# MT90528

# API Specification

**Part Number:** MT90528

**Revision Number:** **1.2**

**Issue Date:** **January 2004**

ZARLINK™ SEMICONDUCTOR

**PURCHASE OF THIS PRODUCT DOES NOT GRANT THE PURCHASER ANY RIGHTS UNDER PATENT NO. 5,260,978. USE OF THIS PRODUCT OR ITS RE-SALE AS A COMPONENT OF ANOTHER PRODUCT MAY REQUIRE A LICENSE UNDER THE PATENT WHICH IS AVAILABLE FROM TELCORDIA TECHNOLOGIES, INC., 445 SOUTH STREET, MORRISTOWN, NEW JERSEY 07960.**

**ZARLINK ASSUMES NO RESPONSIBILITY OR LIABILITY THAT MAY RESULT FROM ITS CUSTOMERS' USE OF ZARLINK PRODUCTS WITH RESPECT TO THIS PATENT. IN PARTICULAR, ZARLINK'S PATENT INDEMNITY IN ITS TERMS AND CONDITIONS OF SALES WHICH ARE SET OUT IN ITS SALES ACKNOWLEDGEMENTS AND INVOICES DOES NOT APPLY TO THIS PATENT.**

# Table of Contents

**1.0 Overview . . . 6**
1.1 Software Revision . . . 6
1.2 System Architecture . . . 6
**2.0 API Function Descriptions . . . 9**
2.1 Initialization Functions . . . 9
2.1.1 Mt90528InitializeApi . . . 9
2.1.2 Mt90528ShutDownApi . . . 10
2.2 ATM Functions . . . 11
2.2.1 Mt90528ConfigureDeviceForUdt . . . 12
2.2.2 Mt90528ConfigureDeviceForSdt . . . 15
2.2.3 Mt90528ConfigureDeviceForUdtAndSdt . . . 17
2.2.4 Mt90528ConfigureDataRxsarModule . . . 20
2.2.5 Mt90528ReadDataRxsarBuffer . . . 21
2.2.6 Mt90528EnableDataRxsar . . . 22
2.2.7 Mt90528DisableDataRxsar . . . 22
2.2.8 Mt90528ConfigureDataTxsarModule . . . 23
2.2.9 Mt90528InitializeDataTxsarWritePointer . . . 24
2.2.10 Mt90528WriteDataTxsarBuffer . . . 24
2.2.11 Mt90528EnableDataTxsar . . . 25
2.2.12 Mt90528DisableDataTxsar . . . 26
2.2.13 Mt90528ConfigureTdmPortForUdt . . . 27
2.2.14 Mt90528DeprogramTdmPortForUdt . . . 28
2.2.15 Mt90528ConfigureTdmPortForSdt . . . 30
2.2.16 Mt90528DeprogramTdmPortForSdt . . . 32
2.2.17 Mt90528SetTdmPortLinkType . . . 33
2.2.18 Mt90528TurnOnUtopiaInterfaceLoopback . . . 34
2.2.19 Mt90528TurnOffUtopiaInterfaceLoopback . . . 35
2.2.20 Mt90528ConfigureTdmPortForLowLatencyLoopback . . . 35
2.2.21 Mt90528DeprogramTdmPortForLowLatencyLoopback . . . 37
2.2.22 Mt90528ConfigureTdmPortForCircularBufferLoopback . . . 37
2.2.23 Mt90528DeprogramTdmPortForCircularBufferLoopback . . . 39
2.2.24 Mt90528EnableTxTdmForUdt . . . 40
2.2.25 Mt90528DisableTxTdmForUdt . . . 41
2.2.26 Mt90528EnableTxVcForSdt . . . 41
2.2.27 Mt90528DisableTxVcForSdt . . . 42
2.2.28 Mt90528OpenTxVcForUdt . . . 43
2.2.29 Mt90528CloseTxVcForUdt . . . 44
2.2.30 Mt90528OpenRxVcForUdt . . . 45
2.2.31 Mt90528CloseRxVcForUdt . . . 47
2.2.32 Mt90528OpenTxVcForSdt . . . 48
2.2.33 Mt90528CloseTxVcForSdt . . . 50
2.2.34 Mt90528OpenRxVcForSdt . . . 51
2.2.35 Mt90528CloseRxVcForSdt . . . 54
2.2.36 Mt90528EnableTxsar . . . 55
2.2.37 Mt90528OpenRxDataVc . . . 56
2.2.38 Mt90528CloseRxDataVc . . . 56
2.2.39 Mt90528UpdateCutVcPeriod . . . 57
2.2.40 Mt90528UpdateCellDelayVariation . . . 58
2.2.41 Mt90528UpdateReplayNSilence . . . 60
2.3 I/O Functions . . . 61
2.3.1 Mt90528WriteExternalMemory . . . 61
2.3.2 Mt90528WriteNoVerifyExternalMemory . . . 62

# Table of Contents


2.3.3 Mt90528ReadExternalMemory . . . 62
2.3.4 Mt90528WriteInternalMemory . . . 63
2.3.5 Mt90528WriteNoVerifyInternalMemory . . . 64
2.3.6 Mt90528ReadInternalMemory . . . 65
2.3.7 Mt90528WriteRegister . . . 65
2.3.8 Mt90528ReadRegister . . . 66
2.4 Interrupt Service Handler . . . 67
2.4.1 Mt90528IsrHandler . . . 67
2.4.2 Mt90528IsrHandlerWith2ndThresholds . . . 69
2.4.3 Mt90528EnableDisableInterrupt . . . 71
2.5 MIB and Performance Management Statistics and Status . . . 73
2.5.1 Statistics Functions . . . 73
2.5.1.1 Mt90528GetDeviceStatistics . . . 73
2.5.1.2 Mt90528GetTdmPortStatistics . . . 74
2.5.1.3 Mt90528GetCbrVcStatistics . . . 75
2.5.1.4 Mt90528GetCbrVcStatisticsWithAppThresholds . . . 76
2.5.1.5 Mt90528EnableDisableCbrVcStatisticsCollection . . . 77
2.5.1.6 Mt90528GetCbrVcStatIndex . . . 78
2.5.1.7 Mt90528SetMibThresholds . . . 78
2.5.1.8 Mt90528SetMibThresholdsWithAppThresholds . . . 79
2.5.1.9 Mt90528SetMibCounters32 . . . 80
2.5.1.10 Mt90528ReadDeviceMibCounters . . . 81
2.5.1.11 Mt90528LoadDeviceMibCounters . . . 82
2.5.1.12 Mt90528ReadUdtRxsarCsVcArrival . . . 83
2.5.1.13 Mt90528ReadSdtRxsarCsVcArrival . . . 83
2.5.1.14 Mt90528ClearUdtRxsarCsVcArrival . . . 84
2.5.1.15 Mt90528ClearSdtRxsarCsVcArrival . . . 84
2.5.1.16 Mt90528ScanForCellLossTimeoutOnVcs . . . 85
2.5.2 Status Functions . . . 86
2.5.2.1 Mt90528GetNextTdmPortStatusIndex . . . 86
2.5.2.2 Mt90528GetNextCbrVcStatusIndex . . . 86
2.5.2.3 Mt90528GetNextCbrVcCellLossStatusIndex . . . 87
2.5.3 Secondary Threshold and Counter Functions . . . 88
2.5.3.1 Mt90528InitializeThresholdTable . . . 88
2.5.3.2 Mt90528AddVcToCbrVcThresholdTable . . . 88
2.5.3.3 Mt90528RemoveVcFromCbrVcThresholdTable . . . 89
2.5.3.4 Mt90528GetCbrVcThresholdIndex . . . 89
2.5.3.5 Mt90528SetAppThresholds . . . 90
2.5.3.6 Mt90528SetAppCounters32 . . . 91
2.5.3.7 Mt90528UpdateAppCounters32 . . . 92
2.5.3.8 Mt90528GetCbrVcAppStatistics . . . 93
2.5.3.9 Mt90528EnableDisableCbrVcAppStatisticsCollection . . . 93
2.5.4 Application Examples using Statistics Counters and Thresholds . . . 94
2.5.4.1 MIB statistics only with no Thresholds set . . . 94
2.5.4.2 MIB statistics only with MIB Thresholds set . . . 95
2.5.4.3 MIB and APP statistics with no Thresholds set . . . 97
2.5.4.4 MIB and APP statistics with MIB and APP Thresholds set . . . 99
**3.0 Constants . . . 102**
**4.0 Configuration Structures . . . 110**
4.1 Structure s_vc . . . 110
4.2 Structure s_device_common_params . . . 110
4.3 Structure s_device_udt_specific_params . . . 111

# Table of Contents

4.4 Structure s_device_sdt_specific_params . . . 111
4.5 Structure s_device_udt_params . . . 112
4.6 Structure s_device_sdt_params . . . 112
4.7 Structure s_device_udt_and_sdt_params . . . 113
4.8 Structure s_data_cell_buffer_params . . . 113
4.9 Structure s_data_tx_sar_enable_params . . . 113
4.10 Structure s_data_rx_sar_enable_params . . . 113
4.11 Structure s_tdm_port_common_params . . . 114
4.12 Structure s_tdm_port_udt_specific_params . . . 114
4.13 Structure s_tdm_port_sdt_specific_params . . . 115
4.14 Structure s_tdm_port_udt_params . . . 116
4.15 Structure s_tdm_port_sdt_params . . . 116
4.16 Structure s_tdm_loopback_params . . . 116
4.17 Structure s_atm_channel . . . 116
4.18 Structure s_atm_channels . . . 117
4.19 Structure s_atm_cell_header_params . . . 117
4.20 Structure s_tx_vc_udt_params . . . 117
4.21 Structure s_tx_vc_sdt_params . . . 117
4.22 Structure s_rx_vc_udt_params . . . 118
4.23 Structure s_rx_vc_sdt_params . . . 118
4.24 Structure s_memory_chunk . . . 119
4.25 Structure a_internal_memory_chunks . . . 119
4.26 Structure a_external_memory_chunks . . . 120
**5.0 Statistics, Status and Threshold Structures . . . 120**
5.1 Structure s_mt90528_device_stats_entry . . . 120
5.2 Structure s_mt90528_tdm_port_stats_entry . . . 121
5.3 Structure s_mt90528_cbr_vc_stats_entry . . . 122
5.4 Structure s_mt90528_statistics . . . 123
5.5 Structure s_stats_flags . . . 123
5.6 Structure s_hw_udt . . . 124
5.7 Structure s_hw_sdt . . . 124
5.8 Structure s_hw_dev . . . 125
5.9 Structure s_hw_load . . . 125
5.10 Structure s_stats_32 . . . 125
5.11 Structure s_stats_params . . . 125
5.12 Structure s_mt90528_device_status_entry . . . 126
5.13 Structure s_mt90528_tdm_port_status_entry . . . 126
5.14 Structure s_mt90528_cbr_vc_status_entry . . . 127
5.15 Structure s_mt90528_status . . . 130
5.16 Structure s_mt90528_cbr_vc_cell_loss_status_entry . . . 130
5.17 Structure s_mt90528_cbr_vc_cell_loss_status . . . 130
5.18 Structure s_mt90528_cbr_vc_statistic_threshold_entry . . . 131
5.19 Structure s_mt90528_threshold . . . 132
**6.0 Glossary . . . 133**

# 1.0 Overview

This document is intended as a reference guide of API calls and structures for the MT90528 family of AAL1 SAR devices.

## 1.1 Software Revision

The MT90528 software API referred to in this document is at release 3.0

## 1.2 System Architecture

There are six components to the MT90528 API: API, ATM, ISR, MIB, MEM and I/O. The API component contains the generic API routines not specifically related to ATM, ISR, MIB, MEM or I/O which includes the interfaces for initializing and shutting down the API. The ATM part consists of interfaces for configuring the UTOPIA interface and TDM ports and opening and closing virtual channels. For release 3.x & up of the API, all functions are in the context of UDT or SDT modes. The ISR part consists of interfaces which allow for handling of the various categories of device interrupts. The MIB part consists of interfaces that handle statistics and thresholds. The MEM part is internal to the API and these interfaces are used by the ATM component. The I/O part consists of routines for reading and writing device registers and internal and external memory. Figure 1 shows the system architecture indicating the API and the supporting low-level software:

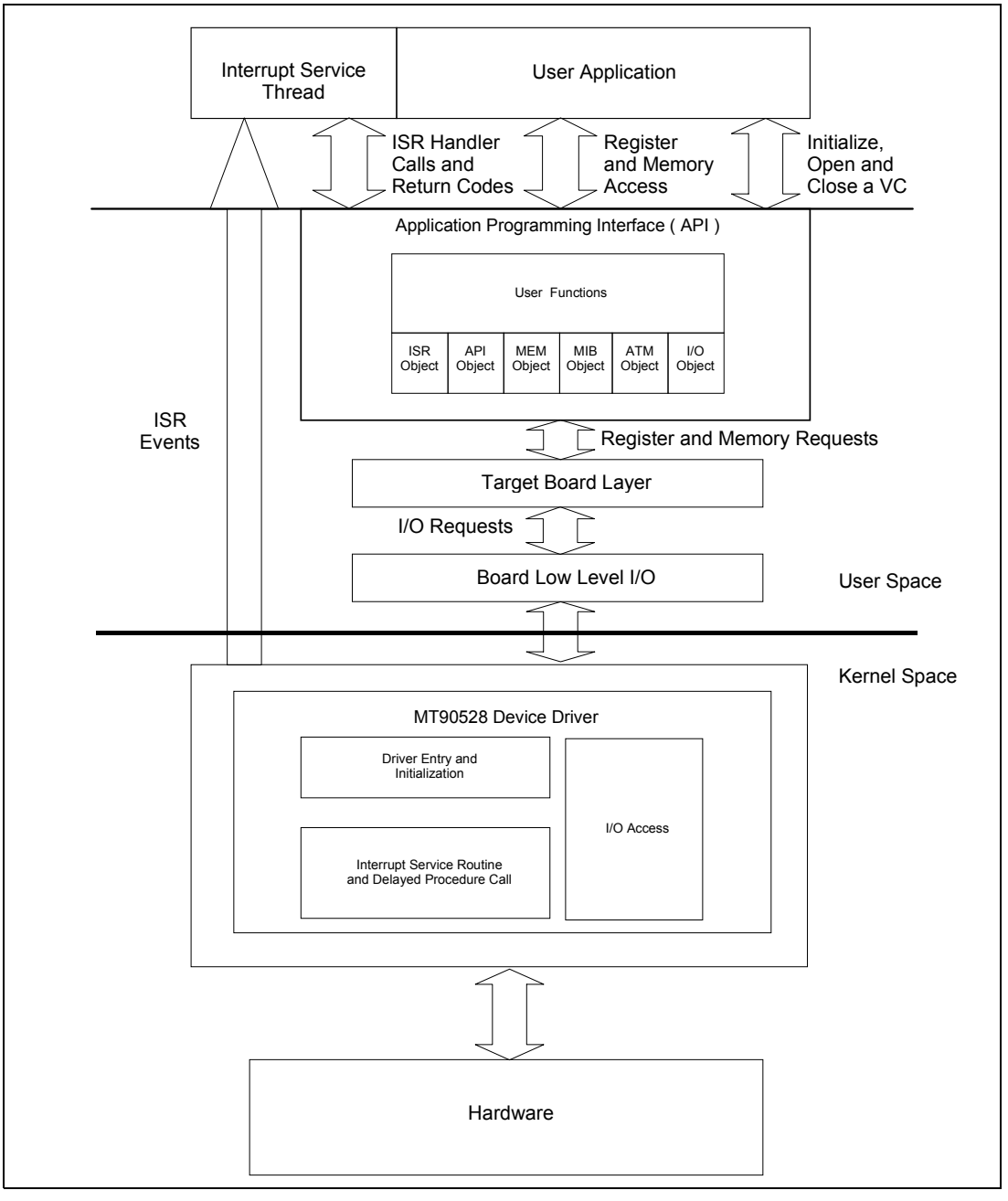


**Figure 1 - System Overview**

| Module | Description |
|---|---|
| User Application | The user application code has to be developed by the user of the API. Example code has been provided on how to initialize the MT90528 device, how to open a VC or multiple VCs, how to close a VC, how to spawn an interrupt service thread and how to shut down the device. |
| Interrupt Service Thread | The interrupt service thread code has to be developed by the user of the API. Example code has been provided on how to create plus wait for an interrupt event and how to service the interrupt events. Some of this code may need to be moved down to the device driver interrupt service routine due to performance issues for real time operating systems. |
| User Functions | These are the API calls that the user application is using. This module is the external interface of the API. |
| API Object | This module in the API initializes the MT90528 device to allow writing to and reading from registers and memory. Specific interrupts are enabled. The module also shuts down the MT90528 device (closes the device driver). |
| ATM Object | This module in the API initializes the device and TDM ports. The module also opens and closes CBR and Data VCs. |
| MIB Object | This module deals with the statistics structures and MIB counters and flags. |
| MEM Object | This module is internal to the API and is used to manage internal and external memory. |
| ISR Object | This module in the API has routines that handle the MT90528 device interrupts for each of the functional modules on the device (e.g. TXSAR, SDT RXSAR) |
| I/O Object | This module in the API handles the register and memory input/output accesses on the MT90528 device. |
| Target Board Layer | This layer handles the input and output accesses for the MT90528 device and the board on which the MT90528 device is used. All of the board-specific definitions reside in this layer. |
| Board Low Level I/O | This layer handles the board address-specific input and output accesses to and from any customer device driver. It also has utility routines for opening and closing a device driver, creating and closing interrupt events on the device driver and acquiring a device driver handle/descriptor. |
| Device Driver Entry and Initialization | The device driver is initialized to reset the board and MT90528 device. The board and MT90528 device interrupts are disabled. The device driver interrupt service routine is enabled. |
| Device Driver I/O Access | The device driver input/output access module provides reading from and writing to the hardware on the board and MT90528 device either using port I/O or memory I/O ISA access. |

**Table 1 - System Overview Module Description**

| Module | Description |
|---|---|
| Device Driver Interrupt Service Routine | The device driver interrupt service routine services the board and MT90528 device hardware interrupts, checking for valid MT90528 device interrupts and initiating an interrupt event to the user's interrupt service thread. |
| Hardware | Hardware accesses to the board and MT90528 device. |

**Table 1 - System Overview Module Description**

# 2.0 API Function Descriptions

## 2.1 Initialization Functions

Initialization functions prepare the chip for operation.

***Mt90528InitializeApi*** is called from the user application main program before any other calls to the API are made. ***Mt90528ShutDownApi*** is called from the main program to clean up the API when the application is shut down.

***Mt90528EnableDisableInterrupt*** is called from ***Mt90528InitializeApi*** to initially enable all categories of interrupts on the device. **Mt90528EnableDisableInterrupt** can be called from the user application to enable/disable specific categories of interrupts.

**NOTE**: the application has to define variables using the structures s_mt90528_statistics, a_internal_memory_chunks and a_external_memory_chunks in order to use the following functions. These common variables are also used by the ATM, ISR and MIB functions.

### 2.1.1 Mt90528InitializeApi

**Prototype**

```
USHORT Mt90528InitializeApi(
      UINT                    Device,
      s_mt90528_statistics    * pStatistics,
      s_memory_chunk          * pIntChunkArray,
      s_memory_chunk          * pExtChunkArray
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pStatistics* | pointer to MIB statistics structure.<br>See 5.4, "Structure s_mt90528_statistics". |
| *pIntChunkArray* | pointer passed by caller of memory manager internal chunk array.<br>See 4.24, "Structure s_memory_chunk". |
| *pExtChunkArray* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, "Structure s_memory_chunk". |

**Return values**

| | |
|---|---|
| *MT90528_API_NO_ERROR* | API is initialized |
| *MT90528_API_INIT_WRITE_ERROR* | failure writing to the chip |
| *MT90528_API_INIT_READ_ERROR* | failure reading from the chip |
| *MT90528_API_INIT_IRQ_ERROR* | failure enabling/disabling device interrupts |
| *MT90528_API_INIT_MEM_MAN_ERROR* | failure to initialize the memory manager |

**Description**

This routine

- initializes the MIB statistics structures
- resets each device on the board
- queries the device revision register and updates the DeviceRev field in the device statistics structure
- initializes the Memory Arbiter Configuration Register with the board specific memory configuration and memory type information for each device on the board
- clears the device internal and external memory (memory clearing takes time)
- disables all PLLs
- enables device specific module interrupts (not TDM port or VC interrupts)
- enables all categories of main interrupts
- initializes the memory manager using the passed in pointers to user created memory chunk arrays in CPU memory (see mem90528.h for memory chunk array structures to define)

**Source Files**

api90528.h, api90528.c

### 2.1.2 Mt90528ShutDownApi

**Prototype**

```
void Mt90528ShutDownApi(
      s_mt90528_statistics    * pStatistics,
      s_memory_chunk          * pIntChunkArray,
      s_memory_chunk          * pExtChunkArray
      )
```

**Parameters**

| | |
|---|---|
| *pStatistics* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunkArray* | pointer passed by caller of memory manager internal chunk array.<br>See 4.24, “Structure s_memory_chunk“. |
| *pExtChunkArray* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, “Structure s_memory_chunk“. |

**Return values**

No return values.

**Description**

The routine

- cleans up after the memory manager
- cleans up the MIB statistics structures

**Source File**

api90528.h, api90528.c

## 2.2 ATM Functions

ATM routines are used to configure the UTOPIA interface and TDM ports and to open and close TX and RX virtual channels. The ATM module also contains routines which support the Data RX_SAR and Data TX_SAR functions of the chip. Routines are available for programming the TDM port and UTOPIA interface loopback modes.

For debugging and board testing the following routines allow the various loopback modes to be turned on and off:

1. ***Mt90528TurnOnUtopiaInterfaceLoopback*, *Mt90528TurnOffUtopiaInterfaceLoopback***
2. ***Mt90528ConfigureTdmPortForLowLatencyLoopback*, *Mt90528DeprogramTdmPortForLowLatencyLoopback***
3. ***Mt90528ConfigureTdmPortForCircularBufferLoopback*, *Mt90528DeprogramTdmPortForCircularBufferLoopback***

For basic ATM cell transmission and reception, the order in which the user application should call the routines is:

1. ***Mt90528ConfigureDeviceForUdt*** or ***Mt90528ConfigureDeviceForSdt*** or ***Mt90528ConfigureDeviceForUdtAndSdt***
2. ***Mt90528ConfigureTdmPortForUdt*** for each UDT port and ***Mt90528ConfigureTdmPortForSdt*** for each SDT port
3. ***Mt90528EnableTxsar***
4. ***Mt90528OpenTxVcForUdt*** for each UDT mode TX VC to be opened
5. ***Mt90528OpenRxVcForUdt*** for each UDT mode RX VC to be opened
6. ***Mt90528OpenTxVcForSdt*** for each SDT mode TX VC to be opened
7. ***Mt90528OpenRxVcForSdt*** for each SDT mode RX VC to be opened
8. ***Mt90528CloseTxVcForUdt*** and ***Mt90528CloseRxVcForUdt*** to close UDT VCs
9. ***Mt90528CloseTxVcForSdt*** and ***Mt90528CloseRxVcForSdt*** to close SDT VCs

For non-CBR data cell transmission these routines should be called sometime after configuring the device:

1. ***Mt90528ConfigureDataTxsarModule*** to allocate memory for the TXSAR data buffer and initialize the Data TXSAR Registers
2. ***Mt90528InitializeDataTxsarWritePointer*** to set the write pointer to the beginning of the buffer
3. ***Mt90528WriteDataTxsarBuffer*** to write one cell to the data buffer
4. ***Mt90528EnableDataTxsar*** to transmit the contents of the data buffer
5. ***Mt90528DisableDataTxsar*** to disable the Data TXSAR Module

For data cell reception these routines should be called sometime after configuring the device:

1. **Mt90528ConfigureDataRxsarModule** to allocate memory for the RXSAR data buffer and initialize the Data RXSAR Registers
2. ***Mt90528EnableDataRxsar*** to enable the Data RXSAR Module
3. ***Mt90528OpenRxDataVc*** to receive F4 OAM cells for a particular VC
4. ***Mt90528CloseRxDataVc*** to stop receiving F4 OAM cells for a particular VC
5. ***Mt90528ReadDataRxsarBuffer*** to read one cell from the data buffer
6. ***Mt90528DisableDataRxsar*** to disable the Data RXSAR Module

Use the following routines to deprogram ports:

1. ***Mt90528DeprogramTdmPortForUdt***
2. ***Mt90528DeprogramTdmPortForSdt***

Other routines are:

1. ***Mt90528SetTdmPortLinkType*** will change the link type of a port
2. ***Mt90528EnableTxTdmForUdt***/***Mt90528DisableTxTdmForUdt*** to enable/disable transmission of UDT cells for a particular port
3. ***Mt90528EnableTxVcForSdt***/***Mt90528DisableTxVcForSdt*** to enable/disable transmission of cells for a particular SDT VC
4. ***Mt90528UpdateCutVcPeriod*** to change the value of the Cut Vc Period for a particular device
5. ***Mt90528UpdateCellDelayVariation*** to change the cell delay variation for a particular VC
6. ***Mt90528UpdateReplayNSilence*** to change the replay and silence setting for a particular SDT RX VC

**NOTE**: the application has to define variables using the structures s_mt90528_statistics, a_internal_memory_chunks and a_external_memory_chunks in order to use the following functions. These common variables are also used by the API, ISR and MIB functions.

### 2.2.1 Mt90528ConfigureDeviceForUdt

**Prototype**

```
USHORT Mt90528ConfigureDeviceForUdt (
      UINT                    Device,
      s_device_udt_params     * pParams,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to a structure containing user-programmable UDT settings for the MT90528 device. See 4.5, “Structure s_device_udt_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array. See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | device is configured for UDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | primary or secondary clock source port is out of range |
| *MT90528_ATM_CHUNK_OUT_OF_RANGE* | memory chunk is out of range |
| *MT90528_ATM_DEVICE_ADDRESS_NEEDS_PHY_MODE* | device address only applicable to phy mode |
| *MT90528_ATM_DEVICE_ADDRESS_OUT_OF_RANGE* | device address is out of range |
| *MT90528_ATM_LUT_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for look-up table |
| *MT90528_ATM_M_PLUS_N_OUT_OF_RANGE* | M and N values are out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine configures the device for UDT mode and if necessary allocates memory for and initializes the entries in the look-up table based on user specified values for the number of VPI and VCI concatenation bits. The LUT will be initialized if external memory is present.

Parameters are verified as follows:

- the device id is in range
- the primary and secondary PLL clock sources specified are valid port ids
- a non-zero device address is only specified with phy mode
- the device address specified is <= 30
- the chunk of external memory specified for the look-up table is <= MAX_ADDRESS_EXT_MEMORY
- the user specified values for the VCI and VPI number of concatenation bits are verified to be in range, i.e., *Params.DeviceCommonParams.UncbM* in [0..12], *Params.DeviceCommonParams.UncbN* in [4..16], and *Params.DeviceCommonParams.UncbM* + *Params.DeviceCommonParams.UncbN* <= 16

The UTOPIA Configuration Register fields are configured as follows:

- SDT_N_UDT is set to UDT mode
- UNI_N_NNI is set to *Params.DeviceCommonParams.UcrUniNNni*
- UTO_CLK_SEL is set to *Params.DeviceCommonParams.UcrUtoClkSel*
- GLOBAL_OAM_SEL is set to *Params.DeviceCommonParams.UcrGlobalOamSel*
- EIGHT_N_SIXTEEN is set to *Params.DeviceCommonParams.UcrEightNSixteen*
- LEVEL1_N_LEVEL2 is set to *Params.DeviceCommonParams.UcrLevel1NLevel2*
- UKSEL is set to *Params.DeviceCommonParams.UcrUkSel*
- PHY_N_ATM is set to *Params.DeviceCommonParams.UcrPhyNAtm*
- DEVICE_ADDRESS is set to *Params.DeviceCommonParams.UcrDeviceAddress*

The Clock Management Configuration Register fields are configured as follows:

- SLV_N_MSTR is set to slave mode
- FNXI1_RATE and FNXI2_RATE are set to 2.43 MHz
- EXT_N_INT is set to *Params.DeviceCommonParams.CmcrExtNInt*
- 8_KHZ_SEL is set to *Params.DeviceSdtSpecificParams.Cmcr8KhzSel*

The External PLL Clock Source Register fields are configured as follows:

- PRI_SEL bits<4:0> are set to *Params.DeviceCommonParams.PllcsPriClkSrcPrt*
- PRI_SEL bit<5> is set to *Params.DeviceCommonParams.PllcsPriClkSrcTyp*
- SEC_SEL bits<4:0> are set to *Params.DeviceCommonParams.PllcsSecClkSrcPrt*
- SEC_SEL bit<5> is set to *Params.DeviceCommonParams.PllcsSecClkSrcTyp*

The VC Match and Match Enable and the VP Match and Match Enable Registers are set to *Params.DeviceCommonParams.VcMatch*, *Params.DeviceCommonParams.VcMatchEnable*, *Params.DeviceCommonParams.VpMatch*, and *Params.DeviceCommonParams.VpMatchEnable* respectively

The look-up table is configured, if and only if external memory is present, as follows:

- $2^{(M + N + 1)}$ bytes are allocated for the look-up table in the chunk of external memory specified by *Params.DeviceCommonParams.ExtMemChunk* where M is *Params.DeviceCommonParams.UncbM* and N is *Params.DeviceCommonParams.UncbN*. If 128 Kbytes of memory are required for the look-up table then *ExtMemChunk* must specify the first of two consecutive empty chunks of memory.
- the look-up table is initialized so that every entry has its Destination of Received Cell field set to LUT_DS_DISCARDED
- the UTOPIA Number of Concatenated bits Register is programmed with *Params.DeviceCommonParams.UncbN* and *Params.DeviceCommonParams.UncbM*
- the 16 most significant bits of the look-up table byte address are stored in the LUT Base Address Register

The UDT Reassembly Control Register fields are configured as follows:

- UDT_DUMMY is set to *Params.DeviceUdtSpecificParams.UrcrUdtDummy*
- UDT_INSERT_LOST is set to *Params.DeviceUdtSpecificParams.UrcrUdtInsertLost*
- CHECK_LATE_ARRIVALS is set to *Params.DeviceUdtSpecificParams.UrcrCheckLateArrivals*

The MIB Timeout Configuration Register fields are configured as follows:

- CUT_VC_PERIOD is set to the user-specified value in *Params.DeviceCommonParams.MtcrCutVcPeriod*

The incoming and outgoing UTOPIA FIFOs are cleared.

The UTO_CLK_CONFIGURED field of the UTOPIA Configuration Register is set at least one clock cycle after setting the UTO_CLK_SEL bit.

**Source Files**

atm90528.h, atm90528.c

### 2.2.2 Mt90528ConfigureDeviceForSdt

**Prototype**

```
USHORT Mt90528ConfigureDeviceForSdt (
      UINT                    Device,
      s_device_sdt_params     * pParams,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks,
      s_memory_chunk          * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to a structure containing user-programmable SDT settings for the MT90528 device. See 4.6, “Structure s_device_sdt_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array. See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | device is configured for SDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | primary or secondary clock source port is out of range |
| *MT90528_ATM_CHUNK_OUT_OF_RANGE* | memory chunk out of range |
| *MT90528_ATM_DEVICE_ADDRESS_NEEDS_PHY_MODE* | device address only applicable to phy mode |
| *MT90528_ATM_DEVICE_ADDRESS_OUT_OF_RANGE* | device address is out of range |
| *MT90528_ATM_LUT_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for look-up table |
| *MT90528_ATM_M_PLUS_N_OUT_OF_RANGE* | M and N values out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine configures the device for SDT mode and allocates memory for and initializes the entries in the look-up table based on user specified values for the number of VPI and VCI concatenation bits.

Parameters are verified as follows:

- the device id is in range
- the primary and secondary PLL clock sources specified are valid port ids
- a non-zero device address is only specified with phy mode
- the device address specified is <= 30

- the chunk of external memory specified for the look-up table is <= MAX_ADDRESS_EXT_MEMORY
- the user specified values for the VCI and VPI number of concatenation bits are verified to be in range, i.e., *Params.DeviceCommonParams.UncbM* in [0..12], *Params.DeviceCommonParams.UncbN* in [4..16], and *Params.DeviceCommonParams.UncbM* + *Params.DeviceCommonParams.UncbN* <= 16

The UTOPIA Configuration Register fields are configured as follows:

- SDT_N_UDT is set to SDT mode
- UNI_N_NNI is set to *Params.DeviceCommonParams.UcrUniNNni*
- UTO_CLK_SEL is set to *Params.DeviceCommonParams.UcrUtoClkSel*
- GLOBAL_OAM_SEL is set to *Params.DeviceCommonParams.UcrGlobalOamSel*
- EIGHT_N_SIXTEEN is set to *Params.DeviceCommonParams.UcrEightNSixteen*
- LEVEL1_N_LEVEL2 is set to *Params.DeviceCommonParams.UcrLevel1NLevel2*
- UKSEL is set to *Params.DeviceCommonParams.UcrUkSel*
- PHY_N_ATM is set to *Params.DeviceCommonParams.DeviceCommonParams.UcrPhyNAtm*
- DEVICE_ADDRESS is set to *Params.DeviceCommonParams.UcrDeviceAddress*

The Clock Management Configuration Register fields are configured as follows:

- SLV_N_MSTR is set to *Params.DeviceSdtSpecificParams.CmcrSlvNMstr*
- F0_MODE is set to *Params.DeviceSdtSpecificParams.CmcrF0Mode*
- 8_KHZ_SEL is set to *Params.DeviceSdtSpecificParams.Cmcr8KhzSel*
- FNXI1_RATE is set to *Params.DeviceSdtSpecificParams.CmcrFnxi1Rate*
- FNXI2_RATE is set to *Params.DeviceSdtSpecificParams.CmcrFnxi2Rate*
- EXT_N_INT is set to *Params.DeviceCommonParams.CmcrExtNInt*

The External PLL Clock Source Register fields are configured as follows:

- PRI_SEL bits<4:0> are set to *Params.DeviceCommonParams.PllcsPriClkSrcPrt*
- PRI_SEL bit<5> is set to *Params.DeviceCommonParams.PllcsPriClkSrcTyp*
- SEC_SEL bits<4:0> are set to *Params.DeviceCommonParams.PllcsSecClkSrcPrt*
- SEC_SEL bit<5> is set to *Params.DeviceCommonParams.PllcsSecClkSrcTyp*

The VC Match and Match Enable and the VP Match and Match Enable Registers are set to Params.DeviceCommonParams.VcMatch, Params.DeviceCommonParams.VcMatchEnable, Params.DeviceCommonParams.VpMatch, and Params.DeviceCommonParams.VpMatchEnable respectively

The look-up table is configured as follows:

- $2^{(M + N + 1)}$ bytes are allocated for the look-up table in the chunk of external memory specified by *Params.DeviceCommonParams.ExtMemChunk* where M is *Params.DeviceCommonParams.UncbM* and N is *Params.DeviceCommonParams.UncbN*. If 128 Kbytes of memory are required for the look-up table then *ExtMemChunk* must specify the first of two consecutive empty chunks of memory.
- the look-up table is initialized so that every entry has its Destination of Received Cell field set to LUT_DS_DISCARDED
- the UTOPIA Number of Concatenated bits Register is programmed with *Params.DeviceCommonParams.UncbN* and *Params.DeviceCommonParams.UncbM*
- the 16 most significant bits of the look-up table byte address are stored in the LUT Base Address Register

The MIB Timeout Configuration Register fields are configured as follows:

- CUT_VC_PERIOD is set to the user-specified value in *Params.DeviceCommonParams.MtcrCutVcPeriod*

The incoming and outgoing UTOPIA FIFOs are cleared.

The SDT Reassembly Control Register fields are configured as follows:

- SDT_DUMMY is set to *Params.DeviceSdtSpecificParams.SrcrSdtDummy*
- SDT_INSERT_LOST is set to *Params.DeviceSdtSpecificParams.SrcrSdtInsertLost*
- CB_BASE_ADD is set to 0

The Main TDM Control Register 1 fields are configured as follows:

- IDLE_DATA is set to *Params.DeviceSdtSpecificParams.Maintdm1IdleData*
- SILENCE_DATA is set to *Params.DeviceSdtSpecificParams.Maintdm1SilenceData*

The Main TDM Control Register 2 fields are configured as follows:

- IDLE_CAS is set to *Params.DeviceSdtSpecificParams.Maintdm2IdleCas*

The UTO_CLK_CONFIGURED field of the UTOPIA Configuration Register is set at least one clock cycle after setting the UTO_CLK_SEL bit.

**Source Files**

atm90528.h, atm90528.c

### 2.2.3 Mt90528ConfigureDeviceForUdtAndSdt

**Prototype**

```
USHORT Mt90528ConfigureDeviceForUdtAndSdt (
      UINT                          Device,
      s_device_udt_and_sdt_params   * pParams,
      s_mt90528_statistics          * pStats,
      s_memory_chunk                * pIntChunks,
      s_memory_chunk                * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to a structure containing user-programmable SDT and UDT settings for the MT90528 device.<br>See 4.7, “Structure s_device_udt_and_sdt_params“. |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array.<br>See 4.24, “Structure s_memory_chunk“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, “Structure s_memory_chunk“ . |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | device is configured for UDT and SDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | primary or secondary clock source port is out of range |
| *MT90528_ATM_CHUNK_OUT_OF_RANGE* | memory chunk is out of range |
| *MT90528_ATM_DEVICE_ADDRESS_NEEDS_PHY_MODE* | device address only applicable to phy mode |
| *MT90528_ATM_DEVICE_ADDRESS_OUT_OF_RANGE* | device address is out of range |
| *MT90528_ATM_LUT_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for look-up table |
| *MT90528_ATM_M_PLUS_N_OUT_OF_RANGE* | M and N values are out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine configures the device for UDT and SDT mode and allocates memory for and initializes the entries in the look-up table based on user specified values for the number of VPI and VCI concatenation bits.

Parameters are verified as follows:

- the device id is in range
- the primary and secondary PLL clock sources specified are valid port ids
- a non-zero device address is only specified with phy mode
- the device address specified is <= 30
- the chunk of external memory specified for the look-up table is <= MAX_ADDRESS_EXT_MEMORY
- the user specified values for the VCI and VPI number of concatenation bits are verified to be in range, i.e., *Params.DeviceCommonParams.UncbM* in [0..12], *Params.DeviceCommonParams.UncbN* in [4..16], and *Params.DeviceCommonParams.UncbM* + *Params.DeviceCommonParams.UncbN* <= 16

The UTOPIA Configuration Register fields are configured as follows:

- SDT_N_UDT is set to UDT mode
- UNI_N_NNI field is set to *Params.DeviceCommonParams.UcrUniNNni*
- UTO_CLK_SEL is set to *Params.DeviceCommonParams.UcrUtoClkSel*
- GLOBAL_OAM_SEL is set to *Params.DeviceCommonParams.UcrGlobalOamSel*
- EIGHT_N_SIXTEEN is set to *Params.DeviceCommonParams.UcrEightNSixteen*
- LEVEL1_N_LEVEL2 is set to *Params.DeviceCommonParams.UcrLevel1NLevel2*
- UKSEL is set to *Params.DeviceCommonParams.UcrUkSel*
- PHY_N_ATM is set to *Params.DeviceCommonParams.DeviceCommonParams.UcrPhyNAtm*
- DEVICE_ADDRESS is set to *Params.DeviceCommonParams.UcrDeviceAddress*

The Clock Management Configuration Register fields are configured as follows:

- SLV_N_MSTR is set to *Params.DeviceSdtSpecificParams.CmcrSlvNMstr*
- F0_MODE is set to *Params.DeviceSdtSpecificParams.CmcrF0Mode*
- 8_KHZ_SEL is set to *Params.DeviceSdtSpecificParams.Cmcr8KhzSel*
- FNXI1_RATE is set to *Params.DeviceSdtSpecificParams.CmcrFnxi1Rate*
- FNXI2_RATE is set to *Params.DeviceSdtSpecificParams.CmcrFnxi2Rate*
- EXT_N_INT is set to *Params.DeviceCommonParams.CmcrExtNInt*

The External PLL Clock Source Register fields are configured as follows:

- PRI_SEL bits<4:0> are set to *Params.DeviceCommonParams.PllcsPriClkSrcPrt*
- PRI_SEL bit<5> is set to *Params.DeviceCommonParams.PllcsPriClkSrcTyp*
- SEC_SEL bits<4:0> are set to *Params.DeviceCommonParams.PllcsSecClkSrcPrt*
- SEC_SEL bit<5> is set to *Params.DeviceCommonParams.PllcsSecClkSrcTyp*

The VC Match and Match Enable and the VP Match and Match Enable Registers are set to *Params.DeviceCommonParams.VcMatch*, *Params.DeviceCommonParams.VcMatchEnable*, *Params.DeviceCommonParams.VpMatch*, and *Params.DeviceCommonParams.VpMatchEnable* respectively

The look-up table is configured as follows:

- $2^{(M + N + 1)}$ bytes are allocated for the look-up table in the chunk of external memory specified by *Params.DeviceCommonParams.ExtMemChunk* where M is *Params.DeviceCommonParams.UncbM* and N is *Params.DeviceCommonParams.UncbN*. If 128 Kbytes of memory are required for the look-up table then *ExtMemChunk* must specify the first of two consecutive empty chunks of memory.
- the look-up table is initialized so that every entry has its Destination of Received Cell field set to LUT_DS_DISCARDED
- the UTOPIA Number of Concatenated bits Register is programmed with *Params.DeviceCommonParams.UncbN* and *Params.DeviceCommonParams.UncbM*
- the 16 most significant bits of the look-up table byte address are stored in the LUT Base Address Register

The UDT Reassembly Control Register fields are configured as follows:

- UDT_DUMMY is set to *Params.DeviceUdtSpecificParams.UrcrUdtDummy*
- UDT_INSERT_LOST is set to *Params.DeviceUdtSpecificParams.UrcrUdtInsertLost*
- CHECK_LATE_ARRIVALS is set to *Params.DeviceUdtSpecificParams.UrcrCheckLateArrivals*

The MIB Timeout Configuration Register fields are configured as follows:

- CUT_VC_PERIOD is set to the user-specified value in *Params.DeviceCommonParams.MtcrCutVcPeriod*

The incoming and outgoing UTOPIA FIFOs are cleared.

The SDT Reassembly Control Register fields are configured as follows:

- SDT_DUMMY is set to *Params.DeviceSdtSpecificParams.SrcrSdtDummy*
- SDT_INSERT_LOST is set to *Params.DeviceSdtSpecificParams.SrcrSdtInsertLost*
- CB_BASE_ADD is set to 0

The Main TDM Control Register 1 fields are configured as follows:

- IDLE_DATA is set to *Params.DeviceSdtSpecificParams.Maintdm1IdleData*
- SILENCE_DATA is set to *Params.DeviceSdtSpecificParams.Maintdm1SilenceData*

The Main TDM Control Register 2 fields are configured as follows:

- IDLE_CAS is set to the *Params.DeviceSdtSpecificParams.Maintdm2IdleCas*

The UTO_CLK_CONFIGURED field of the UTOPIA Configuration Register is set at least one clock cycle after setting the UTO_CLK_SEL bit.

**Source Files**

atm90528.h, atm90528.c

### 2.2.4 Mt90528ConfigureDataRxsarModule

**Prototype**

```
USHORT Mt90528ConfigureDataRxsarModule (
      UINT                          Device,
      s_data_cell_buffer_params     * pParams,
      s_memory_chunk                * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to a structure containing user-programmable settings for configuring the Data RX_SAR Module.<br>See 4.8, “Structure s_data_cell_buffer_params“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data RX_SAR is configured |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_CHUNK_OUT_OF_RANGE* | memory chunk is out of range |
| *MT90528_ATM_DATA_RXSAR_BUFFER_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for buffer |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine allocates memory for the RX_SAR data cell buffer and initializes the registers required for RX_SAR data cell handling.

Parameters are verified as follows:

- the device id is in range
- the chunk of external memory specified for the RX_SAR data buffer is <= MAX_ADDRESS_EXT_MEMORY

The size of the buffer allocated is determined by the user specified value in *Params.BufferSize*. The buffer is allocated in the chunk of external memory specified by *Params.BufferMemoryChunk.*

The size and base address of the buffer is stored in the Data RX_SAR Configuration Register.

The Data RX_SAR Read pointer is initialized to buffer size - 1.

**Source Files**

atm90528.h, atm90528.c

### 2.2.5 Mt90528ReadDataRxsarBuffer

**Prototype**

```
USHORT Mt90528ReadDataRxsarBuffer (
      UINT              Device,
      USHORT            * pData
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pData* | location where data from the RX_SAR buffer is to be read into |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | RX_SAR data cell read successfully, more cells in the buffer |
| *MT90528_ATM_LAST_DATA_CELL* | RX_SAR data cell read successfully, no more cells in the buffer |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_DATA_BUFFER_EMPTY* | no cells to read |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine reads one data cell from the RX_SAR data cell buffer as follows:

- the read pointer is read from the Data RX_SAR Read Pointer Register indicating the last cell read
- the write pointer is read from the Data RX_SAR Write Pointer Register indicating the location where the next cell will be written
- if the read pointer is one less than the write pointer then the Data RX_SAR is about to read a cell which has not been written, the error MT90528_ATM_DATA_BUFFER_EMPTY is returned, otherwise
- the next data cell is read from the RX_SAR data cell buffer into location in CPU memory specified by *pData*
- the read pointer in the Data RX_SAR Read Pointer Register is incremented
- if the cell read is the last cell in the buffer return MT90528_ATM_LAST_DATA_CELL, otherwise return MT90528_ATM_NO_ERROR

Parameters are verified as follows:

- the device id is in range
- the buffer is not empty

If a Data RX_SAR buffer overrun interrupt occurs due to the write pointer catching up to the read pointer, the read pointer will be incremented by one cell by the interrupt handler. This will cause the data cell at the write pointer position to be discarded.

Note: The Data RX_SAR interrupts are disabled in the main interrupt enable register to prevent this routine from being called by the application and the ISR thread at the same time. Otherwise, the read pointer could become corrupted. The interrupts are re-enabled before the routine returns.

**Source Files**

atm90528.h atm90528.c

### 2.2.6 Mt90528EnableDataRxsar

#### Prototype

```
USHORT Mt90528EnableDataRxsar (
      UINT                          Device,
      s_data_rx_sar_enable_params   * pParams,
      s_mt90528_statistics          * pStats,
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to structure containing RX_SAR data interrupt enable flags. See 4.10, “Structure s_data_rx_sar_enable_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

#### Return Values

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data RX_SAR module is enabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

#### Description

This routine

- optionally enables the cell buffer overrun interrupt in the Data RX_SAR module
- optionally enables the cell arrival interrupt in the Data RX_SAR module
- optionally enables the buffer half full interrupt in the Data RX_SAR module
- enables the Data RX_SAR module for cell handling

Parameters are verified as follows:

- the device id is in range

#### Source Files

atm90528.h atm90528.c

### 2.2.7 Mt90528DisableDataRxsar

#### Prototype

```
USHORT Mt90528DisableDataRxsar (
      UINT                    Device
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data RX_SAR module is disabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine disables the Data RX_SAR module by clearing the Data RX_SAR Control Register.

Parameters are verified as follows:

- the device id is in range

**Source Files**

atm90528.h atm90528.c

### 2.2.8 Mt90528ConfigureDataTxsarModule

**Prototype**

```
USHORT Mt90528ConfigureDataTxsarModule (
      UINT                          Device,
      s_data_cell_buffer_params     * pParams,
      s_memory_chunk                * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to a structure containing user-programmable settings for configuring the Data TX_SAR Module.<br>See 4.8, “Structure s_data_cell_buffer_params“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data TX_SAR is configured |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_CHUNK_OUT_OF_RANGE* | memory chunk is out of range |
| *MT90528_ATM_DATA_TXSAR_BUFFER_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for buffer |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine allocates memory for the TX_SAR data cell buffer and initializes the registers required for TX_SAR data cell handling.

Parameters are verified as follows:

- the device id is in range
- the chunk of external memory specified for the TX_SAR data buffer is <= MAX_ADDRESS_EXT_MEMORY

The size of the buffer allocated is determined by the user-specified value in *Params.BufferSize*. The buffer is allocated in the chunk of external memory specified by *Params.BufferMemoryChunk*.

The size and base address of the buffer is stored in the Data TX_SAR Configuration Register.

The Data TX_SAR Write pointer is initialized to 0.

**Source Files**

atm90528.h atm90528.c

### 2.2.9 Mt90528InitializeDataTxsarWritePointer

**Prototype**

```
USHORT Mt90528InitializeDataTxsarWritePointer (
      UINT                    Device
      )
```

**Parameters**

*Device* device identifier, range of values: 0..MAX_MT90528_DEVICES

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | pointer is initialized |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

The Data TX_SAR Write pointer is initialized to point to the first buffer entry.

Parameters are verified as follows:

- the device id is in range

**Source Files**

atm90528.h atm90528.c

### 2.2.10 Mt90528WriteDataTxsarBuffer

**Prototype**

```
USHORT Mt90528WriteDataTxsarBuffer (
      UINT                    Device,
      USHORT                  * pDataCell
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pDataCell* | location of the data to be written to the TX_SAR buffer |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | cell written successfully, room for more cells |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_TXSAR_FULL* | cell written successfully, no room for more cells |
| *MT90528_ATM_DATA_CELL_DISCARDED* | txsar buffer is full, cell has been discarded |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine writes one data cell to the TX_SAR data cell buffer as follows:

- the read and write pointers are read from the Data TX_SAR Read and Write Pointer Registers
- if the write pointer is pointing to the cell imediately before the cell that the read pointer is pointing to, the cell is discarded and an error is returned
- otherwise, the cell is written to the buffer at the location indicated by the write pointer
- the write pointer is incremented and stored in the Write Pointer Register
- if the newly incremented write pointer is pointing to the cell imediately before the cell that the read pointer is pointing to a message is returned indicating that the cell was written successfully and the TX_SAR buffer is now full.

Parameters are verified as follows:

- the device id is in range

**Source Files**

atm90528.h atm90528.c

### 2.2.11 Mt90528EnableDataTxsar

**Prototype**

```
USHORT Mt90528EnableDataTxsar (
      UINT                          Device,
      s_data_tx_sar_enable_params   * pParams
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pParams* | pointer to a structure containing user-programmable settings for enabling the Data TX_SAR module.<br>See 4.9, “Structure s_data_tx_sar_enable_params“ |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data TX_SAR module is enabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_CELL_GENERATION_TIMEOUT_OUT_OF_RANGE* | timeout value is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine enables the Data TX_SAR.

Parameters are verified as follows:

- the device id is in range
- *Params.CellGenerationTimeout* <= 1023

In the Data Cell Generation Timeout Register configure the DCGP field with the user-specified value in *Params.CellGenerationTimeout.*

If the user specified continuous transmission set the write pointer in the Data TX_SAR Write Pointer Register to 1 past the last cell.

Program the Data TX_SAR Control Register fields as follows:

- enable TDSENB
- set AUTO to *Params.Auto*
- set TCBE_SE to *Params.BufferEmptyService*

**Source Files**

atm90528.h atm90528.c

### 2.2.12 Mt90528DisableDataTxsar

**Prototype**

```
USHORT Mt90528DisableDataTxsar (
      UINT                    Device
      )
```

**Parameters**

*Device* device identifier, range of values: 0..MAX_MT90528_DEVICES

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data TX_SAR module is disabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine disables the Data TX_SAR module by clearing the Data TX_SAR Control Register.

Parameters are verified as follows:

- the device id is in range

**Source Files**

atm90528.h atm90528.c

### 2.2.13 Mt90528ConfigureTdmPortForUdt

#### Prototype

```
USHORT Mt90528ConfigureTdmPortForUdt (
      UINT                    Device,
      USHORT                  Port,
      s_tdm_port_udt_params   * pParams,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to structure containing user-programmable settings for configuring a TDM port for UDT. See 4.14, “Structure s_tdm_port_udt_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |

#### Return Values

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | port is configured for UDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE_ERROR* | device id is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE_ERROR* | port id is out of range |
| *MT90528_ATM_PORT_ALREADY_PROGRAMMED* | port is already programmed |
| *MT90528_ATM_PORT_IN_LOOPBACK* | port is in loopback mode |
| *MT90528_ATM_TXSAR_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for the UDT TX_SAR control structure |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

#### Description

This routine configures *Port* for UDT mode.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and not already programmed or in loopback mode

Memory is allocated for *Port’s* UDT TX_SAR Control Structure in the TX_SAR Control part of internal memory.

The Timeout Configuration Register fields for *Port* are configured as follows:

- LATE_CELL_PERIOD is set to *Params.TdmPortCommonParams.TcrLateCellPeriod*

The Clocking Configuration Register fields for *Port* are configured as follows:

- CLKSEL is set to *Params.TdmPortCommonParams.CcrClkSel*
- RTSSEL is set to *Params.TdmPortCommonParams.CcrRtsSel*
- FNXISEL is set to *Params.TdmPortCommonParams.CcrFnxiSel*
- PLL_FREQ_SEL is set to *Params.TdmPortCommonParams.CcrPllFreqSel*
- PLL_MODE_SEL is set to *Params.TdmPortCommonParams.CcrPllModeSel*

- PLL_INPUT_SEL is set to *Params.TdmPortCommonParams.CcrPllInputSel*

The PLL Enable Register fields for *Port* are configured as follows:

- PLL_ENABLE is set to *Params.TdmPortCommonParams.PllenPllEnable*.

The TDM Control Register 1 fields for *Port* are configured as follows:

- TDM_CLK_POL is set to *Params.TdmPortCommonParams.Tdm1TdmClkPol*
- TDM_LINK_TYPE is set to *Params.TdmPortCommonParams.Tdm1TdmLinkType*
- TDM_CLK_RATE is hard-coded to 1.544 MHz for DS1 ports and 2.048 MHz for E1 ports
- TDM_LOS_CLK is set to *Params.TdmPortUdtSpecificParams.Tdm1TdmLosClk*
- TDM_LOS_POL is set to the board-specific constant BOARD_TDM1_TDM_LOS_POL
- TDM_CLK_MODE is set to independent mode
- TDM_DATA_FORMAT is set to UDT mode
- INT_LOS_ENABLE is set to *Params.TdmPortUdtSpecificParams.Tdm1IntLosEnable*

The TDM Control Register 4 fields for *Port* are configured as follows:

- UDT_LOS_SE is set to *Params.TdmPortUdtSpecificParams.Tdm4UdtLosSe*
- UDT_TDM_OUT_BUF_ERROR_SE is set to *Params.TdmPortUdtSpecificParams.Tdm4UdtTdmOutBufErrorSe*

The TDM Control Register 3 fields for *Port* are configured as follows:

- TDM_REASS_CLK_POL is set to *Params.TdmPortCommonParams.Tdm3TdmReassClkPol*
- TDM_REASS_INT_ENB is enabled
- TDM_REASS_PORT_CONTROL is enabled
- TDM_REASS_LOS is set to *Params.TdmPortUdtSpecificParams.Tdm3TdmReassLos*

The TX_SAR Pointer Table Base Register fields for *Port* are configured as follows:

- TXPTB is set to the word offset from the start of TX_SAR control memory of the *Port's* TX_SAR Control Structure
- TXCFG is set to UDT mode

The MIB TDM port statistics table is updated with the TDM port mode and link type.

The MIB device statistics table is updated for number of configured TDM ports.

**Source Files**

atm90528.h atm90528.c

### 2.2.14 Mt90528DeprogramTdmPortForUdt

**Prototype**

```
USHORT Mt90528DeprogramTdmPortForUdt (
      UINT                    Device,
      USHORT                  Port,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array.<br>See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | UDT port is de-programmed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | port id is out of range |
| *MT90528_ATM_PORT_NOT_UDT* | port not programmed for UDT |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine de-programs a UDT mode port.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for UDT
- no VC is opened against *Port*

The UDT TX_SAR Control Structure is cleared and the memory is de-allocated.

The following port specific registers are cleared:

- Timeout Configuration Register
- TDM Control Register 4
- TX_SAR Pointer Table Base Register
- Clocking Configuration Register
- PLL Enable Register
- TDM Control Register 1
- TDM Control Register 3

The MIB TDM port statistics table entry is removed for the TDM port.

The MIB device statistics table is updated for number of configured TDM ports.

**Source Files**

atm90528.h, atm90528.c

### 2.2.15 Mt90528ConfigureTdmPortForSdt

#### Prototype

```
USHORT Mt90528ConfigureTdmPortForSdt (
      UINT                    Device,
      USHORT                  Port,
      s_tdm_port_sdt_params   * pParams,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks,
      s_memory_chunk          * pExtChunks
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to structure containing user-programmable settings for configuring a TDM port for SDT. See 4.15, “Structure s_tdm_port_sdt_params“ |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array. See 4.24, “Structure s_memory_chunk“. |

#### Return Values

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | port is configured for SDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE_ERROR* | device id is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE_ERROR* | port id is out of range |
| *MT90528_ATM_PORT_ALREADY_PROGRAMMED* | port is already programmed |
| *MT90528_ATM_PORT_IN_LOOPBACK* | port is in loopback mode |
| *MT90528_ATM_TXSAR_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for this port's segmentation pointer table |
| *MT90528_ATM_SCB_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for this port's segmentation circular buffers |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

#### Description

This routine configures a port for SDT mode.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and not already programmed or in loopback mode
- *Params->TdmPortSdtSpecificParams.Tdm1TdmDataFormat* is either TDM1_SDT_MODE or TDM1_SDT_MODE_NGREATER46
- the chunk of external memory specified for this *Port’s* segmentation circular buffers <= MAX_ADDRESS_EXT_MEMORY
- the chunk of external memory specified for this *Port’s* reassembly circular buffers <= MAX_ADDRESS_EXT_MEMORY

Memory is allocated for the *Port’s* Segmentation Pointer Table in TX_SAR Control memory.

Memory is allocated for 24 or 32 (depending on whether *Port* is DS1 or E1) consecutive segmentation circular buffers in the chunk of external memory specified by *Params.TdmPortSdtSpecificParams.ExtMemTxCBChunk*.

The Timeout Configuration Register fields for *Port* are configured as follows:

- LATE_CELL_PERIOD is set to *Params.TdmPortCommonParams.TcrLateCellPeriod*

The Clocking Configuration Register fields for *Port* are configured as follows:

- CLKSEL is set to *Params.TdmPortCommonParams.CcrClkSel*
- RTSSEL is set to *Params.TdmPortCommonParams.CcrRtsSel*
- FNXISEL is set to *Params.TdmPortCommonParams.CcrFnxiSel*
- PLL_FREQ_SEL is set to *Params.TdmPortCommonParams.CcrPllFreqSel*
- PLL_MODE_SEL is set to *Params.TdmPortCommonParams.CcrPllModeSel*
- PLL_INPUT_SEL is set to *Params.TdmPortCommonParams.CcrPllInputSel*

The PLL Enable Register fields for *Port* are configured as follows:

- PLL_ENABLE is set to ENABLED

The TDM Control Register 1 fields for *Port* are configured as follows:

- TDM_CLK_POL is set to *Params.TdmPortCommonParams.Tdm1TdmClkPol*
- TDM_LINK_TYPE is set *Params.TdmPortCommonParams.Tdm1TdmLinkType*
- TDM_CLK_RATE is set to *Params.TdmPortCommonParams.Tdm1TdmClkRate*
- TDM_CAS_LOCATION is set *Params.TdmPortSdtSpecificParams.Tdm1CasLocation*
- TDM_MAPPING_SCH is set to *Params.TdmPortSdtSpecificParams.Tdm1MappingSch*
- TDM_PULSE_SEL is set to the *Params.TdmPortSdtSpecificParams.Tdm1TdmPulseSel*
- TDM_PULSE_POL is set to the *Params.TdmPortSdtSpecificParams.Tdm1TdmPulsePol*
- TDM_CLK_MODE is set to the *Params.TdmPortSdtSpecificParams.Tdm1TdmClkMode*
- TDM_BUS_MODE is set to *Params.TdmPortSdtSpecificParams.Tdm1TdmBusMode*
- TDM_DATA_FORMAT is set to *Params.TdmPortSdtSpecificParams.Tdm1TdmDataFormat*

The TDM Control Register 2 fields for *Port* are configured as follows:

- TDM_SEGMEN_BASE_ADD is set to bits <19:11> of the word address in external memroy of the segmentation circular buffers for *Port*
- TDM_SEGMEN_PORT_CONTROL is enabled
- TDM_SEGMEN_INT_ENB is enabled
- TDM_SEGMEN_EXT_ENB is enabled

The TDM Control Register 3 fields for *Port* are configured as follows:

- TDM_REASS_CLK_POL is set to *Params.TdmPortCommonParams.Tdm3TdmReassClkPol*
- TDM_REASS_INT_ENB is enabled
- TDM_REASS_EXT_ENB is enabled
- TDM_REASS_PORT_CONTROL is enabled
- TDM_REASS_BASE_ADD is set to bits <19:15> of the 32 word chunk of external memory where *Port's* reassembly circular buffers will be allocated

The TDM Control Register 4 fields for Port are configured as follows:

- SDT_TDM_OUT_BUF_ERROR_SE is set to *Params.TdmPortSdtSpecificParams.Tdm4SdtTdmOutBufErrorSe*
- SDT_PERM_UNDER_SE is set to *Params.TdmPortSdtSpecificParams.Tdm4SdtPermUnderSe*
- SDT_SIMPLE_UNDER_SE is set to *Params.TdmPortSdtSpecificParams.Tdm4SdtSimpleUnderSe*

If *port* is in backplane mode and the TX_SAR is already enabled then wait for 512ms before enabling the port for SDT.

The TX_SAR Pointer Table Base Register fields for *Port* are configured as follows:

- TXPTB is set to the word offset from the start of the *Port's* segmentation pointer table in TX_SAR control memory
- TXCFG is set to *Params.TdmPortSdtSpecificParams.TxptbTxcfg*

The MIB TDM port statistics table is updated with the TDM port mode and link type.

The MIB device statistics table is updated for number of configured TDM ports.

**Source Files**

atm90528.h atm90528.c

### 2.2.16 Mt90528DeprogramTdmPortForSdt

**Prototype**

```
USHORT Mt90528DeprogramTdmPortForSdt (
      UINT                    Device,
      USHORT                  Port,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks
      s_memory_chunk          * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, "Structure s_mt90528_statistics". |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array.<br>See 4.24, "Structure s_memory_chunk". |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, "Structure s_memory_chunk". |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | SDT port is de-programmed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out or range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | port id is out of range |
| *MT90528_ATM_PORT_NOT_SDT* | port not programmed for SDT |
| *MT90528_PORT_IS_ASSIGNED_A_VC* | port has a vc assigned to it |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine de-programs an SDT mode port.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for SDT mode
- no VCs are opened against *Port*

The segmentation pointer table for *Port* is cleared and the memory is de-allocated.

The following port specific registers are cleared:

- TDM Control Register 2
- Timeout Configuration Register
- TDM Control Register 4
- TX_SAR Pointer Table Base Register
- Clocking Configuration Register
- PLL Enable Register
- TDM Control Register 1
- TDM Control Register 3

The segmentation circular buffers for *Port* are cleared and the memory is de-allocated.

The MIB TDM port statistics table entry is removed for the TDM port.

The MIB device statistics table is updated for number of configured TDM ports.

**Source Files**

atm90528.h, atm90528.c

### 2.2.17 Mt90528SetTdmPortLinkType

**Prototype**

```
USHORT Mt90528SetTdmPortLinkType(
      UINT                    Device,
      USHORT                  Port,
      UCHAR                   LinkType,
      UCHAR                   ClockRate,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *LinkType* | link type to change the port to, i.e., TDM1_DS1_LINK or TDM1_E1_LINK |
| *ClockRate* | clock rate to change the port to, i.e., TDM1_2048_KHZ or TDM1_1544_KHZ |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | port is configured with new link type |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | port id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This adjusts the link type and clock rate for *Port*.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range

The TDM Control Register 1 fields for *Port* are configured as follows:

- *TDM_LINK_TYPE* is set to *LinkType*
- *TDM_CLK_RATE* is set to *ClockRate*

**Source Files**

atm90528.h atm90528.c

### 2.2.18 Mt90528TurnOnUtopiaInterfaceLoopback

**Prototype**

```
USHORT Mt90528TurnOnUtopiaInterfaceLoopback (
      UINT                    Device,
      s_mt90528_statistics    * pStats,
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | utopia interface is in loopback mode |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine clears the UTOPIA FIFOs and puts the UTOPIA interface into loopback mode.

**UTOPIA interface loopback is a debugging mode and should not be used with the normal transmit and receive operations of the chip**.

Parameters are verified as follows:

- *Device* is in range

The MIB device statistics table is updated to indicate UTOPIA is in loopback mode.

**Source Files**

atm90528.h, atm90528.c

### 2.2.19 Mt90528TurnOffUtopiaInterfaceLoopback

**Prototype**

```
USHORT Mt90528TurnOffUtopiaInterfaceLoopback (
      UINT                    Device,
      s_mt90528_statistics    * pStats,
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | utopia interface is not in loopback mode |
| *MT90528_ATM_ DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine clears the UTOPIA FIFOs and takes the UTOPIA interface out of loopback mode.

Parameters are verified as follows:

- *Device* is in range

The MIB device statistics table is updated to indicate UTOPIA is not in loopback mode.

**Source Files**

atm90528.h, atm90528.c

### 2.2.20 Mt90528ConfigureTdmPortForLowLatencyLoopback

**Prototype**

```
USHORT Mt90528ConfigureTdmPortForLowLatencyLoopback (
      UINT                                Device,
      USHORT                              Port,
      s_tdm_loopback_params               * pParams
      s_mt90528_statistics                * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to structure containing user-programmable settings for TDM low-latency loopback. See 4.16, “Structure s_tdm_loopback_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | TDM port is in low-latency loopback |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE_ERROR* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE_ERROR* | Port is out of range |
| *MT90528_ATM_PORT_ALREADY_PROGRAMMED* | Port is already programmed |
| *MT90528_ATM_PORT_IN_LOOPBACK* | Port is already in loopback mode |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine puts the specified port into low-latency loopback.

**TDM low-latency loopback is a debugging mode and should not be used with the normal transmit and receive operations of the chip.**

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and not already programmed or in loopback

The Clocking Configuration Register for *Port* is configured as follows:

- CLKSEL is set to CCR_CLKSEL_STICLK

The TDM Control Register 1 for *Port* is configured as follows:

- TDM_CLK_POL is set to *Params.Tdm1TdmClkPol*
- TDM_LOS_POL is set to the board-specific constant BOARD_TDM1_TDM_LOS_POL
- TDM_LOW_LATENCY_LPBK is enabled

The TDM Control Register 3 *Port* is configured as follows:

- TDM_REASS_CLK_POL is set opposite to the value specified in *Params.Tdm1TdmClkPol*, i.e., if *Params.Tdm1TdmClkPol* is specified as TDM1_RISING_EDGE then TDM_REASS_CLK_POL is set to TDM3_FALLING_EDGE, otherwise, TDM_REASS_CLK_POL is set to TDM3_RISING_EDGE

**Source Files**

atm90528.h, atm90528.c

### 2.2.21 Mt90528DeprogramTdmPortForLowLatencyLoopback

#### Prototype

```
USHORT Mt90528DeprogramTdmPortForLowLatencyLoopback (
      UINT                                Device,
      USHORT                              Port,
      s_mt90528_statistics                * pStats
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

#### Return Values

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Port is not in loopback mode |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range |
| *MT90528_ATM_PORT_NOT_IN_LOOPBACK* | Port is taken out of low-latency loopback mode |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

#### Description

This routine takes the specified port out of low-latency loopback.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and in low-latency loopback mode

The following registers are cleared:

- Clocking Configuration Register for *Port*
- TDM Control Register 1 for *Port*
- TDM Control Register 3 for *Port*

#### Source Files

atm90528.h, atm90528.c

### 2.2.22 Mt90528ConfigureTdmPortForCircularBufferLoopback

#### Prototype

```
USHORT Mt90528ConfigureTdmPortForCircularBufferLoopback (
      UINT                                Device,
      USHORT                              Port,
      s_tdm_loopback_params               * pParams
      s_mt90528_statistics                * pStats,
      s_memory_chunk                      * pIntChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to structure containing user-programmable settings for TDM low-latency loopback. See 4.16, “Structure s_tdm_loopback_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Port is in circular-buffer loopback |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE_ERROR* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE_ERROR* | Port is out of range |
| *MT90528_ATM_PORT_ALREADY_PROGRAMMED* | Port is already programmed |
| *MT90528_ATM_PORT_IN_LOOPBACK* | Port is already in loopback mode |
| *MT90528_ATM_CHUNK_OUT_OF_RANGE* | memory chunk is out of range |
| *MT90528_ATM_SCB_ALLOCATE_MEMORY_ERROR* | failed allocating memory for circular buffers |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine puts the specified port into circular-buffer loopback.

**TDM circular-buffer loopback is a debugging mode and should not be used with the normal transmit and receive operations of the chip.**

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and not programmed and not in loopback mode
- *Params.ExtMemTxCBChunk* is <= MAX_ADDRESS_EXT_MEMORY

Memory is allocated for segmentation circular buffers. In this mode, these buffers are used by the TDM SDT reassembly process as the SDT Reassembly Circular Buffers.

The TDM Control Register 1 for *Port* is configured as follows:

- TDM_MAPPING_SCH is set to *Params.Tdm1MappingSch*
- TDM_PULSE_POL is set to *Params. Tdm1TdmPulsePol*
- TDM_CLK_POL is set to *Params.Tdm1TdmClkPol*
- TDM_LINK_TYPE is set to *Params.Tdm1TdmLinkType*
- TDM_CLK_RATE is set to *Param.Tdm1TdmClkRate*
- TDM_DATA_FORMAT is to SDT mode
- TDM_CIR_BUF_LPBK is enabled

The Clocking Configuration Register for *Port* is configured as follows:

- CLKSEL is set to CCR_CLKSEL_STICLK

The TDM Control Register 3 *Port* is configured as follows:

- TDM_REASS_CLK_POL is set to *Params.Tdm3TdmReassClkPol*
- TDM_REASS_INT_ENB is ENABLED

- TDM_REASS_EXT_ENB is ENABLED
- TDM_REASS_PORT_CONTROL is ENABLED

The Reassembly Circular Buffer Address and Size field for each entry in the TDM SDT Reassembly Control Structure for *Port* is filled in with the corresponding timeslot id in bits<5:1> and a 1 representing buffer size of 64 word entries in bit<0>. The other fields and bits are set to 0.

The TDM Control Register 2 *Port* is configured as follows:

- TDM_SEGMEN_BASE_ADD is set to bits <19:11> of the word address in external memroy of the segmentation circular buffers
- TDM_SEGMEN_PORT_CONTROL is ENABLED
- TDM_SEGMEN_INT_ENB is ENABLED
- TDM_SEGMEN_EXT_ENB is ENABLED

**Source Files**

atm90528.h, atm90528.c

### 2.2.23 Mt90528DeprogramTdmPortForCircularBufferLoopback

**Prototype**

```
USHORT Mt90528DeprogramTdmPortForCircularBufferLoopback (
      UINT                                Device,
      USHORT                              Port,
      s_mt90528_statistics                * pStats,
      s_memory_chunk                      * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Port is not in circular-buffer loopback mode |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range |
| *MT90528_ATM_PORT_NOT_IN_LOOPBACK* | Port is taken out of circular-buffer loopback mode |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine takes the specified port out of circular-buffer loopback.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and in circular-buffer loopback mode

The following registers are cleared:

- Clocking Configuration Register for *Port*
- TDM Control Register 1 for *Port*
- TDM Control Register 2 for *Port*
- TDM Control Register 3 for *Port*

The segmentation circular buffer memory is cleared and deallocated.

The TDM SDT Reassembly Control Structure is cleared for *Port*.

**Source Files**

atm90528.h, atm90528.c

### 2.2.24 Mt90528EnableTxTdmForUdt

**Prototype**

```
USHORT Mt90528EnableTxTdmForUdt (
      UINT                    Device,
      USHORT                  Port,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | process that writes data to the TDM input buffer for Port is enabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range |
| *MT90528_ATM_PORT_NOT_UDT* | Port is not in UDT mode |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine enables transmission of the TDM input stream on a UDT mode TDM port to the TXSAR if previously disabled with Mt90528DisableTxTdmForUdt and returns an error otherwise.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for UDT mode

The TDM Control Register 2 for *Port* is updated as follows:

- TDM_SEGMEN_INT_ENB is enabled

**Source Files**

atm90528.h, atm90528.c

### 2.2.25 Mt90528DisableTxTdmForUdt

**Prototype**

```
USHORT Mt90528DisableTxTdmForUdt (
      UINT                    Device,
      USHORT                  Port,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | process that writes data to the TDM input buffer for Port is disabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range *MT90528_ATM_FATAL_ERROR*<br>API cannot recover |

**Description**

This routine disables transmission of the TDM input stream on a UDT mode TDM port to the TXSAR.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for UDT mode

The TDM Control Register 2 for *Port* is updated as follows:

- TDM_SEGMEN_INT_ENB is cleared

**Source Files**

atm90528.h, atm90528.c

### 2.2.26 Mt90528EnableTxVcForSdt

**Prototype**

```
USHORT Mt90528EnableTxVcForSdt (
      UINT                    Device,
      s_vc                    * pVc,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See Structure s_vc Structure s_vc. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | transmission of cells for *Vc* is enabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_VC_NOT_OPEN* | VC is not open |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine enables transmission of cells for a specific VC.

Parameters are verified as follows:

- *Device* is in range
- *Vc* is opened for SDT in the TX direction

The E bit of the entry in the SDT Segmentation Pointer Table for *Vc* is enabled.

**Source Files**

atm90528.h, atm90528.c

### 2.2.27 Mt90528DisableTxVcForSdt

**Prototype**

```
USHORT Mt90528DisableTxVcForSdt (
      UINT                    Device,
      s_vc                    * pVc,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“ |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | transmission of cells for *Vc* is disabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_VC_NOT_OPEN* | VC not open |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine disables transmission of cells for a specific VC.

Parameters are verified as follows:

- *Device* is in range
- *Vc* is opened for SDT in the TX direction

The E bit of the entry in the SDT Segmentation Pointer Table for *Vc* is disabled.

**Source Files**

atm90528.h, atm90528.c

### 2.2.28 Mt90528OpenTxVcForUdt

**Prototype**

```
USHORT Mt90528OpenTxVcForUdt(
      UINT                    Device,
      UCHAR                   Port,
      s_tx_vc_udt_params      * pParams,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to a structure containing user-programmable settings for opening a TX VC for UDT. See 4.1, “Structure s_vc“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | TX VC is open for UDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range |
| *MT90528_ATM_PORT_NOT_UDT* | Port is not programmed for UDT mode |
| *MT90528_ATM_PORT_IS_ASSIGNED_A_VC* | a TX VC is already assigned to Port |
| *MT90528_ATM_INVALID_VC* | either both VPI and VCI are 0 or the device is in UNI mode and the VPI is greater than 8 bits |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine opens a TX VC for UDT.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for UDT mode
- no VC is opened against *Port*

- *Params.CellHeader.Vc* is valid and not already opened

The TX_SAR Control Structure for the VC is initialized as follows:

- SRTS is set to 0
- SE is set to *Params.Srts*
- SEQ is set to 0
- GFC is set to *Params.CellHeader.Gfc* if the device is programmed for UNI mode and to the 4 MSBs of the 12 bit *Params.CellHeader.Vc.Vpi* if the device is programmed for NNI mode
- VPI is set to the 8 LSBs of *Params.CellHeader.Vc.Vpi*
- VCI is set to *Params.CellHeader.Vc.Vci*
- PTI is set to *Params.CellHeader.Pti*
- C(CLP) is set to *Params.CellHeader.Clp*
- HEC is set to *Params.CellHeader.Hec*
- UDF2 is set *Params.CellHeader.Udf2*
- cell count statistics is set to 0

The TDM Control Register 2 for *Port* is configured as follows:

- TDM_SEGMEN_PORT_CONTROL is enabled
- TDM_SEGMEN_INT_ENB is enabled

Create a TX entry in the CBR VC statistics table and set the number of opened TX CBR VC channels field to 1.

Add 1 to device statistics number of opened TX CBR VC channels.

Add 1 to device statistics number of opened UDT TX CBR VCs.

Add 1 to device statistics number of opened TX CBR VCs.

#### Source Files

atm90528.h, atm90528.c

### 2.2.29 Mt90528CloseTxVcForUdt

#### Prototype

```
USHORT Mt90528CloseTxVcForUdt(
      UINT                    Device,
      UCHAR                   Port,
      s_mt90528_statistics    * pStats
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 4.20, “Structure s_tx_vc_udt_params“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | TX VC is closed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE_ERROR* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE_ERROR* | Port is out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine clears the UDT TX_SAR Control Structure in internal memory and clears TDM Control Register 2 for *Port*.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range
- a UDT TX VC is opened against *Port*

Remove the TX VC entry from the CBR VC statistics table.

Subtract 1 from device statistics for number of opened TX CBR VC channels.

Subtract 1 from device statistics for number of opened UDT TX CBR VCs.

Subtract 1 from device statistics for number of opened TX CBR VCs.

**Source Files**

atm90528.h, atm90528.c

### 2.2.30 Mt90528OpenRxVcForUdt

**Prototype**

```
USHORT Mt90528OpenRxVcForUdt(
      UINT                    Device,
      UCHAR                   Port,
      s_rx_vc_udt_params      * pParams,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to a structure containing user-programmable settings for opening an RX VC for UDT.<br>See 4.22, “Structure s_rx_vc_udt_params“ |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | RX VC is open for UDT |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |

| | |
|---|---|
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range |
| *MT90528_ATM_PORT_NOT_UDT* | Port is not programmed for UDT mode |
| *MT90528_PORT_IS_ASSIGNED_A_VC* | An RX VC is already assigned to Port |
| *MT90528_ATM_INVALID_VC* | either both VPI and VCI are 0 or the device is in UNI mode and the VPI is greater than 8 bits |
| *MT90528_ATM_VC_ALREADY_OPEN* | RX VC is already open |
| *MT90528_ATM_CDV_VALUE_OUT_OF_RANGE* | cell delay variation value is out of range |
| *MT90528_ATM_CLOCK_RECOVERY_ERROR* | more than one clock recovery method specified |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine opens an RX VC for UDT.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for UDT mode
- no VC is opened against *Port*
- *Params.Vc* is valid and not already opened
- SRTS and Adaptive clock recovery methods are not both selected
- the user-specified value for cell delay variation is <= 49 if the *Port* is DS1 and <= 37 if *Port* is E1. The units are in increments of 0.1 milliseconds (100μs)

The UDT RX_SAR Control Structure fields for the VC are initialized as follows:

- A (Adaptive Enabled) is set to *Params.Adaptive*
- S (SRTS Enable) is set to *Params.Srts*
- Maximum Lead is set to (2 * n * *Params.CellDelayVariation/10*) + 94 where n is 256 for E1 ports and 193 for DS1 ports
- VC TDM Port is set to *Port*
- if first cell handling is specified reassembled cells is initialized to 0xffff so that the first cell received will trigger a reassembled cell rollover interrupt, otherwise it is set to 0
- all other fields are set to 0

The Control Structure information is stored at a hardware fixed location in internal memory based on *Port*.

The UDT VCI Register for *Port* is programmed with *Params.Vc.Vci* .

The UDT VPI Register for *Port* is programmed as follows:

- UDT_VPI is set to *Params.Vc.Vpi*
- OAM_SEL is set to *Params.UvpOamSel*

If first cell handling is specified then late cell and/or cut vc interrupts will be enabled after the first cell is received according to the user-specified values in *Params.TcrLateCellSe* and *Params.TcrCutVcSe* respectively, otherwise the interrupts are processed immediately.

Create an RX entry in the CBR VC statistics table and set the number of opened RX CBR VC channels field to 1.

Add 1 to device statistics number of opened RX CBR VC channels.

Add 1 to device statistics number of opened UDT RX CBR VCs.

Add 1 to device statistics number of opened RX CBR VCs.

**Source Files**

atm90528.h, atm90528.c

### 2.2.31 Mt90528CloseRxVcForUdt

**Prototype**

```
USHORT Mt90528CloseRxVcForUdt(
      UINT                    Device,
      UCHAR                   Port,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | RX VC is closed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | Port is out of range |
| *MT90528_ATM_VC_NOT_OPEN* | VC is not open |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine clears the UDT RX_SAR Control Structure in internal memory and the UVC and UVP registers.

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range
- a UDT RX VC is opened against *Port*

Remove the RX VC entry from the CBR VC statistics table.

Subtract 1 from device statistics for number of opened RX CBR VC channels.

Subtract 1 from device statistics for number of opened UDT RX CBR VCs.

Subtract 1 from device statistics for number of opened RX CBR VCs.

**Source Files**

atm90528.h, atm90528.c

### 2.2.32 Mt90528OpenTxVcForSdt

**Prototype**

```
USHORT Mt90528OpenTxVcForSdt(
      UINT                    Device,
      UCHAR                   AssociatedPort,
      s_tx_vc_sdt_params      * pParams,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *AssociatedPort* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to a structure containing user-programmable settings for opening a TX VC for SDT. See 4.21, “Structure s_tx_vc_sdt_params“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | the specified SDT VC is open in the TX direction |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | AssociatedPort is out of range |
| *MT90528_ATM_PORT_NOT_SDT* | *AssociatedPort is* not programmed for SDT mode |
| *MT90628_ATM_TDM_DATA_FORMAT* | tdm data format of *AssociatedPort* is incorrect for greater than 46 channels |
| *MT90528_ATM_INVALID_VC* | either both VPI and VCI are 0 or the device is in UNI mode and the VPI is greater than 8 bits |
| *MT90528_ATM_SRTS_NOT_ALLOWED_WITH_CAS* | SRTS and CAS cannot both be enabled |
| *MT90528_ATM_MAX_VCS_ALREADY_OPENED* | max number of vcs already opened |
| *MT90528_ATM_128_MAX_CHANNELS_FOR_SDT_VC* | maximum 128 channels per SDT VC |
| *MT90528_ATM_32_MAX_CHANNELS_FOR_SDT_VC_WITH_SRTS* | maximum 32 channels allowed when SRTS is enabled |
| *MT90528_ATM_VC_ALREADY_OPEN* | *VC* is already open in the TX direction |
| *MT90528_ATM_TIMESLOT_OUT_OF_RANGE* | a timeslot id is out of range |
| *MT90528_ATM_TIMESLOT_ALREADY_SPECIFIED_FOR_THIS_VC* | a timeslot is specified more than once for this VC |
| *MT90528_ATM_TIMESLOT_ALREADY_ASSIGNED* | a timeslot has already been assigned to some other TX VC |
| *MT90528_ATM_SDT_LINK_TYPE_ERROR* | a timeslot does not match the link type of the Associated-Port |
| *MT90528_ATM_READ_POINTER_OUT_OF_RANGE* | read pointer is out range |
| *MT90528_ATM_TXSAR_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for the SDT segmentation control structure |

*MT90528_ATM_FATAL_ERROR* API cannot recover

**Description**

This routine opens a TX VC for SDT.

Parameters are verified as follows:

- *Device* is in range
- *AssociatedPort* is in range and programmed for SDT mode or SDT n>46 mode if number of channels specified is > 46
- the VC is valid and not already opened in the TX direction
- CAS and SRTS are not both specified
- the number of channels specified is in range
- the timeslots of the channels specified are in range and on ports programmed for SDT with link types matching that of *AssociatedPort*
- none of the timeslots of the channels specified is assigned to another TX VC
- the timeslots of the channels specified are unique
- the read pointer specified is <= 3Fh

Memory is allocated for the SDT Segmentation Control Structure in the TX_SAR Control portion of internal memory.

The SDT Segmentation Control Structure is initialized as follows:

- Number of Channels is set to *Params.Channels.NumberOfChannels*
- Read Pointer is set to *Params.ReadPointer*
- W (wait for multiframe) is to *Params.WaitForMultiframe*
- F (First Cell) is initialized to 1 if read pointer is 0 and cleared if read pointer is non-zero
- Structure Length and M (Mode) are set based on the number of channels specified, whether CAS is enabled or not, and the link type of *AssociatedPort*
- SE (SRTS Enable) is set to *Params.Srts*
- PE (Pointer Enable) and P (Pointer Cell) are disabled for single channel VCs not transporting CAS, otherwise they are enabled
- GFC is set to *Params.CellHeader.Gfc* if the device is programmed for UNI mode and to the 4 MSBs of the 12 bit *Params.CellHeader.Vc.Vpi* if the device is programmed for NNI mode
- VP is set to the 8 LSBs of *Params.CellHeader.Vc.Vpi*
- VCI is set to *Params.CellHeader.Vc.Vci*
- PTI is set to *Params.CellHeader.Pti*
- C(CLP) is set to *Params.CellHeader.Clp*
- HEC is set to *Params.CellHeader.Hec*
- UDF2 is set to *Params.CellHeader.Udf2*
- for every timeslot t specified in *Params.Channels.AtmChannel* the Segmentation Circular Buffer Base Address field is set to the start address of this timeslot's segmentation circular buffer
- all other fields are set to 0 or hard-coded as indicated in Figure 22 of the MT90528 28-Port Primary Rate Circuit Emulation AAL1 SAR Data Sheet

If SRTS is specified the VC_CHANNELS field of the Clocking Configuration Register is set to *Params.Channels.NumberOfChannels* - 1.

Create a TX entry in the CBR VC statistics table and set the number of opened TX CBR VC channels field to to the number of channels specified for this VC.

Update device statistics for number of opened TX CBR VC channels.

Update device statistics for number of opened SDT TX CBR VCs.

Update device statistics for number of opened TX CBR VCs.

If CAS is specified, update device statistics for number of opened E1 or T1 CAS TX CBR VCs.

**Source Files**

atm90528.h, atm90528.c

### 2.2.33 Mt90528CloseTxVcForSdt

**Prototype**

```
USHORT Mt90528CloseTxVcForSdt(
      UINT                    Device,
      s_vc                    * pVc,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | TX VC is closed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE_ERROR* | device id is out of range |
| *MT90528_ATM_INVALID_VC* | either both VPI and VCI are 0 or the device is in UNI mode and the VPI is greater than 8 bits |
| *MT90528_ATM_VC_NOT_OPEN* | VC is not open |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine removes the entry from the Segmentation Pointer Table and clears and deallocates the memory for the SDT Segmentation Control Structure in internal memory for *Vc*. If the VC was receiving RTS nibbles the VC_CHANNELS field of the associated port’s Clocking Configuration Register is cleared.

Parameters are verified as follows:

- *Device* is in range

- *Vc* is valid and open for SDT in the TX direction

Remove the TX VC entry from the CBR VC statistics table.

Update device statistics for number of opened SDT TX CBR VCs.

Update device statistics for number of opened TX CBR VCs.

If this is a CAS VC, update device statistics for number of opened E1 or T1 CAS TX CBR VCs.

**Source Files**

atm90528.h, atm90528.c

### 2.2.34 Mt90528OpenRxVcForSdt

**Prototype**

```
USHORT Mt90528OpenRxVcForSdt(
      UINT                    Device,
      UCHAR                   AssociatedPort,
      s_rx_vc_sdt_params      * pParams,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks,
      s_memory_chunk          * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *AssociatedPort* | TDM port id, range of values: 0..MAX_PORTS |
| *pParams* | pointer to a structure containing user-programmable settings for opening an RX VC for SDT.<br>See 4.23, “Structure s_rx_vc_sdt_params“. |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array.<br>See 4.24, “Structure s_memory_chunk“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array.<br>See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | the specified SDT VC is open in the RX direction |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | Device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | AssociatedPort is out of range |
| *MT90528_ATM_PORT_NOT_SDT* | *AssociatedPort is* not programmed for SDT mode |
| *MT90528_ATM_INVALID_VC* | either both VPI and VCI are 0 or the device is in UNI mode and the VPI is greater than 8 bits |
| *MT90528_ATM_MAX_VCS_ALREADY_OPENED* | max number of vcs already opened |
| *MT90528_ATM_VC_ALREADY_OPEN* | the specified VC is already open in the RX direction |
| *MT90528_ATM_128_MAX_CHANNELS_FOR_SDT_VC* | maximum 128 channels per SDT VC |

| | |
|---|---|
| *MT90528_ATM_32_MAX_CHANNELS_FOR_SDT_VC_WITH_SRTS* | maximum 32 channels allowed when SRTS is enabled |
| *MT90528_ATM_TIMESLOT_OUT_OF_RANGE* | a timeslot id is out of range |
| *MT90528_ATM_TIMESLOT_ALREADY_SPECIFIED_FOR_THIS_VC* | a timeslot is specified more than once for this VC |
| *MT90528_ATM_TIMESLOT_ALREADY_ASSIGNED* | a timeslot has already been assigned to some other RX VC |
| *MT90528_ATM_SDT_LINK_TYPE_ERROR* | a timeslot does not match the link type of the Associated-Port |
| *MT90528_ATM_SRTS_NOT_ALLOWED_WITH_CAS* | SRTS and CAS cannot both be enabled |
| *MT90528_ATM_CLOCK_RECOVERY_ERROR* | more than one clock recovery method specified |
| *MT90528_ATM_INVALID_BUFFER_SIZE* | reassembly circular buffer size is too small for the amount of external memory present |
| *MT90528_ATM_CDV_VALUE_OUT_OF_RANGE* | cell delay variation is out of range |
| *MT90528_ATM_CDV_VALUE_OUT_OF_RANGE_FOR_RCB_SIZE* | cell delay variation is out of range for reassembly circular buffer size |
| *MT90528_ATM_CUT_VC_ERROR* | cut vc interrupt can not be enabled for SDT |
| *MT90528_ATM_LUT_CONFLICT* | VC conflicts with an existing LUT entry |
| *MT90528_ATM_RCB_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for reassembly circular buffers |
| *MT90528_ATM_RXSAR_ALLOCATE_MEMORY_ERROR* | failed to allocate memory for reassembly control structure |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine opens an RX VC for SDT.

Parameters are verified as follows:

- *Device* is in range
- *AssociatedPort* is in range and programmed for SDT mode
- the VC is valid and not already opened in the RX direction
- CAS and SRTS are not both specified
- SRTS and Adaptive clock recovery methods are not both specified
- the user-specified value for cell delay variation is in range. See the table in section Mt90528UpdateCellDelayVariation Mt90528UpdateCellDelayVariation for cell delay variation maximums
- the reassembly circular buffer size > 128 entries if the amount of external memory present > 512Kbytes
- the CUT VC interrupt for the associated port is disabled
- the number of channels specified is in range
- the timeslots of the channels specified are in range and on ports programmed for SDT with link types matching that of *AssociatedPort*
- none of the timeslots of the channels specified is assigned to another RX VC
- the timeslots of the channels specified are unique

Memory is allocated for the SDT reassembly control structure in the SDT RX_SAR Control portion of internal memory.

Memory is allocated for reassembly circular buffers, one for each of the timeslots specified.

The SDT Reassembly Control Structure is initialized as follows:

- Maximum Lead is set to ( 2 * *Params.CellDelayVariation* ) + 2 bytes where *Params.CellDelayVariation* is in increments of 125 microseconds.
- VC TDM Port is set to *AssociatedPort*
- A (Adaptive Enable) is set to *Params.Adaptive*
- S (SRTS Enabled) is set to *Params.Srts*
- bit<2> of the CAS field isset to *Params.CasChangeServiceRequest*
- bit<1> of the CAS field is set to *Params.CasSwControlled*
- bit<0> of the CAS field is set to *Params.Cas*
- BS is *Params.CircularBufferSize*
- Number of Channels is set to *Params.Channels.NumberOfChannels – 1*
- for every timeslot t specified in *Params.Channels.AtmChannel* the CAS[t] field is set to *Params.CasInitialValue[t]*
- for every timeslot t specified in *Params.Channels.AtmChannel* the Reassembly Circular Buffer Base Address field is set to some part of the start address of this timeslot's reassembly circular buffer as outlined in Table 19 of the MT90528 28-Port Primary Rate Circuit Emulation AAL1 SAR Data Sheet
- all other fields are set to 0 or hard-coded as indicated in Figure 27 of the MT90528 28-Port Primary Rate Circuit Emulation AAL1 SAR Data Sheet

For every timeslot specified in *Params.Channels.AtmChannel* the corresponding TDM SDT Reassembly Control Structure entry is filled in as follows:

- the V bit is enabled
- the I bit is set to *Params.Idle*
- the Reassembly Circular Buffer and Size field is filled in based on both the location of the timeslot's reassembly circular buffer and *Params.CircularBufferSize* according to Table 11 in the MT90528 28-Port Primary Rate Circuit Emulation AAL1 SAR Data Sheet

The look-up table is updated for the specified VC as follows:

- the Reassembly Control Structure Address field is set to some portion of the start address of the VC's reassembly control structure
- O is set to *Params.LutOamSel*
- DS is set to TDM bus

If SRTS is specified the VC_CHANNELS field of the Clocking Configuration Register is set to *Params.Channels.NumberOfChannels* - 1.

Create an RX entry in the CBR VC statistics table and set the number of opened RX CBR VC channels field to the number of channels specified for this VC.

Update device statistics for number of opened RX CBR VC channels.

Update device statistics for number of opened SDT RX CBR VCs.

Update device statistics for number of opened RX CBR VCs.

If CAS is specified, update device statistics for number of opened E1 or T1 CAS RX CBR VCs.

**Source Files**

atm90528.h, atm90528.c

### 2.2.35 Mt90528CloseRxVcForSdt

**Prototype**

```
USHORT Mt90528CloseRxVcForSdt(
      UINT                    Device,
      s_vc                    * pVc,
      s_mt90528_statistics    * pStats,
      s_memory_chunk          * pIntChunks,
      s_memory_chunk          * pExtChunks
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See Structure s_vc Structure s_vc. |
| *pStats* | pointer to MIB statistics structure. See 4.1, “Structure s_vc“. |
| *pIntChunks* | pointer passed by caller of memory manager internal chunk array. See 4.24, “Structure s_memory_chunk“. |
| *pExtChunks* | pointer passed by caller of memory manager external chunk array. See 4.24, “Structure s_memory_chunk“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | RX VC is closed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device is out of range |
| *MT90528_ATM_VC_NOT_OPEN* | VC is not open |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine clears the SDT RX_SAR Control Structure in internal memory, clears the VC’s entry in the look-up table, and for each channel clears and deallocates the reassembly circular buffer and clears the TDM SDT Reassembly Control Structure entry. If the VC was receiving RTS nibbles the VC_CHANNELS field of the associated port’s Clocking Configuration Register is cleared.

Parameters are verified as follows:

- *Device* is in range
- the VC is valid and open for SDT in the RX direction

Remove the RX VC entry from the CBR VC statistics table.

Update device statistics for number of opened SDT RX CBR VCs.

Update device statistics for number of opened RX CBR VCs.

If this is a CAS VC, update device statistics for number of opened E1 or T1 CAS RX CBR VCs.

#### Source Files

atm90528.h, atm90528.c

### 2.2.36 Mt90528EnableTxsar

#### Prototype

```
USHORT Mt90528EnableTxsar (
      UINT                    Device,
      s_mt90528_statistics    * pStats,
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

#### Return Values

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | TXENB is enabled |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

#### Description

This routine toggles the BACKPLANE_CLK_CONFIGURED bit of the Clock Management Configuration Register, waits for the frame pulse period of 125μs, and enables the TXENB bit of the TX_SAR Master Enable Register

Parameters are verified as follows:

- *Device* is in range

#### Source Files

atm90528.h, atm90528.c

### 2.2.37 Mt90528OpenRxDataVc

#### Prototype

```
USHORT Mt90528OpenRxDataVc (
      UINT                    Device,
      s_vc                    * pVc,
      s_mt90528_statistics    * pStats
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

#### Return Values

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data VC is open |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id out of range |
| *MT90528_ATM_MAX_DATA_VCS_ALREADY_OPENED* | max number of VCs already opened |
| *MT90528_ATM_LUT_CONFLICT* | VC conflicts with an existing LUT entry |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

#### Description

This routine opens an RX Data VC for cell reception by adding the specified VC to the look-up table and setting the Destination of Received Cell field to Receive Data Cell buffer.

Parameters are verified as follows:

- *Device* is in range
- The maximum number of Data VCs are not already open
- *Vc* does not conflict with an existing LUT entry

Update device statistics for number of opened RX Data VCs.

#### Source Files

atm90528.h, atm90528.c

### 2.2.38 Mt90528CloseRxDataVc

#### Prototype

```
USHORT Mt90528CloseRxDataVc (
      UINT                    Device,
      s_vc                    * pVc,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | Data VC is closed |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_VC_NOT_OPEN* | *Vc* is not open |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

This routine closes an RX Data VC for cell reception by removing the specified VC from the look-up table.

Update device statistics for number of opened RX Data VCs.

Parameters are verified as follows:

- *Device* is in range
- *Vc* is open for data in the RX direction

**Source Files**

atm90528.h, atm90528.c

### 2.2.39 Mt90528UpdateCutVcPeriod

**Prototype**

```
USHORT Mt90528UpdateCutVcPeriod (
      UINT        Device,
      USHORT      CutVcPeriod
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *CutVcPeriod* | new value for the cut vc period in milliseconds |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | cut vc period updated successfully |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id out of range |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

Change the value of the cut vc period for the device.

Parameters are verified as follows:

- *Device* is in range

**Source Files**

atm90528.h, atm90528.c

### 2.2.40 Mt90528UpdateCellDelayVariation

**Prototype**

```
USHORT Mt90528UpdateCellDelayVariation (
      UINT                    Device,
      s_vc                    * pVc,
      USHORT                  CellDelayVariation,
      s_mt90528_statistics    * pStats
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES. |
| *pVc* | pointer to a structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *CellDelayVariation* | new cell delay variation - for UDT mode VCs this value represents units of 100 microseconds, otherwise, the units are in increments of 125 microseconds. |
| *pStats* | pointer to MIB statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | cell delay variation updated successfully |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device id is out of range |
| *MT90528_ATM_CDV_VALUE_OUT_OF_RANGE* | cell delay variation is out of range |
| *MT90528_ATM_CDV_VALUE_OUT_OF_RANGE_FOR_RCB_SIZE* | cell delay variation is out of range for reassembly circular buffer size |
| *MT90528_ATM_VC_NOT_OPEN* | *Vc* not open in the RX direction |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

Change the value of the cell delay variation for an RX VC.

Parameters are verified as follows:

- *Device* is in range
- *Vc* is open in the RX direction
- the user-specified value for the new cell delay variation is <= 37 increments of 0.1 milliseconds (100μs) for UDT VC open against an E1 port

- the user-specified value for the new cell delay variation is <= 49 increments of 0.1 milliseconds (100μs) for UDT VC open against a DS1 port
- the user-specified value for the new cell delay variation is based on the reassembly circular buffer size and the number of channels for the SDT VC. Cell delay variation is in increments of 125 microseconds. The following table identifies the maximum cell delay variation for the different reassembly circular buffer sizes and number of open channels

| Reassembly Circuit Buffer Size | Number of Open Channels | SDT Maximum Cell Delay Variation (125μs increments) |
|---|---|---|
| 64 | 1 | 7 |
| | 2 | 19 |
| | 3 | 23 |
| | 4 | 25 |
| | 5 | 26 |
| | 6-7 | 27 |
| | 8-11 | 28 |
| | 12-23 | 29 |
| | >23 | 30 |
| 128 | 1 | 39 |
| | 2 | 51 |
| | 3 | 55 |
| | 4 | 57 |
| | 5 | 58 |
| | 607 | 59 |
| | 8-11 | 60 |
| | 12-23 | 61 |
| | >23 | 62 |
| 256 | 1 | 103 |
| | 2 | 115 |
| | 3 | 119 |
| | 4 | 121 |
| | 5 | 122 |
| | 6-7 | 123 |
| | 8-11 | 124 |
| | 12-23 | 125 |
| | >23 | 126 |
| 512 | 1 | 231 |
| | 2 | 243 |
| | 3 | 247 |
| | 4 | 249 |
| | 5 | 250 |
| | 6-7 | 251 |
| | 8-11 | 252 |
| | 12-23 | 253 |
| | >23 | 254 |

| Reassembly Circuit Buffer Size | Number of Open Channels | SDT Maximum Cell Delay Variation (125μs increments) |
|---|---|---|
| 1024 | 1 | 487 |
| | 2 | 499 |
| | 3 | 503 |
| | 4 | 505 |
| | 5 | 506 |
| | 6-7 | 507 |
| | 8-11 | 508 |
| | 12-23 | 509 |
| | >23 | 510 |

**Source File**

atm90528.h, atm90528.c

### 2.2.41 Mt90528UpdateReplayNSilence

**Prototype**

```
USHORT Mt90528UpdateReplayNSilence (
      UINT                    Device,
      UCHAR                   Port,
      UCHAR                   ReplayNSilenceValue,
      s_mt90528_statistics    * pStat
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Port* | TDM port id, range of values: 0..MAX_PORTS<br>See 4.1, “Structure s_vc“. |
| *ReplayNSilenceValue* | TDM3_SILENCE or TDM3_REPLAY |
| *pStats* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return Values**

| | |
|---|---|
| *MT90528_ATM_NO_ERROR* | replay and silence field is updated successfully |
| *MT90528_ATM_DEVICE_OUT_OF_RANGE* | device is out of range |
| *MT90528_ATM_PORT_OUT_OF_RANGE* | port is out of range |
| *MT90528_ATM_PORT_NOT_SDT* | port must be programmed for SDT |
| *MT90528_ATM_FATAL_ERROR* | API cannot recover |

**Description**

Change the value of the REPLAY_N_SILENCE field in TDM Control Register 3 for *Port* .

Parameters are verified as follows:

- *Device* is in range
- *Port* is in range and programmed for SDT

**Source Files**

atm90528.h, atm90528.c

## 2.3 I/O Functions

**Description**

Access to the device from the user application is handled by the I/O functions. Separate functions are provided for reading from and writing to registers and internal and external memory.

### 2.3.1 Mt90528WriteExternalMemory

**Prototype**

```
USHORT Mt90528WriteExternalMemory(
      UINT        Device,
      ULONG       Address,
      USHORT      * pData,
      USHORT      Length
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Address* | chip-based external memory address, range of values: 0.. MAX_ADDRESS_EXT_MEMORY |
| *pData* | pointer to block of data to be written |
| *Length* | number of words of data to be written |

**Return Values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_EXTERNAL_ADDRESS* | address specified is out of range |
| *MT90528_IO_WRITE_EXTERNAL_MEMORY_ERROR* | an error was returned from the board layer |

**Description**

This routine writes a block of data to the chip's external memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the amount of external memory configured for the device. The data written is verified.

**WARNING**: writing directly to device registers and memory could detrimentally affect the operation of the API.

**Source Files**

io90528.h, io90528.c

### 2.3.2 Mt90528WriteNoVerifyExternalMemory

**Prototype**

```
USHORT Mt90528WriteNoVerifyExternalMemory(
      UINT        Device,
      ULONG       Address,
      USHORT      * pData,
      USHORT      Length
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Address* | chip-based external memory address, range of values: 0.. MAX_ADDRESS_EXT_MEMORY |
| *pData* | pointer to block of data to be written |
| *Length* | number of words of data to be written |

**Return Values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_EXTERNAL_ADDRESS* | address specified is out of range |
| *MT90528_IO_WRITE_EXTERNAL_MEMORY_ERROR* | an error was returned from the board layer |

**Description**

This routine writes a block of data to the chip's external memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the amount of external memory configured for the device. The data written is not verified.

**WARNING**: writing directly to device registers and memory could detrimentally affect the operation of the API.

**Source Files**

io90528.h, io90528.c

### 2.3.3 Mt90528ReadExternalMemory

**Prototype**

```
USHORT Mt90528ReadExternalMemory(
      UINT        Device,
      ULONG       Address,
      USHORT      * pData,
      USHORT      Length
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Address* | chip based external memory address, range of values: 0 .. MAX_ADDRESS_EXT_MEMORY |
| *pData* | pointer to location where data is to be read into |
| *Length* | number of words of data to be read |

**Return Values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_EXTERNAL_ADDRESS* | address specified is out of range |
| *MT90528_IO_READ_EXTERNAL_MEMORY_ERROR* | an error was returned from the board layer |

**Description**

This routine reads a block of data from the chip's external memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the amount of external memory configured for the device.

**Source Files**

io90528.h, io90528.c

### 2.3.4 Mt90528WriteInternalMemory

**Prototype**

```
USHORT Mt90528WriteInternalMemory(
      UINT        Device,
      ULONG       Address,
      USHORT      * pData,
      USHORT      Length
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Address* | chip-based internal memory address, range of values: MIN_ADDRESS_INT_MEMORY .. MAX_ADDRESS_INT_MEMORY |
| *pData* | pointer to block of data to be written |
| *Length* | number of words of data to be written |

**Return Values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_INTERNAL_ADDRESS* | address specified is out of range |
| *MT90528_IO_WRITE_INTERNAL_MEMORY_ERROR* | an error was returned from the board layer |

**Description**

This routine writes a block of data to the chip's internal memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the amount of internal memory within the device. The data written is verified.

**WARNING**: writing directly to device registers and memory could detrimentally affect the operation of the API.

**Source Files**

io90528.h, io90528.c

### 2.3.5 Mt90528WriteNoVerifyInternalMemory

**Prototype**

```
USHORT Mt90528WriteNoVerifyInternalMemory(
      UINT        Device,
      ULONG       Address,
      USHORT      * pData,
      USHORT      Length
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Address* | chip-based internal memory address, range of values: MIN_ADDRESS_INT_MEMORY .. MAX_ADDRESS_INT_MEMORY |
| *pData* | pointer to block of data to be written |
| *Length* | number of words of data to be written |

**Return Values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_INTERNAL_ADDRESS* | address specified is out of range |
| *MT90528_IO_WRITE_INTERNAL_MEMORY_ERROR* | an error was returned from the board layer |

**Description**

This routine writes a block of data to the chip's internal memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the amount of internal memory within the device. The data written is not verified.

**WARNING**: writing directly to device registers and memory could detrimentally affect the operation of the API.

**Source Files**

io90528.h, io90528.c

### 2.3.6 Mt90528ReadInternalMemory

#### Prototype

```
USHORT Mt90528ReadInternalMemory(
      UINT        Device,
      ULONG       Address,
      USHORT      * pData,
      USHORT      Length
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *Address* | chip-based internal memory address, range of values: MIN_ADDRESS_INT_MEMORY .. MAX_ADDRESS_INT_MEMORY |
| *pData* | pointer to location where data is to be read into |
| *Length* | number of words of data to be read |

#### Return Values

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_INTERNAL_ADDRESS* | address specified is out of range |
| *MT90528_IO_READ_INTERNAL_MEMORY_ERROR* | an error was returned from the board layer |

#### Description

This routine reads a block of data from the chip's internal memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the amount of internal memory within the device.

#### Source Files

io90528.h, io90528.c

### 2.3.7 Mt90528WriteRegister

#### Prototype

```
USHORT Mt90528WriteRegister(
      UINT        Device,
      ULONG       RegisterAddress,
      USHORT      Data
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *RegisterAddress* | chip-based register address, range of values: 0 .. LAST_REGISTER_ON_CHIP |
| *Data* | data word to be written |

**Return Values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_REGISTER_ADDRESS* | register address specified is out of range |
| *MT90528_IO_WRITE_REGISTER_ERROR* | an error was returned from the board layer |

**Description**

This routine writes a word of data to the chip's register memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the register map in the device.

**WARNING**: writing directly to device registers and memory could detrimentally affect the operation of the API.

**Source Files**

io90528.h, io90528.c

### 2.3.8 Mt90528ReadRegister

**Prototype**

```
USHORT Mt90528ReadRegister(
      UINT        Device,
      ULONG       RegisterAddress,
      USHORT      *pData
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *RegisterAddress* | chip-based register address, range of values: 0 .. LAST_REGISTER_ON_CHIP |
| *Data* | pointer to the word address to read the data into |

**Return values**

| | |
|---|---|
| *MT90528_IO_NO_ERROR* | no error |
| *MT90528_IO_INVALID_DEVICE_ID* | device specified is out of range |
| *MT90528_IO_INVALID_REGISTER_ADDRESS* | register address specified is out of range |
| *MT90528_IO_READ_REGISTER_ERROR* | an error was returned from the board layer |

**Description**

This routine reads a word of data from the chip's register memory. Range checking is performed on the device id based on the number of devices appearing on the board. Address range checking is performed based on the register map in the device.

**Source Files**

io90528.h, io90528.c

## 2.4 Interrupt Service Handler

Interrupts coming from the device are handled by a master interrupt handler routine, which services the nine device interrupts. The handler function gathers all the information related to the interrupts received and after clearing the interrupts, passes back the information to the calling routine via a structure s_mt90528_status. MIB and performance management statistic information is updated in the structure s_mt90528_statistics.

**NOTE**: the application has to define variables using the structures s_mt90528_statistics, s_mt90528_status and the optional s_mt90528_threshold in order to use the following functions. These common variables are also used by the API, ATM and MIB functions.

### 2.4.1 Mt90528IsrHandler

#### Prototype

```
USHORT Mt90528IsrHandler(
      UINT                    Device,
      s_mt90528_statistics    * pStatistics,
      s_mt90528_status        * pStatus
      )
```

#### Parameters

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pStatus* | pointer to user supplied interrupt, threshold and wrap (rollover) status structure.<br>See 5.15, “Structure s_mt90528_status“. |

#### Return values

| | |
|---|---|
| *MT90528_ISR_NO_ERROR* | interrupts handled successfully |
| *MT90528_ISR_READ_ERROR* | failure reading an interrupt register |
| *MT90528_ISR_TXSAR_READ_ERROR* | failure reading during TXSAR handling |
| *MT90528_ISR_TXSAR_WRITE_ERROR* | failure writing during TXSAR handling |
| *MT90528_ISR_SDT_RXSAR_READ_ERROR* | failure reading during SDT RXSAR handling |
| *MT90528_ISR_SDT_RXSAR_WRITE_ERROR* | failure writing during SDT RXSAR handling |
| *MT90528_ISR_UDT_RXSAR_READ_ERROR* | failure reading during UDT RXSAR handling |
| *MT90528_ISR_UDT_RXSAR_WRITE_ERROR* | failure writing during UDT RXSAR handling |
| *MT90528_ISR_TDM_READ_ERROR* | failure reading during TDM handling |
| *MT90528_ISR_TDM_WRITE_ERROR* | failure writing during TDM handling |
| *MT90528_ISR_REASS_SIDE_TIMEOUT_READ_ERROR* | failure reading during Reassembly Side Timeout |

| | |
|---|---|
| | handling |
| *MT90528_ISR_REASS_SIDE_TIMEOUT_WRITE_ERROR* | failure writing during Reassembly Side Timeout handling |
| *MT90528_ISR_UTOPIA_READ_ERROR* | failure reading during Utopia handling |
| *MT90528_ISR_UTOPIA_WRITE_ERROR* | failure writing during Utopia Handling |
| *MT90528_ISR_XMEMA_READ_ERROR* | failure reading during External Memory Arbiter handling |
| *MT90528_ISR_XMEMA_WRITE_ERROR* | failure writing during External Memory Arbiter handling |
| *MT90528_ISR_DATA_RXSAR_READ_ERROR* | failure reading during Data RXSAR handling |
| *MT90528_ISR_DATA_RXSAR_WRITE_ERROR* | failure writing during Data RXSAR handling |
| *MT90528_ISR_CPU_READ_ERROR* | failure reading during CPU service handling |
| *MT90528_ISR_CPU_ACC_ERROR* | HAIC access cycle complete bit not cleared. |

**Description**

This device servicing routine should be called when an interrupt has occurred. A separate application defined task or thread should deal with this servicing requirement by calling this routine when an interrupt occurs on an MT90528 device.

All interrupts from an MT90528 device will be serviced by this routine. MIB and performance management (PM) statistic information will be updated in the statistics structure. Status information will be passed back via the status structure about what interrupts occurred, if there are MIB / PM software counter threshold alerts and if there are 32 bit MIB / PM software counter wraps (rollovers) about to occur (within the next 2 interrupts for the same CBR VC).

**NOTE:** when a Reassembly Side Timeout cut VC interrupt occurs, the "V" bit (VC arrival) is cleared in the UDT Reassembly Control Structure for the affected TDM port. The “V” bit (VC arrival) is not cleared in the SDT Reassembly Control Structure.

Late Cell and Cut VC interrupts for UDT, if enabled, are software configured to start timing on arrival of the first cell for a CBR VC. The reassembled cell rollover interrupt is used as an indicator of first cell arrival and therefore is enabled temporarily for this purpose (even if the interrupt was disabled).

**Source files**

isr90528.h, isr90528.c

### 2.4.2 Mt90528IsrHandlerWith2ndThresholds

**Prototype**

```
USHORT Mt90528IsrHandlerWith2ndThresholds(
      UINT                    Device,
      s_mt90528_threshold     * pThreshold,
      s_mt90528_statistics    * pStatistics,
      s_mt90528_status        * pStatus
      )
```

**Parameters**

| | |
|---|---|
| *Device* | device identifier, range of values: 0..MAX_MT90528_DEVICES |
| *pThreshold* | pointer to user supplied threshold structure containing a second set of threshold values and MIB / PM software counters.<br>See 5.19, “Structure s_mt90528_threshold“. |
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pStatus* | pointer to user supplied interrupt, threshold and wrap (rollover) status structure.<br>See 5.15, “Structure s_mt90528_status“. |

**Return values**

| | |
|---|---|
| *MT90528_ISR_NO_ERROR* | interrupts handled successfully |
| *MT90528_ISR_READ_ERROR* | failure reading an interrupt register |
| *MT90528_ISR_TXSAR_READ_ERROR* | failure reading during TXSAR handling |
| *MT90528_ISR_TXSAR_WRITE_ERROR* | failure writing during TXSAR handling |
| *MT90528_ISR_SDT_RXSAR_READ_ERROR* | failure reading during SDT RXSAR handling |
| *MT90528_ISR_SDT_RXSAR_WRITE_ERROR* | failure writing during SDT RXSAR handling |
| *MT90528_ISR_UDT_RXSAR_READ_ERROR* | failure reading during UDT RXSAR handling |
| *MT90528_ISR_UDT_RXSAR_WRITE_ERROR* | failure writing during UDT RXSAR handling |
| *MT90528_ISR_TDM_READ_ERROR* | failure reading during TDM handling |
| *MT90528_ISR_TDM_WRITE_ERROR* | failure writing during TDM handling |
| *MT90528_ISR_REASS_SIDE_TIMEOUT_READ_ERROR* | failure reading during Reassembly Side Timeout handling |
| *MT90528_ISR_REASS_SIDE_TIMEOUT_WRITE_ERROR* | failure writing during Reassembly Side Timeout handling |
| *MT90528_ISR_UTOPIA_READ_ERROR* | failure reading during Utopia handling |

| | |
|---|---|
| *MT90528_ISR_UTOPIA_WRITE_ERROR* | failure writing during Utopia Handling |
| *MT90528_ISR_XMEMA_READ_ERROR* | failure reading during External Memory Arbiter handling |
| *MT90528_ISR_XMEMA_WRITE_ERROR* | failure writing during External Memory Arbiter handling |
| *MT90528_ISR_DATA_RXSAR_READ_ERROR* | failure reading during Data RXSAR handling |
| *MT90528_ISR_DATA_RXSAR_WRITE_ERROR* | failure writing during Data RXSAR handling |
| *MT90528_ISR_CPU_READ_ERROR* | failure reading during CPU service handling |
| *MT90528_ISR_CPU_ACC_ERROR* | HAIC access cycle complete bit not cleared. |

**Description**

This device servicing routine should be called when an interrupt has occurred. A separate application defined task or thread should deal with this servicing requirement by calling this routine when an interrupt occurs on an MT90528 device.

The difference between this function and Mt90528IsrHandler is that this function handles a second set of thresholds for a second set of 32 bit MIB / PM software counters. In some performance management scenarios, thresholds for both 15-minute and 24-hour periods are required. The standard thresholds in the s_mt90528_statistics structure would handle the 15-minute period and the thresholds in the s_mt90528_threshold structure would handle the 24-hour period. See the MIB / PM section in this document on how to manage the s_mt90528_threshold structure.

**NOTE:** if 24-hour thresholds are not required, use the Mt90528IsrHandler function for handling interrupts which don’t require the s_mt90528_threshold structure.

All interrupts from an MT90528 device will be serviced by this routine. MIB and performance management statistic information will be updated in the statistics structure. Status information will be passed back via the status structure about what interrupts occurred, if there are MIB / PM software counter threshold alerts for both sets of thresholds and if there are 32 bit MIB / PM software counter wraps (rollovers) about to occur (within the next 2 interrupts for the same CBR VC).

**NOTE:** when a Reassembly Side Timeout cut VC interrupt occurs, the "V" bit (VC arrival) is cleared in the UDT Reassembly Control Structure for the affected TDM port. The “V” bit (VC arrival) is not cleared in the SDT Reassembly Control Structure.

Late Cell and Cut VC interrupts for UDT, if enabled, are software configured to start timing on arrival of the first cell for a CBR VC. The reassembled cell rollover interrupt is used as an indicator of first cell arrival and therefore is enabled temporarily for this purpose (even if the interrupt was disabled).

**Source files**

isr90528.h, isr90528.c

### 2.4.3 Mt90528EnableDisableInterrupt

#### Prototype

```
USHORT Mt90528EnableDisableInterrupt(
      UINT                    Device,
      USHORT                  Port,
      USHORT                  IrqIndex,
      USHORT                  ADValue,
      s_mt90528_statistics    * pStatistics
      )
```

#### Parameters

*Device* device identifier, range of values: 0..MAX_MT90528_DEVICES
*Port* TDM port id, range of values: 0..MAX_PORTS for port registers. Assign “0” for non-port registers
*IrqIndex* interrupt table index as defined in isr90528.h – see table in this functions description
*ADValue* assertion or de-assertion value (1 or 0)
*pStatistics* pointer to user supplied statistics structure. See 5.4, “Structure s_mt90528_statistics“.

#### Return Values

*MT90528_ATM_NO_ERROR* interrupt or service was enabled or disabled
*MT90528_ISR_MODULE_INT_READ_ERROR* failure reading from the chip
*MT90528_ISR_MODULE_INT_WRITE_ERROR* failure writing to the chip

#### Description

This routine enables or disables interrupt or service enable bits in specified registers on an MT90528 device; returns an error otherwise.

The following table lists the *IrqIndex* constants that can used with this function:

| IrqIndex Constant | Description |
|---|---|
| TX_SAR_IE | main TX_SAR interrupt |
| SDT_RXSAR_IE | main SDT RX_SAR interrupt |
| UDT_RXSAR_IE | main UDT RX_SAR interrupt |
| TDM_IE | main TDM interrupt |
| UTOPIA_IE | main UTOPIA interrupt |
| XMEMA_IE | main XMEMA interrupt |
| DATA_RXSAR_IE | main DATA RX_SAR interrupt |
| RX_TIMEOUT_IE | main RX Timeout interrupt |
| CPU_IE | main CPU service interrupt |
| MIER_ALL | all main interrupt bits * |

| IrqIndex Constant | Description |
|---|---|
| ACC_DONE_SE | HAIC access complete |
| TCBE_SE | Data TX_SAR buffer empty |
| UDT_REASS_ROLL_SE | UDT RX_SAR reassembled cells rollover |
| UDT_HDR_ROLL_SE | UDT RX_SAR header errors rollover |
| UDT_SEQ_ROLL_SE | UDT RX_SAR sequence errors rollover |
| UDT_LOST_ROLL_SE | UDT RX_SAR lost cells rollover |
| UDT_MIS_ROLL_SE | UDT RX_SAR misinserted cells rollover |
| UDT_UNDER_ROLL_SE | UDT RX_SAR underrun rollover |
| UDT_OVER_ROLL_SE | UDT RX_SAR overrun rollover |
| UDT_LATE_ROLL_SE | UDT RX_SAR late cells rollover |
| UDT_CELL_COUNTER_RO_SE | UDT RX_SAR cell counter rollover |
| UDT_RXSAR_SE | UDT RX_SAR service enable |
| DROR_SE | Data RX_SAR cell buffer overrun |
| DRCA_SE | Data RX_SAR cell arrival |
| DRBHF_SE | Data RX_SAR buffer half full |
| DRCCR_SE | Data RX_SAR cell counter rollover |
| SDT_REASS_ROLL_SE | SDT RX_SAR reassembled cells rollover |
| SDT_HDR_ROLL_SE | SDT RX_SAR header errors rollover |
| SDT_SEQ_ROLL_SE | SDT RX_SAR sequence errors rollover |
| SDT_LOST_ROLL_SE | SDT RX_SAR lost cells rollover |
| SDT_MIS_ROLL_SE | SDT RX_SAR misinserted cells rollover |
| SDT_REFRAME_ROLL_SE | SDT RX_SAR reframes rollover |
| SDT_PARITY_ROLL_SE | SDT RX_SAR parity rollover |
| SDT_UNDER_ROLL_SE | SDT RX_SAR underrun rollover |
| SDT_OVER_ROLL_SE | SDT RX_SAR overrun rollover |
| SDT_PTR_OUT_OF_RANGE_SE | SDT RX_SAR pointer out of range |
| SDT_CAS_CHANGE_SE | SDT RX_SAR CAS changed |
| SDT_RXSAR_SE | SDT RX_SAR service enable |
| SDT_CELL_COUNTER_RO_SE | SDT RX_SAR reassembled cells counter rollover |
| LATE_CELL_SE | MIB timeout late cell (per TDM port) |
| CUT_VC_SE | MIB timeout cut vc (per TDM port) |

| IrqIndex Constant | Description |
|---|---|
| CELL_CNT_ROLL_SE | Utopia cell count rollover |
| PARITY_ROLL_SE | Utopia parity rollover |
| UDT_LOS_SE | TDM port loss of signal (per TDM port) |
| UDT_TDM_OUT_BUF_ERROR_SE | TDM port UDT output buffer error (per TDM port) |
| SDT_TDM_OUT_BUF_ERROR_SE | TDM port SDT output buffer error (per TDM port) |
| SDT_PERM_UNDER_SE | TDM port SDT permanent underrun (per TDM port) |
| SDT_SIMPLE_UNDER_SE | TDM port SDT simple underrun (per TDM port) |
| PESE | Memory Arbiter parity error |

***** Special note for MIER_ALL:
- to enable all main interrupts use MIER_ALL_IE for the ADValue
- to disable all main interrupts use MIER_NONE_IE for the ADValue

**Source Files**

isr90528.h, isr90528.c

## 2.5 MIB and Performance Management Statistics and Status

Requirements for statistics come from the ATM forum, Telcordia and the ITU. Statistics provide entities on networks with performance management data. The following MT90528 functions provide statistical data to the application in the s_mt90528_statistics, s_mt90528_status and the optional s_mt90528_threshold data structures.

**NOTE**: The API keeps track of the data in the statistics structures whereas the application has to keep track of the data in the application statistic structure (s_mt90528_threshold). The application statistic structure is an optional statistic for tracking a second set of thresholds and counters.

**NOTE**: the application has to define variables using the structures s_mt90528_statistics, s_mt90528_status, the optional s_mt90528_threshold and the optional s_mt90528_cbr_vc_cell_loss_status in order to use the following functions. These common variables are also used by the API, ATM and ISR functions.

### 2.5.1 Statistics Functions

#### 2.5.1.1 Mt90528GetDeviceStatistics

**Prototype**

```
void Mt90528GetDeviceStatistics(
      s_mt90528_statistics          * pStatistics,
      s_mt90528_device_stats_entry  * pDevStats
      )
```

**Parameters**

*pStatistics* pointer to user supplied statistics structure.
See 5.4, “Structure s_mt90528_statistics“.

*pDevStats* pointer to user supplied device statistics structure.
See 5.1, “Structure s_mt90528_device_stats_entry“.

**Return values**

None.

**Description**

This routine collects the device statistics from the *pStatistics* structure of the MT90528 device. The structure is copied to the *pDevStats* structure. There is an option field in the *pDevStats* structure named *ResetStatistics* that if set for TRUE (1) will reset any interrupt counters that are device specific in the *pStatistics* structure after collecting the statistics.

**Source files**

mib90528.h, mib90528.c

### 2.5.1.2 Mt90528GetTdmPortStatistics

**Prototype**

```
void Mt90528GetTdmPortStatistics(
      USHORT                              TdmPort,
      s_mt90528_statistics                * pStatistics,
      s_mt90528_tdm_port_stats_entry      * pTdmStats
      )
```

**Parameters**

| | |
|---|---|
| *TdmPort* | TDM port id, range of values: 0..MAX_PORTS |
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pTdmStats* | pointer to user supplied TDM port statistics structure entry.<br>See 5.2, “Structure s_mt90528_tdm_port_stats_entry“. |

**Return values**

None.

**Description**

This routine collects the TDM port statistics from the *pStatistics* structure of the MT90528 device. The TDM port structure is copied to the *pTdmStats* structure. There is an option field in the *pTdmStats* structure named *ResetStatistics* that if set for TRUE (1) will reset any interrupt counters that are TDM port specific for the specified *TdmPort* in the *pStatistics* structure after collecting the statistics.

**Source files**

mib90528.h, mib90528.c

### 2.5.1.3 Mt90528GetCbrVcStatistics

#### Prototype

```
USHORT Mt90528GetCbrVcStatistics(
      USHORT                        VcIndex,
      s_mt90528_statistics          * pStatistics,
      s_mt90528_cbr_vc_stats_entry  * pCbrVcStats
      )
```

#### Parameters

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pCbrVcStats* | pointer to user supplied CBR VC statistics structure entry.<br>See 5.3, “Structure s_mt90528_cbr_vc_stats_entry“. |

#### Return values

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the statistics were successfully copied |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the MT90528 control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the MT90528 control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

#### Description

This routine collects the CBR VC statistics from the *pStatistics* structure of the MT90528 device. Use this function if a second set of thresholds and counters (i.e. 24-hour period) are not being used. A CBR VC structure entry is copied to the *pCbrVcStats* structure. There is an option field in the *pCbrVcStats* structure named *ResetStatistics* that if set for TRUE (1) will reset any interrupt counters that are CBR VC specific for the specified *VcIndex* in the *pStatistics* structure after collecting the statistics.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex

#### Source files

mib90528.h, mib90528.c

### 2.5.1.4 Mt90528GetCbrVcStatisticsWithAppThresholds

**Prototype**

```
USHORT Mt90528GetCbrVcStatisticsWithAppThresholds (
      USHORT                        VcIndex,
      USHORT                        VcIndex2,
      s_mt90528_statistics          * pStatistics,
      s_mt90528_threshold           * pThreshold,
      s_mt90528_cbr_vc_stats_entry  * pCbrVcStats
      )
```

**Parameters**

*VcIndex*CBR VC index pointing to the CBR VC entry in the pStatistics structure

| | |
|---|---|
| *VcIndex2* | CBR VC index pointing to the CBR VC entry in the pThreshold structure |
| *pStatistics* | pointer to user supplied Mib statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pThreshold* | pointer to user supplied App threshold structure. See 5.19, “Structure s_mt90528_threshold“. |
| *pCbrVcStats* | pointer to user supplied CBR VC statistics structure entry. See 5.3, “Structure s_mt90528_cbr_vc_stats_entry“ |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the statistics were successfully copied |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the MT90528 control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the MT90528 control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

**Description**

This routine collects the CBR VC statistics from the *pStatistics* structure of the MT90528 device. Use this function instead of Mt90528GetCbrVcStatistics if a second set of thresholds and counters (i.e. 24-hour period) are being used. A CBR VC structure entry is copied to the *pCbrVcStats* structure. There is an option field in the *pCbrVcStats* structure named *ResetStatistics* that if set for TRUE (1) will reset any interrupt counters that are CBR VC specific for the specified *VcIndex* in the *pStatistics* structure after collecting the statistics.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex and *VcIndex2* can be acquired using the function Mt90528GetCbrVcThresholdIndex.

**Source files**

mib90528.h, mib90528.c

### 2.5.1.5 Mt90528EnableDisableCbrVcStatisticsCollection

**Prototype**

```
USHORT Mt90528EnableDisableCbrVcStatisticsCollection(
      USHORT                  VcIndex,
      Flag                    Status,
      s_mt90528_statistics    * pStatistics
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *Status* | ENABLED (1) or DISABLED (0) flag |
| *pStatistics* | pointer to user supplied statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the statistics were successfully enabled or disabled |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the MT90528 control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the MT90528 control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

**Description**

This routine enables or disables collection of statistics for the specified CBR VC index in the *pStatistics* structure of the MT90528 device using the *Status* flag. If collection is disabled, the 32 bit MIB / PM software counters will not be updated and no threshold alerts will occur during interrupts which are in the *pStatistics* structure for the CBR VC. **The system default for collection of CBR VC statistics is DISABLED**. If collection is enabled after previously being disabled, using this function, the 32 bit MIB / PM software counters are cleared. Enabling collection after already being enabled will not cause the 32 bit MIB / PM software counters to be cleared. If collection is disabled, the device MIB hardware counters are cleared.

If the statistics collection is disabled for the CBR VC by this function and the application statistics collection is enabled for the same CBR VC by Mt90528EnableDisableCbrVcAppStatisticsCollection, the application statistics will not be collected.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.1.6 Mt90528GetCbrVcStatIndex

#### Prototype

```
USHORT Mt90528GetCbrVcStatIndex(
      s_vc                    Vc,
      s_mt90528_statistics    * pStatistics,
      USHORT                  * pVcIndex
      )
```

#### Parameters

| | |
|---|---|
| *Vc* | user supplied structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *pStatistics* | pointer to user supplied statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pVcIndex* | pointer to CBR VC index pointing to the CBR VC entry in the pStatistics structure |

#### Return values

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | VC index found in statistics table |
| *MT90528_MIB_INVALID_VC* | specified VC not found in the statistics table |

#### Description

This routine finds the VC index into the *pStatistics* structure of the MT90528 device using the user supplied VC structure and passes back the index.

#### Source files

mib90528.h, mib90528.c

### 2.5.1.7 Mt90528SetMibThresholds

#### Prototype

```
USHORT Mt90528SetMibThresholds(
      USHORT                  VcIndex,
      s_stats_params          * pParam,
      s_mt90528_statistics    * pStatistics
      )
```

#### Parameters

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pParam* | pointer to user supplied threshold parameter structure. See 5.11, “Structure s_stats_params“. |
| *pStatistics* | pointer to user supplied statistics structure. See 5.4, “Structure s_mt90528_statistics“ . |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the thresholds were successfully set |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the MT90528 control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the MT90528 control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

**Description**

Thresholds determine if a MIB counter has exceeded a threshold value within a given period of time.

This routine sets the threshold values in the *pStatistics* structure for the first set of thresholds (i.e. 15-minute period). Use the function Mt90528SetMibThresholdsWithAppThresholds instead of this function when setting a first set of thresholds (i.e. 15-minute period) if a second set of thresholds (i.e. 24-hour period) is required.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.1.8 Mt90528SetMibThresholdsWithAppThresholds

**Prototype**

```
USHORT Mt90528SetMibThresholdsWithAppThresholds (
      USHORT                  VcIndex,
      USHORT                  VcIndex2,
      s_stats_params          * pParam,
      s_mt90528_statistics    * pStatistics,
      s_mt90528_threshold     * pThreshold
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *VcIndex2* | CBR VC index pointing to the CBR VC entry in the pThreshold structure |
| *pParam* | pointer to user supplied threshold parameter structure. See 5.11, “Structure s_stats_params“. |
| *pStatistics* | pointer to user supplied Mib statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pThreshold* | pointer to user supplied App threshold structure. See 5.19, “Structure s_mt90528_threshold“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the thresholds were successfully set |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the MT90528 control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the MT90528 control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

**Description**

Thresholds determine if a MIB counter has exceeded a threshold value within a given period of time.

This routine sets the threshold values in the *pStatistics* structure for the first set of thresholds (i.e. 15-minute period) when a second set of thresholds is required. Use the function Mt90528SetMibThresholds instead of this function if a second set of thresholds (i.e. 24-hour period) is not required.

See the function Mt90528SetAppThresholds to set a second set of thresholds (i.e. 24-hour period).

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex and *VcIndex2* can be acquired using the function Mt90528GetCbrVcThresholdIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.1.9 Mt90528SetMibCounters32

**Prototype**

```
USHORT Mt90528SetMibCounters32(
      USHORT                  VcIndex,
      s_stats_params          * pParam,
      s_mt90528_statistics    * pStatistics
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pParam* | pointer to user supplied counter parameter structure. See 5.11, “Structure s_stats_params“. |
| *pStatistics* | pointer to user supplied statistics structure. See 5.4, “Structure s_mt90528_statistics“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the counter values were successfully set |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the statistics table |

**Description**

The 32 bit MIB software counters can be preset to specified values.

This routine sets the counter values for a given VC in the *pStatistics* structure for the first set of counters (i.e. 15-minute period). This routine will affect the MIB / PM statistics, so caution is advised. Using Mt90528GetCbrVcStatistics with reset counters enabled is the normal method of setting the MIB software counters. This routine is intended for debug purposes and not intended for normal operation.

See the function Mt90528SetAppCounters32 to set a second set of counters (i.e. 24-hour period).

**WARNING**: If thresholds are being used, do not use this routine to set the MIB software counters or else threshold crossings will not occur when expected.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.1.10 Mt90528ReadDeviceMibCounters

**Prototype**

```
USHORT Mt90528ReadDeviceMibCounters(
      USHORT                  VcIndex,
      s_mt90528_statistics    * pStatistics,
      s_hw_load               * pMibLoad
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pMibLoad* | pointer to user supplied parameter structure for the MIB hardware counters.<br>See 5.9, “Structure s_hw_load“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the device MIB values were successfully read |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the statistics table |
| *MT90528_MIB_READ_ERROR* | error reading MIB counters from the MT90528 device |

**Description**

This routine reads the MIB device counters for a specified VC and passes back the counter values in the *pMibLoad* structure.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.1.11 Mt90528LoadDeviceMibCounters

**Prototype**

```
USHORT Mt90528LoadDeviceMibCounters(
      USHORT                  VcIndex,
      s_mt90528_statistics    * pStatistics,
      s_hw_load               * pMibLoad
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pMibLoad* | pointer to user supplied parameter structure for the MIB hardware counters.<br>See 5.9, “Structure s_hw_load“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the device MIB values were successfully set |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the statistics table |
| *MT90528_MIB_READ_ERROR* | error reading MIB counters from the MT90528 device |
| *MT90528_MIB_WRITE_ERROR* | error writing MIB counter values to the MT90528 device |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active - load Mib failed as Reassembled Cell Counter cannot be changed |

**Description**

This routine loads the MIB device counters for a specified VC with counter values supplied in the *pMibLoad* structure.

This routine will affect the MIB / PM statistics, so caution is advised.

**WARNING**: If thresholds are being used, do not use this routine to load the device hardware counters. If this routine is used, threshold crossings will not occur when expected.

**NOTE**: the VcIndex can be acquired using the function Mt90528GetCbrVcStatIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.1.12 Mt90528ReadUdtRxsarCsVcArrival

#### Prototype

```
USHORT Mt90528ReadUdtRxsarCsVcArrival(
      USHORT                  Port,
      s_mt90528_statistics    * pStatistics,
      UCHAR                   * pVcArrival
      )
```

#### Parameters

| | |
|---|---|
| *Port* | TDM port id |
| *pStatistics* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pVcArrival* | pointer to VC arrival bit value |

#### Return values

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | entry is updated |
| *MT90528_MIB_READ_ERROR* | could not read the control structure vc arrival bit |
| *MT90528_MIB_PORT_OUT_OF_RANGE* | specified TDM port is not in range for device |

#### Description

Reads the MT90528 device UDT Rxsar control structure V bit (VC arrival) for the specified TDM port.

#### Source files

mib90528.h, mib90528.c

### 2.5.1.13 Mt90528ReadSdtRxsarCsVcArrival

#### Prototype

```
USHORT Mt90528ReadSdtRxsarCsVcArrival(
      USHORT                  VcIndex,
      s_mt90528_statistics    * pStatistics,
      UCHAR                   * pVcArrival
      )
```

#### Parameters

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pStatistics* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pVcArrival* | pointer to VC arrival bit value |

#### Return values

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | entry is updated |
| *MT90528_MIB_READ_ERROR* | could not read the control structure vc arrival bit |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the statistics table |

**Description**

Reads the MT90528 device SDT Rxsar control structure V bit (VC arrival) for the specified VcIndex.

**Source files**

mib90528.h, mib90528.c

### 2.5.1.14 Mt90528ClearUdtRxsarCsVcArrival

**Prototype**

```
USHORT Mt90528ClearUdtRxsarCsVcArrival(
      USHORT                  Port,
      s_mt90528_statistics    * pStatistics
      )
```

**Parameters**

| | |
|---|---|
| *Port* | TDM port id |
| *pStatistics* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | entry is updated |
| *MT90528_MIB_READ_ERROR* | could not read the control structure vc arrival bit |
| *MT90528_MIB_WRITE_ERROR* | could not write the control structure vc arrival bit |
| *MT90528_MIB_PORT_OUT_OF_RANGE* | specified TDM port is not in range for device |

**Description**

Clears the MT90528 device UDT Rxsar control structure V bit (VC arrival) for the specified TDM port.

**Source files**

mib90528.h, mib90528.c

### 2.5.1.15 Mt90528ClearSdtRxsarCsVcArrival

**Prototype**

```
USHORT Mt90528ClearSdtRxsarCsVcArrival(
      USHORT                  VcIndex,
      s_mt90528_statistics    * pStatistics
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *pStatistics* | pointer to MIB statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | entry is updated |
| *MT90528_MIB_READ_ERROR* | could not read the control structure vc arrival bit |
| *MT90528_MIB_WRITE_ERROR* | could not write the control structure vc arrival bit |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the statistics table |

**Description**

Clears the MT90528 device SDT Rxsar control structure V bit (VC arrival) for the specified VcIndex.

**Source files**

mib90528.h, mib90528.c

### 2.5.1.16 Mt90528ScanForCellLossTimeoutOnVcs

**Prototype**

```
USHORT Mt90528ScanForCellLossTimeoutOnVcs(
      s_mt90528_statistics                * pStatistics,
      s_mt90528_cbr_vc_cell_loss_status   * pStatus
      )
```

**Parameters**

| | |
|---|---|
| *pStatistics* | pointer to user supplied statistics structure.<br>See 5.4, “Structure s_mt90528_statistics“. |
| *pStatus* | pointer to cell loss status structure<br>NOTE: the quick status flag pStatus->CellLossSet indicates<br>TRUE(1) at least one CBR VC has continued cell loss or<br>FALSE(0) no CBR VC's had continued cell loss.<br>See 5.17, “Structure s_mt90528_cbr_vc_cell_loss_status“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | - scan was successful |
| *MT90528_MIB_NO_ROOM* | - no room left for new entry in cell loss status table |
| *return_code* | - return_codes from Mt90528ReadUdtRxsarCsVcArrival or<br>Mt90528ReadSdtRxsarCsVcArrival or<br>Mt90528ClearUdtRxsarCsVcArrival or<br>Mt90528ClearSdtRxsarCsVcArrival |

**Description**

This device servicing routine should be called typically every 100 milliseconds to poll for CBR VC cell loss timeouts on opened VCs. The s_mt90528_statistics structure is used by this function to keep track of cell loss. This function updates the *AtmfCesCellLossStatus* field in the CBR VC statistics structure for a VC to either *STATS_CELL_LOSS* (1) if continued cell loss has occurred for the *AtmfCesCellLossIntegrationPeriod* (a field in the CBR VC statistics structure – default is 2500 milliseconds) or *STATS_CELL_NOLOSS* (2) if cells were received within the *AtmfCesCellLossIntegrationPeriod*. The application defined *pStatus* structure which is updated and passed back to the user indicates the CBR VCs that have continued cell loss for the cell loss integration period.

**Source files**

mib90528.h, mib90528.c

### 2.5.2 Status Functions

#### 2.5.2.1 Mt90528GetNextTdmPortStatusIndex

**Prototype**

```
UCHAR Mt90528GetNextTdmPortStatusIndex(
      s_mt90528_status  * pStatus,
      USHORT            * pTdmIndex
      )
```

**Parameters**

| | |
|---|---|
| *pStatus* | pointer to user supplied status structure for interrupt, threshold and wrap notification.<br>See 5.15, “Structure s_mt90528_status“. |
| *pTdmIndex* | pointer to TDM index pointing to an entry in the TDM port status table |

**Return values**

| | |
|---|---|
| *TRUE* (1) | next tdm port index is a valid index in the status table |
| *FALSE* (0) | no more entries in the status table |

**Description**

This routine finds the next entry in the TDM port status table in the *pStatus* structure. If the end of the table is reached, the routine returns false. By calling Mt90528IsrHandler or Mt90528IsrHandlerWith2ndThresholds, the TDM port index is automatically reset.

**Source files**

mib90528.h, mib90528.c

#### 2.5.2.2 Mt90528GetNextCbrVcStatusIndex

**Prototype**

```
UCHAR Mt90528GetNextCbrVcStatusIndex(
      s_mt90528_status  * pStatus,
      USHORT            * pVcIndex
      )
```

**Parameters**

| | |
|---|---|
| *pStatus* | pointer to user supplied status structure for interrupt, threshold and wrap notification.<br>See 5.15, “Structure s_mt90528_status“. |
| *pVcIndex* | pointer to CBR VC index pointing to an entry in the CBR VC status table |

**Return values**

| | |
|---|---|
| *TRUE* (1) | next CBR VC index is a valid index in the status table |
| *FALSE* (0) | no more entries in the status table |

**Description**

This routine finds the next entry in the CBR VC status table in the *pStatus* structure. If the end of the table is reached, the routine returns false. By calling Mt90528IsrHandler or Mt90528IsrHandlerWith2ndThresholds, the CBR VC index is automatically reset.

**Source files**

mib90528.h, mib90528.c

### 2.5.2.3 Mt90528GetNextCbrVcCellLossStatusIndex

**Prototype**

```
UCHAR Mt90528GetNextCbrVcCellLossStatusIndex(
      s_mt90528_cbr_vc_cell_loss_status   * pStatus,
      USHORT                              * pVcIndex
      )
```

**Parameters**

| | |
|---|---|
| *pStatus* | pointer to user supplied cell loss status structure for CBR VC cell loss notification. See 5.17, “Structure s_mt90528_cbr_vc_cell_loss_status“. |
| *pVcIndex* | pointer to CBR VC index pointing to an entry in the CBR VC cell loss status table |

**Return values**

| | |
|---|---|
| *TRUE* (1) | next CBR VC index is a valid index in the status table |
| *FALSE* (0) | no more entries in the status table |

**Description**

This routine finds the next entry in the CBR VC cell loss status table in the *pStatus* structure. If the end of the table is reached, the routine returns false. By calling *Mt90528ScanForCellLossTimeoutOnVcs* the *pStatus* structure is automatically reset.

**Source files**

mib90528.h, mib90528.c

### 2.5.3 Secondary Threshold and Counter Functions

#### 2.5.3.1 Mt90528InitializeThresholdTable

**Prototype**

```
void Mt90528InitializeThresholdTable(
      s_mt90528_threshold     * pThreshold
      )
```

**Parameters**

*pThreshold* pointer to user supplied secondary threshold and MIB / PM software counter structure.
See 5.19, “Structure s_mt90528_threshold“.

**Return values**

None.

**Description**

This routine initializes the secondary threshold and counter structure.

**Source files**

mib90528.h, mib90528.c

#### 2.5.3.2 Mt90528AddVcToCbrVcThresholdTable

**Prototype**

```
USHORT Mt90528AddVcToCbrVcThresholdTable(
      s_vc                    Vc,
      s_mt90528_threshold     * pThreshold,
      USHORT                  * pVcIndex
      )
```

**Parameters**

*Vc* user supplied structure containing the VPI and VCI values for this VC.
See 4.1, “Structure s_vc“.

*pThreshold* pointer to user supplied secondary threshold and MIB / PM software counter structure.
See 5.19, “Structure s_mt90528_threshold“.

*pVcIndex* pointer to CBR VC index pointing to the CBR VC entry in the pThreshold structure

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | VC added to threshold table |
| *MT90528_MIB_VC_ALREADY_OPEN* | VC was previously added to the threshold table |
| *MT90528_MIB_NO_ROOM* | no more entries can be added to the threshold table |

**Description**

This routine adds an entry in the *pThreshold* structure for the specified VC and passes back the index.

**Source files**

mib90528.h, mib90528.c

### 2.5.3.3 Mt90528RemoveVcFromCbrVcThresholdTable

**Prototype**

```
USHORT Mt90528RemoveVcFromCbrVcThresholdTable(
      s_vc                    Vc,
      s_mt90528_threshold     * pThreshold
      )
```

**Parameters**

| | |
|---|---|
| *Vc* | user supplied structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *pThreshold* | pointer to user supplied secondary threshold and MIB / PM software counter structure. See 5.19, “Structure s_mt90528_threshold“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | VC removed from threshold table |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the threshold table |
| *MT90528_MIB_INDEX_OUT_OF_RANGE* | VC index is out of range for table |

**Description**

This routine removes an entry from the *pThreshold* structure for the specified VC.

**Source files**

mib90528.h, mib90528.c

### 2.5.3.4 Mt90528GetCbrVcThresholdIndex

**Prototype**

```
USHORT Mt90528GetCbrVcThresholdIndex(
      s_vc                    Vc,
      s_mt90528_threshold     * pThreshold,
      USHORT                  * pVcIndex
      )
```

**Parameters**

| | |
|---|---|
| *Vc* | user supplied structure containing the VPI and VCI values for this VC. See 4.1, “Structure s_vc“. |
| *pThreshold* | pointer to user supplied secondary threshold and MIB / PM software counter structure. See 5.19, “Structure s_mt90528_threshold“. |
| *pVcIndex* | pointer to CBR VC index pointing to the CBR VC entry in the pThreshold structure |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | VC removed from threshold table |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the threshold table |

**Description**

This routine finds an entry in the *pThreshold* structure for the specified VC and passes back the index.

**Source files**

mib90528.h, mib90528.c

### 2.5.3.5 Mt90528SetAppThresholds

**Prototype**

```
USHORT Mt90528SetAppThresholds(
      USHORT                  VcIndex,
      USHORT                  VcIndex2,
      s_stats_params          * pParam,
      s_mt90528_statistics    * pStatistics,
      s_mt90528_threshold     * pThreshold
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *VcIndex2* | CBR VC index pointing to the CBR VC entry in the pThreshold structure |
| *pParam* | pointer to user supplied threshold parameter structure. See 5.11, “Structure s_stats_params“. |
| *pStatistics* | pointer to user supplied Mib statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pThreshold* | pointer to user supplied secondary threshold and MIB / PM software counter structure. See 5.19, “Structure s_mt90528_threshold“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the thresholds were successfully set |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

**Description**

This routine sets the threshold values in the *pThreshold* structure for the second set of thresholds (i.e. 24-hour period).

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex and *VcIndex2* can be acquired using the function Mt90528GetCbrVcThresholdIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.3.6 Mt90528SetAppCounters32

**Prototype**

```
USHORT Mt90528SetAppCounters32(
      USHORT                  VcIndex,
      s_stats_params          * pParam,
      s_mt90528_threshold     * pThreshold
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pThreshold structure |
| *pParam* | pointer to user supplied counter parameter structure. See 5.11, “Structure s_stats_params“. |
| *pThreshold* | pointer to user supplied secondary threshold and MIB / PM software counter structure. See 5.19, “Structure s_mt90528_threshold“. |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the counter values were successfully set |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the threshold table |

**Description**

The 32 bit APP software counters can be preset to specified values.

This routine sets the software counter values in the *pThreshold* structure for the second set of counters (i.e. 24-hour period). ). Using Mt90528GetCbrVcAppStatistics with reset counters enabled is the normal method of setting the APP software counters. This routine is intended for debug purposes and not intended for normal operation.

**WARNING**: If thresholds are being used, do not use this routine to set the App software counters or else threshold crossings will not occur when expected.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcThresholdIndex

#### Source files

mib90528.h, mib90528.c

### 2.5.3.7 Mt90528UpdateAppCounters32

#### Prototype

```
USHORT Mt90528UpdateAppCounters32(
      USHORT                        VcIndex,
      s_mt90528_cbr_vc_stats_entry  * pCbrVcStat,
      s_mt90528_threshold           * pThreshold
      )
```

#### Parameters

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pThreshold structure |
| *pCbrVcStats* | pointer to user supplied CBR VC statistics structure entry.<br>See 5.3, “Structure s_mt90528_cbr_vc_stats_entry“ |
| *pThreshold* | pointer to user supplied secondary threshold and MIB / PM software counter structure.<br>See 5.19, “Structure s_mt90528_threshold“. |

#### Return values

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the counter values were successfully updated |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the threshold table |

#### Description

This routine updates the counter values in the pThreshold structure for the second set of counters (i.e. 24-hour period) from the first set of counters (i.e. 15-minute period). This routine would be called after Mt90528GetCbrVcStatistics function is called for the same CBR VC.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcThresholdIndex

#### Source files

mib90528.h, mib90528.c

### 2.5.3.8 Mt90528GetCbrVcAppStatistics

**Prototype**

```
USHORT Mt90528GetCbrVcAppStatistics(
      USHORT                                        VcIndex,
      s_mt90528_threshold                           * pThreshold,
      s_mt90528_cbr_vc_statistic_threshold_entry    * pCbrVcAppStats
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pThreshold structure. |
| *pThreshold* | pointer to user supplied secondary threshold and MIB / PM softwarecounter structure. See 5.19, “Structure s_mt90528_threshold“. |
| *pCbrVcAppStats* | pointer to user supplied CBR VC MIB / PM statistics structure entry |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the statistics were successfully copied |
| *MT90528_MIB_INVALID_VC* | specified VC index not found in the threshold table |

**Description**

This routine collects the CBR VC secondary MIB / PM threshold and counter statistics from the pThreshold structure of the MT90528 device. A CBR VC secondary threshold and counter structure entry is copied to the pCbrVcAppStats structure. There is an option field in the pCbrVAppStats structure named ResetStatistics that if set for TRUE (1) will reset any interrupt counters that are CBR VC specific for the specified VcIndex in the pThreshold structure after collecting the statistics.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcThresholdIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.3.9 Mt90528EnableDisableCbrVcAppStatisticsCollection

**Prototype**

```
USHORT Mt90528EnableDisableCbrVcAppStatisticsCollection(
      USHORT                VcIndex,
      USHORT                VcIndex2,
      Flag                  Status,
      s_mt90528_statistics  * pStatistics,
      s_mt90528_threshold   * pThreshold
      )
```

**Parameters**

| | |
|---|---|
| *VcIndex* | CBR VC index pointing to the CBR VC entry in the pStatistics structure |
| *VcIndex2* | CBR VC index pointing to the CBR VC entry in the pThreshold structure |
| *Status* | ENABLED (1) or DISABLED (0) flag |
| *pStatistics* | pointer to user supplied Mib statistics structure. See 5.4, “Structure s_mt90528_statistics“. |
| *pThreshold* | pointer to user supplied App threshold structure |

**Return values**

| | |
|---|---|
| *MT90528_MIB_NO_ERROR* | the statistics were successfully enabled or disabled |
| *MT90528_MIB_INVALID_VC* | no valid VC, indexed by VcIndex found in the statistics table |
| *MT90528_MIB_READ_ERROR* | could not read the control structure mib counters |
| *MT90528_MIB_WRITE_ERROR* | could not write the control structure mib counters |
| *MT90528_MIB_REASS_FIRST_CELL_HANDLING_ACTIVE* | first cell handling active, cannot change reassembled cell count |

**Description**

This routine enables or disables collection of statistics for the specified CBR VC index in the *pThreshold* structure of the MT90528 device using the *Status* flag. If collection is disabled, the 32 bit MIB / PM software counters will not be updated and no threshold alerts will occur for the CBR VC during interrupts which are in the *pThreshold* structure for the CBR VC. **The system default for collection of CBR VC application statistics is DISABLED**. If collection is enabled after previously being disabled, using this function, the 32 bit MIB / PM and APP software counters are cleared. Enabling collection after already being enabled will not cause the 32 bit MIB / PM and APP software counters to be cleared. If collection is disabled, the device MIB hardware counters are cleared.

**NOTE**: the *VcIndex* can be acquired using the function Mt90528GetCbrVcStatIndex and *VcIndex2* can be acquired using the function Mt90528GetCbrVcThresholdIndex

**Source files**

mib90528.h, mib90528.c

### 2.5.4 Application Examples using Statistics Counters and Thresholds

#### 2.5.4.1 MIB statistics only with no Thresholds set

The following is an example of using only the MIB (i.e. 15 minute) statistics collection.

```
UINT Device;
s_vc Vc;
s_mt90528_statistics Statistics[MAX_MT90528_DEVICES];
s_mt90528_cbr_vc_stats_entry vc_mib_stats;
USHORT VcIndex;
USHORT return_code;


Device = 0;
```

```
/* CBR VC opened here – the open creates the CBR VC entry in the MIB statistics table */

/* get the VC index into the MIB statistics table */
if ( (return_code = Mt90528GetCbrVcStatIndex( Vc, &Statistics[Device], &VcIndex ))
        == MT90528_MIB_NO_ERROR )
        return return_code;

/* enable CBR VC statistics for MIB only */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, ENABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* counters incremented by device */

/* get the MIB software counters for this VC */
vc_mib_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcStatistics( VcIndex, &Statistics[Device],
        &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB data - vc_mib_stats has a copy of the software counters values */

/* counters incremented by device */

/* 15 minute period is up – application collects and resets counters for next period */
vc_mib_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcStatistics( VcIndex, &Statistics[Device],
        &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB data - vc_mib_stats has a copy of the software counters values */

/* disable CBR VC statistics for MIB */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, DISABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* CBR VC closed here – the close removes the CBR VC entry from the MIB statistics table */
```

### 2.5.4.2 MIB statistics only with MIB Thresholds set

The following is an example of using only the MIB (i.e. 15 minute) statistics collection with thresholds.

```
UINT Device;
s_vc Vc;
s_mt90528_statistics Statistics[MAX_MT90528_DEVICES];
s_mt90528_cbr_vc_stats_entry vc_mib_stats;
USHORT VcIndex;
s_stats_params set_mib;
USHORT return_code;

Device = 0;

/* SDT CBR VC opened here – the open creates the CBR VC entry in the MIB statistics table */
```

```
/* get the VC index into the MIB statistics table */
if ( (return_code = Mt90528GetCbrVcStatIndex( Vc, &Statistics[Device], &VcIndex ))
        == MT90528_MIB_NO_ERROR )
        return return_code;

/* enable CBR VC statistics for MIB only */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, ENABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* set the status flags to indicate that all thresholds should be set */
set_mib.Status.Aal1HdrErrors = TRUE;
set_mib.Status.Aal1SeqErrors = TRUE;
set_mib.Status.BufOverflows = TRUE;
set_mib.Status.BufUnderflows = TRUE;
set_mib.Status.LostCells = TRUE;
set_mib.Status.MisinsertedCells = TRUE;
set_mib.Status.PointerReframes = TRUE;
set_mib.Status.PointerParityErrors = TRUE;

/* set MIB thresholds for all counters for the VC */
set_mib.Value.Aal1HdrErrors = 20;
set_mib.Value.Aal1SeqErrors = 21;
set_mib.Value.BufOverflows = 22;
set_mib.Value.BufUnderflows = 23;
set_mib.Value.LostCells = 24;
set_mib.Value.MisinsertedCells = 25;
set_mib.Value.PointerReframes = 26;
set_mib.Value.PointerParityErrors = 27;

/* set MIB thresholds for the VC */
if ( (return_code = Mt90528SetMibThresholds( VcIndex, &set_mib, &Statistics[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* counters incremented by device – threshold crossings occur */
/* ISR handler returns threshold crossing indicators to application */

/* get the MIB software counters for this VC */
vc_mib_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcStatistics( VcIndex, &Statistics[Device],
        &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB data - vc_mib_stats has a copy of the software counters values */

/* counters incremented by device – threshold crossings occur */
/* ISR handler returns threshold crossing indicators to application */

/* 15 minute period is up – application collects and resets counters for next period */
vc_mib_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcStatistics( VcIndex, &Statistics[Device],
        &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB data - vc_mib_stats has a copy of the software counters values */
```

```
/* disable CBR VC statistics for MIB */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, DISABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* SDT CBR VC closed here – the close removes the CBR VC entry from the MIB statistics table */
```

### 2.5.4.3 MIB and APP statistics with no Thresholds set

The following is an example of using the MIB (i.e. 15 minute) and APP (i.e. 24 hour) statistics collection.

```
UINT Device;
s_vc Vc;
s_mt90528_statistics Statistics[MAX_MT90528_DEVICES];
s_mt90528_threshold AppStats[MAX_MT90528_DEVICES];
s_mt90528_cbr_vc_stats_entry vc_mib_stats;
s_mt90528_cbr_vc_statistic_threshold_entry vc_app_stats;
USHORT VcIndex;
USHORT AppVcIndex;
USHORT return_code;

Device = 0;

/* during initialization of the API – initialize the application statistics table */
Mt90528InitializeThresholdTable( &AppStats[Device] );

/* CBR VC opened here – the open creates the CBR VC entry in the MIB statistics table */

/* get the VC index into the MIB statistics table */
if ( (return_code = Mt90528GetCbrVcStatIndex( Vc, &Statistics[Device], &VcIndex ))
        == MT90528_MIB_NO_ERROR )
        return return_code;

/* add the VC to the application stats table */
if ( (return_code = Mt90528AddVcToCbrVcThresholdTable( Vc, &AppStats[Device],
        &AppVcIndex )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* enable CBR VC statistics for MIB */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, ENABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* enable CBR VC statistics for APP */
if ( (return_code = Mt90528EnableDisableCbrVcAppStatisticsCollection( VcIndex,
        AppVcIndex, ENABLED, &Statistics[Device], &AppStats[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* counters incremented by device */

/* get the MIB software counters for this VC but don't reset the counters */
vc_mib_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcStatistics( VcIndex, &Statistics[Device],
```

```
        &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* get the APP software counters for this VC but don't reset the counters */
vc_app_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB and APP data - vc_mib_stats and vc_app_stats have a copy of the software counters values */

/* counters incremented by device */

/* 15 minute period is up – application collects and resets counters for next period */
vc_mib_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcStatistics( VcIndex, &Statistics[Device],
        &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* update the counters in the APP threshold counter table */
if ( Mt90528UpdateAppCounters32( AppVcIndex, &vc_mib_stats, &AppStats[Device] )
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* get the APP software counters for this VC but don't reset the counters */
vc_app_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB and APP data - vc_mib_stats and vc_app_stats have a copy of the software counters values */

/* counters incremented by device */
/* ISR handler returns threshold crossing indicators to application */

/* 15 minute period is up and at 24-hour mark - application collects and resets counters for next period */
vc_mib_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* update the counters in the APP counter table */
if ( Mt90528UpdateAppCounters32( AppVcIndex, &vc_mib_stats, &AppStats[Device] )
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* CBR VC counters for APP needs to be collected and reset at the 24-hour mark */
vc_app_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the APP data for 24 hour reporting - vc_app_stats has a copy of the software counters values */

/* disable CBR VC statistics for MIB */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, DISABLED,
```

```
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* disable CBR VC statistics for APP */
if ( (return_code = Mt90528EnableDisableCbrVcAppStatisticsCollection( VcIndex,
        AppVcIndex, DISABLED, &Statistics[Device], &AppStats[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* CBR VC closed here – the close removes the CBR VC entry from the MIB statistics table */

/* remove the VC from the application stats table */
if ( (return_code = Mt90528RemoveVcFromCbrVcThresholdTable( Vc, &AppStats[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;
```

#### 2.5.4.4 MIB and APP statistics with MIB and APP Thresholds set

The following is an example of using the MIB (i.e. 15 minute) and APP (i.e. 24 hour) statistics collection with thresholds.

```
UINT Device;
s_vc Vc;
s_mt90528_statistics Statistics[MAX_MT90528_DEVICES];
s_mt90528_threshold AppStats[MAX_MT90528_DEVICES];
s_mt90528_cbr_vc_stats_entry vc_mib_stats;
s_mt90528_cbr_vc_statistic_threshold_entry vc_app_stats;
s_stats_params set_mib;
s_stats_params app_set_mib;
USHORT VcIndex;
USHORT AppVcIndex;
USHORT return_code;

Device = 0;

/* during initialization of the API – initialize the application statistics table */
Mt90528InitializeThresholdTable( &AppStats[Device] );

/* SDT CBR VC opened here – the open creates the CBR VC entry in the MIB statistics table */

/* get the VC index into the MIB statistics table */
if ( (return_code = Mt90528GetCbrVcStatIndex( Vc, &Statistics[Device], &VcIndex ))
        == MT90528_MIB_NO_ERROR )
        return return_code;

/* add the VC to the application stats table */
if ( (return_code = Mt90528AddVcToCbrVcThresholdTable( Vc, &AppStats[Device],
        &AppVcIndex )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* enable CBR VC statistics for MIB */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, ENABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* enable CBR VC statistics for APP */
if ( (return_code = Mt90528EnableDisableCbrVcAppStatisticsCollection( VcIndex,
```

```
        AppVcIndex, ENABLED, &Statistics[Device], &AppStats[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* set the status flags to indicate that all MIB thresholds should be set */
set_mib.Status.Aal1HdrErrors = TRUE;
set_mib.Status.Aal1SeqErrors = TRUE;
set_mib.Status.BufOverflows = TRUE;
set_mib.Status.BufUnderflows = TRUE;
set_mib.Status.LostCells = TRUE;
set_mib.Status.MisinsertedCells = TRUE;
set_mib.Status.PointerReframes = TRUE;
set_mib.Status.PointerParityErrors = TRUE;

/* set MIB thresholds for all counters for the VC */
set_mib.Value.Aal1HdrErrors = 20;
set_mib.Value.Aal1SeqErrors = 21;
set_mib.Value.BufOverflows = 22;
set_mib.Value.BufUnderflows = 23;
set_mib.Value.LostCells = 24;
set_mib.Value.MisinsertedCells = 25;
set_mib.Value.PointerReframes = 26;
set_mib.Value.PointerParityErrors = 27;

/* set MIB thresholds for the VC */
if ( (return_code = Mt90528SetMibThresholdsWithAppThresholds( VcIndex, AppVcIndex,
        &set_mib, &Statistics[Device], &AppStats[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* set the status flags to indicate that all APP thresholds should be set */
app_set_mib.Status.Aal1HdrErrors = TRUE;
app_set_mib.Status.Aal1SeqErrors = TRUE;
app_set_mib.Status.BufOverflows = TRUE;
app_set_mib.Status.BufUnderflows = TRUE;
app_set_mib.Status.LostCells = TRUE;
app_set_mib.Status.MisinsertedCells = TRUE;
app_set_mib.Status.PointerReframes = TRUE;
app_set_mib.Status.PointerParityErrors = TRUE;

/* set APP thresholds for all counters for the VC */
app_set_mib.Value.Aal1HdrErrors = 40;
app_set_mib.Value.Aal1SeqErrors = 41;
app_set_mib.Value.BufOverflows = 42;
app_set_mib.Value.BufUnderflows = 43;
app_set_mib.Value.LostCells = 44;
app_set_mib.Value.MisinsertedCells = 45;
app_set_mib.Value.PointerReframes = 46;
app_set_mib.Value.PointerParityErrors = 47;

/* set APP thresholds for the VC */
if ( (return_code = Mt90528SetAppThresholds( VcIndex, AppVcIndex, &app_set_mib,
        &Statistics[Device], &AppStats[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* counters incremented by device */
/* ISR handler returns threshold crossing indicators to application */
```

```
/* get the MIB software counters for this VC but don't reset the counters */
vc_mib_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcStatisticsWithAppThresholds( VcIndex, AppVcIndex,
        &Statistics[Device], &AppStats[Device], &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* get the APP software counters for this VC but don't reset the counters */
vc_app_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB and APP data - vc_mib_stats and vc_app_stats have a copy of the software counters values */

/* counters incremented by device */
/* ISR handler returns threshold crossing indicators to application */

/* 15 minute period is up – application collects and resets counters for next period */
vc_mib_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcStatisticsWithAppThresholds( VcIndex, AppVcIndex,
        &Statistics[Device], &AppStats[Device], &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* update the counters in the APP threshold counter table */
if ( Mt90528UpdateAppCounters32( AppVcIndex, &vc_mib_stats, &AppStats[Device] )
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* get the APP software counters for this VC but don't reset the counters */
vc_app_stats.ResetStatistics = FALSE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* application uses the MIB and APP data - vc_mib_stats and vc_app_stats have a copy of the software counters values */

/* counters incremented by device */
/* ISR handler returns threshold crossing indicators to application */

/* 15 minute period is up and at 24-hour mark - application collects and resets counters for next period */
vc_mib_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcStatisticsWithAppThresholds( VcIndex, AppVcIndex,
        &Statistics[Device], &AppStats[Device], &vc_mib_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* update the counters in the APP threshold counter table */
if ( Mt90528UpdateAppCounters32( AppVcIndex, &vc_mib_stats, &AppStats[Device] )
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* CBR VC counters for APP needs to be collected and reset at the 24-hour mark */
vc_app_stats.ResetStatistics = TRUE;
if ( (return_code = Mt90528GetCbrVcAppStatistics( AppVcIndex, &AppStats[Device],
        &vc_app_stats )) != MT90528_MIB_NO_ERROR )
        return return_code;
```

```
/* application uses the APP data for 24 hour reporting - vc_app_stats has a copy of the software counters values */

/* disable CBR VC statistics for MIB */
if ( (return_code = Mt90528EnableDisableCbrVcStatisticsCollection( VcIndex, DISABLED,
        &Statistics[Device] )) != MT90528_MIB_NO_ERROR )
        return return_code;

/* disable CBR VC statistics for APP */
if ( (return_code = Mt90528EnableDisableCbrVcAppStatisticsCollection( VcIndex,
        AppVcIndex, DISABLED, &Statistics[Device], &AppStats[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;

/* SDT CBR VC closed here – the close removes the CBR VC entry from the MIB statistics table */

/* remove the VC from the application stats table */
if ( (return_code = Mt90528RemoveVcFromCbrVcThresholdTable( Vc, &AppStats[Device] ))
        != MT90528_MIB_NO_ERROR )
        return return_code;
```

## 3.0 Constants

| Constant Name | Value |
|---|---|
| CCR_1544_KHZ | 0 |
| CCR_2048_KHZ | 1 |
| CCR_4096_KHZ | 2 |
| CCR_ADAPTIVE_MODE | 3 |
| CCR_CLKSEL_COMMON_CLK | 3 |
| CCR_CLKSEL_MCLK_DIV_2 | 0 |
| CCR_CLKSEL_PLLCLK | 2 |
| CCR_CLKSEL_STICLK | 1 |
| CCR_CPU_MODE | 3 |
| CCR_FNXI1_RATE | 0 |
| CCR_FNXI2_RATE | 1 |
| CCR_FREERUN_MODE | 2 |
| CCR_HOLDOVER_MODE | 1 |
| CCR_LINE_CLOCKING_MODE | 0 |
| CCR_NETWORK_MODE | 2 |
| CCR_NORMAL_MODE | 0 |
| CCR_RTSSEL_C4M_C2M | 1 |

**Table 2 - Register Constants and Defaults**

| Constant Name | Value |
|---|---|
| CCR_RTSSEL_PLLCLK | 2 |
| CCR_RTSSEL_STICLK | 0 |
| CCR_SRTS_MODE | 1 |
| CHUNK0 | 0 |
| CHUNK1 | 10000h |
| CHUNK2 | 20000h |
| CHUNK3 | 30000h |
| CHUNK4 | 40000h |
| CHUNK5 | 50000h |
| CHUNK6 | 60000h |
| CHUNK7 | 70000h |
| CHUNK8 | 80000h |
| CHUNK9 | 90000h |
| CHUNK10 | A0000h |
| CHUNK11 | B0000h |
| CHUNK12 | C0000h |
| CHUNK13 | D0000h |
| CHUNK14 | E0000h |
| CHUNK15 | F0000h |
| CHUNK16 | 100000h |
| CHUNK17 | 110000h |
| CHUNK18 | 120000h |
| CHUNK19 | 130000h |
| CHUNK20 | 140000h |
| CHUNK21 | 150000h |
| CHUNK22 | 160000h |
| CHUNK23 | 170000h |
| CHUNK24 | 180000h |
| CHUNK25 | 190000h |
| CHUNK26 | 1A0000h |

**Table 2 - Register Constants and Defaults**

| Constant Name | Value |
|---|---|
| CHUNK27 | 1B0000h |
| CHUNK28 | 1C0000h |
| CHUNK29 | 1D0000h |
| CHUNK30 | 1E0000h |
| CHUNK31 | 1F0000h |
| CMCR_1215_KHZ | 4 |
| CMCR_2430_KHZ | 5 |
| CMCR_151875_HZ | 1 |
| CMCR_303750_HZ | 2 |
| CMCR_4860_KHZ | 6 |
| CMCR_607500_HZ | 3 |
| CMCR_75937_HZ | 0 |
| CMCR_9720_KHZ | 7 |
| CMCR_DIRECT_PHY_CLK | 0 |
| CMCR_DIVIDED_PHY_CLK | 1 |
| CMCR_EXTERNAL | 1 |
| CMCR_GENERIC_NEG_TRIG_NEG_POL | 0 |
| CMCR_GENERIC_NEG_TRIG_POS_POL | 1 |
| CMCR_GENERIC_POS_TRIG_NEG_POL | 2 |
| CMCR_GENERIC_POS_TRIG_POS_POL | 3 |
| CMCR_INTERNAL | 0 |
| CMCR_MASTER_MODE | 0 |
| CMCR_SLAVE_MODE | 1 |
| CMCR_ST_BUS | 4 |
| DATA_CELL_BUFFER_SIZE_16 | 0 |
| DATA_CELL_BUFFER_SIZE_32 | 1 |
| DATA_CELL_BUFFER_SIZE_64 | 2 |
| DATA_CELL_BUFFER_SIZE_128 | 3 |
| DEF_CCR_CLKSEL | CCR_CLKSEL_PLLCLK |
| DEF_CCR_RTSSEL | CCR_RTSSEL_STICLK |

**Table 2 - Register Constants and Defaults**

| Constant Name | Value |
|---|---|
| DEF_MTCR_CUT_VC_PERIOD | 9C4h |
| DEF_PLL_FREQ_SEL | CCR_1544_KHZ |
| DEF_PLL_INPUT_SEL | CCR_LINE_CLOCKING_MODE |
| DEF_PLL_MODE_SEL | CCR_FREERUN_MODE |
| DEF_TCR_LATE_CELL_PERIOD | 7FFh |
| DEF_TXSAR_C | 0 |
| DEF_TXSAR_PTI | 0 |
| DEF_URCR_CHECK_LATE_ARRIVALS | DISABLED |
| DEF_URCR_UDT_DUMMY | FFh |
| DEF_URCR_UDT_INSERT_LOST | URCR_MAX_TWO_DUMMY_CELLS |
| DISABLED | 0 |
| E1_TIMESLOTS | 32 |
| ENABLED | 1 |
| FALSE | 0 |
| LAST_REGISTER_ON_CHIP | PERS |
| MAX_ADDRESS_INT_MEMORY | BE3FFh |
| MAX_DCGTOR_DCGP | 3FFh |
| MAX_PORTS | See [C313BCS4] MT90528 API Porting Guide and Software Design Specification |
| MAX_VC | See [C313BCS4] MT90528 API Porting Guide and Software Design Specification |
| MAX_DATA_VC | See [C313BCS4] MT90528 API Porting Guide and Software Design Specification |
| MAX_RXSAR_UDT_MAXIMUM_LEAD | 7FFh |
| MAX_SDT_CHANNELS | 128 |
| MAX_UCR_DEVICE_ADDRESS | 1Eh |
| MAX_UNCB_M | Ch |
| MAX_UNCB_N | 10h |
| MIN_ADDRESS_INT_MEMORY | 80000h |
| MIN_DCGTOR_DCGP | 0 |
| MIN_INT_MEM_TXSAR | 80000h |
| MIN_INT_MEM_TXSAR_REV3 | 80010h |

**Table 2 - Register Constants and Defaults**

| Constant Name | Value |
|---|---|
| MIN_RXSAR_UDT_MAXIMUM_LEAD | 5Fh |
| MIN_UCR_DEVICE_ADDRESS | 0 |
| MIN_UNCB_M | 0 |
| MIN_UNCB_N | 4 |
| PLLCS_PLLCLK | 1 |
| PLLCS_STICLK | 0 |
| RXSAR_BS_64 | 0 |
| RXSAR_BS_128 | 2 |
| RXSAR_BS_256 | 4 |
| RXSAR_BS_512 | 8 |
| RXSAR_BS_1024 | 10h |
| TDM1_3_CHANNELS_OUT_OF_4 | 1 |
| TDM1_1544_KHZ | 0 |
| TDM1_2048_KHZ | 1 |
| TDM1_BACKPLANE_MODE | 1 |
| TDM1_DS1_LINK | 1 |
| TDM1_E1_LINK | 0 |
| TDM1_FALLING_EDGE | 1 |
| TDM1_FIRST_24_CHANNELS | 0 |
| TDM1_FRAME_PULSE | 0 |
| TDM1_GENERIC | 0 |
| TDM1_INDEPENDENT_MODE | 0 |
| TDM1_LS_FOUR_BITS | 0 |
| TDM1_MS_FOUR_BITS | 1 |
| TDM1_MULTIFRAME_PULSE | 1 |
| TDM1_NEGATIVE_POLARITY | 0 |
| TDM1_NORMAL_OPERATION | 0 |
| TDM1_POSITIVE_POLARITY | 1 |
| TDM1_RISING_EDGE | 0 |
| TDM1_SDT_MODE | 2 |

**Table 2 - Register Constants and Defaults**

| Constant Name | Value |
|---|---|
| TDM1_SDT_MODE_NGREATER46 | 3 |
| TDM1_ST_BUS | 1 |
| TDM1_SWITCH_TO_INTERNAL_CLOCK | 1 |
| TDM3_FALLING_EDGE | 0 |
| TDM3_LOS_SIGNAL_ONES | 1 |
| TDM3_LOS_SIGNAL_ZEROS | 0 |
| TDM3_REPLAY | 1 |
| TDM3_RISING_EDGE | 1 |
| TDM3_SILENCE | 0 |
| TRUE | 1 |
| UCR_ATM_MODE | 0 |
| UCR_DISCARD_OAM_CELLS | 0 |
| UCR_DISCARD_UNKNOWN_CELLS | 0 |
| UCR_EIGHT_BIT_OPERATION | 1 |
| UCR_EXTERNAL_CLOCK | 1 |
| UCR_INTERNAL_CLOCK | 0 |
| UCR_KEEP_OAM_CELLS | 1 |
| UCR_KEEP_UNKNOWN_CELLS | 1 |
| UCR_LEVEL1 | 1 |
| UCR_LEVEL2 | 0 |
| UCR_NNI | 0 |
| UCR_PHY_MODE | 1 |
| UCR_SIXTEEN_BIT_OPERATION | 0 |
| UCR_UNI | 1 |
| URCR_MAX_TWO_DUMMY_CELLS | 0 |
| URCR_NUMBER_LOST_CELLS | 1 |

**Table 2 - Register Constants and Defaults**

| Label | Address | Name |
|---|---|---|
| CWRR | 0x0000 | Chip Wide Reset Register |
| MSR | 0x0002 | Main Status Register |
| LAWI | 0x0004 | Low Address Word Indirection Register |
| HAIC | 0x0006 | High Address Indirection Command Register |
| IDR | 0x0008 | Indirection Data Register |
| MIER | 0x000A | Main Interrupt Enable Register |
| MRR | 0x000C | MT90528 Revision Register |
| TXPTB_P(p) | (0x1000+(p)*0x2) | TX_SAR Pointer Table Base Register for port p |
| DTCON | 0x1038 | Data TX_SAR Configuration Register |
| DTWPR | 0x103A | Data TX_SAR Write Pointer Register |
| DTRPR | 0x103C | Data TX_SAR Read Pointer Register |
| DTCR | 0x103E | Data TX_SAR Control Register |
| DCGTOR | 0x1040 | Data Cell Generation Time Out Register |
| DTSR | 0x1042 | Data TX_SAR Status Register |
| TXEN | 0x1044 | TX_SAR Master Enable Register |
| URCR | 0x2000 | UDT Reassembly Control Register |
| URSER | 0x2002 | UDT Reassembly Service Enable Register |
| URSR | 0x2004 | UDT Reassembly Status Register |
| URCCR | 0x2006 | UDT Reassembly Cell Counter Register |
| DRCR | 0x2020 | Data RX_SAR Control Register |
| DRSR | 0x2022 | Data RX_SAR Status Register |
| DRCON | 0x2024 | Data RX_SAR Configuration Register |
| DRWPR | 0x2026 | Data RX_SAR Write Pointer Register |
| DRRPR | 0x2028 | Data RX_SAR Read Pointer Register |
| DRCCR | 0x202A | Data RX_SAR Cell Counter Register |
| SRCR | 0x2040 | SDT Reassembly Control Register |
| SRSER | 0x2042 | SDT Reassembly Service Enable Register |
| SRSR | 0x2044 | SDT Reassembly Status Register |
| SRCSR | 0x2046 | SDT Reassembly Cell Counter Status Register |
| SRCCR | 0x2048 | SDT Reassembly Cell Counter Register |

**Table 3 - Register Constants**

| Label | Address | Name |
|---|---|---|
| MTCR | 0x3000 | MIB Timeout Configuration Register |
| MTSR1 | 0x3002 | MIB Timeout Status Register 1 |
| MTSR2 | 0x3004 | MIB Timeout Status Register 2 |
| TCR_P(p) | (0x3200+(p)*0x2) | Timeout Configuration Register for port p |
| UCR | 0x4000 | UTOPIA Configuration Register |
| UNCB | 0x4002 | UTOPIA Number of Concatenated Bits Register |
| ULBA | 0x4004 | LUT Base Address Register |
| UVCM | 0x4006 | VC Match Register |
| UVCME | 0x4008 | VC Match Enable Register |
| UVPM | 0x400A | VP Match Register |
| UVPME | 0x400C | VP Match Enable Register |
| UPM | 0x400E | UTOPIA Parity Mismatches Register |
| UFS | 0x4010 | UTOPIA FIFO Status Register |
| USR | 0x4012 | UTOPIA Status Register |
| USER | 0x4014 | UTOPIA Service Enable Register |
| UICC | 0x4016 | UTOPIA Incoming Cell Counter |
| UVC_P(p) | (0x4200+(p)*0x4) | UDT VCI for Port p |
| UVP_P(p) | (0x4202+(p)*0x4) | UDT VPI for Port p |
| CMCR | 0x5000 | Clock Management Configuration Register |
| PLLCS | 0x5002 | External PLL Clock Source Register |
| CCR_P(p) | (0x5200+(p)*0x10) | Clocking Configuration Register for port p |
| CPAR_P(p) | (0x5202+(p)*0x10) | Clocking Phase Accumulator Register for port p |
| CDDR_P(p) | (0x5204+(p)*0x10) | Clocking DCO Difference Register for port p |
| CSFSR_P(p) | (0x5206+(p)*0x10) | SRTS FIFO Status Register for port p |
| PLLEN_P(p) | (0x5208+(p)*0x10) | PLL Enable Register for port p |
| MAINTDM1 | 0x6000 | Main TDM Control Register 1 |
| MAINTDM2 | 0x6002 | Main TDM Control Register 2 |
| TDM1_P(p) | (0x6200+(p)*0x10) | TDM Control Register 1 for port p |
| TDM2_P(p) | (0x6202+(p)*0x10) | TDM Control Register 2 for port p |
| TDM3_P(p) | (0x6204+(p)*0x10) | TDM Control Register 3 for port p |

**Table 3 - Register Constants**

| Label | Address | Name |
|---|---|---|
| TDM4_P(p) | (0x6206+(p)*0x10) | TDM Control Register 4 for port p |
| TDM5_P(p) | (0x6208+(p)*0x10) | TDM Control Register 5 for port p |
| TDM6_P(p) | (0x620A+(p)*0x10) | TDM Control Register 6 for port p |
| MACR | 0x7000 | Memory Arbiter Configuration Register |
| PERS | 0x7004 | Parity Error Status Register |

**Table 3 - Register Constants**

# 4.0 Configuration Structures

## 4.1 Structure s_vc

| Field Name | Definition | Range of Values |
|---|---|---|
| Vpi | USHORT | 0h..FFFh |
| Vci | USHORT | 0h..FFFFh |

## 4.2 Structure s_device_common_params

| Field Name | Definition | Range of Values |
|---|---|---|
| UcrUniNNni | UCHAR | UCR_NNI,<br>UCR_UNI |
| UcrUtoClkSel | UCHAR | UCR_INTERNAL_CLOCK,<br>UCR_EXTERNAL_CLOCK |
| UcrGlobalOamSel | UCHAR | UCR_DISCARD_OAM_CELLS,<br>UCR_KEEP_OAM_CELLS |
| UcrEightNSixteen | UCHAR | UCR_SIXTEEN_BIT_OPERATION,<br>UCR_EIGHT_BIT_OPERATION |
| UcrLevel1NLevel2 | UCHAR | UCR_LEVEL2,<br>UCR_LEVEL1 |
| UcrUkSel | UCHAR | UCR_DISCARD_UNKNOWN_CELLS,<br>UCR_KEEP_UNKNOWN_CELLS |
| UcrPhyNAtm | UCHAR | UCR_ATM_MODE,<br>UCR_PHY_MODE |
| VcMatch | USHORT | 0..FFFFh |
| VcMatchEnable | USHORT | 0..FFFFh |
| VpMatch | USHORT | 0..FFFh |
| VpMatchEnable | USHORT | 0..FFFh |

| Field Name | Definition | Range of Values |
|---|---|---|
| MtcrCutVcPeriod | USHORT | 0..FFFFh |
| CmcrExtNInt | UCHAR | CMCR_INTERNAL,<br>CMCR_EXTERNAL |
| Cmcr8KhzSel | UCHAR | CMCR_DIRECT_PHY_CLK,<br>CMCR_DIVIDED_PHY_CLK |
| PllcsPriClkSrcPrt | UCHAR | 0..1Bh |
| PllcsPriClkSrcTyp | UCHAR | PLLCS_STICLK,<br>PLLCS_PLLCLK |
| PllcsSecClkSrcPrt | UCHAR | 0..1Bh |
| PllcsSecClkSrcTyp | UCHAR | PLLCS_STICLK,<br>PLLCS_PLLCLK |
| ExtMemLutChunk | ULONG | CHUNK0..CHUNK31 |
| UncbM | USHORT | MIN_UNCB_M..MAX_UNCB_M |
| UncbN | USHORT | MIN_UNCB_N..MAX_UNCB_N |

### 4.3 Structure s_device_udt_specific_params

| Field Name | Definition | Range of Values |
|---|---|---|
| UrcrUdtDummy | UCHAR | 0..FFh |
| UrcrUdtInsertLost | UCHAR | URCR_MAX_TWO_DUMMY_CELLS ,<br>URCR_NUMBER_LOST_CELLS |
| UrcrCheckLateArrivals | UCHAR | DISABLED,<br>ENABLED |

### 4.4 Structure s_device_sdt_specific_params

| Field Name | Definition | Range of Values |
|---|---|---|
| CmcrSlvNMstr | UCHAR | CMCR_MASTER_MODE,<br>CMCR_SLAVE_MODE |
| CmcrF0Mode | UCHAR | CMCR_GENERIC_NEG_TRIG_NEG_POL,<br>CMCR_GENERIC_NEG_TRIG_POS_POL,<br>CMCR_GENERIC_POS_TRIG_NEG_POL,<br>CMCR_GENERIC_POS_TRIG_POS_POL,<br>CMCR_ST_BUS |

| Field Name | Definition | Range of Values |
|---|---|---|
| CmcrFnxi1Rate | UCHAR | CMCR_75937_HZ,<br>CMCR_151875_HZ,<br>CMCR_303750_HZ,<br>CMCR_607500_HZ,<br>CMCR_1215_KHZ,<br>CMCR_2430_KHZ,<br>CMCR_4860_KHZ,<br>CMCR_9720_KHZ |
| CmcrFnxi2Rate | UCHAR | CMCR_75937_HZ,<br>CMCR_151875_HZ,<br>CMCR_303750_HZ,<br>CMCR_607500_HZ,<br>CMCR_1215_KHZ,<br>CMCR_2430_KHZ,<br>CMCR_4860_KHZ,<br>CMCR_9720_KHZ |
| SrcrSdtDummy | UCHAR | 0..FFh |
| SrcrSdtInsertLost | UCHAR | DISABLED,<br>ENABLED |
| Maintdm1IdleData | UCHAR | 0..FFh |
| Maintdm1SilenceData | UCHAR | 0..FFh |
| Maintdm2IdleCas | UCHAR | 0..Fh |

### 4.5 Structure s_device_udt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| DeviceCommonParams | s_device_common_params | n/a |
| DeviceUdtSpecificParams | s_device_udt_specific_params | n/a |

### 4.6 Structure s_device_sdt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| DeviceCommonParams | s_device_common_params | n/a |
| DeviceSdtSpecificParams | s_device_sdt_specific_params | n/a |

### 4.7 Structure s_device_udt_and_sdt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| DeviceCommonParams | s_device_common_params | n/a |
| DeviceUdtSpecificParams | s_device_udt_specific_params | n/a |
| DeviceSdtSpecificParams | s_device_sdt_specific_params | n/a |

### 4.8 Structure s_data_cell_buffer_params

| Field Name | Definition | Range of Values |
|---|---|---|
| BufferMemoryChunk | ULONG | CHUNK0 .. CHUNK31 |
| BufferSize | UCHAR | DATA_CELL_BUFFER_SIZE_16,<br>DATA_CELL_BUFFER_SIZE_32,<br>DATA_CELL_BUFFER_SIZE_64,<br>DATA_CELL_BUFFER_SIZE_128, |

### 4.9 Structure s_data_tx_sar_enable_params

| Field Name | Definition | Range of Values |
|---|---|---|
| ContinuousTx | BOOLEAN | FALSE,<br>TRUE |
| Auto | UCHAR | FALSE,<br>TRUE |
| CellGenerationTimeout | USHORT | 0.. MAX_DCGTOR_DCGP |
| BufferEmptyService | UCHAR | DISABLED,<br>ENABLED |

### 4.10 Structure s_data_rx_sar_enable_params

| Field Name | Definition | Range of Values |
|---|---|---|
| CellBufferOverrunService | UCHAR | DISABLED,<br>ENABLED |
| CellArrivalService | UCHAR | DISABLED,<br>ENABLED |
| BufferHalfFullService | UCHAR | DISABLED,<br>ENABLED, |

### 4.11 Structure s_tdm_port_common_params

| Name | Description | Range of Values |
|---|---|---|
| TcrLateCellPeriod | USHORT | 0..7FFh |
| CcrClkSel | UCHAR | CCR_CLKSEL_MCLK_DIV_2,<br>CCR_CLKSEL_STICLK,<br>CCR_CLKSEL_PLLCLK,<br>CCR_CLKSEL_COMMON_CLK, |
| CcrRtsSel | UCHAR | CCR_RTSSEL_STICLK,<br>CCR_RTSSEL_C4M_C2M,<br>CCR_RTSSEL_PLLCLK, |
| CcrPllFreqSel | UCHAR | CCR_1544_KHZ,<br>CCR_2048_KHZ,<br>CCR_4096_KHZ |
| CcrPllModeSel | UCHAR | CCR_NORMAL_MODE,<br>CCR_HOLDOVER_MODE,<br>CCR_FREERUN_MODE,<br>CCR_CPU_MODE |
| CcrPllInputSel | UCHAR | CCR_LINE_CLOCKING_MODE,<br>CCR_SRTS_MODE,<br>CCR_NETWORK_MODE,<br>CCR_ADAPTIVE_MODE |
| CcrFnxiSel | UCHAR | CCR_FNXI1_RATE,<br>CCR_FNXI2_RATE |
| Tdm1TdmClkPol | UCHAR | TDM1_RISING_EDGE,<br>TDM1_FALLING_EDGE |
| Tdm1TdmLinkType | UCHAR | TDM1_E1_LINK,<br>TDM1_DS1_LINK |
| Tdm3TdmReassClkPol | UCHAR | TDM3_FALLING_EDGE,<br>TDM3_RISING_EDGE |
| PllenPllEnable | UCHAR | ENABLED,<br>DISABLED |

### 4.12 Structure s_tdm_port_udt_specific_params

| Name | Description | Range of Values |
|---|---|---|
| Tdm1TdmLosClk | UCHAR | TDM1_NORMAL_OPERATION,<br>TDM1_SWITCH_TO_INTERNAL_CLOCK |
| Tdm1IntLosEnable | UCHAR | DISABLED,<br>ENABLED |
| Tdm3TdmReassLos | UCHAR | TDM3_LOS_SIGNAL_ZEROS,<br>TDM3_LOS_SIGNAL_ONES |

| Name | Description | Range of Values |
|---|---|---|
| Tdm4UdtLosSe | UCHAR | DISABLED,<br>ENABLED |
| Tdm4UdtTdmOutBufErrorSe | UCHAR | DISABLED,<br>ENABLED |

### 4.13 Structure s_tdm_port_sdt_specific_params

| Name | Description | Range of Values |
|---|---|---|
| ExtMemTxCBChunk | ULONG | CHUNK0 .. CHUNK31 |
| ExtMemRxCBChunk | ULONG | CHUNK0 .. CHUNK31 |
| Tdm1TdmClkRate | UCHAR | TDM1_1544_KHZ,<br>TDM1_2048_KHZ |
| Tdm1CasLocation | UCHAR | TDM1_LS_FOUR_BITS,<br>TDM1_MS_FOUR_BITS |
| Tdm1MappingSch | UCHAR | TDM1_FIRST_24_CHANNELS,<br>TDM1_3_CHANNELS_OUT_OF_4 |
| Tdm1TdmPulseSel | UCHAR | TDM1_FRAME_PULSE,<br>TDM1_MULTIFRAME_PULSE |
| Tdm1TdmPulsePol | UCHAR | TDM1_NEGATIVE_POLARITY,<br>TDM1_POSITIVE_POLARITY |
| Tdm1TdmClkMode | UCHAR | TDM1_INDEPENDENT_MODE,<br>TDM1_BACKPLANE_MODE |
| Tdm1TdmBusMode | UCHAR | TDM1_GENERIC,<br>TDM1_ST_BUS |
| Tdm1TdmDataFormat | UCHAR | TDM1_SDT_MODE,<br>TDM1_SDT_MODE_NGREATER46 |
| Tdm3ReplayNSilence | UCHAR | TDM3_SILENCE,<br>TDM3_REPLAY |
| Tdm4SdtTdmOutBufErrorSe | UCHAR | DISABLE,<br>ENABLE |
| Tdm4SdtPermUnderSe | UCHAR | DISABLE,<br>ENABLE |
| Tdm4SdtSimpleUnderSe | UCHAR | DISABLE,<br>ENABLE |

### 4.14 Structure s_tdm_port_udt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| TdmPortCommonParams | s_tdm_port_common_params | n/a |
| TdmPortUdtSpecificParams | s_tdm_port_udt_specific_params | n/a |

### 4.15 Structure s_tdm_port_sdt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| TdmPortCommonParams | s_tdm_port_common_params | n/a |
| TdmPortSdtSpecificParams | s_tdm_port_sdt_specific_params | n/a |

### 4.16 Structure s_tdm_loopback_params

| Field Name | Definition | Range of Values |
|---|---|---|
| ExtMemTxCBChunk | ULONG | CHUNK0 .. CHUNK31 |
| Tdm1TdmLinkType | UCHAR | TDM1_E1_LINK,<br>TDM1_DS1_LINK |
| Tdm1TdmClkRate | UCHAR | TDM1_1544_KHZ,<br>TDM1_2048_KHZ |
| Tdm1MappingSch | UCHAR | TDM1_FIRST_24_CHANNELS,<br>TDM1_3_CHANNELS_OUT_OF_4 |
| Tdm1TdmClkPol | UCHAR | TDM1_RISING_EDGE,<br>TDM1_FALLING_EDGE |
| Tdm1TdmPulsePol | UCHAR | TDM1_NEGATIVE_POLARITY,<br>TDM1_POSITIVE_POLARITY |
| Tdm3TdmReassClkPol | UCHAR | TDM3_FALLING_EDGE,<br>TDM3_RISING_EDGE |

### 4.17 Structure s_atm_channel

| Field Name | Definition | Range of Values |
|---|---|---|
| Port | UCHAR | 0..MAX_PORTS |
| Timeslot | UCHAR | 0..E1_TIMESLOTS |

### 4.18 Structure s_atm_channels

| Field Name | Definition | Range of Values |
|---|---|---|
| NumberOfChannels | UCHAR | 0.. MAX_SDT_CHANNELS |
| AtmChannel | array [MAX_SDT_CHANNELS] of s_atm_channel | n/a |

### 4.19 Structure s_atm_cell_header_params

| Field Name | Definition | Range of Values |
|---|---|---|
| Vc | s_vc | n/a |
| Gfc | UCHAR | 0..Fh |
| Pti | UCHAR | 0..7h |
| Clp | UCHAR | 0..1h |
| Hec | UCHAR | 0..FFh |
| Udf2 | UCHAR | 0..FFh |

### 4.20 Structure s_tx_vc_udt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| CellHeader | s_atm_cell_header_params | n/a |
| Srts | UCHAR | DISABLED, ENABLED |

### 4.21 Structure s_tx_vc_sdt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| CellHeader | s_atm_cell_header_params | n/a |
| Channels | s_atm_channels | n/a |
| Cas | UCHAR | DISABLED, ENABLED |
| ReadPointer | UCHAR | 0..3Fh |
| Srts | UCHAR | DISABLED, ENABLED |
| WaitForMultiframe | UCHAR | DISABLED, ENABLED |

### 4.22 Structure s_rx_vc_udt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| Vc | s_vc | n/a |
| UvpOamSel | UCHAR | DISABLED,<br>ENABLED |
| Adaptive | UCHAR | DISABLED,<br>ENABLED |
| Srts | UCHAR | DISABLED,<br>ENABLED |
| CellDelayVariation | UCHAR | 0..49 increments of 100 microseconds |
| TcrLateCellSe | UCHAR | DISABLED,<br>ENABLED |
| TcrCutVcSe | UCHAR | DISABLED,<br>ENABLED |
| FirstCellHandling | UCHAR | DISABLED,<br>ENABLED |

### 4.23 Structure s_rx_vc_sdt_params

| Field Name | Definition | Range of Values |
|---|---|---|
| Vc | s_vc | n/a |
| Channels | s_atm_channels | n/a |
| LutOamSel | UCHAR | DISABLED,<br>ENABLED |
| CellDelayVariation | UCHAR | 0..510 increments of 125 microseconds |
| Adaptive | UCHAR | DISABLED,<br>ENABLED |
| CircularBufferSize | UCHAR | RXSAR_BS_64,<br>RXSAR_BS_128,<br>RXSAR_BS_256,<br>RXSAR_BS_512,<br>RXSAR_BS_1024 |
| Cas | UCHAR | DISABLED,<br>ENABLED |
| CasSwControlled | UCHAR | FALSE,<br>TRUE |
| CasChangeServiceRequest | UCHAR | FALSE,<br>TRUE |

| Field Name | Definition | Range of Values |
|---|---|---|
| CasInitialValue | UCHAR [MAX_SDT_CHANNELS] | 0..Fh |
| SimpleUnderrun | UCHAR | DISABLED, ENABLED |
| Idle | UCHAR | DISABLED, ENABLED |
| PermanentUnderrun | UCHAR | DISABLED, ENABLED |
| Srts | UCHAR | DISABLED, ENABLED |

### 4.24 Structure s_memory_chunk

This structure is used and filled in by the memory manager to keep track of device internal and external memory. Arrays a_internal_memory_chunks and a_external_memory_chunks are derived from the s_memory_chunk structure, which is user defined for the memory manager.

| Field Name | Definition | Description |
|---|---|---|
| StartAddress | ULONG | start address of 32K word memory chunk |
| EndAddress | ULONG | end address of 32K word memory chunk |
| Allocated | BOOLEAN | whether entire memory chunk is allocated or not |
| StraddlingChunks | BOOLEAN | whether current memory chunk allocation straddles next memory chunk or not |
| Head | s_memory_buffer * | head linked list node of 32K word memory chunk |

### 4.25 Structure a_internal_memory_chunks

This structure is used and filled in by the memory manager to keep track of device internal memory. Array a_internal_memory_chunks is derived from the s_memory_chunk structure. **IMPORTANT**: the array InternalMemoryChunkArray[MAX_MT90528_DEVICES] is application defined for the memory manager.

| Field Name | Definition | Description |
|---|---|---|
| InternalMemoryChunkArray[MAX_MT90528_DEVICES] | a_internal_memory_chunks | array defined by the application for use by the API memory manager |

### 4.26 Structure a_external_memory_chunks

This structure is used and filled in by the memory manager to keep track of device external memory. Array a_external_memory_chunks is derived from the s_memory_chunk structure. **IMPORTANT**: the array ExternalMemoryChunkArray[MAX_MT90528_DEVICES] is application defined for the memory manager.

| Recommended Array Name | Definition | Description |
|---|---|---|
| ExternalMemoryChunkArray[MAX_MT90528_DEVICES] | a_external_memory_chunks | array defined by the application for use by the API memory manager |

## 5.0 Statistics, Status and Threshold Structures

The structures listed in this section also contain hidden fields internal to the API. If the field is not listed in this section then it should not be used by the application.

### 5.1 Structure s_mt90528_device_stats_entry

This structure is used by s_mt90528_statistics to keep track of device specific statistics and counters. Any other fields not listed in this table are internal to the operation of the API.

| Field Name | Definition | Description |
|---|---|---|
| Device | UINT | device id of an MT90528 device |
| DeviceRev | USHORT | device revision DEVICE_REVn |
| OpenedCbrTxVcs | USHORT | total number of TX CBR VC's currently opened |
| OpenedCbrRxVcs | USHORT | total number of RX CBR VC's currently opened |
| OpenedUdtTxVcs | USHORT | total number of TX UDT VC's currently opened |
| OpenedUdtRxVcs | USHORT | total number of RX UDT VC's currently opened |
| OpenedSdtTxVcs | USHORT | total number of TX SDT VC's currently opened |
| OpenedSdtRxVcs | USHORT | total number of RX SDT VC's currently opened |
| OpenedDataRxVcs | USHORT | total number of RX DATA VC's currently opened |
| OpenedT1CasTxVcs | USHORT | total number of TX T1 CAS VC's currently opened |
| OpenedE1CasTxVcs | USHORT | total number of TX E1 CAS VC's currently opened |
| OpenedT1CasRxVcs | USHORT | total number of RX T1 CAS VC's currently opened |
| OpenedE1CasRxVcs | USHORT | total number of RX E1 CAS VC's currently opened |
| OpenedCbrTxChannels | ULONG | total number of TX CBR channels currently opened |
| OpenedCbrRxChannels | ULONG | total number of RX CBR channels currently opened |
| ConfiguredTdmPorts | UCHAR | total number of TDM ports currently configured |
| ResetStatistics | UCHAR | reset the interrupt device specific counters (FALSE or TRUE) |
| DataTxsarTxCellBufferEmpty | USHORT | total number of IRQ Data TX Cell buffer empties |

| Field Name | Definition | Description |
|---|---|---|
| DataRxsarBufferOverrunErrors | USHORT | total number of IRQ Data RX buffer overruns |
| DataRxsarBufferHalfFull | USHORT | total number of IRQ Data RX buffer half full |
| DataRxsarCellArrival | USHORT | total number of IRQ Data RX cell arrivals |
| DataRxsarCells | ULONG | total number of IRQ Data RX cells |
| UdtRxsarCells | ULONG | total number of IRQ Udt RX cells |
| SdtRxsarCells | ULONG | total number of IRQ Sdt RX cells |
| UtopiaCells | ULONG | total number of IRQ Utopia cells |
| UtopiaParity | ULONG | total number of IRQ Utopia parity errors |
| XmemLowByteParityErrors | USHORT | total number of IRQ external memory low byte parity errors |
| XmemHighByteParityErrors | USHORT | total number of IRQ external memory high byte parity errors |

## 5.2 Structure s_mt90528_tdm_port_stats_entry

This structure is used by s_mt90528_statistics to keep track of TDM port specific statistics and counters. Any other fields not listed in this table are internal to the operation of the API.

| Field Name | Definition | Description |
|---|---|---|
| ResetStatistics | UCHAR | reset the interrupt TDM port specific counters (FALSE or TRUE) |
| TimeoutCutVcs | USHORT | total number of IRQ reassembly side timeout cut VCs |
| TimeoutLateCells | USHORT | total number of IRQ reassembly side timeout late cells - UDT only |
| TdmLossOfSignals | USHORT | total number of IRQ TDM loss of signals - UDT only |
| TdmUdtOutputBufferErrors | USHORT | total number of IRQ TDM UDT output buffer errors |
| TdmSdtOutputBufferErrors | USHORT | total number of IRQ TDM SDT output buffer errors |
| TdmSimpleUnderruns | USHORT | total number of IRQ TDM simple underruns - SDT only |
| TdmPermanentUnderruns | USHORT | total number of IRQ TDM permanent underruns - SDT only |
| TdmSimpleUnderrunReport | UCHAR | latest IRQ TDM simple underrun report – SDT only |
| TdmPermanentUnderrunReport | ULONG | latest IRQ TDM permanent underrun report - SDT only |

### 5.3 Structure s_mt90528_cbr_vc_stats_entry

This structure is used by s_mt90528_statistics to keep track of CBR VC specific statistics and counters. Any other fields not listed in this table are internal to the operation of the API.

| Field Name | Definition | Description |
|---|---|---|
| Vc | s_vc | Vpi and Vci values |
| OpenedTxChannels | UCHAR | number of opened TX channels for this VC |
| OpenedRxChannels | UCHAR | number of opened RX channels for this VC |
| AtmfCesCas | INTEGER | CONF_CAS_BASIC (1) non-CAS or<br>CONF_CAS_E1CAS (2) e1 CAS or<br>CONF_CAS_DS1ESFCAS (4) ds1 CAS extended superframe |
| AtmfCesCbrService | INTEGER | CONF_SERVICE_UNSTRUCT (1) udt or<br>CONF_SERVICE_STRUCT (2) sdt |
| AtmfCesCellLossIntegrationPeriod | INTEGER | cell loss integration period in milliseconds - default is CONF_DEFAULT_CELL_LOSS_PERIOD (2500) |
| AtmfCesCellLossStatus | INTEGER | indicates if there was continuous cell loss for the cell loss integration period - STATS_CELL_LOSS (1) or STATS_CELL_NOLOSS (2) |
| CollectStatistics | UCHAR | ENABLED (1) or DISABLED (0) statistics collection in this structure for this VC. The default is DISABLED (0). |
| ResetStatistics | UCHAR | reset the interrupt CBR VC specific counters (FALSE or TRUE) |
| ThresAal1HdrErrors | ULONG | threshold setting for aal1 header errors |
| ThresAal1SeqErrors | ULONG | threshold setting for aal1 sequence errors |
| ThresLostCells | ULONG | threshold setting for lost cells |
| ThresMisinsertedCells | ULONG | threshold setting for misinserted cells |
| ThresBufUnderflows | ULONG | threshold setting for buffer underruns |
| ThresBufOverflows | ULONG | threshold setting for buffer overflows |
| ThresPointerReframes | ULONG | threshold setting for pointer reframes |
| ThresPointerParityErrors | ULONG | threshold setting for pointer parity errors |
| AtmfCesReassCells | ULONG | total number of reassembled cells for this CBR VC |
| AtmfCesAal1HdrErrors | ULONG | total number of aal1 header errors for this CBR VC |
| AtmfCesAal1SeqErrors | ULONG | total number of aal1 sequence errors for this CBR VC |
| AtmfCesLostCells | ULONG | total number of lost cells for this CBR VC |
| AtmfCesMisinsertedCells | ULONG | total number of misinserted cells for this CBR VC |

| Field Name | Definition | Description |
|---|---|---|
| AtmfCesPointerReframes | ULONG | total number of pointer reframes for this CBR VC - SDT only |
| AtmfCesPointerParityErrors | ULONG | total number of pointer parity errors for this CBR VC - SDT only |
| AtmfCesBufUnderflows | ULONG | total number of buffer underruns for this CBR VC |
| AtmfCesBufOverflows | ULONG | total number of buffer overflows for this CBR VC |
| AtmfCesLateCells | ULONG | total number of late cells for this CBR VC - UDT only |
| PointerOutOfRangeErrors | USHORT | total number of pointer out of range errors for this CBR VC - SDT only |
| CasChanged | USHORT | total number of Cas changes for this CBR VC - SDT only |

### 5.4 Structure s_mt90528_statistics

This structure is used or updated by the ATM, ISR and MIB functions. Statistics are retrieved by the application for MIB and performance management.

| Field Name | Definition | Description |
|---|---|---|
| Mt90528DeviceStatisticsTable | s_mt90528_device_stats_entry | device specific statistics |
| Mt90528TdmPortStatisticsTable[MAX_PORTS] | s_mt90528_tdm_port_stats_entry | table of TDM port specific statistics |
| Mt90528CbrVcStatisticsTable[MAX_VC] | s_mt90528_cbr_vc_stats_entry | table of CBR VC specific statistics |

**IMPORTANT**: the array Statistics[MAX_MT90528_DEVICES] is application defined for MIB statistics.

| Recommended Array Name | Definition | Description |
|---|---|---|
| Statistics[MAX_MT90528_DEVICES] | s_mt90528_statistics | array defined by the application for use by the API for statistics |

### 5.5 Structure s_stats_flags

This structure is used by s_hw_load and s_stats_params structures.

| Field Name | Definition | Description |
|---|---|---|
| ReassCell | UCHAR | reassembled cells counter is to be loaded (FALSE or TRUE) |
| Aal1HdrErrors | UCHAR | aal1 header errors threshold or counter is to be loaded (FALSE or TRUE) |
| Aal1SeqErrors | UCHAR | aal1 sequence errors threshold or counter is to be loaded (FALSE or TRUE) |
| LostCells | UCHAR | lost cells threshold or counter is to be loaded (FALSE or TRUE) |

| Field Name | Definition | Description |
|---|---|---|
| MisinsertedCells | UCHAR | misinserted cells threshold or counter is to be loaded (FALSE or TRUE) |
| PointerReframes | UCHAR | pointer reframes counter is to be loaded (FALSE or TRUE) |
| PointerParityErrors | UCHAR | pointer parity errors counter is to be loaded (FALSE or TRUE) |
| BufUnderflows | UCHAR | buffer underruns threshold or counter is to be loaded (FALSE or TRUE) |
| BufOverflows | UCHAR | buffer overflows threshold or counter is to be loaded (FALSE or TRUE) |
| LateCells | UCHAR | late cells counter is to be loaded (FALSE or TRUE) |

### 5.6 Structure s_hw_udt

This structure is used by s_hw_dev structure.

| Field Name | Definition | Description |
|---|---|---|
| ReassCell | USHORT | MIB device counter value for reassembled cells |
| Aal1HdrErrors | UCHAR | MIB device counter value for aal1 header errors |
| Aal1SeqErrors | UCHAR | MIB device counter value for aal1 sequence errors |
| LostCells | UCHAR | MIB device counter value for lost cells |
| MisinsertedCells | UCHAR | MIB device counter value for misinserted cells |
| BufUnderflows | UCHAR | MIB device counter value for buffer underruns |
| BufOverflows | UCHAR | MIB device counter value for buffer overflows |
| LateCells | UCHAR | MIB device counter value for late cells |

### 5.7 Structure s_hw_sdt

This structure is used by s_hw_dev structure.

| Field Name | Definition | Description |
|---|---|---|
| ReassCell | USHORT | MIB device counter value for reassembled cells |
| Aal1HdrErrors | UCHAR | MIB device counter value for aal1 header errors |
| Aal1SeqErrors | UCHAR | MIB device counter value for aal1 sequence errors |
| LostCells | UCHAR | MIB device counter value for lost cells |
| MisinsertedCells | UCHAR | MIB device counter value for misinserted cells |
| PointerReframes | UCHAR | MIB device counter value for pointer reframes |
| PointerParityErrors | UCHAR | MIB device counter value for pointer parity errors |
| BufUnderflows | UCHAR | MIB device counter value for buffer underruns |
| BufOverflows | UCHAR | MIB device counter value for buffer overflows |

### 5.8 Structure s_hw_dev

This union structure is used by s_hw_load structure.

| Field Name | Definition | Description |
|---|---|---|
| Udt | s_hw_udt | intermediate field pointing to s_hw_udt |
| Sdt | s_hw_sdt | intermediate field pointing to s_hw_sdt |

### 5.9 Structure s_hw_load

This structure is used to read or load the MIB device counters for a specified VC.

| Field Name | Definition | Description |
|---|---|---|
| Status | s_stats_flags | flags to indicate which MIB device counters should be loaded |
| Count | s_hw_dev | counters to read from or load into MIB device counters for a specified VC |

### 5.10 Structure s_stats_32

This structure is used by s_stats_params structure.

| Field Name | Definition | Description |
|---|---|---|
| ReassCell | ULONG | count of reassembled cells |
| Aal1HdrErrors | ULONG | count of aal1 header errors |
| Aal1SeqErrors | ULONG | count of aal1 sequence errors |
| LostCells | ULONG | count of lost cells |
| MisinsertedCells | ULONG | count of misinserted cells |
| PointerReframes | ULONG | count of pointer reframes |
| PointerParityErrors | ULONG | count of pointer parity errors |
| BufUnderflows | ULONG | count of buffer underruns |
| BufOverflows | ULONG | count of buffer overflows |
| LateCells | ULONG | count of late cells |

### 5.11 Structure s_stats_params

This structure is used to load the 32 bit MIB counters for a specified VC.

| Field Name | Definition | Description |
|---|---|---|
| Status | s_stats_flags | flags to indicate which 32 bit counters should be loaded/set |
| Value | s_stats_32 | counter values to load into the 32 bit counters |

### 5.12 Structure s_mt90528_device_status_entry

This structure is used by s_mt90528_status to keep track of device specific interrupts.

| Field Name | Definition | Description |
|---|---|---|
| IrqSet | UCHAR | flag to indicate at least one of the device specific interrupts occurred (FALSE or TRUE) |
| IrqDataTxsarTxCellBufferEmpty | UCHAR | flag to indicate a data txsar tx cell buffer empty interrupt occurred (FALSE or TRUE) |
| IrqDataRxsarBufferOverrunError | UCHAR | flag to indicate a data rxsar buffer overrun error interrupt occurred (FALSE or TRUE) |
| IrqDataRxsarBufferHalfFull | UCHAR | flag to indicate a data rxsar buffer half full interrupt occurred (FALSE or TRUE) |
| IrqDataRxsarCellArrival | UCHAR | flag to indicate a data rxsar cell arrival interrupt occurred (FALSE or TRUE) |
| IrqDataRxsarCell | UCHAR | flag to indicate a data rxsar cell counter rollover interrupt occurred (FALSE or TRUE) |
| IrqUdtRxsarCell | UCHAR | flag to indicate a udt rxsar cell counter rollover interrupt occurred (FALSE or TRUE) |
| IrqSdtRxsarCell | UCHAR | flag to indicate a sdt rxsar cell rollover counter interrupt occurred (FALSE or TRUE) |
| IrqUtopiaCell | UCHAR | flag to indicate a Utopia cell counter rollover interrupt occurred (FALSE or TRUE) |
| IrqUtopiaParity | UCHAR | flag to indicate a Utopia parity error rollover interrupt occurred (FALSE or TRUE) |
| IrqXmemLowByteParityError | UCHAR | flag to indicate an external memory low byte parity error interrupt occurred (FALSE or TRUE) |
| IrqXmemHighByteParityError | UCHAR | flag to indicate an external memory high byte parity error interrupt occurred (FALSE or TRUE) |
| IrqCpuHaicAccessComplete | UCHAR | flag to indicate the HAIC access complete interrupt occurred (FALSE or TRUE) |

### 5.13 Structure s_mt90528_tdm_port_status_entry

This structure is used by s_mt90528_status to keep track of TDM port specific interrupts.

| Field Name | Definition | Description |
|---|---|---|
| TdmPort | UCHAR | TDM port id (0 .. 27 or 0 .. 7) |
| IrqTimeoutCutVc | UCHAR | flag to indicate a timeout cut VC interrupt occurred on this TDM port (FALSE or TRUE) |
| IrqTimeoutLateCell | UCHAR | flag to indicate a timeout late cell interrupt occurred on this TDM port (FALSE or TRUE) - UDT only |

| Field Name | Definition | Description |
|---|---|---|
| IrqTdmLossOfSignal | UCHAR | flag to indicate a TDM loss of signal interrupt occurred on this TDM port (FALSE or TRUE) - UDT only |
| IrqTdmOutputBufferError | UCHAR | flag to indicate a TDM output buffer error interrupt occurred on this TDM port (FALSE or TRUE) |
| IrqTdmSimpleUnderrun | UCHAR | flag to indicate a TDM simple underrun interrupt occurred on this TDM port (FALSE or TRUE) - SDT only |
| IrqTdmPermanentUnderrun | UCHAR | flag to indicate a TDM permanent underrun interrupt occurred on this TDM port (FALSE or TRUE) - SDT only |

### 5.14 Structure s_mt90528_cbr_vc_status_entry

This structure is used by s_mt90528_status to keep track of CBR VC specific interrupts, thresholds and wraps (rollovers).

| Field Name | Definition | Description |
|---|---|---|
| Vc | s_vc | Vpi and Vci values |
| IrqReassCell | UCHAR | flag to indicate a reassembled cell interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqHdrError | UCHAR | flag to indicate an aal1 header error interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqAal1SeqError | UCHAR | flag to indicate an aal1 sequence error interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqLostCell | UCHAR | flag to indicate a lost cell interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqMisinsertedCell | UCHAR | flag to indicate a misinserted cell interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqPointerReframe | UCHAR | flag to indicate a pointer reframe interrupt occurred on this CBR VC (FALSE or TRUE) - SDT only |
| IrqPointerParityError | UCHAR | flag to indicate a pointer parity error interrupt occurred on this CBR VC (FALSE or TRUE) - SDT only |
| IrqBufUnderflow | UCHAR | flag to indicate a buffer underrun interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqBufOverflow | UCHAR | flag to indicate a buffer overrun interrupt occurred on this CBR VC (FALSE or TRUE) |
| IrqLateCell | UCHAR | flag to indicate a late cell interrupt occurred on this CBR VC (FALSE or TRUE) - UDT only |
| IrqPointerOutOfRangeError | UCHAR | flag to indicate a pointer out of range interrupt occurred on this CBR VC (FALSE or TRUE) - SDT only |
| IrqCasChanged | UCHAR | flag to indicate a CAS changed interrupt occurred on this CBR VC (FALSE or TRUE) - SDT only |

| Field Name | Definition | Description |
|---|---|---|
| WrapCounter32 | UCHAR | flag to indicate at least one 32 bit counter is about to wrap on this CBR VC (FALSE or TRUE) |
| WrapReassCell | UCHAR | flag to indicate a reassembled cell 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapHdrError | UCHAR | flag to indicate an aal1 header error 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapAal1SeqError | UCHAR | flag to indicate an aal1 sequence error 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapLostCell | UCHAR | flag to indicate a lost cell 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapMisinsertedCell | UCHAR | flag to indicate a misinserted cell 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapPointerReframe | UCHAR | flag to indicate a pointer reframe 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) - SDT only |
| WrapPointerParityError | UCHAR | flag to indicate a pointer parity error 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) - SDT only |
| WrapBufUnderflow | UCHAR | flag to indicate a buffer underrun 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapBufOverflow | UCHAR | flag to indicate a buffer overflow 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) |
| WrapLateCell | UCHAR | flag to indicate a late cell 32 bit counter wrap is about to occur within the next 2 interrupts on this CBR VC (FALSE or TRUE) - UDT only |
| ThresAlert | UCHAR | flag to indicate at least one 32 bit counter has reached a threshold crossover (FALSE or TRUE) |
| TAlertAal1HdrErrors | UCHAR | flag to indicate an aal1 header error 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) |
| TAlertAal1SeqErrors | UCHAR | flag to indicate an aal1 sequence error 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) |
| TAlertLostCells | UCHAR | flag to indicate a lost cell 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) |
| TAlertMisinsertedCells | UCHAR | flag to indicate a misinserted cell 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) |

| Field Name | Definition | Description |
|---|---|---|
| TAlertBufUnderflows | UCHAR | flag to indicate a buffer underrun 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) |
| TAlertBufOverflows | UCHAR | flag to indicate a buffer overflow 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) |
| TAlertPointerReframes | UCHAR | flag to indicate a pointer reframe 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) - SDT only |
| TAlertPointerParityErrors | UCHAR | flag to indicate a pointer parity error 32 bit counter threshold crossing has occured on this CBR VC (FALSE or TRUE) - SDT only |
| TAlert2Aal1HdrErrors | UCHAR | flag to indicate an aal1 header error 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) |
| TAlert2Aal1SeqErrors | UCHAR | flag to indicate an aal1 sequence error 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) |
| TAlert2LostCells | UCHAR | flag to indicate a lost cell 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) |
| TAlert2MisinsertedCells | UCHAR | flag to indicate a misinserted cell 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) |
| TAlert2BufUnderflows | UCHAR | flag to indicate a buffer underrun 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) |
| TAlert2BufOverflows | UCHAR | flag to indicate a buffer overflow 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) |
| TAlert2PointerReframes | UCHAR | flag to indicate a pointer reframe 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) - SDT only |
| TAlert2PointerParityErrors | UCHAR | flag to indicate a pointer parity error 32 bit counter 2nd set (i.e. 24-hours) threshold crossing has occurred on this CBR VC (FALSE or TRUE) - SDT only |

### 5.15 Structure s_mt90528_status

This structure is updated by the ISR handler functions. Status information is used by the application for interrupts, thresholds and wrap (rollover) management. Any other fields not listed in this table are internal to the operation of the API.

| Field Name | Definition | Description |
|---|---|---|
| Mt90528DeviceStatusTable | s_mt90528_device_status_entry | device specific status of interrupts |
| Mt90528TdmPortStatusTable[MAX_PORTS] | s_mt90528_tdm_port_status_entry | table of TDM port specific status of interrupts |
| Mt90528CbrVcStatusTable[MAX_VC] | s_mt90528_cbr_vc_status_entry | table of CBR VC specific status of interrupts, thresholds and wraps |

**IMPORTANT**: the array Status[MAX_MT90528_DEVICES] is application defined for IRQ and Threshold reporting.

| Recommended Array Name | Definition | Description |
|---|---|---|
| Status[MAX_MT90528_DEVICES] | s_mt90528_status | array defined by the application for use by the ISR handler and application |

### 5.16 Structure s_mt90528_cbr_vc_cell_loss_status_entry

This structure is used by s_mt90528_cbr_vc_cell_loss_status to keep track of CBR VC specific cell loss.

| Field Name | Definition | Description |
|---|---|---|
| StatisticsCbrVcIndex | USHORT | VC index pointing to a CBR VC entry in the CBR VC statistics table |

### 5.17 Structure s_mt90528_cbr_vc_cell_loss_status

This structure is updated by the scan for cell loss on VCs function. Status information is used by the application for CBR VC cell loss management.

| Field Name | Definition | Description |
|---|---|---|
| CellLossSet | UCHAR | Indicates that at least one CBR VC had continuous cell loss exceeding the cell loss integration period - TRUE(1) or FALSE(0) |
| Mt90528CbrVcCellLossStatusTable[MAX_VC] | s_mt90528_cbr_vc_cell_loss_status_entry | table of CBR VC indexes pointing to CBR VCs in the CBR VC statistics table |

**IMPORTANT**: the array CellLossStatus[MAX_MT90528_DEVICES] is application defined for CBR VC cell loss timeout reporting.

| Recommended Array Name | Definition | Description |
|---|---|---|
| CellLossStatus[MAX_MT90528_DEVICES] | s_mt90528_cbr_vc_cell_loss_status | array defined by the application for use by the Mt90528ScanForCellLossTimeoutOnVcs function and the application |

### 5.18 Structure s_mt90528_cbr_vc_statistic_threshold_entry

This structure is used by s_mt90528_threshold structure to keep track of a second set of CBR VC thresholds and 32 bit counters (i.e. 24-hour period).

| Field Name | Definition | Description |
|---|---|---|
| Vc | s_vc | Vpi and Vci values |
| CollectStatistics | UCHAR | ENABLED (1) or DISABLED (0) statistics collection in this structure for this VC. The default is DISABLED (0). |
| ResetStatistics | UCHAR | reset the secondary CBR VC specific counters |
| ThresAal1HdrErrors | ULONG | second threshold setting for aal1 header errors (i.e. 24-hour period) |
| ThresAal1SeqErrors | ULONG | second threshold setting for aal1 sequence errors (i.e. 24-hour period) |
| ThresLostCells | ULONG | second threshold setting for lost cells (i.e. 24-hour period) |
| ThresMisinsertedCells | ULONG | second threshold setting for misinserted cells (i.e. 24-hour period) |
| ThresBufUnderflows | ULONG | second threshold setting for buffer underruns (i.e. 24-hour period) |
| ThresBufOverflows | ULONG | second threshold setting for buffer overflows (i.e. 24-hour period) |
| ThresPointerReframes | ULONG | second threshold setting for pointer reframes (i.e. 24-hour period) |
| ThresPointerParityErrors | ULONG | second threshold setting for pointer parity errors (i.e. 24-hour period) |
| ReassCells | ULONG | second counter for reassembled cells (i.e. 24-hour period) |
| Aal1HdrErrors | ULONG | second counter for aal1 header errors (i.e. 24-hour period) |
| Aal1SeqErrors | ULONG | second counter for aal1 sequence errors (i.e. 24-hour period) |
| LostCells | ULONG | second counter for lost cells (i.e. 24-hour period) |
| MisinsertedCells | ULONG | second counter for misinserted cells (i.e. 24-hour period) |
| PointerReframes | ULONG | second counter for pointer reframes (i.e. 24-hour period) - SDT only |

| Field Name | Definition | Description |
|---|---|---|
| PointerParityErrors | ULONG | second counter for lost cells (i.e. 24-hour period) - SDT only |
| BufUnderflows | ULONG | second counter for buffer underruns (i.e. 24-hour period) |
| BufOverflows | ULONG | second counter for buffer overflows (i.e. 24-hour period) |
| LateCells | ULONG | second counter for late cells (i.e. 24-hour period) - UDT only |

### 5.19 Structure s_mt90528_threshold

This structure is used by the application to keep track of a second set of CBR VC thresholds and 32 bit counters (i.e. 24-hour period).

| Field Name | Definition | Description |
|---|---|---|
| Mt90528CbrVcThresholdTable[MAX_VC] | s_mt90528_cbr_vc_statistic_threshold_entry | table of CBR VC entries to track a second set of thresholds and 32 bit counters (i.e. 24-hour period) |

**IMPORTANT**: the array AppStats[MAX_MT90528_DEVICES] is application defined for APP statistics.

| Recommended Array Name | Definition | Description |
|---|---|---|
| AppStats[MAX_MT90528_DEVICES] | s_mt90528_threshold | array defined by the application for use by the API for statistics |

## 6.0 Glossary

**AAL -** ATM Adaptation Layer; standardized protocols used to translate higher layer services from multiple applications into the size and format of an ATM cell.
**AAL1 -** ATM Adaptation Layer 1 used for the transport of constant bit rate, time-dependent traffic (e.g., voice, video); requires transfer of timing information between source and destination; maximum of 47-bytes of user data permitted in payload as an additional header byte is required to provide sequencing information.
**Asynchronous -** 1. Not synchronous; not periodic. 2. The temporal property of being sourced from independent timing references. Asynchronous signals have different frequencies, and no fixed phase relationship. 3. In telecom, data which is not synchronized to the public network clock. 4. The condition or state when an entity is unable to determine, prior to its occurrence, exactly when an event transpires.
**ATM -** Asynchronous Transfer Mode; a method in which information to be transferred is organized into fixed-length cells; asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic. (While ATM cells are transmitted synchronously to maintain clock between sender and receiver, the sender transmits data cells when it has something to send and transmits empty cells when idle, and is not limited to transmitting data every Nth cell.)
**Cell -** Fixed-size information package consisting of 53 bytes (octets) of data; of these, 5 bytes represent the cell header and 48 bytes carry the user payload and required overhead. **Note:** If a 16-bit UTOPIA interface is used, each cell is 54 bytes long, composed of 6 bytes of cell header and 48 bytes carrying user payload and required overhead.
**CBR -** Constant Bit Rate; an ATM service category supporting a constant or guaranteed rate, with timing control and strict performance parameters. Used for services such as voice, video, or circuit emulation.
**CDV -** Cell Delay Variation; a QoS parameter that measures the peak-to-peak cell delay through the network; results from buffering and cell scheduling.
**CES -** Circuit Emulation Service; ATM Forum service providing a virtual circuit which emulates the characteristics of a constant bit rate, dedicated-bandwidth circuit (e.g., DS1).
**CLP -** Cell Loss Priority; a 1-bit field in the ATM cell header that corresponds to the loss priority of a cell; cells with CLP = 1 can be discarded in a congestion situation.
**CSI -** Convergence Sublayer Indication bit in the AAL1 header byte; when present in an even-numbered cell using SDT, indicates the presence of a pointer byte; used to transport RTS values in odd-numbered cells using SRTS for clock recovery.
**DBCES -** Dynamic Bandwidth Circuit Emulation Service (also referred to as Dynamic Bandwidth Utilization); ATM Forum service providing a virtual circuit which follows the CES specification for the transport of only the active time slots of a constant bit rate, dedicated-bandwidth circuit.
**E1 carrier -** Similar to the North American T1, E1 is the European format for digital transmission.
E1 carries signals at 2.048 Mbps (32 channels at 64kbps), versus the T1, which carries signals at 1.544 Mbps (24 channels at 64kbps). E1 and T1 lines may be interconnected for international use.
**GFC -** Generic Flow Control; 4-bit field in the ATM header used for local functions (not carried end-to-end); when cells are configured for NNI formatting, this value forms the four MSBs of the VPI field.
**HEC -** Header Error Control; using the fifth octet in the ATM cell header, ATM equipment (usually the PHY) may check for an error and correct the contents of the header; CRC algorithm allows for single-error correction and multiple-error detection.
**MIB** – Management Information Base. A collection of data presented in objects to the network via SNMP about entities in the network.
**NNI -** ATM Network Node Interface.
**OAM -** Operations, Administration and Maintenance; MSB within the PTI field of the ATM cell header which indicates if the ATM cell carries management information such as fault indications.
**PHY -** Physical Layer; bottom layer of the ATM Reference Model; provides ATM cell transmission over the physical interfaces that interconnect the various ATM devices.

**PTI -** Payload Type Identifier; 3-bit field in the ATM cell header - MSB indicates if the cell contains OAM information or user data.
**RTS -** Residual Time Stamp; see SRTS.
**SAR -** Segmentation and Reassembly; method of partitioning, at the source, frames into ATM cells and reassembling, at the destination, these cells back into information frames; lower sublayer of the AAL which inserts data from the information frames into cells and then adds the required header, trailer, and/or padding bytes to create 48-byte payloads to be transmitted to the ATM layer.
**SDT -** Structured Data Transfer; format used within AAL1 for blocks consisting of N * 64 kbps channels; blocks are segmented into cells for transfer and additional overhead bytes (pointers) are used to indicate structure boundaries within cells (therefore aiding data recovery).
**SNMP** – Simple Network Management Protocol. This protocol uses the MIB information (objects).
**SRTS -** Synchronous Residual Time Stamp; method for clock recovery in which difference signals between a source clock and the network reference clock (time stamps) are transmitted to allow reconstruction of the source clock. The destination reconstructs the source clock based on the time stamps and the network reference clock. (Note that the same network reference clock is required at both ends.)
**Synchronous -** 1. The temporal property of being sourced from the same timing reference. Synchronous signals have the same frequency, and a fixed (often implied to be zero) phase offset. 2. A mode of transmission in which the sending and receiving terminal equipment are operating continually at the same rate and are maintained in a desired phase relationship by an appropriate means.
**T1 carrier -** A dedicated phone connection supporting data rates of 1.544Mbps. A T1 line actually consists of 24 individual channels, each of which supports 64kbps. Each 64kbps channel can be configured to carry voice or data traffic. Most telephone companies allow you to buy just some of these individual channels, known as fractional T1 access. T1 lines are sometimes referred to as DS1 lines.
**UDT -** Unstructured Data Transfer; format used within AAL1 for transmission of user data without regard for structure boundaries (e.g., circuit emulation).
**UNI –** ATM User Node Interface.
**UTOPIA -** Universal Test and Operations Physical Interface for ATM; a PHY-level interface to provide connectivity between ATM components.
**VC -** Virtual Channel; one of several logical connections defined within a virtual path (VP) between two ATM devices; provides sequential, unidirectional transport of ATM cells. Also Virtual Circuit.
**VCI -** Virtual Channel Identifier; 16-bit value in the ATM cell header that provides a unique identifier for the virtual channel (VC) within a virtual path (VP) that carries a particular cell.
**VP** - Virtual Path; a unidirectional logical connection between two ATM devices; consists of a set of virtual channels (VC).
**VPI -** Virtual Path Identifier; 8-bit value (in UNI, 12 bits in NNI) in the ATM cell header that indicates the virtual path (VP) to which a cell belongs.

**For more information about all Zarlink products**
**visit our Web Site at**

## www.zarlink.com



TECHNICAL DOCUMENTATION - NOT FOR RESALE