRUE the function will not return a cell, and more_cells will be set to FALSE.

Direction: IN Type: ULONG

Default: FALSE

header 32 bit field

The header of the cell. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: OUT Type: ULONG

Default: MT90502_NULL_HEADER

payload[12] 12 element array of 32 bit fields

An array of the 48 payload bytes of the cell. The payload bytes of the cell are arranged in the array as follows:

	b31 - b24	b23 - b16	B15 -b8	b7 - b0
payload[0]	payload byte 0	payload byte 1	payload byte 2	payload byte 3
payload[1]	payload byte 4	payload byte 5	payload byte 6	payload byte 7
…	…	…	…	…
payload[11]	payload byte 44	payload byte 45	payload byte 46	payload byte 47

Direction: OUT Type: ULONG[12]

Default: 0

rx_utopia_port

MT90502_PORT_A
MT90502_PORT_B
MT90502_PORT_C

The UTOPIA port on which the cell was received.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

rx_cell_routing

MT90502_AAL2_VC_LUT_ENTRY
MT90502_DATA_VC_LUT_ENTRY
MT90502_UNKNOWN_CELL

Indicates how the cell was routed to the data cell FIFO. The cell can have been routed by the LUT entry of a CBR or data VC, or it can have been routed by the port if the cell was declared as unknown (see the u_rxp_ncr and u_rxp_ocr cell routing parameters in the MT90502_CONF structure). If a LUT entry routed the cell then the handle to the VC is contained in the rx_vc_hndl parameter.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_ROUTING

rx_vc_hndl

If the cell was routed to the data cell FIFO via a VC's LUT entry then this field contains the handle to that VC. See rx_cell_routing.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

more_cells TRUE / FALSE

True if there are more cells buffered to be read.

Direction: OUT Type: ULONG

Default: FALSE

2.9 AAL2 Mini-Packet Functions

2.9.1 mt90502_send_aal2_mini_pkt

This function transmits a CPU generated AAL2 mini-packet. The mini-packet is inserted in a cell, on the VC to which the channel indicated by cid_hndl is associated.

The mt90502_send_aal2_mini_pkt_def function inserts default values into the MT90502_AAL2_SEND_MINI_PKT structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_send_aal2_mini_pkt_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_AAL2_SEND_MINI_PKT* paal2_send_mini_pkt );

ULONG mt90502_send_aal2_mini_pkt( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_AAL2_SEND_MINI_PKT* paal2_send_mini_pkt );
```

Return Values

MT90502ER_GENERIC_OK
Indicates success.

MT90502ER_SEND_MINI_PKT_BUFFER_FULL
there is no room in the send AAL2 mini-packet buffer. The packet was not queued to be sent.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
pointer to an instance structure of the chip

paal2_send_mini_pkt

pointer to an MT90502_AAL2_SEND_MINI_PKT structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.9.1.1 Structure MT90502_AAL2_SEND_MINI_PKT

cid_hndl handle

the handle indicating the mini packet stream on which the mini-packet is to be sent.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

li 6 bit field

The LI value to be sent in the mini-packet.

Direction: IN Type: ULONG

Default: MT90502_INVALID_LI

uui 5 bit field

The UUI value to be sent in the mini-packet.

Direction: IN Type: ULONG

Default: MT90502_INVALID_UUI

payload[16] 16 element array of 32 bit fields

An array of the 64 payload bytes of the mini-packet. The payload bytes of the mini-packet are arranged in the array as follows:

	b31 - b24	b23 - b16	b15 -b8	b7 - b0
payload[0]	payload byte 0	payload byte 1	payload byte 2	payload byte 3
payload[1]	payload byte 4	payload byte 5	payload byte 6	payload byte 7
…	…	…	…	…
payload[15]	payload byte 60	payload byte 61	payload byte 62	payload byte 63

Only the payload bytes up to, and including payload byte "LI" need to be filled.

Direction: IN Type: ULONG[16]

Default: 0

2.9.2 mt90502_receive_aal2_mini_pkt

This function retrieves the oldest AAL2 mini-packet from the RX CPU mini-packet buffer.

Note that a mini-packet of a channel will be routed to the RX CPU mini-packet buffer only for certain values of the packets UUI field (see rx_uui_mapping field of mt90502_map_cid).

The mt90502_receive_aal2_mini_pkt_def function inserts default values into the MT90502_AAL2_RCV_MINI_PKT structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_receive_aal2_mini_pkt_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_AAL2_RCV_MINI_PKT* paal2_rcv_mini_pkt );

ULONG mt90502_receive_aal2_mini_pkt(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_AAL2_RCV_MINI_PKT* paal2_rcv_mini_pkt );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_RX_MINI_PKT_BUFFER_EMPTY`
when there are no packets in the RX CPU mini-packet buffer. The returned structure is invalid.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
pointer to an instance structure of the chip

paal2_rcv_mini_pkt
pointer to an MT90502_AAL2_RCV_MINI_PKT structure containing the received mini-packet and associated information. The user allocates this structure. The definitions of the structure's elements are listed below.

2.9.2.1 Structure MT90502_AAL2_RCV_MINI_PKT

reset_buffers TRUE / FALSE

If set to TRUE, the hardware and software buffers for the mini packets will be emptied. When set to TRUE the function will not return a mini packet, and more_packets will be set to FALSE.

Direction: IN Type: ULONG

Default: FALSE

cid_hndl handle

The handle indicating the mini packet stream on which the mini-packet was received.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header 32 bit field

The ATM cell header defined in mt90502_open_aal2_vc for the VC on which the mini-packet was received. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: OUT Type: ULONG

Default: MT90502_NULL_HEADER

utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

The UTOPIA port that the mini-packet was received on.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

more_packets TRUE / FALSE

True if there are more packets buffered to be read.

Direction: OUT Type: ULONG

Default: FALSE

cid 8 bit field

The CID value of the received mini-packet.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

li 6 bit field

The LI value of the received mini-packet.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_LI

uui 5 bit field

The UUI value of the received mini-packet.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_UUI

payload[16] 16 element array of 32 bit fields

An array of the 64 payload bytes of the received mini-packet. The payload bytes of the mini-packet are arranged in the array as follows:

	b31 - b24	b23 - b16	b15 -b8	b7 - b0
payload[0]	payload byte 0	payload byte 1	payload byte 2	payload byte 3
payload[1]	payload byte 4	payload byte 5	payload byte 6	payload byte 7
…	…	…	…	…
payload[15]	payload byte 60	payload byte 61	payload byte 62	payload byte 63

Only the payload bytes up to, and including payload byte “LI” are valid.

Direction: OUT Type: ULONG[16]

Default: 0

2.9.3 mt90502_send_cas_pkt

This function creates a Type 3 packet to transmit the user specified CAS value. The type 3 packet is inserted on a CID connection indicated by the user. The API will insert the LI, Redundancy, Time Stamp, and CRC-10 fields as required per ITU 366.2. The API will also send redundant and/or refresh packets as specified in ITU 366.2.

If a CID has not finished to send the 3 redundant packets of a redundancy sequence and an attempt is made to send a new CAS value, the old CAS value would not be sent anymore and the new CAS value will be sent, in order to avoid the interleaving of two different time stamps.

The mt90502_send_cas_pkt_def function inserts default values into the MT90502_SEND_CAS_PKT structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_send_cas_pkt_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_SEND_CAS_PKT* paal2_send_cas_pkt );

ULONG mt90502_send_cas_pkt (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_SEND_CAS_PKT* paal2_send_cas_pkt );
```

Return Values

MT90502ER_GENERIC_OK
Indicates success.

MT90502ER_SEND_CAS_PKT_BUFFER_FULL
there is no room in the send CAS packet buffer. The packet was not queued to be sent.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
pointer to an instance structure of the chip

paal2_send_cas_pkt
pointer to an MT90502_SEND_CAS_PKT structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.9.3.1 Structure MT90502_SEND_CAS_PKT

cid_hndl handle

Indicates the mini packet-stream on which the CAS mini-packet is to be sent.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

cas_redundancy_interval MT90502_NO_REDUNDANCY
MT90502_5MS_ INTERVAL
MT90502_10MS_ INTERVAL

MT90502_15MS_ INTERVAL
MT90502_20MS_ INTERVAL

If not set to MT90502_NO_REDUNDANCY then the CAS mini-packet will be sent with triple redundancy with the specified time interval separating each packet transmission. The triple redundancy is represented by an incrementing value from 0 to 2 in the redundancy field of each redundant packet. If MT90502_NO_REDUNDANCY is selected, only one mini-packet will be sent with a redundancy field of 3.

Direction: IN Type: ULONG

Default: MT90502_NO_REDUNDANCY

cas_refresh MT90502_NO_REFRESH
MT90502_REFRESH

Determines if the CAS packet is to be refreshed. If set to MT90502_REFRESH the API will queue a CAS mini-packet (with a redundancy field of 3 and proper timestamp) once per refresh cycle (See mt90502_cas_refresh function). This CAS is refreshed until: this function is used to send a different CAS packet on the same CID, the mt90502_cancel_cas function is used for this CID, the function is canceled by the mt90502_map_cid function or the CID is closed. If set to MT90502_NO_REFRESH no refresh CAS packets will be transmitted.

Direction: IN Type: ULONG

Default: MT90502_NO_REFRESH

cas 4 bit field

The CAS value to be sent in the CAS packet.

Direction: IN Type: ULONG

Default: MT90502_INVALID_CAS

2.9.4 mt90502_cas_refresh

This function is called periodically by the user application. It is used to refresh the state of all CAS packets for which the refresh was specified in a mt90502_send_cas_pkt function. The time between refreshes is determined by the interval at which this function completes cycles.

Thus, this function must be called regularly to maintain a periodic refresh interval. The CAS packets should be refreshed at a 5 sec. interval per ITU-T 366.2.

The user must specify the maximum time the function can execute. The function will return when the allocated execution time (see max_refresh_time parameter) has expired, or once all queued CAS packets has been refreshed (see update_complete parameter). The typical usage is to initiate a cycle as a background operation every 5 seconds with a reasonably short execution time and call it repeatedly until update_complete is TRUE.

The mt90502_cas_refresh_def function inserts default values into the fields of the MT90502_CAS_REFRESH structure. The default value of a structure field is indicated below the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_cas_refresh_def (
            MT90502_INSTANCE_API *pmt90502_api,
            MT90502_CAS_REFRESH *prefresh );
```

```
ULONG mt90502_cas_refresh (
            MT90502_INSTANCE_API *pmt90502_api,
            MT90502_CAS_REFRESH *prefresh );
```

Return Values

MT90502ER_GENERIC_OK	Indicates success

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api

a pointer to the MT90502_INSTANCE_API structure of the chip for which the refresh CAS packets are to be sent.

prefresh

a pointer to an MT90502_CAS_REFRESH structure. The definitions of the elements of the structure are provided below.

2.9.4.1 Structure MT90502_CAS_REFRESH

max_refresh_time 0 – 1048575 ms

The maximum time in ms this function will run. The refresh function will return when one cycle is complete (i.e. update_complete=TRUE) or this time is expired. The function will begin refreshing where it left off in the cycle on subsequent calls.

Direction: IN Type: ULONG

Default: 500

refresh_complete TRUE / FALSE

If TRUE a cycle of refreshes (i.e. all packets have been refreshed once) has been completed during this function execution. If FALSE not all refreshes were completed in the allowed max_refresh_time. A subsequent call will return TRUE when a cycle has been completed. Once a cycle has been completed a new cycle will be started on the next call of mt90502_cas_refresh.

Direction: OUT Type: ULONG

Default: FALSE

2.9.5 mt90502_cancel_cas

This function cancels the CAS refresh packet transmissions by the API if they are currently enabled. See cas_refresh parameter of the mt90502_send_cas_pkt function.
The mt90502_cancel_cas_def function inserts default values into the MT90502_CANCEL_CAS structure. The default value of a structure field is indicated following the field’s description.

Usage

```
#include “mt90502_api.h”

ULONG mt90502_cancel_cas_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CANCEL_CAS* paal2_cancel_cas );
```

```
ULONG mt90502_cancel_cas (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CANCEL_CAS* paal2_cancel_cas );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success.

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api

pointer to an instance structure of the chip

paal2_cancel_cas

pointer to an MT90502_CANCEL_CAS structure. The user allocates this structure. The definitions of the structure’s elements are listed below.

2.9.5.1 Structure MT90502_CANCEL_CAS

cid_hndl handle

Indicates the mini packet-stream on which CAS mini-packet refreshing is to be canceled.

Direction: IN Type: ULONG

Default: MT90502_INVALID_HANDLE

2.9.6 mt90502_receive_cid_event

This function retrieves the oldest CID event from the RX CID event buffer. CID events are processed by the API only when enable_cas is set to TRUE in the MT90502_CONF structure. The API will perform will present only one event to the user when voice is first received or the CAS value changes.

A CID can generate one of two events: the reception of a CAS packet (UUI 24 and LI 4) or the reception of a voice packet (UUIs 0 – 15). The events are generated only if these packets are routed to the RX CPU mini-packet buffer for processing. For the API to generate these events on unopen CIDs the default rx UUI mapping given to all unopen CIDs must be set to route the mini-packets to the RX CPU mini-packet buffer (see default_rx_uui_mapping parameter in the MT90502_CONF structure). To generate the events for open CIDs the UUIs 0-15 and 24 must be mapped to the RX CPU mini-packet buffer (see the rx_uui_mapping parameter of the MT90502_OPEN_CID and MT90502_MAP_CID structures).

If UUIs 0 – 15 are routed to the RX CPU mini-packet buffer for the API to generate an event corresponding to the first voice packet subsequent voice packets will also generate events. To reduce the traffic reaching the buffer the routing of UUIs 0 – 15 the API can be configured to modify the routing automatically upon voice packet reception to discard all mini-packets following the first received mini-packet. For closed CIDs this is specified by the rx_voice_one_only parameter of the MT90502_CONF structure. For open CIDs it is specified by the rx_voice_one_only parameter of the MT90502_OPEN_CID and MT90502_MAP_CID structures. Note that once the CID is open and mapped to channel(s), an RX channel, the voice stream routing will not be changed by the API.

The mt90502_receive_cid_event_def function inserts default values into the MT90502_RCV_CID_EVENT structure. The default value of a structure field is indicated following the field’s description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_receive_cid_event_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_RCV_CID_EVENT* prcv_cid_event );

ULONG mt90502_receive_cid_event (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_RCV_CID_EVENT* prcv_cid_event );
```

Return Values

MT90502ER_GENERIC_OK
: Indicates success

MT90502ER_RX_CID_EVENT_BUFFER_EMPTY
: when there are no events in the RX CID event buffer. The returned structure is invalid.

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api
: pointer to an instance structure of a chip

prcv_cid_event
: pointer to an MT90502_RCV_CID_EVENT structure containing the generated CID event and associated information. The user allocates this structure. The definitions of the structure's elements are listed below.

2.9.6.1 Structure MT90502_RCV_CID_EVENT

reset_buffers TRUE / FALSE

If set to TRUE, the software buffer for the CID events will be emptied. When set to TRUE the function will not return a CID event, and more_events will be set to FALSE.

Direction: IN Type: ULONG

Default: FALSE

cid_hndl handle

The handle indicating the mini packet stream on which the event was generated. If the received CID is closed on the VC receiving the packet for this event, the value will be MT90502_INVALID_HANDLE

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_hndl handle

The handle of the VC on which the packet generating this event was received.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_HANDLE

vc_header 32 bit field

The ATM cell header defined in mt90502_open_aal2_vc for the VC on which the mini-packet was received. Header fields are in the following order (starting from bit 31): GFC, VPI, VCI, PT, CLP.

Direction: OUT Type: ULONG

Default: MT90502_NULL_HEADER

utopia_port MT90502_PORTA
MT90502_PORTB
MT90502_PORTC

The UTOPIA port that the packet generating the event was received on.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_UTOPIA_PORT

more_events TRUE / FALSE

True if there are more packets buffered to be read.

Direction: OUT Type: ULONG

Default: FALSE

cid 8 bit field

The CID value of the received packet generating this event.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CID

cas_voice MT90502_CAS_EVENT
MT90502_VOICE_EVENT

Indicates if this event was generated by a received CAS packet or by the reception of voice in the CPU mini-packet FIFO. If MT90502_VOICE_EVENT then the value of old_cas_value and new_cas_value will be MT90502_INVALID_CAS. If MT90502_CAS_EVENT then the parameter new_cas_value contains the CAS bits of the received CAS packet, and the parameter old_cas_value contains the previously received CAS bits if the CID was open when they were received otherwise it will be MT90502_INVALID_CAS.

Direction: OUT Type: ULONG

Default: FALSE

new_cas_value MT90502_INVALID_CAS,
4 bit field

The CAS bits received in the CAS packet. This parameter will be invalid (MT90502_INVALID_CAS) if cas_voice is set to MT90502_VOICE_EVENT.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CAS

old_cas_value MT90502_INVALID_CAS,
4 bit field

The CAS bits of the previously received CAS packet. This parameter will be invalid (MT90502_INVALID_CAS) if cas_voice is set to MT90502_VOICE_EVENT or the CID was not open during the reception of the prior CAS value.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_CAS

2.10 GPIO Functions

2.10.1 mt90502_set_gpio_value

This function sets the output enable of gpio_pin to gpio_oe and the driven value of gpio_pin to gpio_value.

The mt90502_set_gpio_value_def function inserts default values into the MT90502_SET_GPIO_PARMS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_set_gpio_value_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_SET_GPIO_PARMS* pset_gpio_parms );

ULONG mt90502_set_gpio_value( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_SET_GPIO_PARMS* pset_gpio_parms );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an API instance structure of the chip

pset_gpio_parms

pointer to an MT90502_SET_GPIO_PARMS structure. The definitions of the structure's elements are listed below.

2.10.1.1 Structure MT90502_SET_GPIO_PARMS

gpio_pin

MT90502_GPIO_PIN_0
MT90502_GPIO_PIN_1
. . .
MT90502_GPIO_PIN_7

MT90502_CT_NETREF_1
MT90502_CT_NETREF_2

The pin to which the output enable and value are to be set.

Direction: IN Type: ULONG

Default: MT90502_INVALID_GPIO

gpio_oe TRUE/FALSE

If TRUE the output will be enabled. If FALSE the pin is tri-stated.

Direction: IN Type: ULONG

Default: FALSE

gpio_value 0 / 1

The value to be driven on the pin specified by gpio_pin. If the pin output is not enabled by gpio_oe, it will remain tri-stated regardless of the value set.

Direction: IN Type: ULONG

Default: 0

2.10.2 mt90502_get_gpio_value

This function returns the value, as well as the rise and fall values of the GPI/GPIO specified by gpio_pin.

The mt90502_get_gpio_value_def function inserts default values into the MT90502_GET_GPIO_PARMS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_api.h"

ULONG mt90502_get_gpio_value_def( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_GET_GPIO_PARMS* pget_gpio_parms );

ULONG mt90502_get_gpio_value( MT90502_INSTANCE_API* pmt90502_api,
            MT90502_GET_GPIO_PARMS* pget_gpio_parms );
```

Return Values

`MT90502ER_GENERIC_OK` Indicates success

Also see Section 4.0 "Return Codes" for non-successful codes.

Parameters

pmt90502_api

pointer to an API instance structure of the chip

pget_gpio_parms

pointer to an MT90502_GET_GPIO_PARMS structure. The user allocates this structure. The definitions of the structure's elements are listed below.

2.10.2.1 Structure MT90502_GET_GPIO_PARMS

gpio_pin

MT90502_GPIO_PIN_0
MT90502_GPIO_PIN_1
...
MT90502_GPIO_PIN_7

MT90502_CT_NETREF_1
MT90502_CT_NETREF_2

Specifies the pin to be sampled.

Direction: IN Type: ULONG

Default: MT90502_INVALID_GPIO

input_value 0 / 1

The value currently received on gpio_pin.

Direction: OUT Type: ULONG

Default: 0

rise TRUE / FALSE

If TRUE, the pin has transitioned from ‘0’ to ‘1’ since the last time this function was called.

Direction: OUT Type: ULONG

Default: FALSE

fall TRUE / FALSE

If TRUE, the pin has transitioned from ‘1’ to ‘0’ since the last time this function was called.

Direction: OUT Type: ULONG

Default: FALSE

2.11 Clock Recovery Functions

2.11.1 mt90502_get_adaptive_clk_recov_pnt

This function retrieves a clock recovery point from the API’s soft buffer. An adaptive clock recovery point can be retrieved from either the A or B buffers. This function can be used to create a Stratum 4 enhanced clock signal with the MT90502.

The mt90502_get_adaptive_clk_recov_pnt_def function inserts default values into the MT90502_ADAP_PNT structure. The default value of a structure field is indicated following the field’s description.

Usage

```
#include “mt90502_api.h”

ULONG mt90502_get_adaptive_clk_recov_pnt_def (
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_ADAP_PNT* padap_pnt );

ULONG mt90502_get_adaptive_clk_recov_pnt(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_ADAP_PNT* padap_pnt );
```

Return Values

MT90502ER_GENERIC_OK
Indicates success

MT90502ER_ADAPTIVE_CLK_RECOV_BUFFER_EMPTY
Indicates that the selected buffer is empty.

Also see Section 4.0 “Return Codes” for non-successful codes.

Parameters

pmt90502_api
pointer to an API instance structure of the chip

padap_pnt

pointer to an MT90502_ADAP_PNT structure. The definitions of the structure's elements are listed below.

2.11.1.1 Structure MT90502_ADAP_PNT

reset_buffers TRUE/FALSE

If set to TRUE, the specified hardware and software buffers for the clock recovery points will be emptied. When set to TRUE the function will not return a point, and more_points will be set to FALSE. Which hardware and software buffers will be emptied is indicated by buffer_select.

Direction: IN Type: ULONG

Default: FALSE

buffer_select MT90502_ADAP_BUF_A
MT90502_ADAP_BUF_B

Indicates which hardware and software buffers the function is to access.

Direction: IN Type: ULONG

Default: MT90502_ADAP_BUF_A

mem_clk_i_cnt 32 bit counter

Value of free running mem_clk_i_cycle counter.

Direction: OUT Type: ULONG

Default: 0

ref_cnt 32 bit counter

Value of free running reference clock cycle counter.

Direction: OUT Type: ULONG

Default: 0

mem_clk_i_last_ref_inc 16 bit counter

The number of mem_clk_i cycles since the last increment of the reference clock count.

Direction: OUT Type: ULONG

Default: 0

uui 5 bit counter

If the clock recovery buffer is sourced by either an RX xxPCM channel or an RX HDLC channel, this field is the value of the UUI of the mini packet which generated the clock recovery point. This field is not used in the case where the buffer is sourced by a VC.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_UUI

li 5 bit counter

If the clock recovery buffer is sourced by either an RX xxPCM channel or an RX HDLC channel, this field is the value of the LI of the mini packet which generated the clock recovery point. This field is not used in the case where the buffer is sourced by a VC.

Direction: OUT Type: ULONG

Default: MT90502_INVALID_LI

more_points TRUE/FALSE

Indicates whether there are more clock recovery points of the specified type pending in either the chip's buffer or the soft buffer maintained by the API.

Direction: IN Type: ULONG

Default: FALSE

2.12 Interrupt Functions

See System Architecture for the flow of interrupt treatment.

The interrupts are divided into five categories:

Fatal – indicates that the chip has encountered a fatal error, and must be reset to operate correctly once again.

Data Error – indicates that the chip has detected an error that leads to bad data integrity. The chip and all connections will continue to operate.

Error – indicates that the chip has detected an error that must be handled by the user application. There is no recovery required by the chip, the severity and or recovery, if any, can only be determined by the application.

H100 Error – indicates an error with the H100 bus clock or frame signals. If the bus is configured for recovery, it will happen automatically. Once a recovery has taken place the set_h100_master or set_h100_slave functions must be called to re-establish a recovery configuration. For example if it H100 were originally configured as MT90502_H100_TRACKA_FALLBACKB and subsequently "falls back" to B, once A is repaired the set_h100_slave function may be called with MT90502_H100_TRACKB_FALLBACKA to "re-arm" auto recovery.

If auto recovery is not configured these errors flag a continuous data integrity disruption on all voice connections if it is indicated on the current master clock or frame signal. To attempt recovery of the H100 bus the user must call either the set_h100_master or set_h100_slave functions.

HDLC Error – indicates an error in HDLC packets received in the TX direction (i.e. received from the H.100 bus).

Alarm – Alarms indicate an event has occurred like reception of a mini packet in the CPU mini packet receive buffer.

API Sync – this interrupt is used by the API to maintain synchronization with the chip. This is provided for information only and there is no user action required. If disabled the API ISR must be called a minimum of every 20 sec to prevent complete loss of synchronization which can lead to system failure if the API attempts any operations against the device. (i.e. upon loss of synchronization, channels will remain open and functioning but the API will no longer be able to open or close channels without the risk of corrupting the chip.)

The category to which an interrupt belongs is indicated by the interrupt's name's prefix.

The behavior of all of the chip's interrupts can be configured independently. An interrupt can be enabled or disabled. In the case where an interrupt is enabled, the interrupt can behave in one of three ways once it is active. The interrupt can remain active and will not be reset by the APIISR. The interrupt can be kept enabled and be reset immediately by the APIISR. Or, the interrupt can be reset and temporarily disabled by the APIISR. See Interrupt Configuration Parameters.

For the alarms, an additional interrupt configuration mode exists. The interrupt can be disabled, and the servicing of the FIFO to which the alarm is tied can also be disabled. That is, the APIISR will not service the FIFO, regardless of the state of the FIFO.

2.12.1 mt90502_interrupt_service_routine

It is to be called by the user provided function mt90502_access_apiisr to service interrupts. This function lies in the APIISR code entity (see System Architecture). Because this function can be called by both the OS ISR and the API code entity, accesses to the function must be serialized. This function will take the appropriate action to treat any active interrupts when called by the OS ISR.

All interrupts are enabled by the user via the mt90502_open function call. Disabled interrupts are still serviced by this routine but they will not generate a hardware interrupt on the interrupt pin of the device.

If the user wants to create an entirely polled system, all interrupts can be set to "disabled" and the user becomes responsible for calling this routine often enough for proper operation of the device.

This function will reset all conditions causing the interrupt such that the interrupt pin typically will be inactive when it returns.

The mt90502_interrupt_service_routine_def function can be used by the OS ISR to request typical APIISR operation. The function will insert the appropriate values into the fields of the MT90502_INT_STRUCT structure.

Usage

```
#include "mt90502_apiisr.h"

void mt90502_interrupt_service_routine(
            MT90502_INSTANCE_APIISR* pmt90502_apiisr,
            MT90502_INT_STRUCT* pint_strct );

void mt90502_interrupt_service_routine_def(
            MT90502_INSTANCE_APIISR* pmt90502_apiisr,
            MT90502_INT_STRUCT* pint_strct );
```

Parameters

pmt90502_apiisr

a pointer to the MT90502_INSTANCE_APIISR structure of the chip to be serviced.

pint_strct

a structure indicating the type of servicing to be performed. The parameter is used to differentiate between OS ISR calls and various API calls to this function. The API may need to perform tasks which are normally performed by the interrupt service routine. For example, the API may need to empty the chip's RX data cell FIFO. Or, the API may need a resource which is kept in the APIISR instance structure (for example, retrieving a data cell from the software FIFO). Thus, this parameter permits the API to force certain operations to be performed by the ISR. Other parameters within the structure are used to provide information needed by the API ISR to perform an operation. These parameters are only used by the API code entity. The definitions of the structure's elements are supplied below.

2.12.1.1 Structure MT90502_INT_STRUCT

The structure is composed of two sub-structures. The first is the structure used by the API to access the APIISR block via a communication pipe (see mt90502_access_apiisr). The second is used to retrieve indications, from the APIISR block, of events which were flagged by the chip.

ppipe_strct MT90502_PIPE_STRUCT

See structure MT90502_PIPE_STRUCT.

Direction: IN/IO Type: POINTER

Default: NULL

pint_flags MT90502_INT_FLAGS

Pointer to a structure indicating the events flagged by the chip. See structure MT90502_INT_FLAGS.

Direction: IN, IN/IO Type: POINTER

Default: NULL

2.12.1.2 Structure MT90502_INT_FLAGS

The following parameters indicate what events were detected during the operation of the ISR.

fatal_general TRUE / FALSE

If TRUE an internal fatal chip error has been detected.

Direction: OUT Type: ULONG

Default: FALSE

fatal_ssrama_parity TRUE / FALSE

If TRUE a parity error has been detected on bank A of the SSRAM interface.

Direction: OUT Type: ULONG

Default: FALSE

fatal_ssramb_parity TRUE / FALSE

If TRUE a parity error has been detected on bank B of the SSRAM interface.

Direction: OUT Type: ULONG

Default: FALSE

fatal_sdrama_parity TRUE / FALSE

If TRUE a parity error has been detected on bank A of the SDRAM interface.

Direction: OUT Type: ULONG

Default: FALSE

fatal_sdramb_parity TRUE / FALSE

If TRUE a parity error has been detected on bank B of the SDRAM interface.

Direction: OUT Type: ULONG

Default: FALSE

data_err_sdrama_too_late TRUE / FALSE

If TRUE bank A of the SDRAM may have exceeded the configured refresh period. Information in the bank may be corrupt.

Direction: OUT Type: ULONG

Default: FALSE

data_err_sdramb_too_late TRUE / FALSE

If TRUE bank B of the SDRAM may have exceeded the configured refresh period. Information in the bank may be corrupt.

Direction: OUT Type: ULONG

Default: FALSE

data_err_utopia_parity_a TRUE / FALSE

If TRUE a parity error has been detected on the receive direction of UTOPIA port A.

Direction: OUT Type: ULONG

Default: FALSE

data_err_utopia_parity_b TRUE / FALSE

If TRUE a parity error has been detected on the receive direction of UTOPIA port B.

Direction: OUT Type: ULONG

Default: FALSE

data_err_utopia_parity_c TRUE / FALSE

If TRUE a parity error has been detected on the receive direction of UTOPIA port C.

Direction: OUT Type: ULONG

Default: FALSE

error_phy_alarm_a TRUE / FALSE

If TRUE PHY device A has generated an alarm via the phya_alm pin.

Direction: OUT Type: ULONG

Default: FALSE

error_phy_alarm_b TRUE / FALSE

If TRUE PHY B device has generated an alarm via the phyb_alm pin.

Direction: OUT Type: ULONG

Default: FALSE

error_rxsar_cell_loss TRUE / FALSE

If TRUE then one or many cells have been lost at the internal RX SAR cell FIFO of the UTOPIA module due to an overflow.

Direction: OUT Type: ULONG

Default: FALSE

error_txa_cell_loss TRUE / FALSE

If TRUE then one or many cells have been lost at the internal TX A cell FIFO of the UTOPIA module due to an overflow.

Direction: OUT Type: ULONG

Default: FALSE

error_txb_cell_loss TRUE / FALSE

If TRUE then one or many cells have been lost at the internal TX B cell FIFO of the UTOPIA module due to an overflow.

Direction: OUT Type: ULONG

Default: FALSE

error_txc_cell_loss TRUE / FALSE

If TRUE then one or many cells have been lost at the internal TX C cell FIFO of the UTOPIA module due to an overflow.

Direction: OUT Type: ULONG

Default: FALSE

error_mini_pkt_fifo TRUE / FALSE

If TRUE the receive mini packet FIFO in the external control memory has overflowed, causing some mini packets to be lost. Note: This may include CAS packets.

Direction: OUT Type: ULONG

Default: FALSE

error_rx_data_cell_fifo TRUE / FALSE

If TRUE the RX data cell FIFO in the external control memory has overflowed, causing some data cells to be lost.

Direction: OUT Type: ULONG

Default: FALSE

error_rx_event_fifo TRUE / FALSE

If TRUE the RX event FIFO in the external control memory has overflowed, causing some statistics to not be updated and one or more mini packets or LI or UUI change indications may have been lost.

Direction: OUT Type: ULONG

Default: FALSE

error_adap_a_fifo TRUE / FALSE

If TRUE the adaptive clk recovery A FIFO in the external control memory has overflowed, causing some clk recovery points to be lost.

Direction: OUT Type: ULONG

Default: FALSE

error_adap_b_fifo TRUE / FALSE

If TRUE the adaptive clk recovery B FIFO in the external control memory has overflowed, causing some clk recovery points to be lost.

Direction: OUT Type: ULONG

Default: FALSE

h100_error_out_of_sync TRUE / FALSE

If TRUE the H.100 slave has lost its framing on the bus, causing the chip's H.100 data pins to be tri-stated.

Direction: OUT Type: ULONG

Default: FALSE

h100_error_clk_a TRUE / FALSE

If TRUE the clock CT_C8_A is not behaving in accordance to the H100 specification.

Direction: OUT Type: ULONG

Default: FALSE

h100_error_clk_b TRUE / FALSE

If TRUE the clock CT_C8_B is not behaving in accordance to the H100 specification.

Direction: OUT Type: ULONG

Default: FALSE

h100_error_frame_a TRUE / FALSE

If TRUE the clock CT_FRAME_A is not behaving in accordance to the H100 specification.

Direction: OUT Type: ULONG

Default: FALSE

h100_error_frame_b TRUE / FALSE

If TRUE the clock CT_FRAME_B is not behaving in accordance to the H100 specification.

Direction: OUT Type: ULONG

Default: FALSE

hdlc_error_misaligned_flag TRUE / FALSE

If TRUE an HDLC packet was received with a misaligned flag (i.e. not on a byte boundary).

Direction: OUT Type: ULONG

Default: FALSE

hdlc_error_bad_idle_code TRUE / FALSE

If TRUE an HDLC packet was received with an idle code in the middle of the packet.

Direction: OUT Type: ULONG

Default: FALSE

hdlc_error_long_packet TRUE / FALSE

If TRUE an HDLC packet was received with a packet length greater than 64 bytes.

Direction: OUT Type: ULONG

Default: FALSE

hdlc_error_short_packet TRUE / FALSE

If TRUE an HDLC packet was received with 0 data bytes.

Direction: OUT Type: ULONG

Default: FALSE

alarm_li_uui_change_event TRUE / FALSE

If TRUE an LI or UUI change event has been detected and is pending in the soft LI change buffer (see soft_li_uui_change_buffer_size).

Direction: OUT Type: ULONG

Default: FALSE

alarm_mini_pkt_rcvd_event TRUE / FALSE

If TRUE a mini packet has been received and is pending in the soft RX mini packet buffer (see soft_rx_mini_pkt_buffer_size).

Direction: OUT Type: ULONG

Default: FALSE

alarm_cid_event TRUE / FALSE

If TRUE a CID event has occurred and is pending in the CID event FIFO (see mt90502_receive_cid_event).

Direction: OUT Type: ULONG

Default: FALSE

alarm_data_cell_fifo TRUE / FALSE

If TRUE a data cell has been received and is pending in the soft RX data cell buffer (see soft_rx_data_buffer_size).

Direction: OUT Type: ULONG

Default: FALSE

api_sync TRUE / FALSE

If TRUE the chip interrupted for purposes of maintaining synchronization with the API.

Direction: OUT Type: ULONG

Default: FALSE

2.12.2 mt90502_mask_interrupt

This function is to be used by the operating system's interrupt service routine. This function disables the chip's interrupt pin. When the chip generates an interrupt, the OS starts its interrupt service routine (see System Architecture). The API's ISR must be called to treat the interrupt. Either the OS calls the API's ISR directly from its ISR, or it defers the treatment of the ISR to a later time, and at a lower CPU priority level. In the latter case, the interrupt pin of the chip must be disabled until the current interrupt has been treated. This function serves this purpose. The function first performs a read to the chip's interrupt register to determine if the chip is the source of the interrupt (many devices can share the same interrupt line). If the chip is the source of the interrupt, the function performs a single read and write to the chip's interrupt register, which disables the interrupt pins from generating another interrupt for up to 16 ms. After the disable timer has expired the interrupt pin will function normally. If the conditions causing the original interrupt still exist or a new event has occurred, the chip will interrupt immediately when the timer expires. The API's ISR will re-enable the interrupt pin when it completes allowing new interrupts to occur in potentially less than 16ms.

The mt90502_mask_interrupt_def function inserts default values into the MT90502_MASK_INT_PARMS structure. The default value of a structure field is indicated following the field's description.

Usage

```
#include "mt90502_apimi.h"

ULONG mt90502_mask_interrupt_def(MT90502_MASK_INT_PARMS* pmask_int_parms );

ULONG mt90502_mask_interrupt( MT90502_MASK_INT_PARMS* pmask_int_parms );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success.

`MT90502ER_INT_NOT_ACTIVE`
Indicates that the interrupt of the pin was not active.

`MT90502ER_INT_RW_ERROR`
Indicates that an error occurred while trying to read from / write to the chip.

Parameters

pmask_int_parms pointer to an MT90502_MASK_INT_PARMS structure. The definitions of the structure's elements are listed below.

2.12.2.1 Structure MT90502_MASK_INT_PARMS

user_chip_number 0 – ??

The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

Direction: IN Type: ULONG

Default: MT90502_INVALID_CHIP_NUMBER

2.12.3 mt90502_configure_interrupts

This function is used to change the current configuration of interrupt servicing. Before calling this function the mt90502_configure_interrupts_def function should be called. This function will insert the current interrupt configuration into the MT90502_CONF_INTERRUPTS structure. From this, only the fields corresponding to the desired interrupts need be changed.

Usage

```
#include "mt90502_api.h"

void mt90502_configure_interrupts(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONF_INTERRUPTS* pconf_interrupts );

void mt90502_configure_interrupts_def(
            MT90502_INSTANCE_API* pmt90502_api,
            MT90502_CONF_INTERRUPTS* pconf_interrupts );
```

Parameters

pmt90502_api

pointer to the MT90502_INSTANCE_API structure of the chip for which the interrupts are to be reconfigured.

pconf_interrupts

pointer to an interrupt configuration structure. See structure MT90502_CONF_INTERRUPTS.

3.0 User Supplied Function Descriptions

The API functions make all accesses to the physical device through user supplied functions. This gives the user full control over how accesses to the device are accomplished in any given implementation. In order to write these access routines specific implementation details of the device are given here.

Details to be supplied.

The performance of the CPU accesses depends on the cache_cpu_accesses parameter of the MT90502_CONF structure. This parameter is provided to the mt90502_open function call at start up. If this parameter is set to TRUE then no caching of the CPU accesses will be done in the chip. This results in a higher average access time, but a lower worst-case access time. If the parameter is set to FALSE, then the CPU accesses are cached.

3.1 Write Functions

3.1.1 mt90502_driver_write_api, _apiisr, _osisr

Performs a single word write to the chip. Any error returned by this function is considered a fatal error. Two or three versions of the function are needed because the function may be accessed from two or three different software layers depending on the user system architecture, See System Architecture. Thus, each function must have a different name, but the functionality remains identical.

Usage

```
#include "mt90502_apiud.h"

ULONG mt90502_driver_write_api( ULONG user_chip_number,
            MT90502_WRITE_PARMS* pwrite_parms );

ULONG mt90502_driver_write_apiisr(ULONG user_chip_number,
            MT90502_WRITE_PARMS* pwrite_parms );

ULONG mt90502_driver_write_osisr(ULONG user_chip_number,
            MT90502_WRITE_PARMS* pwrite_parms );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_DRIVER_WRITE_FAILED`
return values from 0xFFFF0000-0xFFFF000F are reserved for write routine return values. This return value will be passed to by the API function to the calling user routine. Any error returned by this function is considered a fatal error.

Parameters

user_chip_number
The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

pwrite_parms
pointer to an MT90502_WRITE_PARMS structure. The definitions of the structure's elements are listed below.

3.1.1.1 Structure MT90502_WRITE_PARMS

write_address 0 – 0x007FFFFE

Start address of the word access. This is a byte address that always points to words and must be even.

Direction: IN Type: ULONG

Default: N/A

write_data 16 bit field

The word to be written.

Direction: IN Type: ULONG

Default: N/A

3.1.2 mt90502_driver_write_smear_api, _apiisr, osisr

Performs a write of a data word to multiple addresses of the chip. Any error returned by this function is considered a fatal error. Two or three versions of the function are needed because the function may be accessed from two or three different software layers depending on the user system architecture, See System Architecture. Thus, each function must have a different name, but the functionality remains identical.

Usage

```
#include "mt90502_apiud.h"

ULONG mt90502_driver_write_smear_api( ULONG user_chip_number,
            MT90502_WRITE_SMEAR_PARMS* write_smear_parms );

ULONG mt90502_driver_write_smear_apiisr( ULONG user_chip_number,
            MT90502_WRITE_SMEAR_PARMS* write_smear_parms );

ULONG mt90502_driver_write_smear_osisr( ULONG user_chip_number,
            MT90502_WRITE_SMEAR_PARMS* write_smear_parms );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_DRIVER_WRITE_FAILED`
return values from 0xFFFF0000-0xFFFF000F are reserved for write routine return values. This return value will be passed to by the API function to the calling user routine. Any error returned by this function is considered a fatal error.

Parameters

user_chip_number
The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

pwrite_smear_parms
pointer to an MT90502_WRITE_SMEAR_PARMS structure. The definitions of the structure's elements are listed below.

3.1.2.1 Structure MT90502_WRITE_SMEAR_PARMS

write_address 0 – 0x007FFFFE

Start address of the writes. This is a byte address that always points to words and must be even. This is the address of the first location to write to. For each subsequent word the address is incremented by two.

Direction: IN Type: ULONG

Default: N/A

write_data 16 bit field

The word to be written.

Direction: IN Type: ULONG

Default: N/A

auto_parity TRUE / FALSE

When true the write is to a structure which can generate parity automatically. If TRUE the write_parity pointer is null.

Direction: IN Type: ULONG

Default: N/A

write_parity 2 bit field

The parity bits to be written. This buffer contains the parity bits to be written with the word if required (see auto_parity).

Direction: IN Type: ULONG

Default: N/A

write_length 1 – ??

The number of locations to write the data to. (length in words).

Direction: IN Type: ULONG

Default: N/A

write_increment 1 – ??

The address increment (in bytes) for each successive write to be performed. For example if the initial write_address is 4, the write_increment is 6, and the write_length is 2, the first write will occur at address 4 and the second write will occur at address 10. Note this will always be an even value.

Direction: IN Type: ULONG

Default: N/A

we 2 bit field

The write enable bits to be applied to every write. This is a two-bit field. Bit 1 enables the writing of MSB byte of the data word, and bit 0 enables LSB byte. The enables are active high.

Direction: IN Type: ULONG

Default: N/A

3.1.3 mt90502_driver_write_burst_api, _apiisr, osisr

Performs a write of an array of data words to consecutive addresses of the chip. Any error returned by this function is considered a fatal error. Two or three versions of the function are needed because the function may be accessed from two or three different software layers depending on the user system architecture, See System Architecture. Thus, each function must have a different name, but the functionality remains identical.

Usage

```
#include "mt90502_apiud.h"

ULONG mt90502_driver_write_burst_api( ULONG user_chip_number,
            MT90502_WRITE_BURST_PARMS* write_burst_parms );

ULONG mt90502_driver_write_burst_apiisr( ULONG user_chip_number,
            MT90502_WRITE_BURST_PARMS* write_burst_parms );

ULONG mt90502_driver_write_burst_osisr( ULONG user_chip_number,
            MT90502_WRITE_BURST_PARMS* write_burst_parms );
```

Return Values

MT90502ER_GENERIC_OK
: Indicates success

MT90502ER_DRIVER_WRITE_FAILED
: return values from 0xFFFF0000-0xFFFF000F are reserved for write routine return values. This return value will be passed to by the API function to the calling user routine. Any error returned by this function is considered a fatal error.

Parameters

user_chip_number
: The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

pwrite_burst_parms
: pointer to an MT90502_WRITE_BURST_PARMS structure. The definitions of the structure's elements are listed below.

3.1.3.1 Structure MT90502_WRITE_BURST_PARMS

write_address 0 – 0x007FFFFE

Start address of the writes. This is a byte address that always points to words and must be even. This is the address of the first location to write to. For each subsequent word the address is incremented by two.

Direction: IN Type: ULONG

Default: N/A

write_data The word to be written.

Direction: IN Type: POINTER

Default: N/A

write_length 1 – ??

The number of locations to write the data to. (length in words).

Direction: IN Type: ULONG

Default: N/A

3.2 Read Functions

3.2.1 mt90502_driver_read_api, _apiisr, _osisr

Performs a single word read from the chip. Any error returned by this function is considered a fatal error. Two or three versions of the function are needed because the function may be accessed from two or three different software layers depending on the user system architecture, See System Architecture. Thus, each function must have a different name, but the functionality remains identical.

Usage

```
#include "mt90502_apiud.h"

ULONG mt90502_driver_read_api( ULONG user_chip_number,
            MT90502_READ_PARMS* pread_parms );

ULONG mt90502_driver_read_apiisr( ULONG user_chip_number,
            MT90502_READ_PARMS* pread_parms );

ULONG mt90502_driver_read_osisr( ULONG user_chip_number,
            MT90502_READ_PARMS* pread_parms );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_DRIVER_READ_FAILED`
return values from 0xFFFF0010-0xFFFF001F are reserved for read routine return values. This return value will be passed to by the API function to the calling user routine. Any error returned by this function is considered a fatal error.

Parameters

user_chip_number
The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

pread_parms
pointer to an MT90502_READ_PARMS structure. The definitions of the structure's elements are listed below.

3.2.1.1 Structure MT90502_READ_PARMS

read_address 0 – 0x007FFFFE

Start address of the read. This is a byte address that always points to words and must be even. This is the address of the word to be read.

Direction: IN Type: ULONG

Default: N/A

pread_data Pointer to a single ULONG to receive the read data.

Direction: IN/OUT Type: POINTER

Default: N/A

3.2.2 mt90502_driver_read_burst_api, _apiisr, _osisr

Performs a burst of reads from the chip. Any error returned by this function is considered a fatal error. Two or three versions of the function are needed because the function may be accessed from two or three different software layers depending on the user system architecture, See System Architecture. Thus, each function must have a different name, but the functionality remains identical.

Usage

```
#include "mt90502_apiud.h"

ULONG mt90502_driver_read_burst_api( ULONG user_chip_number,
            MT90502_READ_BURST_PARMS* pread_burst_parms );

ULONG mt90502_driver_read_burst_apiisr( ULONG user_chip_number,
            MT90502_READ_BURST_PARMS* pread_burst_parms );

ULONG mt90502_driver_read_burst_osisr( ULONG user_chip_number,
            MT90502_READ_BURST_PARMS* pread_burst_parms );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_DRIVER_READ_FAILED`
return values from 0xFFFF0010-0xFFFF001F are reserved for read routine return values. This return value will be passed to by the API function to the calling user routine. Any error returned by this function is considered a fatal error.

Parameters

user_chip_number
The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

pread_burst_parms
pointer to an MT90502_READ_BURST_PARMS structure. The definitions of the structure's elements are listed below.

3.2.2.1 Structure MT90502_READ_BURST_PARMS

read_address 0 – 0x007FFFFE

Start address of the burst. This is a byte address that always points to words and must be even. This is the address of the first word in the burst. For each subsequent word the address is incremented by two.

Direction: IN Type: ULONG

Default: N/A

pread_data
Pointer to a list of ULONGs to receive the read data. Each element is one word.

Direction: I N/OUT Type: POINTER

Default: N/A

read_length 1 – ??

Length of the pread_data (burst length in words).

Direction: IN Type: ULONG

Default: N/A

3.2.3 mt90502_driver_read_debug_api, _apiisr, _osisr

Performs a burst of reads from the chip with parity. Any error returned by this function is considered a fatal error. Two or three versions of the function are needed because the function may be accessed from two or three different software layers depending on the user system architecture, See System Architecture. Thus, each function must have a different name, but the functionality remains identical.

Usage

```
#include "mt90502_apiud.h"

ULONG mt90502_driver_read_debug_api( ULONG user_chip_number,
            MT90502_READ_DEBUG_PARMS* read_debug_parms );

ULONG mt90502_driver_read_debug_apiisr(ULONG user_chip_number,
            MT90502_READ_DEBUG_PARMS* read_debug_parms );

ULONG mt90502_driver_read_debug_osisr(ULONG user_chip_number,
            MT90502_READ_DEBUG_PARMS* read_debug_parms );
```

Return Values

`MT90502ER_GENERIC_OK`
Indicates success

`MT90502ER_DRIVER_READ_FAILED`
return values from 0xFFFF0010-0xFFFF001F are reserved for read routine return values. This return value will be passed to by the API function to the calling user routine. Any error returned by this function is considered a fatal error.

Parameters

user_chip_number
The chip identifier parameter provided to the mt90502_open function. (see System Architecture).

pread_debug_parms
pointer to an MT90502_READ_DEBUG_PARMS structure. The definitions of the structure's elements are listed below.

3.2.3.1 Structure MT90502_READ_DEBUG_PARMS

read_address 0 – 0x007FFFFE

Start address of the burst. This is a byte address that always points to words and must be even. This is the address of the first word in the burst. For each subsequent word the address is incremented by two.

Direction: IN Type: ULONG

Default: N/A

pread_data
Pointer to a list of ULONGs to receive the read data. Each element is one word.

Direction: IN/OUT Type: POINTER

Default: N/A

pread_parity

Pointer to a list of ULONGs to receive the associated parity of each word. Each element is 2 bits where the most significant bit is the parity of the associated word's most significant byte.

Direction: IN/OUT Type: POINTER

Default: N/A

read_length 1 – ??

Length of the pread_data and pread_parity buffers (burst length in words).

Direction: IN Type: ULONG

Default: N/A

3.3 Interrupt Service Routine Called From API

The system architecture is depicted in System Architecture Section. As illustrated, the API's interrupt service routine can be accessed by both the OS's interrupt service routine or deferred procedure call, and other functions within the API.

3.3.1 mt90502_access_apiisr

Because the API's ISR is accessed by the OS in kernel space (in the case of the embedded system), the code for the API's ISR must lie in the kernel's space. However, the API in the user's space must also have access to the API's ISR. This routine serves as the bridge for the API from user space to kernel space.

The API calls this function, and passes to it a chip number (user_chip_number) and a pointer to a structure (ppipe_strct). This structure must be passed to the interrupt service routine, along with the pointer to the APIISR structure corresponding to the chip number provided. The APIISR instance structure pointer, can be retrieved from an instance structure pointer array maintained by the user, using the chip number as the index. For the structure passed via the pointer pint_strct, one of two actions must be taken:

If the API's ISR is located in user space, then the memory pointed to by ppipe_strct is accessible, and the pointer can be passed directly.

If the API's ISR is located in kernel space, then the memory pointed to by ppipe_strct is inaccessible because it points to memory in the user space. In this case, the contents of the structure are copied into a new kernel-space memory buffer. The pointer to this new buffer is passed to the interrupt service routine.

The definitions of the contents of the structure pointed to by pint_strct are provided in MT90502_INT_STRUCT Structure. Within the MT90502_INT_STRUCT structure is a void pointer pint_buf. This pointer points to a buffer of contiguous memory. The size of the buffer is indicated by pipe_buf_size (in bytes). The contents of this buffer must also be passed (copied if necessary) to the interrupt service routine.

Before the function terminates, the contents of the kernel space buffer must be copied back to the user space buffer passed to this function.

Some sample code implementing this piping mechanism follows:

```
ULONG mt90502_access_apiisr( ULONG user_chip_number,
                    MT90502_PIPE_STRUCT* puser_pipe_strct)
{
        MT90502_INSTANCE_APIISR*pmt90502_apiisr;
        MT90502_PIPE_STRUCT     kernel_pipe_strct;
        void*                   puser_int_buf;
        …
```

```
        // Copy the contents of the structure if the interrupt service routine is located in
        // kernel space.

        memcpy(&kernel_pipe_strct, puser_pipe_strct,
                sizeof(MT90502_PIPE_STRUCT));

        // Copy the contents of the buffer pointer within the structure if the API's ISR is
        // located in kernel space.

        if (puser_pipe_strct->ppipe_buf_size > 0)
        {
                kernel_pipe_strct.ppipe_buf = malloc(puser_pipe_strct->ppipe_buf_size);
                memcpy(kernel_pipe_strct.ppipe_buf,
                        puser_pipe_strct->ppipe_buf,
                        puser_pipe_strct->ppipe_buf_size);
        }
        …
        // Select the corresponding APIISR instance structure according to chip number.
        pmt90502_apiisr = user_pointer_array[user_chip_number];
        …
        // Call the serialization function. This function will then call the API's ISR.
        apiisr_serialization(pmt90502_apiisr, pkernel_pipe_buf);

        // The function has returned, so copy back buffers to user space memory buffer.
        // Keep a copy of the pointer to the user's void buffer pointer.
        puser_pipe_buf = puser_pipe_strct->ppipe_buf;
        memcpy(puser_pipe_strct, &kernel_pipe_strct,
                        sizeof(MT90502_PIPE_STRUCT));
        puser_pips_strct->pint_buf = puser_pips_buf;
        if (kernel_pips_strct.pipe_buf_size > 0)
        {
                memcpy(puser_pipe_strct->ppipe_buf,
                        kernel_pipe_strct.ppipe_buf,
                        kernel_pipe_strct.ppipe_buf_size);
                free(kernel_pipe_strct.ppipe_buf);
        }

}
```

Once the buffers are copied, the mt90502_serialize_interrupt_service_routine function must be called, which in turn calls the API's ISR. See System Architecture.

Usage

```
#include "mt90502_apiud.h"

void mt90502_run_interrupt_service_routine(ULONG user_chip_number,
            MT90502_PIPE_STRUCT* ppipe_strct );
```

Parameters

user_chip_number The chip identifier parameter provided to the mt90502_open function. (see System Architecture)

ppipe_strct Pointer to an MT90502_PIPE_STRUCT structure indicating what servicing is to be performed by the APIISR. The definitions of the structure's fields are provided below.

3.3.1.1 Structure MT90502_PIPE_STRUCT

The following parameters are used to determine what operations are to be performed by the API's ISR.

isr_type MT90502_ISR_TYPE_NORMAL

Must be set to MT90502_ISR_TYPE_NORMAL by the OS ISR calling this function.

Direction: IN Type: ULONG

Default: MT90502_ISR_TYPE_NORMAL

result

This field is used by the APIISR block to return the error code of the functions performed by it due to this pipe message. This field is set to MT90502ER_GENERIC_OK if no errors occurred.

Direction: OUT Type: ULONG

Default: MT90502ER_GENERIC_OK

ppipe_buf

Pointer to a buffer of bytes (unsigned 8-bit fields) used by the APIISR code entity to perform the operations indicated by the isr_type parameter. The pointer points to a block of contiguous memory. This parameter is only used if isr_type is not set to MT90502_ISR_TYPE_NORMAL.

Direction: IN, IN/IO Type: POINTER

Default: NULL

pipe_buf_size 0 – ??

Indicates the number of bytes pointed to by ppipe_buf.

Direction: IN Type: ULONG

Default: 0

4.0 Return Codes

The description for error return codes can be found in the mt90502_def.h file of the API release.

Errors in the range 0x1000-0x1FFF indicate the API software has detected an internal fatal error. These errors should be reported to the vendor for resolution.

5.0 Configuration Structures

5.1 Structure MT90502_CONF

5.1.1 General Parameters

user_chip_number identifier

This number is carried down to the user-supplied read/write routines to distinguish which chip the API is servicing. This can be used as an array index of the chip to be serviced to retrieve the correct instance pointer. Of course, if only one chip is being serviced by the API, then this parameter can be ignored. See System Architecture.

Direction: IN Type: ULONG

Default: MT90502_INVALID_CHIP_NUMBER

max_rw_access 8 – 1024

The maximum number of device addresses that the API will attempt to read or write in a single call of a user read or write function (e.g. mt90502_driver_write_burst_api).

Direction: IN Type: ULONG

Default: 8

enable_cas TRUE / FALSE

When TRUE the API will process any mini-packets received in the RX CPU mini-packet buffer with UUI 24, LI 4 and message type field of 3 (Type 3 packets per ITU I366.2). The API will also create a TX CAS packet buffer and RX CID event buffer (see soft_cid_event_buffer_size and soft_tx_cas_pkt_buffer_size) and filter the redundant CAS packets and check the CRC per ITU I366.2. See mt90502_receive_cas_pkt function). Also, any mini-packets received on UUIs 0 – 15 of a CID that is either closed or open but not mapped to any RX channel will generate a CID event in the RX CID event buffer. Furthermore, if the rx_voice_one_only parameter of this structure is set to TRUE then all subsequent mini-packets received on UUIs 0 – 15 for closed CIDs will be discarded. See the rx_voice_one_only parameter of the MT90502_OPEN_CID structure for open CID behavior.

Direction: IN Type: ULONG

Default: FALSE

max_aal2_vc 1 – 1023

The maximum number of AAL2 VCs that this chip instance will open concurrently. This parameter is used (in conjunction with max_channel) in part to determine the minimum amount of SSRAM (RX and TX) needed.

The minimum amount of SSRAM needed by the chip can be 1 of 4 settings (see Typical Memory Configurations). These 4 settings are dependant on the channel capacity of the chip.

In the chip, it must be possible to open max_aal2_vc number of VCs with a minimum of 1 channel per VC. Thus, the largest of the values max_aal2_vc and max_channel determines the maximum number of channels that must be supported. This number, rounded up to 1 of the 4 settings mentioned above determines the channel capacity of the chip.

This field also determines the amount of memory needed by the instance structure of the chip to keep track of all VCs, and thus affects the required size of the instance structure.

Direction: IN Type: ULONG

Default: 1023

max_cid 1 – 65536

The maximum number of CIDs that this chip instance will open concurrently.

This field determines the amount of memory needed by the instance structure of the chip to keep track of all CIDs, and thus affects the required size of the instance structure.

Direction: IN Type: ULONG

Default: 1023

max_cid_value 1 – 255

The maximum CID value of an open CID. This field determines the amount of memory needed by the instance structure of the chip to map incoming CPS packets to open connections, and thus affects the required size of the instance structure.

Direction: IN Type: ULONG

Default: 31

max_data_vc_a 1 - ??

This field indicates the maximum number of data VCs which can be opened concurrently on port A. The maximum number is dependent upon the number of header bits used to perform header concatenation on port A. This field is used to minimize the amount of memory needed to manage the data VCs within the API.

Direction: IN Type: ULONG

Default: 1024

max_data_vc_b 1 - ??

This field indicates the maximum number of data VCs which can be opened concurrently on port B. The maximum number is dependent upon the number of header bits used to perform header concatenation on port B. This field is used to minimize the amount of memory needed to manage the data VCs within the API.

Direction: IN Type: ULONG

Default: 1024

max_data_vc_c 1 - ??

This field indicates the maximum number of data VCs which can be opened concurrently on port C. The maximum number is dependent upon the number of header bits used to perform header concatenation on port C. This field is used to minimize the amount of memory needed to manage the data VCs within the API.

Direction: IN Type: ULONG

Default: 1024

max_channel 1 – 1023

The maximum number of channels this chip instance will open concurrently. This parameter is used (in conjunction with max_vc) in part to determine the minimum amount of external SSRAM needed by the chip (see max_vc).

This field also determines the amount of memory needed by the instance structure of the chip to keep track of all open channels, and thus affects the required size of the instance structure. Note that in the instance structure the number of channels is not rounded up to 1 of 4 settings, such as done with the chip's structures stored in external memory (see max_vc). Thus, the amount of memory used by the instance structure for keeping track of the open channels is directly proportional to the max_channel.

Direction: IN Type: ULONG

Default: 1023

max_stream {4, 8, 16, 32}

The maximum number of H.100 streams that this chip instance will allocate timeslots on concurrently. This parameter is used to allow the chip to operate at lower clock frequencies. When less than 32 streams are specified the most significant streams are removed first. For example, if max_stream = 8 then only streams ct_d[7:0] may be used by this chip instance.

Direction: IN Type: ULONG

Default: 32

max_cas_refresh 4 – 65536

The maximum number of CIDs simultaneously sending CAS refresh. All CIDs requiring refresh are refreshed at the same interval. The refresh is done by a user call to the mt90502_cas_refresh function. The interval of refresh is determined by how often the user calls the function. See the mt90502_cas_refresh function.
This value is used to determine the amount of memory needed for the CAS refresh packet buffer.

Direction: IN Type: ULONG

Default: 1024

upclk_freq 25000000 – 50000000

The frequency of upclk, in Hz.

Direction: IN Type: ULONG

Default: 25000000

mclk_freq 12500000 – 60000000

The frequency of mclk, in Hz.

Direction: IN Type: ULONG

Default: 60000000

led_flash_freq 1 – 10

LED flashing frequency in Hz. This is used to indicate link activity.

Direction: IN Type: ULONG

Default: 3

cache_cpu_accesses TRUE / FALSE

Configures whether the device will cache accesses or not. In general, if the device is being read using direct accesses this should be set to TRUE to avoid CPU access timeouts. If reads

are indirect then this parameter, in general, should be set to FALSE to increase the performance of the device. See User Supplied Function Descriptions for more information.

Direction: IN Type: ULONG

Default: FALSE

pdv_min_monitor_period 1-20000 ms

The minimum amount of time elapsed between two PDV monitoring periods, in milliseconds. This parameter is applied to all RX xxPCM channels. The PDV is monitored by calling the mt90502_update_counters function. If the mt90502_update_counters function is called more frequently than the specified period, the PDV monitoring statistics will not be updated (i.e. MAX and MIN PDV will be monitored for at least the monitor period).

Direction: IN Type: ULONG

Default: 10000 (10 sec)

adap_a_enb TRUE / FALSE

Whether adaptive clock recovery A FIFO is to be enabled.

Direction: IN Type: ULONG

Default: FALSE

adap_b_enb see adap_a_enb

Default: FALSE

rx_circular_buffer_size MT90502_CIRC_BUF_SIZE_256B
MT90502_CIRC_BUF_SIZE_512B
MT90502_CIRC_BUF_SIZE_1024B

The size of the receive circular buffer associated to each open channel, in bytes. The amount of total memory used by the chip for the circular buffers is determined as follows:

rx_circular_buffer_memory = max_channel * rx_cicular_buffer_size

Direction: IN Type: ULONG

Default: MT90502_CIRC_BUF_SIZE_256B

rx_mini_pkt_buffer_size MT90502_CPU_MP_BUF_SIZE_16KB
MT90502_CPU_MP_BUF_SIZE_32KB
MT90502_CPU_MP_BUF_SIZE_64KB
MT90502_CPU_MP_BUF_SIZE_128KB

The required size of RX mini packet buffer contained in SSRAM (in bytes). Each stored mini packet requires enough bytes for the header and payload of the received packet rounded up to the nearest 2 bytes. The RX mini packet buffer is filled by received mini packets that are routed to the CPU interface. The device will assert an interrupt when the buffer is ½ full and the API will transfer the cells to the soft RX mini packet buffer during interrupt servicing. The required size is determined by the maximum rate that mini packets destined to the CPU interface are expected to be received and the amount of time after the interrupt is asserted until the API interrupt service routine is called by the user software. If this buffer overflows mini packets will be dropped and the rx_mini_pkt_buf_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes.

Direction: IN Type: ULONG

Default: MT90502_CPU_MP_BUF_SIZE_16KB

tx_mini_pkt_buffer_size

MT90502_CPU_MP_BUF_SIZE_16KB
MT90502_CPU_MP_BUF_SIZE_32KB
MT90502_CPU_MP_BUF_SIZE_64KB
MT90502_CPU_MP_BUF_SIZE_128KB

The required size of TX mini packet buffer contained in SSRAM (in bytes). Each stored mini packet requires enough bytes for the payload of the transmitted packet rounded up to the nearest 2 bytes. The TX mini packet buffer is filled by transmitted mini packets that are routed by the CPU interface. The required size is determined by the maximum rate that the CPU is expected to generate mini packets.

Direction: IN Type: ULONG

Default: MT90502_CPU_MP_BUF_SIZE_16KB

rx_data_buffer_size

MT90502_RX_DATA_BUF_64_CELLS
MT90502_RX_DATA_BUF_128_CELLS
MT90502_RX_DATA_BUF_256_CELLS
MT90502_RX_DATA_BUF_512_CELLS
MT90502_RX_DATA_BUF_1024_CELLS
MT90502_RX_DATA_BUF_2048_CELLS

The required size of RX data buffer contained in SSRAM in cells (each cell requires 64 bytes). The RX data cell buffer is filled by received cells that are routed to the CPU interface (e.g. data). The device will assert an interrupt when the buffer is ½ full, and the API will transfer the cells to the soft RX data buffer during interrupt servicing. The required size is determined by the maximum rate that cells destined to the CPU interface are expected to be received and the amount of time after the interrupt is asserted until the API interrupt service routine is called by the user software. If this buffer overflows cells will be dropped and the rx_data_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes.

Direction: IN Type: ULONG

Default: MT90502_RX_DATA_BUF_64_CELLS

rx_event_buffer_size

MT90502_EVENT_BUF_SIZE_512_EVENTS
MT90502_EVENT_BUF_SIZE_1K_EVENTS
MT90502_EVENT_BUF_SIZE_2K_EVENTS
MT90502_EVENT_BUF_SIZE_4K_EVENTS
MT90502_EVENT_BUF_SIZE_8K_EVENTS
MT90502_EVENT_BUF_SIZE_16K_EVENTS

The required size of RX event buffer contained in SSRAM, in events (each event requires 8 bytes). The buffer is filled by events that occur on open AAL2 VCs. The device will assert an interrupt when the buffer is ½ full and the API will remove the events and update the AAL2 VC statistics or retrieve indicated CPU mini-packets, CAS changes, or LI change indications during interrupt servicing. The required size is determined by the maximum rate of events expected and the amount of time after the interrupt is asserted until the API interrupt service routine is called by the user software. If this buffer overflows RX events will be dropped and the rx_event_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes.

Direction: IN Type: ULONG

Default: MT90502_EVENT_BUF_SIZE_1K_EVENTS

adap_a_buffer_size MT90502_ADAPTIVE_BUF_256_PNTS
MT90502_ADAPTIVE_BUF_512_PNTS
MT90502_ADAPTIVE_BUF_1K_PNTS
MT90502_ADAPTIVE_BUF_2K_PNTS
MT90502_ADAPTIVE_BUF_4K_PNTS
MT90502_ADAPTIVE_BUF_8K_PNTS

The required size of clock recovery buffer A contained in SSRAM in units of one adaptive clock recovery point (16 bytes). The buffer is filled by points generated from either cells received on a selected clock recovery VC, mini packets received on a selected xxPCM channel, or mini packets received on a selected HDLC stream. The selection of the sourcing VC/channel is done when the VC/channel is opened. The device will assert an interrupt when the buffer is ½ full and the API will transfer the points to the soft clock recovery A buffer during interrupt servicing. The required size is determined by the maximum rate cells/mini packets on the selected VC/channel are expected to be received and the amount of time after the interrupt is asserted until the API interrupt service routine is called by the user software. If this buffer overflows points will be dropped and the adap_a_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes.

Direction: IN Type: ULONG

Default: MT90502_ADAPTIVE_BUF_256_PNTS

adap_b_buffer_size see adap_a_buffer_size

Default: MT90502_ADAPTIVE_BUF_256_PNTS

soft_rx_mini_pkt_buffer_size 4 – 131072

The required size of the soft RX mini packet buffer contained in program memory in bytes. Each stored mini packet requires enough bytes for the packet header and payload plus an overhead of 12 bytes rounded up to the nearest 2 bytes. The soft RX mini packet buffer is filled from the RX mini packet buffer in SSRAM by the API interrupt service routine and emptied by user calls to the mt90502_receive_aal2_mini_pkt function. The required size of the buffer is determined by the rate that mini packets destined to the CPU interface are expected to be received and the frequency of user software removing mini packets from the buffer. If this buffer overflows, mini packets will be dropped and the soft_rx_mini_pkt_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes.

Direction: IN Type: ULONG

Default: 16384

soft_cid_event_buffer_size 4 – 131072

The required size of the soft CID event buffer contained in program memory in bytes. Each stored CID event requires 12 bytes. The soft RX CID event buffer is filled from the RX mini packet buffer in SSRAM by the API interrupt service routine and emptied by user calls to the mt90502_receive_cid_event function. The required size of the buffer is determined by the rate that CID events are expected to be generated and the frequency of user software removing the events from the buffer. If this buffer overflows, CID events will be dropped and the soft_cid_event_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes. If enable_cas is set to FALSE this parameter is ignored and no buffer is allocated.

Direction: IN Type: ULONG

Default: 16384

soft_rx_data_buffer_size 4 – 16384

The required size of the soft RX data buffer contained in program memory in cells (each cell requires 72 bytes). The soft RX data buffer is filled from the RX data buffer in SSRAM by the API interrupt service routine and emptied by user calls to the mt90502_receive_data_cell function. The required size of the buffer is determined by the rate that cells destined to the CPU interface are expected to be received and the frequency of user software removing cells from the buffer. If this buffer overflows, cells will be dropped and the soft_rx_data_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes.

Direction: IN Type: ULONG

Default: 1024

soft_tx_mini_pkt_buffer_size 4 – 131072

The required size of the soft TX mini packet buffer contained in program memory in bytes. Each stored mini packet requires enough bytes for the packet header and payload plus an overhead of 12 bytes rounded up to the nearest 2 bytes. The soft TX mini packet buffer allows the API to receive mini packets from the function mt90502_send_aal2_mini_pkt and buffer them when the device TX mini packet buffer is full. When the soft buffer becomes full the mt90502_send_aal2_mini_pkt function returns the MT90502ER_SEND_MINI_PKT_BUFFER_FULL result.

Direction: IN Type: ULONG

Default: 16384

soft_tx_cas_pkt_buffer_size 4 – 131072

The required size of the soft TX CAS packet buffer contained in program memory in bytes. Each stored CAS packet requires 20 bytes. The soft TX CAS packet buffer allows the API to send CAS packets from the function mt90502_send_cas_pkt and buffer them when the device TX mini packet buffer is full. When the soft buffer becomes full the mt90502_send_cas_pkt function returns the MT90502ER_SEND_CAS_PKT_BUFFER_FULL result. If enable_cas is set to FALSE this parameter is ignored and no buffer is allocated.

Direction: IN Type: ULONG

Default: 16384

soft_tx_data_buffer_size 4 – 16384

The required size of the soft TX data buffer contained in program memory in cells (each cell requires 60 bytes). The soft TX data buffer allows the API to receive data cells from the function mt90502_send_data_cell and mt90502_send_test_cell, and buffer them when the device TX buffer is full. When the soft buffer becomes full the mt90502_send_data_cell and mt90502_send_test_cell functions returns the MT90502ER_SEND_DATA_BUFFER_FULL result.

Direction: IN Type: ULONG

Default: 256

soft_li_uui_change_buffer_size 0 – 32768

The required size of the soft LI change buffer contained in program memory, in LI change events (each event requires 20 bytes). The soft LI change buffer allows the API to collect LI or UUI changes from the chip. The LI changes are retrieved by the function

mt90502_get_li_uui_change_event. If no channels are configured with rx_detect_li_change or rx_detect_uui_change set to TRUE, then the buffer may be configured with a size of 0.

Direction: IN Type: ULONG

Default: 256

soft_adap_a_buffer_size 4 – 16384

The required size of the soft clock recovery A buffer contained in program memory, in clock recovery points (each point requires 24 bytes). The soft clock recovery A buffer is filled from the clock recovery A buffer in SSRAM by the API interrupt service routine and emptied by user calls to the mt90502_get_adaptive_clk_recov_point function. The required size is determined by the maximum rate cells/mini packets on the selected VC/channel are expected to be received and the frequency of user software removing points from the buffer. If this buffer overflows clock recovery points will be lost and the soft_adap_a_a_buffer_overflow parameter will be set in the MT90502_CHIP_STATS structure returned by the function mt90502_get_chip_statistics. This parameter is used for optimizing structure sizes. This soft buffer must be at least as large as the buffer in the device (soft_adap_a_buffer_size >= adap_a_buffer_size)

Direction: IN Type: ULONG

Default: 1024

soft_adap_b_buffer_size see **soft_adap_a_buffer_size**

Default: 1024

soft_console_buffer_size 0 – 262144

The required size of the soft console message buffer contained in program memory, in bytes. Messages are stored as text in a circular buffer. They are retrieved by the function get_console_messages. If the size is configured as 0, console messaging will be disabled. When the buffer becomes full, newer messages are lost until it is emptied by the function get_console_messages. If the buffer ever overflows the last message will indicate the overflow.

Direction: IN Type: ULONG

Default: 16384

5.1.2 Interrupt Configuration Parameters

Interrupt_period_granularity 5 - 1000 ms

The granularity of the specified minimum period for an internally generated interrupt, in ms. An interrupt can be disabled for a short period of time following its activation. If configured for a given interrupt this field indicates the granularity of time for the timeout period. For example, if 10 ms is chosen then an interrupt can be disabled for 10, 20, 30, … ms.

Direction: IN Type: UINT32

Default: 10 (10 ms)

interrupt_polarity MT90502_INT_ACTIVE_LOW_OC
MT90502_INT_ACTIVE_HIGH_OC

Polarity and active status of interrupt line 1. The line can be active high or low and is in tri-state (open collector) when not active. Interrupt line 2 is not used by the chip to signal interrupts.

Direction: IN Type: ULONG

Default: MT90502_INT_ACTIVE_LOW_OC

interrupt_configuration see structure **MT90502_CONF_INTERRUPTS**.

Direction: IN Type: MT90502_CONF_INTERRUPTS

Default: see structure

5.1.3 Memory Configuration Parameters

mem_banka_chip_size MT90502_BANK_CHIP_SIZE_128KB
MT90502_BANK_CHIP_SIZE_256KB
MT90502_BANK_CHIP_SIZE_512KB
MT90502_BANK_CHIP_SIZE_1MB

Indicates the size of each SSRAM memory chip used for bank A.

Direction: IN Type: ULONG

Default: MT90502_BANK_CHIP_SIZE_1MB

mem_banka_num_chips 1, 2

The number of SSRAM memory chips used to form bank A.

Direction: IN Type: ULONG

Default: 1

mem_bankb_chip_size MT90502_BANK_CHIP_SIZE_128KB
MT90502_BANK_CHIP_SIZE_256KB
MT90502_BANK_CHIP_SIZE_512KB
MT90502_BANK_CHIP_SIZE_1MB

Indicates the size of each SSRAM memory chip used for bank B.

Direction: IN Type: ULONG

Default: MT90502_BANK_CHIP_SIZE_1MB

mem_bankb_num_chips 1, 2

The number of SSRAM memory chips used to from bank B.

Direction: IN Type: ULONG

Default: 1

mem_bankb_present TRUE / FALSE

If TRUE bank B is present.

Direction: IN Type: ULONG

Default: TRUE

rx_circ_buf_parity_use MT90502_RX_CIRC_BUF_PARITY_RW_ERRORS
MT90502_RX_CIRC_BUF_PARITY_UNDERRUNS

Determines how the parity bits on the RX side of all circular buffers are used. The parity bits can be used to detect read and write errors, or to detect SID and TDM underruns.

Direction: IN Type: ULONG

Default: MT90502_RX_CIRC_BUF_PARITY_UNDERRUNS

sdram_size MT90502_SDRAM_SIZE_8MB
MT90502_SDRAM_SIZE_16MB

The size of the SDRAM, in bytes.

Direction: IN Type: ULONG

Default: MT90502_SDRAM_SIZE_8MB

sdram_refresh_time 2 – 32

The maximum interval between two CBR (Auto) refresh cycles, in microseconds.

Direction: IN Type: ULONG

Default: 8

5.1.4 UTOPIA Port Physical Configuration Parameters

u_pa_level2_mode TRUE / FALSE

If TRUE, UTOPIA port A is configured to operate with address select lines as described in UTOPIA level 2 specification (PHY mode only). The address is determined by u_pa_level2_add. Note that in this mode port B is disabled. While u_txa_always_drive may be TRUE or FALSE, using UTOPIA level 2 addresses is only useful if it is FALSE.

Direction: IN Type: ULONG

Default: FALSE

u_pa_level2_add 0 – 30

This is the address used by port A when operation under UTOPIA level 2 multi-PHY mode with address lines. (u_pa_level2_mode = TRUE).

Direction: IN Type: ULONG

Default: 0

u_pa_enb TRUE / FALSE

Determines whether UTOPIA port A is enabled.

Direction: I N Type: ULONG

Default: TRUE

u_pa_sar_mode MT90502_PHY_LAYER
MT90502_ATM_LAYER

Determines whether UTOPIA port A is in PHY or ATM mode.

Direction: IN Type: ULONG

Default: MT90502_ATM_LAYER

u_txa_clk_oe TRUE / FALSE

If TRUE the txa_clk clock is driven by the chip. If FALSE, the clock is tri-stated and should be driven into the chip by external components.

Direction: IN Type: ULONG

Default: TRUE

u_rxa_clk_oe TRUE / FALSE

If TRUE the rxa_clk clock is driven by the chip. If FALSE, the clock is tri-stated and should be driven into the chip by external components.

Direction: IN Type: ULONG

Default: TRUE

u_txa_always_drive TRUE / FALSE

If TRUE, the data, parity, and soc lines are always driven. If FALSE, the pins are driven only when UTOPIA TX port A is selected, and tri-stated otherwise.

Direction: IN Type: ULONG

Default: TRUE

u_txa_led_conf MT90502_PHY_LED_CONF_LED
MT90502_PHY_LED_CONF_TRISTATE
MT90502_PHY_LED_CONF_DRIVE_LOW
MT90502_PHY_LED_CONF_DRIVE_HIGH

Determines whether the txa_led pin is used to drive a LED or to drive a constant value.

Direction: IN Type: ULONG

Default: MT90502_PHY_LED_CONF_LED

u_rxa_led_conf MT90502_PHY_LED_CONF_LED
MT90502_PHY_LED_CONF_TRISTATE
MT90502_PHY_LED_CONF_DRIVE_LOW
MT90502_PHY_LED_CONF_DRIVE_HIGH

Determines whether the rxa_led pin is used to drive a LED or to drive a constant value.

Direction: IN Type: ULONG

	Default:	MT90502_PHY_LED_CONF_LED
u_pb_enb		see **u_pa_enb**
	Default:	FALSE
u_pb_sar_mode		see **u_pa_sar_mode**
	Default:	MT90502_PHY_LAYER
u_txb_clk_oe		see **u_txa_clk_oe**
	Default:	FALSE
u_rxb_clk_oe	see u_rxa_clk_oe	
	Default:	FALSE
u_txb_always_drive		see **u_txa_always_drive**
	Default:	FALSE
u_txb_led_conf		see **u_txa_led_conf**
	Default:	MT90502_PHY_LED_CONF_LED
u_rxb_led_conf		see **u_rxa_led_conf**
	Default:	MT90502_PHY_LED_CONF_LED
u_pc_enb		see **u_pa_enb**
	Default:	TRUE
u_pc_sar_mode		see **u_pa_sar_mode**
	Default:	MT90502_PHY_LAYER
u_txc_clk_oe		see **u_txa_clk_oe**
	Default:	FALSE
u_rxc_clk_oe		see **u_rxa_clk_oe**
	Default:	FALSE
u_txc_always_drive		see **u_txa_always_drive**
	Default:	TRUE

5.1.4.1 Utopia Clock Divider Configuration Parameters

u_divtxa_source

MT90502_TXA_CLK_IN
MT90502_TXB_CLK_IN
MT90502_TXC_CLK_IN
MT90502_RXA_CLK_IN
MT90502_RXB_CLK_IN
MT90502_RXC_CLK_IN
MT90502_MCLK

Selects the input of clock divisor UTOPIA TX A. The sources can be the TX clock of ports A, B, C, the RX clock of port A, B, C, and mclk. Note that the clock divisor must not drive the same clock that feeds it. For example, u_divtxa_source must not be set to MT90502_TXA_CLK_IN.

Direction: IN Type: ULONG

Default: MT90502_TXC_CLK_IN

u_divtxa_clk_divisor 1 – 63

The integer value of clock divisor UTOPIA TX A.

Direction: IN Type: ULONG

Default: 1

u_divtxa_clk_invert TRUE / FALSE

If TRUE then the output of the clock divisor is inverted.

Direction: IN Type: ULONG

Default: TRUE

u_divtxb_source see **u_divtxa_source**

Default: MT90502_TXC_CLK_IN

u_divtxb_clk_divisor see **u_divtxa_clk_divisor**

Default: 1

u_divtxb_clk_invert see **u_divtxa_clk_invert**

Default: FALSE

u_divtxc_source see **u_divtxa_source**

Default: MT90502_RXC_CLK_IN

u_divtxc_clk_divisor see **u_divtxa_clk_divisor**

Default: 1

u_divtxc_clk_invert see **u_divtxa_clk_invert**

Default: FALSE

u_divrxa_source see **u_divtxa_source**

Default: MT90502_TXC_CLK_IN

u_divrxa_clk_divisor see **u_divtxa_clk_divisor**

Default: 1

u_divrxa_clk_invert see **u_divtxa_clk_invert**

Default: TRUE

u_divrxb_source see **u_divtxa_source**

Default: MT90502_TXC_CLK_IN

u_divrxb_clk_divisor see **u_divtxa_clk_divisor**

Default: 1

u_divrxb_clk_invert see **u_divtxa_clk_invert**

Default: FALSE

u_divrxc_source see **u_divtxa_source**

Default: MT90502_TXC_CLK_IN

u_divrxc_clk_divisor see **u_divtxa_clk_divisor**

Default: 1

u_divrxc_clk_invert see **u_divtxa_clk_invert**

Default: FALSE

5.1.5 UTOPIA Operational Characteristics Parameters

5.1.5.1 General

The general parameters apply to all UTOPIA ports.

u_hec_mask 8 bit field

This mask is XORed with the accumulated header CRC result to form the HEC value of all cells sent on a TX UTOPIA port.

Direction: IN Type: ULONG

Default: 0x55

u_phy_alarm_mode MT90502_PHY_ALARM_DISABLED
MT90502_PHY_ALARM_ACTIVE_LOW
MT90502_PHY_ALARM_ACTIVE_HIGH

Determines the polarity of all PHY alarms.

Direction: IN Type: ULONG

Default: MT90502_PHY_ALARM_ACTIVE_HIGH

5.1.5.2 Cell Routing

u_rxa_null_cell_elim TRUE / FALSE

Whether to eliminate null cells entering the chip via port UTOPIA RX A.

Direction: IN Type: ULONG

Default: TRUE

u_txa_network_mask 28 bit field

This mask indicates which bits of a cell's GFC, VPI and VCI fields will be used to identify a VC connection by the network device connected to this UTOPIA port. All bits that are high will be used by the network device, and all bits that are low will be ignored.

This field is used by the API software to detect conflicts during the opening of a VC that will be exiting the chip through UTOPIA port A. If the requested header will not be unique to the network according to this mask the mt90502_open_aal2_vc or mt90502_open_data_vc function will be unsuccessful.

Direction: IN Type: ULONG

Default: 0x000FFFF

u_rxa_default_ncr

0, or the OR of any or all of:
MT90502_CELL_ROUTE_TXA
MT90502_CELL_ROUTE_TXB
MT90502_CELL_ROUTE_TXC
MT90502_CELL_ROUTE_DATA_FIFO

Determines how non-OAM cells entering the chip on port UTOPIA RX A are routed if they pass the mask and match but no VC is currently open with the corresponding ATM header. The cells can be sent to any or all of the ports by using the OR of the values listed above. If this field is set to 0, the cells will be discarded. Note that the cells cannot be routed on the same port they were received on.

Direction: IN Type: ULONG

Default: 0

u_rxa_default_ocr

0, or the OR of any or all of:
MT90502_CELL_ROUTE_TXA
MT90502_CELL_ROUTE_TXB
MT90502_CELL_ROUTE_TXC
MT90502_CELL_ROUTE_DATA_FIFO

Determines how OAM cells entering the chip on port UTOPIA RX A are routed if they pass the mask and match but no VC is currently open with the corresponding ATM header. The cells can be sent to any or all of the ports by using the OR of the values listed above. If this field is set to 0, the cells will be discarded. Note that the cells cannot be routed on the same port they were received on.

Direction: IN Type: ULONG

Default: 0

u_rxa_ncr

0, or the OR of any or all of:
MT90502_CELL_ROUTE_TXA
MT90502_CELL_ROUTE_TXB
MT90502_CELL_ROUTE_TXC
MT90502_CELL_ROUTE_DATA_FIFO

Determines how unknown non-OAM cells entering the chip on port UTOPIA RX A are routed. The cells can be sent to any or all of the ports by using the OR of the values listed above. If this field is set to 0, the cells will be discarded.

Direction: IN Type: ULONG

Default: MT90502_CELL_ROUTE_DATA_FIFO

u_rxa_ocr

0, or the OR of any or all of:
MT90502_CELL_ROUTE_TXA
MT90502_CELL_ROUTE_TXB
MT90502_CELL_ROUTE_TXC
MT90502_CELL_ROUTE_DATA_FIFO

Determines how unknown OAM cells entering the chip on port UTOPIA RX A are routed. The cells can be sent to any or all of the ports by using the OR of the values listed above. If this field is set to 0, the cells will be discarded.

Direction: IN Type: ULONG

Default: MT90502_CELL_ROUTE_DATA_FIFO

u_rxa_header_mask 28 bit field

This mask is used to determine which bits of a cell's GFC, VPI and VCI fields will be used with the u_rxa_header_match parameter to determine if a cell received on UTOPIA RX port A is known. If it is known then it will be passed to the LUT. Otherwise the cell will be routed according to the u_rxa_ncr and u_rxa_ocr fields.

The 28 bit field represents the GFC, VPI and VCI fields of the header where the GFC bits are the most significant. A "1" in a bit position indicates the corresponding bit position of the cell header should be compared with the corresponding bit of the u_pa_header_match parameter. All compared bits must match the u_pa_header_match parameter value to be passed to the LUT.

Note that this field must mask all bits which are not used by the LUT for header concatenation (see fields u_rxa_lut_index_vpi_bits and u_rxa_lut_index_vci_bits).Direction: IN Type: ULONG

Default: 0xF00FC00

u_rxa_header_match 28 bit field

This parameter is used in conjunction with the u_rxa_header_mask parameter to determine the required value of selected bits of a cell's GFC, VPI and VCI fields for a cell received on UTOPIA RX port A to be passed to the LUT. If the corresponding bits of the GFC,VPI, or VCI of the received header do not match the value set in this parameter and the mis-matched bits are not masked by the u_rxa_header_mask parameter the cell is treated as unknown and routed via the u_rxa_ncr and u_rxa_ocr fields.

The 28 bit field represents the concatenated GFC, VPI and VCI fields of the header where the GFC bits are the most significant. Only the result of bits not masked by the u_rxa_header_mask parameter will determine if the cell should be passed to the LUT.

For example:

GFC | VPI | VCI (from cell header)	0010 10001010 00000000 10110010	0010 10001010 00000000 10110**1**10
u_rxa_header_match	0000 00001010 00000000 10110010	0000 00001010 00000000 10110**0**10
Match Result (1=mismatch)	0010 10000000 00000000 00000000	0010 10000000 00000000 00000**1**00
u_rxa_header_mask	0000 00001111 00000000 11111111	0000 00001111 00000000 11111**1**11
Masked result (1=mismatched cell)	0000 00000000 00000000 00000000	0000 00000000 00000000 00000**1**00
Result	Routed according to LUT entry	Routed as unknown cell

Direction: IN Type: ULONG

Default: 0x0000000

u_rxa_lut_index_vpi_bits 0 – 12

This parameter determines how many bits of the concatenated GFC and VPI fields are to be used to index the LUT entry for cells received on UTOPIA RX port A. The specified number of bits are selected from LSB to MSB of the 12 bit field formed by the concatenation of the GFC and VPI fields of the header where the GFC bits are the most significant. The selected bits of the cell header are used to form the index of the LUT entry. The LUT index can be a minimum

of 1 bit and a maximum of 20 bits; so a maximum of 20 bits may be selected by the combination of this parameter and the u_rxa_lut_index_vci_bits parameter. The more total bits that are selected the larger the LUT structure is required to be. See u_rxa_lut_index_vci_bits for an example.

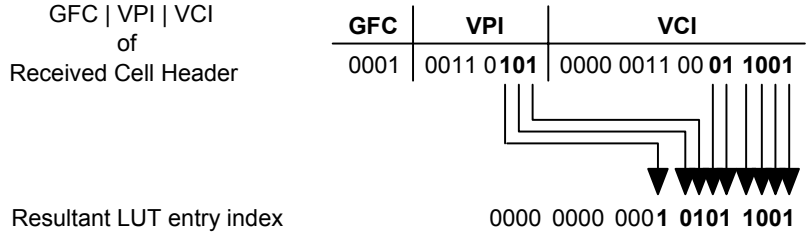
Direction: IN Type: ULONG

Default: 8

u_rxa_lut_index_vci_bits 0 – 16

This parameter determines how many bits of the VCI field are to be used to index the LUT entry for cells received on UTOPIA RX port A. The specified number of bits are selected from LSB to MSB of the 16 bit VCI field of the header. The selected bits of the cell header are used to form the index of the LUT entry. The LUT index can be a maximum of 20 bits so a maximum of 20 bits may be selected by the combination of this parameter and the u_rxa_lut_index_vpi_bits parameter. The more total bits that are selected the larger the LUT structure is required to be. If u_rxa_lut_index_vpi_bits and u_rxa_lut_index_vci_bits are both set to 0 then no VCs will be able to be opened on UTOPIA port A.

Example LUT entry index:
u_rxa_lut_index_vpi_bits = 3
u_rxa_lut_index_vci_bits = 6



Direction: IN Type: ULONG

Default: 10

u_rxb_null_cell_elim see **u_rxa_null_cell_elim**

Default: TRUE

u_txb_network_mask see **u_txa_network_mask**

Default: 0x0000000

u_rxb_default_ncr see **u_rxa_default_ncr**

Default: 0

u_rxb_default_ocr see **u_rxa_default_ocr**

Default: 0

u_rxb_ncr see **u_rxa_ncr**

Default: 0

u_rxb_ocr see **u_rxa_ocr**

Default: 0

u_rxb_header_mask see **u_rxa_header_mask**

Default: 0x0000000

u_rxb_header_match see **u_rxa_header_match**

Default: 0x0000000

u_rxb_lut_index_vpi_bits see **u_rxa_lut_index_vpi_bits**

Default: 0

u_rxb_lut_index_vci_bits see **u_rxa_lut_index_vci_bits**

Default: 0

u_rxc_null_cell_elim see **u_rxa_null_cell_elim**

Default: TRUE

u_txc_network_mask see **u_txa_network_mask**

Default: 0x0000FFF

u_rxc_default_ncr see **u_rxa_ncr**

Default: 0

u_rxc_default_ocr see **u_rxa_ocr**

Default: 0

u_rxc_ncr see **u_rxa_ncr**

Default: MT90502_CELL_ROUTE_DATA_FIFO

u_rxc_ocr see **u_rxa_ocr**

Default: MT90502_CELL_ROUTE_DATA_FIFO

u_rxc_header_mask see **u_rxa_header_mask**

Default: 0x FF0FFF0

u_rxc_header_match see **u_rxa_header_match**

Default: 0x0000000

u_rxc_lut_index_vpi_bits see **u_rxa_lut_index_vpi_bits**

Default: 4

u_rxc_lut_index_vci_bits see **u_rxa_lut_index_vci_bits**

Default: 4

Where as port A and B can support a maximum of 20 look-up bits, port C can only support up to 16. Thus, u_rxc_lut_index_vpi_bits + u_rxc_lut_index_vci_bits cannot exceed 16.

5.1.5.3 Flow Control

u_txa_rxa_cell_max 0 – 14
MT90502_NO_BACK_PRESSURE

If the cell fill of the UTOPIA TX port A output FIFO becomes greater than this value, cells from the UTOPIA RX port A input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the UTOPIA RX port A input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

u_txa_rxb_cell_max 0 – 14
MT90502_NO_BACK_PRESSURE

If the cell fill of the UTOPIA TX port A output FIFO becomes greater than this value, cells from the UTOPIA RX port B input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the UTOPIA RX port B input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

u_txa_rxc_cell_max 0 – 14
MT90502_NO_BACK_PRESSURE

If the cell fill of the UTOPIA TX port A output FIFO becomes greater than this value, cells from the UTOPIA RX port C input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the UTOPIA RX port C input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: 4

u_txa_txsar_cell_max 0 – 14
MT90502_NO_BACK_PRESSURE

If the cell fill of the UTOPIA TX port A output FIFO becomes greater than this value, cells from the TX SAR output FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then TXSAR output FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: 14

u_txa_data_cell_max 0 – 14
MT90502_NO_BACK_PRESSURE

If the cell fill of the UTOPIA TX port A output FIFO becomes greater than this value, cells from the data input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the data cell input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: 4

u_txb_rxa_cell_max see **u_txa_rxa_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txb_rxb_cell_max see **u_txa_rxb_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txb_rxc_cell_max see **u_txa_rxc_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txb_txsar_cell_max see **u_txa_txsar_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txb_data_cell_max see **u_txa_data_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txc_rxa_cell_max see **u_txa_rxa_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txc_rxb_cell_max see **u_txa_rxb_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txc_rxc_cell_max see **u_txa_rxc_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txc_txsar_cell_max see **u_txa_txsar_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_txc_data_cell_max see **u_txa_data_cell_max**

Default: MT90502_NO_BACK_PRESSURE

u_rxsar_rxa_cell_max 0 – 62
MT90502_NO_BACK_PRESSURE

If the cell fill of the RX SAR input FIFO becomes greater than this value, cells from the UTOPIA RX port A input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the UTOPIA RX port A input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

u_rxsar_rxb_cell_max 0 – 62
MT90502_NO_BACK_PRESSURE

If the cell fill of the RX SAR input FIFO becomes greater than this value, cells from the UTOPIA RX port B input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the UTOPIA RX port B input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

u_rxsar_rxc_cell_max 0 – 62
MT90502_NO_BACK_PRESSURE

If the cell fill of the RX SAR input FIFO becomes greater than this value, cells from the UTOPIA RX port C input FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the UTOPIA RX port C input FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

u_rxsar_txsar_cell_max 0 – 62
MT90502_NO_BACK_PRESSURE

If the cell fill of the RX SAR input FIFO becomes greater than this value, cells from the TXSAR output FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the TXSAR output FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

u_rxsar_data_cell_max 0 – 62
MT90502_NO_BACK_PRESSURE

If the cell fill of the RX SAR input FIFO becomes greater than this value, cells from the data cell output FIFO will be blocked. If MT90502_NO_BACK_PRESSURE is selected then the data cell output FIFO will not be blocked by this FIFO (cells may be dropped if the TX FIFO is full).

Direction: IN Type: ULONG

Default: MT90502_NO_BACK_PRESSURE

5.1.6 CID Configuration Parameters

rx_voice_one_only TRUE / FALSE

If TRUE, when the default_rx_uui_mapping for UUIs 0–15 is MT90502_RX_MINI_PKT_BUFFER, the first mini-packet received on UUIs 0–15 will generate an entry in the CID event buffer and the mapping will be modified by the API to MT90502_DELETE_MINI_PKT. Voice packets will be discarded until the CID is opened or the VC is closed and re-opened with a new mapping.

Direction: IN Type: ULONG

Default: FALSE

default_rx_uui_mapping[17] MT90502_DELETE_MINI_PKT
MT90502_RX_MINI_PKT_BUFFER

Each element of the array indicates the mapping of a received UUI value(s) for unopened CID values of an open VC. The indices correspond to UUI values in the following way:

index = 0 => UUIs = 0 – 15
index = 1 => UUI = 16
index = 2 => UUI = 17
… …
index = 16 => UUI = 31

Each mini-packet received can be either: deleted (MT90502_DELETE_MINI_PKT)or sent to the CPU RX mini-packet buffer (MT90502_RX_MINI_PKT_BUFFER). If enable_type3_processing is set to TRUE, UUIs 24 and 31 should be set to MT90502_RX_MINI_PKT_BUFFER to allow Type 3 reception per ITU standard I366.2.

By configuring unopened CID traffic to the RX CPU mini packet buffer the processor can receive packets for all CIDs on the VC even though the CIDs are not opened. If traffic is detected that indicates a voice connection is required the user software can map the CID to the appropriate channel. This allows oversubscription of CIDs.

Direction: IN Type: ULONG[17]

Default: MT90502_DELETE_MINI_PKT

5.1.7 TDM Configuration Parameters

underrun_padding_law MT90502_NO_PADDING_TRANSLATION
MT90502_PADDING_ULAW
MT90502_PADDING_ALAW

Determines the handling of underrun padding for all underrun sources. (See underrun_padding parameter of the MT90502_RX_XXPCM_CHAN structure) MT90502_PADDING_ULAW causes the MT90502 to treat the padding samples as u-Law and apply the proper translation to match the Law configured for the channel. (See pcm_law_translation parameter of the MT90502_RX_XXPCM_CHAN structure) MT90502_PADDING_ALAW causes the MT90502 to treat the padding samples as a-Law and apply the proper translation to match the Law configured for the channel. MT90502_NO_PADDING_TRANSLATION causes the MT90502 to do no translation of the padding samples before placing them on the bus, regardless of the configuration of the channel.

Direction: IN Type: ULONG

Default: MT90502_NO_PADDING_TRANSLATION

u_law_no_zero MT90502_PCM_A_LAW
MT90502_PCM_U_LAW

When TRUE, the u-law value 00h will be replaced by 02h in all a-law to u-law conversions.

Direction: IN Type: ULONG

Default: MT90502_PCM_U_LAW

adpcm_bit_positions MT90502_ADPCM_IN_LOWER_BITS
MT90502_ADPCM_IN_HIGHER_BITS

If a byte on the H100 bus contains ADPCM data, this field determines which bits of the byte contain the data. The data can either be placed in the most significant bits of the byte, or the least significant.

Direction: IN Type: ULONG

Default: MT90502_ADPCM_IN_LOWER_BITS

dstream_0_3_format MT90502_FORMAT_A
MT90502_FORMAT_B

This parameter determines the format of TDM TSSTs for streams on ct_d[3:0]. MT90502_FORMAT_A requires only one TSST to transport PCM or ADPCM but only allows

the device to auto-switch between ADPCM types, switching a stream between PCM and ADPCM requires SW intervention. MT90502_FORMAT_B requires 2 TSSTs to transport PCM or ADPCM but allows the device to auto-switch between PCM and ADPCM. See TDM Formats section.

Direction: IN Type: ULONG

Default: MT90502_FORMAT_A

dstream_0_3_freq MT90502_H100_FREQ_2MHZ
MT90502_H100_FREQ_4MHZ
MT90502_H100_FREQ_8MHZ

The frequency at which the lines ct_d[3:0] are operating.

Direction: IN Type: ULONG

Default: MT90502_H100_FREQ_8MHZ

dstream_4_7_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_4_7_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

dstream_8_11_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_8_11_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

dstream_12_15_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_12_15_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

dstream_16_19_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_16_19_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

dstream_20_23_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_20_23_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

dstream_24_27_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_24_27_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

dstream_28_31_format see **dstream_0_3_format**

Default: MT90502_FORMAT_A

dstream_28_31_freq see **dstream_0_3_freq**

Default: MT90502_H100_FREQ_8MHZ

h100_pll_clk_in_freq

MT90502_PLL_FREQ_8MHZ
MT90502_PLL_FREQ_16MHZ
MT90502_PLL_FREQ_32MHZ
MT90502_PLL_FREQ_64MHZ

The frequency of the pll_clk pin. This is the clock source for the H100 master block of the MT90502.

Direction: IN Type: ULONG

Default: MT90502_PLL_FREQ_16MHZ

h100_fr_comp_polarity

MT90502_POL_ACTIVE_HIGH
MT90502_POL_ACTIVE_LOW

Polarity of the H100 frame fr_comp.

Direction: IN Type: ULONG

Default: MT90502_POL_ACTIVE_LOW

h100_fr_comp_type

MT90502_FRCOMP_STRADDLE_FR_BOUNDARY
MT90502_FRCOMP_LAST_BIT
MT90502_FRCOMP_FIRST_BIT

Determines when the H100 fr_comp signal is active relative to the frame. The fr_comp signal can be configured in three ways:

- it can straddle the boundary between the last bit of one frame and first bit of the next.

- it can be active during the first bit of data of each H100 frame (i.e. the inactive to active transition is aligned with the frame boundary).

- it can be active during the last bit of data of each H100 frame. (i.e. the active to inactive edge is aligned with the frame boundary).

Note that the duration of the active period is determined by fr_comp_width. See examples below.

Direction: IN Type: ULONG

Default: MT90502_FRCOMP_STRADDLE_FR_BOUNDARY

h100_fr_comp_width

MT90502_FRCOMP_WIDTH_500NS
MT90502_FRCOMP_WIDTH_250NS
MT90502_FRCOMP_WIDTH_125NS

The width of the fr_comp signal. See examples below.

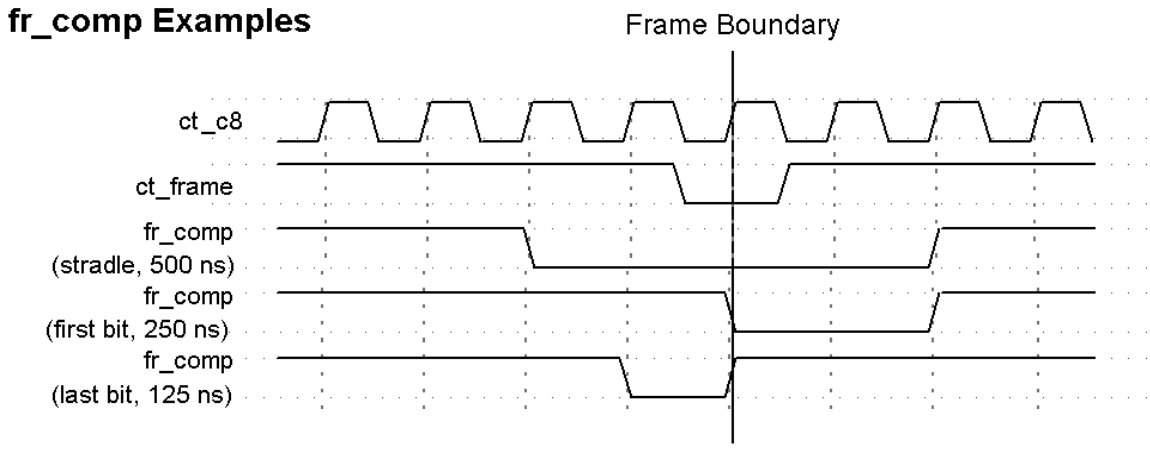


Figure 5 - fr_comp Examples

Direction: IN Type: ULONG

Default: MT90502_FRCOMP_WIDTH_125NS

h100_sclk_invert TRUE / FALSE

If FALSE the sclk signal has the same polarity as depicted in the H100 specification. Else, the signal's polarity is the inverse of that depicted in the H100 specification.

Direction: IN Type: ULONG

Default: FALSE

h100_sclkx2_invert TRUE / FALSE

If FALSE the sclkx2 signal has the same polarity as depicted in the H100 specification. Else, the signal's polarity is the inverse of that depicted in the H100 specification.

Direction: IN Type: ULONG

Default: FALSE

h100_sclk_freq MT90502_SCLK_FREQ_2MHZ
MT90502_SCLK_FREQ_4MHZ
MT90502_SCLK_FREQ_8MHZ

The frequency of the sclk signal.

Direction: IN Type: ULONG

Default: MT90502_SCLK_FREQ_8MHZ

h100_sampling MT90502_H100_SAMPLE_AT_3_QUARTERS
MT90502_H100_SAMPLE_AT_RISING_EDGE
MT90502_H100_SAMPLE_AT_FALLING_EDGE

The point at which a bit is sampled from the ct_d[31:0] lines. The bit can be sampled at the rising edge of the clock, the falling edge of the clock, or at 3/4ths of the clock cycle.

Direction: IN Type: ULONG

Default: MT90502_H100_SAMPLE_AT_3_QUARTERS

5.1.8 HDLC Configuration Parameters

hdlc_type MT90502_HDLC_FRAMING_BITWISE
MT90502_HDLC_FRAMING_BYTEWISE

Determines the type of framing used for HDLC packets on the H100 bus.

Direction: IN Type: ULONG

Default: MT90502_HDLC_FRAMING_BYTEWISE

hdlc_packaging_type MT90502_HDLC_WITHOUT_AAL2_HEADER
MT90502_HDLC_WITH_AAL2_HEADER

Determines if all HDLC packets entering / leaving the chip contain the AAL2 header.

Direction: IN Type: ULONG

Default: MT90502_HDLC_WITH_AAL2_HEADER

crc_preset 16 bit field

The initial value that is fed into the CRC generator. This value should be set to 0xFFFF so as to be compliant with CCITT HDLC.

Direction: IN Type: ULONG

Default: 0xFFFF

crc_mask 16 bit field

This field is masked with the accumulated CRC before being sent. This value should be set to 0xF0B8 so as to be compliant with CCITT HDLC.

Direction: IN Type: ULONG

Default: 0xF0B8

5.1.9 Tone Buffer Configuration Parameters

tone_buf_sizes[24] 0 – 65535

Member n of this array indicates the size, in bytes, of the tone buffer pair tone_buf_patterns[2n] and tone_buf_patterns[2n+1]. A value of 0 implies that the tone buffer pair is disabled.

Direction: IN Type: ULONG[24]

Default: (0,0,0…0)

tone_buf_patterns[48] array of pointers

An array of pointers pointing to tone buffers to be loaded in the RAM of the chip. The buffers are paired in the following way:

pair0	tone_buf_patterns[0] & tone_buf_patterns[1]
pair1	tone_buf_patterns[2] & tone_buf_patterns[3]
	……
pair31	tone_buf_patterns[46] & tone_buf_patterns[47]

Tone buffers of the same pair have the same size. The size of each pair can be found using the pair index (e.g. the buffers of pair1 have a size of tone_buf_sizes[1]). Each pointer points to an

array of bytes. The values of these buffers can be sent on the H100 bus in the case of underruns / silence suppression.

Direction: IN Type: POINTER[48]

Default: NULL

5.1.10 Silence Suppression Configuration Parameters

ADPCM channels can be configured to perform one of two silence suppression methods: simple and complex. Simple silence suppression involves an external device indicating to the chip which samples of the channel are silent and which are voice. Mini-packets are discarded if all samples that would comprise a given packet are indicated as silent by the external device.

In complex silence suppression, the chip determines which mini-packets are silent and which are voice. The method used is that the energy is calculated on all the received TDM bytes of a mini-packet to determine if the mini-packet should be treated as voice or silence.

For all channels configured to perform silence suppression, a silence suppression profile must be selected, irrespectively of the silence suppression method used. Depending on the silence suppression method, not all parameters of the profile are relevant (see MT90502_SIL_SUPPR_PROFILE structure description below). If a parameter is applicable in only one method, then it is indicated in the parameter's description. The method is selected by the sil_suppr_type parameter below.

For complex silence suppression, all energy calculations are done using linear samples converted from a PCM stream. When a channel is configured with silence suppression the chip requires PCM data for the transmitted channel, and if required by the profile the received channel, to be on specific TSSTs. For PCM configured channels the TX channel must be placed on a timeslot of an even stream number and the associated RX timeslot, if required by the profile, must be placed on the same timeslot as it's TX channel on the stream 1 greater than the TX channel's.

For ADPCM configured channels the chip requires equivalent PCM streams of the ADPCM data to be supplied. To accomplish this the TX ADPCM channel must be placed on a timeslot of stream 3, 7, 11, 15, 19, 24, 28, or 31. The equivalent TX PCM data must placed in the same timeslot as the TX ADPCM channel on the stream 3 less than the TX ADPCM channel's. The RX PCM data, if required by the profile, must be placed on the same timeslot as the TX ADPCM channel on the stream 2 less than the TX ADPCM channel's.

For example if an ADPCM channel is opened by mt90502_open_tx_xxpcm_channel and tx_sil_suppr_profile is not MT90502_SIL_SUPPR_DISABLED then the tx_stream value must be one of {3, 7, 11, 15, 19, 24, 28, 31} for this example we will use 7. If the tx_timeslot is 42 then timeslot 42 of stream 4 is automatically reserved for the TX PCM data and timeslot 42 of stream 5 is reserved for the RX PCM data if required by the profile selected by tx_sil_suppr_profile.

These streams do not need to be precisely synchronized but should be as close as possible. Any discrepancies can be compensated for by the additional_delay, silence_to_voice_time, and voice_to_silence_time parameters of the profile used.

sil_suppr_type

MT90502_SIMPLE_SIL_SUPPR
MT90502_COMPLEX_SIL_SUPPR
MT90502_PROFILE_SIL_SUPPR

Determines the silence suppression method used for all channels which are configured to suppress silence. When set to MT90502_SIMPLE_SIL_SUPPR the value of the profile_sil_suppr_type parameter will be ignored and will be treated as specifying simple silence suppression for all profiles. When set to MT90502_COMPLEX_SIL_SUPPR the value of the profile_sil_suppr_type parameter will be ignored and will be treated as specifying complex silence suppression for all profiles. When set to MT90502_PROFILE_SIL_SUPPR

the value of the profile_sil_suppr_type parameter determines the type of suppression for each profile.

Direction: IN Type: ULONG

Default: MT90502_SIMPLE_SIL_SUPPR

simple_assoc_tsst_bit MT90502_ASSOC_TSST_BIT_0
MT90502_ASSOC_TSST_BIT_7

The field determines which bit of an associated TSST for silence suppression is used to indicate silence. The two possibilities are bit 0 and bit 7. This filed is used when the specified suppression type is simple and simple_suppr_method is MT90502_ASSOC_TSST.

Direction: IN Type: ULONG

Default: MT90502_ASSOC_TSST_BIT_7

simple_a_match

This field is only used when sil_suppr_type is SIMPLE_SIL_SUPPR and simple_suppr_method, of the MT90502_SIL_SUPPR_PROFILE structure, is MT90502_A_MATCH. The field is used in conjunction with simple_a_mask to determine if received bytes are silent or not. An incoming byte is masked with simple_a_mask. The result of the masking is then compared to the value of simple_a_match. If the two are identical then the byte is deemed as silent. If all received bytes of a mini-packet are deemed silent, then the mini-packet is deemed silent.

Direction: IN Type: ULONG

Default: 0x80

simple_a_mask

This field is only used when sil_suppr_type is SIMPLE_SIL_SUPPR and simple_suppr_method, of the MT90502_SIL_SUPPR_PROFILE structure, is MT90502_A_MATCH. Each bit of this field is ANDed with the corresponding bit of the each received sample of a channel. The result is compared with simple_a_match. See simple_a_match.

Direction: IN Type: ULONG

Default: 0x80

simple_b_match see **simple_a_match**

Direction: IN Type: ULONG

Default: 0x80

simple_b_mask see **simple_a_mask**

Direction: IN Type: ULONG

Default: 0x80

sil_pcm_law MT90502_PCM_A_LAW
MT90502_PCM_U_LAW

Determines the default format of the PCM channels used for silence suppression. This parameter is only used when sil_suppr_type is set to MT90502_COMPLEX_SIL_SUPPR.

Direction: IN Type: ULONG

Default: MT90502_PCM_U_LAWsilent_buf_size 0
MT90502_BUF_SIZE_16KB
MT90502_BUF_SIZE_32KB
MT90502_BUF_SIZE_64KB

This field indicates the size of all silent buffers pointed to by the elements of the array silent_buf_patterns. A value of 0 implies that all silent buffers are disabled.

Direction: IN Type: ULONG

Default: 0

silent_buf_patterns[32] array of pointers

An array of pointers. Each pointer points to a silent pattern buffer (bytes). The values of these buffers can be sent on the H100 bus in the case of underruns / silence suppression.

Direction: IN/IN Type: POINTER[32]

Default: NULL

null_bytes[32] 32 8-bit fields

An array of 32 possible single byte values that can be inserted on the H100 bus in the case of underruns / silence suppression.

Direction: IN Type: ULONG

Default: [0-29] = 0, [30] = 0x55, [31] = 0xFF

sid_to_silence_pcm[128] array of SID silence sources

This parameter specifies the handling of I.366.2 Annex I, generic silence insertion descriptor ID values. The array is used like a lookup table for the SID ID value except that for all the reserved values 0–29, 79–126 the lookup will use entry 0 (i.e. indices 1–29, 79–126 are not required to be valid). Each element of the array specifies the source of silent noise to use for it's corresponding SID ID. Since the sources each have several buffers the specific source identifier is formed by ORing the define for the source (e.g. SID_TONE_BUFFER) with the desired buffer number. The source defines are:

SID_TONE_BUFFER	(buffers 0–47 available)
SID_SILENT_PATTERN_BUFFER	(buffers 0–31 available)
SID_NULL_BYTE	(buffers 0–31 available)
SID_USE_PREVIOUS_SID_SOURCE	no buffers)

A lookup entry set to SID_USE_PREVIOUS_SID_SOURCE requires no buffer number and should not be ORed with any value. This define will cause any SID received with this ID to use the source identified by the previous received SID value.

Direction: IN Type: ULONG[128]

Default: SID_USE_PREVIOUS_SID_SOURCE

sil_suppr_profiles[256] array of MT90502_SIL_SUPPR_PROFILE pointers

Array of 256 possible profiles to use for silence suppression of XXPCM channels. If an element of sil_suppr_profiles is set to NULL then that profile is considered invalid. This array is processed consecutively beginning with entry 0 until the first NULL pointer is encountered, all remaining entries are considered invalid. The descriptions of the elements of the structure

MT90502_SIL_SUPPR_PROFILE are listed below. To initialize a silence suppression profile see mt90502_init_sil_suppr_profile function.

Direction: IN/IN Type: POINTER[256]

Default: NULL

5.1.10.1 Structure MT90502_SIL_SUPPR_PROFILE

The following parameters are used to determine how to suppress silence on a given channel. Each mini packet received has a state of either silence or voice. This state is determined differently for each silence suppression method. In simple silence suppression, an external DSP is responsible for indicating if a mini packet is silent or not. In complex silence suppression, the chip performs energy calculations on the received bytes. The default values are inserted by the mt90502_init_sil_suppr_profile function.

profile_sil_suppr_type MT90502_SIMPLE_SIL_SUPPR
MT90502_COMPLEX_SIL_SUPPR

When the sil_suppr_type parameter is set to MT90502_PROFILE_SIL_SUPPR this parameter determines the silence suppression method used for all channels which are configured to suppress silence with this profile. Otherwise this parameter is ignored.

Direction: IN Type: ULONG

Default: MT90502_SIMPLE_SIL_SUPPR

additional_delay 0 – 25 (ms)

Indicates the additional delay inserted in the transmission direction of the channel to offer a better quality of voice. The additional delay is used as a look-ahead on the channel.

Assume that additional_delay is set to 25 ms, that each AAL2 mini-packet carries 5 EDUs of voice (5 ms), and refer to the figure below:

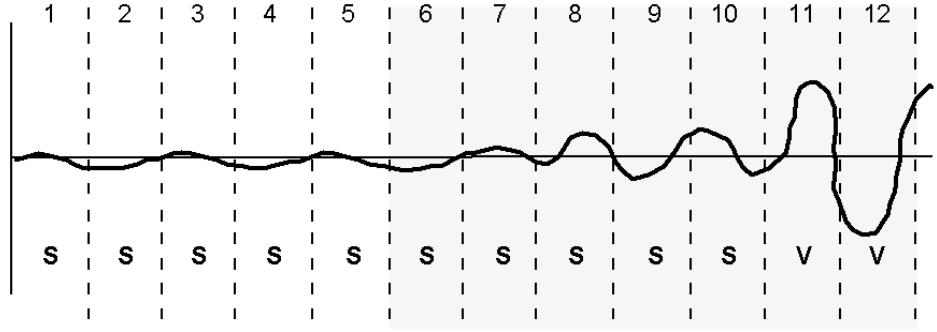


Figure 6 - Silence Suppression Additional Delay

In the figure above the mini-packets 1 – 7 are silent, 8 – 10 are considered close to voice but still silent, and the remaining voice The transmitted mini-packets (i.e. not suppressed) are indicated by the shaded region.

If there is no additional delay inserted in the channel then mini-packets 1 through 10 would surely be suppressed. The transmission of the TDM bytes on UTOPIA would resume with mini-packets 11 and 12 because they are deemed as voice. The problem with this scenario is that the first few mini-packets preceding the voice (8 – 10) are probably the beginning of a

voice spurt. Suppressing these mini-packets gives a “cutting” effect between the silence padding and voice.

If there is additional delay inserted in the channel then the determination of whether a mini-packet is silent or not is based on the state of the mini-packet to follow in additional_delay ms. Thus, setting additional_delay to 25 ms for the example in the figure above implies that mini-packets 6 – 10 are deemed as voice and transmitted. Thus the beginning of the voice spurt, and a few mini-packets of silence preceding it, are also transmitted. This provides a much smoother transition between silence and voice.

The value of this field is a compromise between inserted delay and quality of voice. A greater value implies more delay, where as a smaller value results in a discontinuous transition between silence and voice.

Direction: IN Type: ULONG

Default: 10

silence_to_voice_time 0 - 25 (ms)

At any moment a channel is in one of two silence suppression states: voice or silence. This field determines how much time, in ms, of continuous voice must be observed on the channel before the state of the channel is changed from silence to voice. In silence mode, silence suppression is performed. Typically, this value is kept low so as to cease suppressing as soon as voice is detected on the channel. Thus, a typical value for this field is 0 ms, which would cause any mini-packet exceeding the energy threshold to force the voice state and be sent.

Assume that each AAL2 mini-packet carries 5 EDUs of voice (5 ms), and silence_to_voice_time is set to 5 ms (1 AAL2 mini-packet). Refer to the figure below. The energy level of each mini-packet (i.e. whether it is considered as voice or silence) is indicated at the bottom of the figure. The silence suppression state (i.e. whether a mini-packet is sent or suppressed) is indicated by the shading in the figure: shaded indicates the voice state, and no shading the silent state.

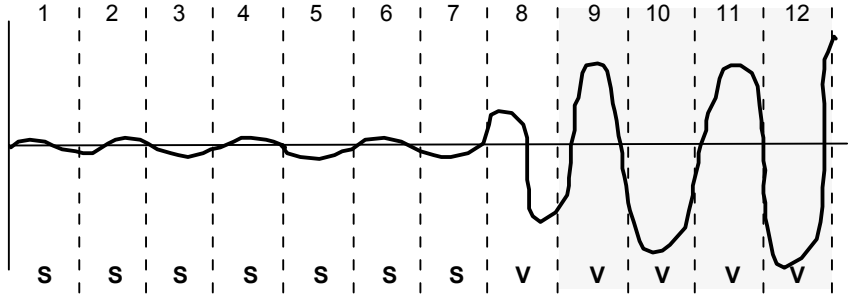


Figure 7 - Silence Suppression Silence to Voice Time

The channel is in silence state at the beginning of this example. Mini-packets 1 – 7 are deemed as silence. Mini-packet 8 is determined to be voice by the configured silence suppression algorithm (simple or complex). Despite this, the state of the channel remains as silent because 0 ms of continuous voice has preceded this mini-packet. Mini-packet 9 however does have 5 ms of voice preceding it (mini-packet 8) is deemed as voice.

The use of this field is to suppress very short periods of voice when the channel is in a silent state. These short periods might result from background noise.

The field is intimately related to additional_delay. In the example above mini-packet 8 is suppressed. If additional delay is inserted into the channel then the silence suppression state

will change sooner (see additional_delay). As a rule of thumb this field should at least be equal to the value of additional_delay. However, the greater additional_delay is with respect to silence_to_voice_time the better the quality of sound. A typical value for this field is 10 ms.

Direction: IN Type: ULONG

Default: 10

voice_to_silence_time 0 - 500 (ms)

At any moment a channel is in one of two silence suppression states: voice or silence. This field determines how much time, in ms, of continuous silence must be observed on the channel before the state of the channel is changed from voice to silence. This value is typically high because silence suppression is desired only when a long amount of continuous silence has been observed. A short amount of silence might represent a short pause between 2 sentences. Suppressing this short period would result in a sharp transition between silence and voice, providing a poor quality of sound. A typical value for this field is 100 ms.

Assume that each AAL2 mini-packet carries 5 EDUs of voice (5 ms), and voice_to_silence_time is set to 15 ms (3 AAL2 mini-packets). Refer to the figure below. The energy level of each mini-packet (i.e. whether it is considered as voice or silence) is indicated at the bottom of the figure. The silence suppression state (i.e. whether a mini-packet is sent or suppressed) is indicated by the shading in the figure: shaded indicates the voice state, and no shading the silent state.

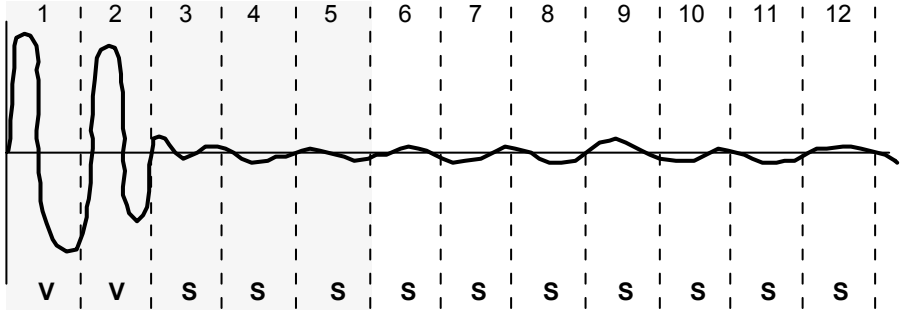


Figure 8 - Silence Suppression Voice to Silence Time

The channel is in voice state at the beginning of this example. Mini-packets 3, 4, and 5 are deemed silent, but the state of the channel remains as silent because the mini-packets do not have 15 ms (3 mini-packets) of continuous silence preceding them. However, as of mini-packet 6 all conditions are met and the state of the channel is switched to silent. Mini-packet will not be sent. Following the change of states from voice to silent a SID packet will be sent.

This field is somewhat related to additional_delay. That is, if additional delay is inserted into the channel then the state of the channel can be switched sooner (see additional_delay). If the state is switched too soon (based on the data to come later on the channel) then part of the voice could be suppressed as well. Thus, voice_to_silence_time must at least be equal to additional_delay. Though, the greater the value of this field the better the quality of voice (recall that, as stated above, short values of this field could suppress short pauses in a conversation). This field will typically be much larger than additional_delay, so the requirement mentioned above will always be met.

When complex silence suppression is used, this field is also related to the sample_length_for_sid_calc field. It must be at least twice as large as

sample_length_for_sid_calc to ensure that no voice is present when the SID energy calculation is done.

Direction: IN Type: ULONG

Default: 100

simple_suppr_method MT90502_ASSOC_TSST
MT90502_A_MATCH
MT90502_B_MATCH
MT90502_FORMAT_B_CODEPOINT

The field determines how the state of each sample, silence or voice, is specified. There are 2 methods that can be used to indicate whether a sample should be treated as voice or silence by the MT90502. A bit of the associated odd stream of the channel's TSST may be used where a '1' in that bit indicates that the sample in the associated TSST is to be treated as silence and a '0' indicates voice (see simple_assoc_tsst_bit). In the second method the received sample is masked and matched to one of two global bytes of the MT90502 where a valid match result indicates that the sample is to be treated as silence. This second method is only useful when ADPCM is employed on the channel and the unused bits of the ADPCM TSST can be used to indicate silence.

The MT90502_FORMAT_B_CODEPOINT method requires all channels using this profile are on streams in Format B (see TDM Configuration parameters of the MT90502_CONF Structure) and the compression mode must be set to MT90502_COMP_AUTO_DETECT. This method allows an external VAD to suppress packets with a single codepoint at the end of the packet and the SID power is calculated from an externally supplied PCM unsigned magnitude value supplied by an external device.

This parameter is only valid when the specified suppression type is simple.

Direction: IN Type: ULONG

Default: MT90502_ASSOC_TSST

sample_length_for_sid_calc 1 – 64 (ms)

The sample length used to calculate the SID energy for the channel. This value must be less than twice the value of voice_to_silence_time. If it isn't then some voice mini-packets will be used in the SID calculation, and thus corrupt the result. A typical value for this field is 25 ms. This parameter is only valid when the specified suppression type is simple.

Direction: IN Type: ULONG

Default: 25

sil_suppr_state_table pointer to table

A pointer to a 2 dimensional array of byte fields. Each dimension of the array is of length 79. The array represents a silence suppression state table, such as the one shown below:

Typical Silence Suppression State Table:

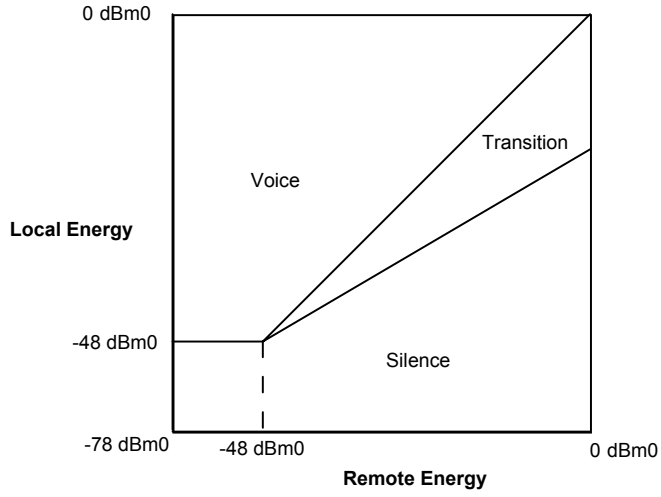


Figure 9 - Typical Silence Suppression State Table

The first dimension is the index of the local energy, and the second the remote energy. For each dimension, index 0 represents the 0 dB energy level, and 78 the -78 dB energy level. Thus, the element table[4][56] is the silence suppression state corresponding to -4dB of local energy and –56 dB remote energy of that table.

The silence suppression table indicates the threshold between voice and silence. Thus, the energy level on the reception side of the channel is used to determine whether a mini-packet to be transmitted on UTOPIA is voice or silence. To change from one state to another the channel's energy must completely cross the transition area. That is, 2 successive mini-packets with energy in the silence and transition areas, respectively, will not cause the silence suppression state of the channel to change.

This parameter is only valid when the specified suppression type is simple.

Each element of the table can have 1 of the following values:

MT90502_STATE_VOICE
MT90502_STATE_TRANSITION
MT90502_STATE_SILENT

Direction: IN/IN Type: POINTER

Default: NULL

5.2 Structure MT90502_CONF_INTERRUPTS

The following parameters determine which events will trigger an interrupt, and how that event will be treated by the API's ISR. See structure MT90502_INT_STRUCT for descriptions of what the interrupts indicate.

fatal_general_conf

MT90502_INT_DISABLE
MT90502_INT_NO_TIMEOUT
MT90502_INT_TIMEOUT

Indicates the configuration of the general fatal interrupt. The interrupt can be disabled from asserting the hardware interrupt pin (MT90502_INT_DISABLE). If the interrupt is enabled, it

can behave in one of two ways once the interrupt has been treated. It can be reset and kept enabled (MT90502_INT_NO_TIMEOUT) or it can be cleared and disabled for a timeout period time (MT90502_INT_TIMEOUT). In the latter case, the timeout period is specified by the fatal_general_timeout parameter. The configuration of this interrupt can be changed dynamically, see mt90502_configure_interrupts.

Direction: IN Type: ULONG

Default: MT90502_INT_NO_TIMEOUT

fatal_ssrama_parity_conf see **fatal_general_conf**

Default: MT90502_INT_NO_TIMEOUT

fatal_ssramb_parity_conf see **fatal_general_conf**

Default: MT90502_INT_NO_TIMEOUT

fatal_sdrama_parity_conf see **fatal_general_conf**

Default: MT90502_INT_NO_TIMEOUT

fatal_sdramb_parity_conf see **fatal_general_conf**

Default: MT90502_INT_NO_TIMEOUT

data_err_sdrama_too_late_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

data_err_sdramb_too_late_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

data_err_utopia_parity_a_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

data_err_utopia_parity_b_conf see **fatal_general_conf**

Default MT90502_INT_TIMEOUT

data_err_utopia_parity_c_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

error_phy_alarm_a_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

error_phy_alarm_b_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

error_rxsar_cell_loss_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

error_txa_cell_loss_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

error_txb_cell_loss_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

error_txc_cell_loss_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
error_mini_pkt_fifo_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
error_rx_data_cell_fifo_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
error_rx_event_fifo_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
error_adap_a_fifo_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
error_adap_b_fifo_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
h100_error_out_of_sync_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
h100_error_clk_a_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
h100_error_frame_a_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
h100_error_clk_b_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
h100_error_frame_b_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
hdlc_error_misaligned_flag_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
hdlc_error_bad_idle_code_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
hdlc_error_long_packet_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
hdlc_error_short_packet_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT
alarm_li_uui_change_event_conf	see **fatal_general_conf**
Default:	MT90502_INT_TIMEOUT

alarm_mini_pkt_rcvd_event_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

alarm_cid_event_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

alarm_data_cell_fifo_int_conf see **fatal_general_conf**

Default: MT90502_INT_TIMEOUT

api_sync_conf MT90502_INT_DISABLE
MT90502_INT_NO_TIMEOUT

This configuration bit is provided for debug purposes. This interrupt is used by the API to maintain synchronization with the device.

Default: MT90502_INT_NO TIMEOUT

fatal_general_timeout 10 – 10000 ms

This parameter specifies the timeout period of the fatal_general interrupt when the fatal_general_conf parameter specifies MT90502_INT_TIMEOUT. This parameter should be a multiple of the 10 ms. If not, it is rounded up to the next nearest multiple of 10 ms before being applied.

Direction: IN Type: ULONG

Default: 1000

fatal_ssrama_parity_timeout see **fatal_general_timeout**

Default: 1000

fatal_ssramb_parity_timeout see fatal_general_timeout

Default: 1000

fatal_sdrama_parity_timeout see **fatal_general_timeout**

Default: 1000

fatal_sdramb_parity_timeout see **fatal_general_timeout**

Default: 1000

data_err_sdrama_too_late_timeout see **fatal_general_timeout**

Default: 1000

data_err_sdramb_too_late_timeout see **fatal_general_timeout**

Default: 1000

data_err_utopia_parity_a_timeout see **fatal_general_timeout**

Default: 1000

data_err_utopia_parity_b_timeout see **fatal_general_timeout**

Default: 1000

data_err_utopia_parity_c_timeout	see **fatal_general_timeout**
Default:	1000
data_err_scheduler_bw_timeout	see **fatal_general_timeout**
Default:	1000
error_phy_alarm_a_timeout	see **fatal_general_timeout**
Default:	1000
error_phy_alarm_b_timeout	see **fatal_general_timeout**
Default:	1000
error_rxsar_cell_loss_timeout	see **fatal_general_timeout**
Default:	1000
error_txa_cell_loss_timeout	see **fatal_general_timeout**
Default:	1000
error_txb_cell_loss_timeout	see **fatal_general_timeout**
Default:	1000
error_txc_cell_loss_timeout	see **fatal_general_timeout**
Default:	1000
error_mini_pkt_fifo_timeout	see **fatal_general_timeout**
Default:	1000
error_rx_data_cell_fifo_timeout	see **fatal_general_timeout**
Default:	1000
error_rx_event_fifo_timeout	see **fatal_general_timeout**
Default:	1000
error_adap_a_fifo_timeout	see **fatal_general_timeout**
Default:	1000
error_adap_b_fifo_timeout	see **fatal_general_timeout**
Default:	1000
h100_error_out_of_sync_timeout	see **fatal_general_timeout**
Default:	10
h100_error_clk_a_timeout	see **fatal_general_timeout**
Default:	10
h100_error_frame_a_timeout	see **fatal_general_timeout**
Default:	10

h100_error_clk_b_timeout see **fatal_general_timeout**

Default: 10

h100_error_frame_b_timeout see **fatal_general_timeout**

Default: 10

hdlc_error_misaligned_flag_timeout see **fatal_general_timeout**

Default: 1000

hdlc_error_bad_idle_code_timeout see **fatal_general_timeout**

Default: 1000

hdlc_error_long_packet_timeout see **fatal_general_timeout**

Default: 1000

hdlc_error_short_packet_timeout see **fatal_general_timeout**

Default: 1000

alarm_li_uui_change_event_timeout 100-100000

This parameter specifies the timeout period of the alarm_li_uui_change_event interrupt when the alarm_li_uui_change_event_conf parameter specifies MT90502_INT_TIMEOUT. This parameter should be a multiple of the 100 us. If not, it is rounded up to the next nearest multiple of 100 us before being applied. For values larger than 10 ms it is rounded up to the next nearest multiple of 1 ms before being applied.

Direction: IN Type: ULONG

Default: 500

alarm_mini_pkt_rcvd_event_timeout see **alarm_li_uui_change_event_timeout**

Default: 500

alarm_cid_event_timeout see **alarm_li_uui_change_event_timeout**

Default: 500

alarm_data_cell_fifo_timeout see **alarm_li_uui_change_event_timeout**

Default: 500

For more information about all Zarlink products
visit our Web Site at

www.zarlink.com

TECHNICAL DOCUMENTATION - NOT FOR RESALE