IPC Notify Drivers and Transports

Contents

Introduction

The IPC product supports different notification drivers and transports to support the higher-level user APIs. These drivers and transport implementations may take advantage of specific hardware and/or trade-off performance for features. This article summarizes the notification drivers and transports available with IPC and their limitations. It also describes how to switch between different notification drivers and transports, and common pitfalls associated with doing so.

> **NOTE**
>
> Throughout this article, there are references to the SysLink product. While the architecture of SysLink was similar to IPC, and the details hold for SysLink-based products, **the HLOS side** of IPC 3.x was completely rewritten, and this article does **not** apply to the HLOS side of IPC.

> **NOTE**
>
> SysLink added the ability to change the notify drivers/transports via products.mak in the SysLink 2.20.01.18 release.

Notify Drivers

The **Notify** module is an Inter-processor communication (IPC) mechanism that allows notification of events from one processor to another. The underlying 'Notify Driver' used to implement this capability can vary based on functionality and performance factors. The SysLink and IPC products provide several Notify driver implementations. All processors involved in the notification process must agree on a single driver and configure it consistently.

Shared Memory Notify Driver

By default, both SysLink and IPC use a shared-memory Notify driver (**NotifyDriverShm**). This shared-memory Notify driver offers room for a single pending notification in shared-memory per event. This is the most flexible as it supports disabling/enabling of notification events on a specified interrupt line. There is also the capability of ensuring that the sender of a notification has someone listening in on an event line.

Configuration

Since this is the default configuration for both SysLink and IPC, nothing special is needed to use this driver.

Circular Buffer Notify Driver

This Notify driver uses a circular buffer (**NotifyDriverCirc**) in shared-memory to store notifications. Unlike **NotifyDriverShm**, this driver stores all notifications in the same circular buffer (whose size is configurable) therefore disabling of single event is **not** supported. Another limitation is that the sender of an event won't know if an event was actually processed, so an event might be dropped if global notifications are disabled by the receiver. Therefore when using this driver, the sender always assumes that the receiving core(s) have initialized and registered for the event and that the event hasn't been globally disabled.

Though this does cause limitations in the usage of the Notify module, its compensated by a reduction in latency.

Configuration

The Notify driver configuration feature in SysLink is controlled by a build options variable (SYSLINK_NOTIFYDRIVER). The variable will only configure the GPP SysLink driver (syslink.ko). In IPC, its controlled via the application's configuration file.

To configure the SysLink-side driver, you can set the variable as follows in the top-level products.mak file and re-build SysLink.

<syntaxhighlight lang='make'> SYSLINK_NOTIFYDRIVER=NOTIFYDRIVERCIRC </syntaxhighlight>

For more option information see the SysLink Install Guide Build Options section.

To configure the IPC/BIOS-side driver (e.g. DSP, M3, etc.), you'll need to add the following lines to the slave's application configuration file (*.cfg):

For OMAP-L138: <syntaxhighlight lang='javascript'> var Notify = xdc.useModule('ti.sdo.ipc.Notify'); Notify.SetupProxy = xdc.useModule('ti.sdo.ipc.family.da830.NotifyCircSetup'); </syntaxhighlight>

For OMAP3: <syntaxhighlight lang='javascript'> var Notify = xdc.useModule('ti.sdo.ipc.Notify'); Notify.SetupProxy = xdc.useModule('ti.sdo.ipc.family.omap3530.NotifyCircSetup'); </syntaxhighlight>

For TI81XX: <syntaxhighlight lang='javascript'> var Notify = xdc.useModule('ti.sdo.ipc.Notify'); Notify.SetupProxy = xdc.useModule('ti.sdo.ipc.family.ti81xx.NotifyCircSetup'); </syntaxhighlight>

MessageQ Transports

The **MessageQ** module is an Inter-processor communication (IPC) mechanism that supports the structured sending and receiving of variable length messages. As with the Notify Driver, an underlying 'Transport' is used to pass messages and its implementation can vary based on functionality and performance factors. The SysLink and IPC products provide various transport implementations. All processors involved in message passing must agree on a single transport and configure it consistently.

Shared Memory Transport

This shared-memory MessageQ transport (**TransportShm**) uses ListMP to temporarily queue messages in shared memory before the messages are moved to the destination queue. This transport is typically the most fully featured but the slowest because of the overhead of queuing messages using a multi-processor-safe linked list.

Configuration

This is the default configuration on both SysLink and IPC, nothing special is needed to use this transport.

Shared Memory Circular Buffer Transport

This shared-memory circular buffer based MessageQ transport (**TransportShmCirc**) uses a fixed-length circular buffer to temporarily queue messages in shared memory before the messages are moved to the destination queue. This transport is typically faster than TransportShm because of the efficiencies gained by using a circular buffer instead of a linked list.

Configuration

The transport configuration feature in SysLink is controlled by a build options variable (SYSLINK_TRANSPORT). The variable will only configure the GPP SysLink driver (syslink.ko). In IPC, its controlled via the application's configuration file.

To configure the GPP core driver, you can set the variable as follows in the top-level products.mak file and re-build the SysLink kernel driver.

<syntaxhighlight lang='make'> SYSLINK_TRANSPORT=TRANSPORTSHMCIRC </syntaxhighlight>

For more option information see the SysLink Install Guide Build Options section.

To configure the IPC/BIOS-side, you'll need to add the following lines to the slave's application configuration file (*.cfg) and re-build the application:

<syntaxhighlight lang='javascript'> var MessageQ = xdc.module('ti.sdo.ipc.MessageQ'); MessageQ.SetupTransportProxy = xdc.module('ti.sdo.ipc.transports.TransportShmCircSetup'); </syntaxhighlight>

Shared Memory Notify Transport

This shared-memory Notify-based MessageQ transport (**TransportShmNotify**) does no buffering before the messages are moved to the destination queue. Because of the lack of buffering, this transport tends to offer lower MessageQ latency than either TransportShm or TransportShmCirc. However, if messages aren't received quickly enough by the receiver, the sender may spin while waiting for the receiver to move the message to its local queue.

Note: This transport is recommended to be used with the NotifyDriverCirc as the buffering will be preformed in the notify driver. Using NotifyDriverShm will only allow one message at a time to be placed in the MessageQ.

Configuration

The transport configuration feature in SysLink is controlled by a build options variable (SYSLINK_TRANSPORT). The variable will only configure the GPP SysLink driver (syslink.ko). In SYS/BIOS, its controlled via the application's configuration file.

To configure the GPP core driver, you can set the variable as follows in the top-level **products.mak** file and re-build the SysLink kernel driver.

<syntaxhighlight lang='make'> SYSLINK_TRANSPORT=TRANSPORTSHMNOTIFY </syntaxhighlight>

For more option information see the SysLink Install Guide Build Options section.

To configure the IPC/BIOS-side, you'll need to add the following lines to the slave's application configuration file (*.cfg) and re-build the application:

<syntaxhighlight lang='javascript'> var MessageQ = xdc.module('ti.sdo.ipc.MessageQ'); MessageQ.SetupTransportProxy = xdc.module('ti.sdo.ipc.transports.TransportShmNotifySetup'); </syntaxhighlight>

Common Pitfalls

Misconfiguration

The most common issue with changing the Notify Driver or MessageQ Transports is misconfiguration from one of the cores. A SysLink limitation is that all cores in the system must be configured to used the same Notify Driver and MessageQ transport. (When only using the IPC product without SysLink - for example, for BIOS-to-BIOS communication - while they must agree between each core, different drivers/transports can be used between different cores.)

A common symptom of misconfiguration is that GPP application hangs and eventually times out in SysLink's Ipc_control() STARTCALLBACK call. Misconfiguration can lead to data corruption in Shared Memory Region 0 (SR0) and can be difficult to determine; the memory layout within SR0 is affected by which drivers/transports are being used, so misconfiguration leads to undefined behavior. If you are experiencing unknown data corruption issues after changing these setting ensure that all cores are properly configured.

Build errors

A common build error occurs when changing to a different Notify driver. Some Linux kernels (e.g. TI81XX PSPs) have the notify module incorporated into the kernel, and that notify driver uses the NotifyDriverShm protocol. Changing to a different Notify driver requires that the Notify driver used be the one provided by SysLink, not the one provided by the kernel. SysLink tries to detect misconfigurations like this at build time, failing during the build when possible.

For more information see the SysLink Notify page.

Note: If other cores are using the Notify module in the kernel for other purposes (e.g. TI81XX HDVPSS driver), SysLink applications must use the default Notify driver (NotifyDriverShm) to ensure all cores are using the same Notify driver.

See Also

- IPC Users Guide - describes the OS-independent IPC APIs and their usage. Also provides information on various Notify drivers and MessageQ transports provided for SYS/BIOS-based environments.

{{ 1. switchcategory:MultiCore= • For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum • For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **SysLink Notify Drivers and Transports** here.	Keystone= • For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum • For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **SysLink Notify Drivers and Transports** here.	C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***SysLink Notify Drivers and Transports*** *here.*	DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***SysLink Notify Drivers and Transports*** *here.*	MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***SysLink Notify Drivers and Transports*** *here.*	OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***SysLink Notify Drivers and Transports*** *here.*	OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***SysLink Notify Drivers and Transports*** *here.*	MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article* ***SysLink Notify Drivers and Transports*** *here.*	*For technical s please post yo questions at http://e2e.ti.co Please post o comments abo article* ***SysLinl Drivers and Transports*** *he* }}

Links

Amplifiers & Linear	DLP & MEMS	Processors	Switches & Multiplexers
Audio	High-Reliability	• ARM Processors	Temperature Sensors & Control ICs
Broadband RF/IF & Digital Radio	Interface	• Digital Signal Processors (DSP)	Wireless Connectivity
Clocks & Timers	Logic	• Microcontrollers (MCU)	
Data Converters	Power Management	• OMAP Applications Processors	

Retrieved from "https://processors.wiki.ti.com/index.php?title=IPC_Notify_Drivers_and_Transports&oldid=194387"

This page was last edited on 24 February 2015, at 12:38.