ENT-AN1169 Application Note
Handling Interrupts for VSC8575-11/14, VSC8582-11/14, and VSC8584-11/14 Devices

Microsemi
Power Matters.™

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Contents

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

Revision 1.0 was the first publication of this document.

2 Introduction

Revision B of the affected family of 1G devices described in this document support Serial Management Interface (SMI) interrupts. The SMI interrupts include an MDINT output for signaling the station manager (STA) when certain events occur in the PHY. When a PHY generates a main interrupt status interrupt through the MDINT bit (register 26.15), the MDINT pin asserts low if the interrupt pin enable bit (MII register 25.15) is set.

The PHYs support more functionality than can be addressed by a single 16-bit interrupt status register (26), therefore an extended interrupt status register (29E2) was added to handle more interrupt options. Interrupt masking options can be set to enable or disable extended interrupt status events to be propagated to the main interrupt status register.

The affected devices have an issue with the clearing of the extended interrupt status bit (register 26.5) from the interrupt status register. This failure to correctly clear the interrupt status causes the MDINT bit (register 26.15) to remain set. If the MDINT bit remains set, the appropriate bit(s) in register 29G continue to indicate an interrupt, and the MDINT pin monitored by the station manager remains asserted.

The devices also have an issue with the clearing of the AMS media change interrupt bit (register 26.4) from the interrupt status register. This failure to correctly clear the interrupt status causes the MDINT bit (register 26.15) to remain set, the appropriate bit(s) in register 29G to continue to indicate an interrupt, and the MDINT pin remains asserted.

This document describes the workaround steps provided in the PHY API and provides guidance for user application modifications to ensure accurate status information can be extracted from the extended interrupt status and AMS media change interrupt bits of the affected devices.

2.1 Affected Devices

- VSC8575XKS-11
- VSC8575XKS-14
- VSC8582XKS-11
- VSC8582XKS-14
- VSC8584XKS-11
- VSC8584XKS-14

3 Extended Interrupt Issue

3.1 Interrupt Processing

To enable interrupts, the user application code sets the appropriate interrupt enable masks using the PHY API. This is accomplished by configuring the interrupt mask register (register 25). To enable the SMI MDINT output, register 25.15 must be set. This register also contains a mask for the extended interrupts (register 25.5 – extended interrupt mask). When register 25.15 and register 25.5 are set, any interrupt from the extended interrupt status will be indicated by the MDINT signaling.

Once the MDINT indication is received by the station manager, an Interrupt Service Routine (ISR) can be called to poll the PHY registers and process the interrupts. Normally, the ISR sequence is as follows:

1. Read PHY register 29G (global interrupt status) to determine port and type
2. Read PHY register 26. Process interrupts are indicated by bits set
    - If PHY register 26.5 is set, read the extended interrupt status register (29E2)
    - If PHY register 29E2 has any bits set, process interrupts are indicated by set bits

The PHY API provides API functions to set interrupt masks and to process or handle the interrupt status indications from the hardware. PHY API revision 4.67.01 (and later) contains a workaround and changes to address this issue.

3.2 Setting Interrupts Masks

To enable interrupts, the software sets the appropriate interrupt enable masks by configuring the interrupt mask register (register 25), setting the MDINT (register 25.15), and configuring the extended interrupts mask register (28E2) appropriately. To enable the extended interrupts to propagate to MDINT, the API software automatically sets the extended interrupt mask (register 25.5), and then sets the appropriate interrupts in the extended interrupt mask register (28E2).

3.2.1 Workaround when Setting Interrupts

To work around the extended interrupt issue, the interrupt must never be triggered by the PHY hardware. This means that the interrupt enable mask bits must never be set for this interrupt. If the interrupt is triggered, the only way to clear the interrupt status is through hardware or software resets. Therefore, the API software must not allow the setting of the extended interrupt mask (register 25.5), which is normally set to propagate the extended interrupt status indication to the main interrupt status register. This action prevents the extended interrupt status (register 26.5) from being set in the PHY hardware.

The PHY API Revision 4.67.01 (and later) contains the described workaround in the following function used to configure the PHY interrupt events:

`vtss_phy_event_enable_set()`

3.2.2 Workaround when Processing Interrupts

The workaround forces the extended interrupt enable mask bit (register 25.5) to always be clear, which also means that extended interrupt status bit (register 26.5) will never trigger and it will always be clear. Therefore, there is no way for the device to indicate that an extended interrupt has occurred (see the description of register 29E2 in the device datasheet). The solution to this issue requires frequent polling of register 29E2.

As a result, the user application software must be modified to poll for the events described in register 29E2 at some interval that is acceptable to the user application. The user application may use the existing polling function (`vtss_phy_event_poll`) which is used to poll both the interrupt status register (register 26) and the extended interrupt status register (register 29E2), or the user application may use a new API function (`vtss_phy_ext_event_poll`) to only poll the extended interrupt status register (register 29E2).

A new function (`vtss_phy_is_viper_revB`) is available to determine if the device(s) are revision B and thus require the workaround. If `vtss_phy_is_viper_revB` indicates TRUE, then the user

application is required to poll for the extended interrupt status. Revision A silicon does not have this issue and a workaround is not applicable. PHY API revision 4.67.01 and later contains the workaround described in the following functions used to process the PHY interrupt events:

- `vtss_phy_event_poll()`– Poll register 26 and register 29E2
- `vtss_phy_ext_event_poll()`– Poll register 29E2 only
- `vtss_phy_is_viper_revB()`– Returns a Boolean (T/F) for revision B

4 AMS Media Change Interrupt Issue

4.1 Setting Interrupt Mask Workaround

To work around the AMS media change interrupt bit (register 26.4) issue, the interrupt must never be triggered. If the interrupt is triggered, the only way to clear the interrupt status is through hardware or software resets. Therefore, the API software must not allow the setting of the AMS media change interrupt enable bit in the interrupt mask register (register 25.4).

The `vtss_phy_event_enable_set()`function, available in PHY API revision 4.67.01 and later, contains the described workaround to prevent the setting of register 25.4 for the affected devices.

4.2 Processing Interrupts Workaround

The workaround forces the AMS media change interrupt enable mask bit (register 25.4) to always be clear, which also means that an AMS media change interrupt event will never trigger and it will always be clear. Therefore, there is no way for the hardware to indicate than an AMS media change interrupt has occurred. However, this issue can be resolved using the PHY status.

Systems must include software to check the values in register 28.1:0 or the values in register 20E1.7:6 upon link-up events, if the port is configured for AMS operation. These values indicate the current media status. The current media status must then be compared to the stored media status to determine if something has changed or if the previous status was something other than “No Media.” The current media status mapping is as follows:

- 00 – No Media Type
- 01 – Copper Media Type
- 10 – SerDes Media Type
- 11 – Reserved

When the port is configured for AMS operation, the user application software must use the `vtss_phy_status_get` function to poll for the PHY status immediately following a link event.This function returns the structure `vtss_port_status_t`. The application must then determine if the AMS mode has been configured for this port by checking the media interface. The media interface for the AMS media operating mode is configured in one of the following API functions:

- `VTSS_PHY_MEDIA_IF_AMS_CU_PASSTHRU`
- `VTSS_PHY_MEDIA_IF_AMS_FI_PASSTHRU`
- `VTSS_PHY_MEDIA_IF_AMS_CU_1000BX`
- `VTSS_PHY_MEDIA_IF_AMS_FI_1000BX`
- `VTSS_PHY_MEDIA_IF_AMS_CU_100FX`
- `VTSS_PHY_MEDIA_IF_AMS_FI_100FX`

If one of the listed media interface modes has been selected, then the application can check the value of the element “fiber”, which is a Boolean (T/F), and the element “copper”, which is also a Boolean (T/F) in the `vtss_port_status_t` structure. If the value of the element “fiber” is set to TRUE, then the media type is SerDes media. If the value of the element “copper” is set to TRUE, then the media type is Copper media. If the value of the element “fiber” is FALSE and the value of the element “copper” is FALSE, then the media type is “No Media.”

5 PHY API Functions

This section contains descriptions of the enhanced functions made available in PHY API revision 4.67.01 that are needed to properly operate the affected devices.

User application code must also be modified to poll for the events described in the extended interrupt status register (register 29E2) using either **vtss_phy_event_poll** or **vtss_phy_ext_event_poll**.

```
/**
 * \brief Enabling / Disabling of events
 *
 * \param inst [IN]     Target instance reference.
 * \param port_no [IN]  Port number
 * \param ev_mask [IN]  Mask containing events that are enabled/disabled
 * \param enable [IN]   Enable/disable of event
 *
 * \return Return code.
 **/

vtss_rc vtss_phy_event_enable_set(const vtss_inst_t         inst,
                                  const vtss_port_no_t      port_no,
                                  const vtss_phy_event_t    ev_mask,
                                  const BOOL                enable);



/**
 * \brief Getting current interrupt event state
 *
 * \param inst [IN]     Target instance reference.
 * \param port_no [IN]  Port number
 * \param ev_mask [IN]  Mask containing events that are enabled/disabled
 *
 * \return Return code.
 **/
vtss_rc vtss_phy_event_enable_get(const vtss_inst_t         inst,
                                  const vtss_port_no_t      port_no,
                                  vtss_phy_event_t          *ev_mask);



/**
 * \brief Polling for active events
 *
 * \param inst [IN]     Target instance reference.
 * \param port_no [IN]  Port number
 * \param ev_mask [OUT] Mask containing events that are active
 *
 * \return Return code.
 **/
vtss_rc vtss_phy_event_poll(const vtss_inst_t     inst,
                            const vtss_port_no_t  port_no,
                            vtss_phy_event_t      *const ev_mask);



/**
```

```
 * \brief Polling for active EXT Interrupt events
 *
 * \param inst [IN]     Target instance reference.
 * \param port_no [IN]  Port number
 * \param ev_mask [OUT] Mask containing events that are active
 *
 * \return Return code.
 *
 * NOTE: Viper Rev. B Self-Clearing Interrupt Stuck ON Work-Around
 *       Normally, the API function: vtss_phy_event_poll() handles ALL
Interrupts.
 *       This API is a work-around for Viper family
 *                          (VSC8584/VSC8582/VSC8575/VSC8564/VSC8562/VSC8586)
 *       Viper Rev_B has a Bug which prevents EXT INT (Reg26.5) and
 *                         AMS INT (Reg26.4) from Clearing properly (MDINT
stays asserted),
 *     This results in MDINT Stuck ON if one of these INT's are ever triggered,
 *                        putting the system into a Stuck Interrupt situation
 *       This API can be used to directly Poll for the events in Extended
Interrupt Status Reg. 29E2.
 **/
vtss_rc vtss_phy_ext_event_poll(const vtss_inst_t     inst,
                                const vtss_port_no_t  port_no,
                                vtss_phy_event_t      *const ev_mask);



/**
 * \brief Polling for to determine if the Chip Type and revision is Viper Rev_B
 *
 * \param inst [IN]           Target instance reference.
 * \param port_no [IN]        Port number
 * \param is_viper_revB [OUT] Boolean to indicate that the Chip/Rev is Viper
RevB
 *
 * \return Return code.
 *
 **/
vtss_rc vtss_phy_is_viper_revB(const vtss_inst_t     inst,
                               const vtss_port_no_t  port_no,
                               BOOL                  *is_viper_revB);
```