



MSP-H8 Embedded SDK

The [MSP-H8](#) is an 8-line PCM interface PCI board that supports both embedded and host-level signal-processing applications. The Embedded SDK gives the developer all the tools needed for the productive development of the call-stream processing required by applications such as terminating voice, fax, and data, as well as PSTN-IP gateway and IAD applications.

The MSP-H8 is ideal as a platform for low-density multi-line integrated-media stream processing using the Commetrex' [OpenMedia](#) streams environment. But, should the developer require a proprietary board-level environment, the MSP-H8 offers the hardware foundation for use with this Embedded Developer's Kit to support the development and market deployment of proprietary board-level environments and media-processing software.

Prior to the MSP-H8, digital-media stream-processing-resource boards were optimized for one media-processing task, usually voice or fax. The board's architecture and resources were limited and did not support integrated-media applications. Moreover, the architectures were closed, keeping the OEM from adding a media-processing resource, even if the board's MIPS and RAM could support it.

Now, the MSP-H8 has changed the rules by providing an open high-performance and scalable PCM interface that allows the developer to take advantage of the relatively infinite MIPS of today's general-purpose processors. And, if the MIPS available on the on-board TI TMS320C5402 are adequate, it can be used for the call-stream processing.



Features

- Open, embedded developer's kit
- Complete source code
- Windows 2000 based
- WDM driver
- Comprehensive diagnostics
- Board-level tasking executive
- Trunk-interface state machines
- Security button controller and interface
- Power-on self test and loader software
- JTAG interface
- Makefiles and project files
- Field-proven trunk-protocol software

Benefits

- Avoid 18-month hardware development effort
- Avoid 36-month software effort
- Total control of your software
- Supports your proprietary environment
- Productive system development
- Customer satisfaction

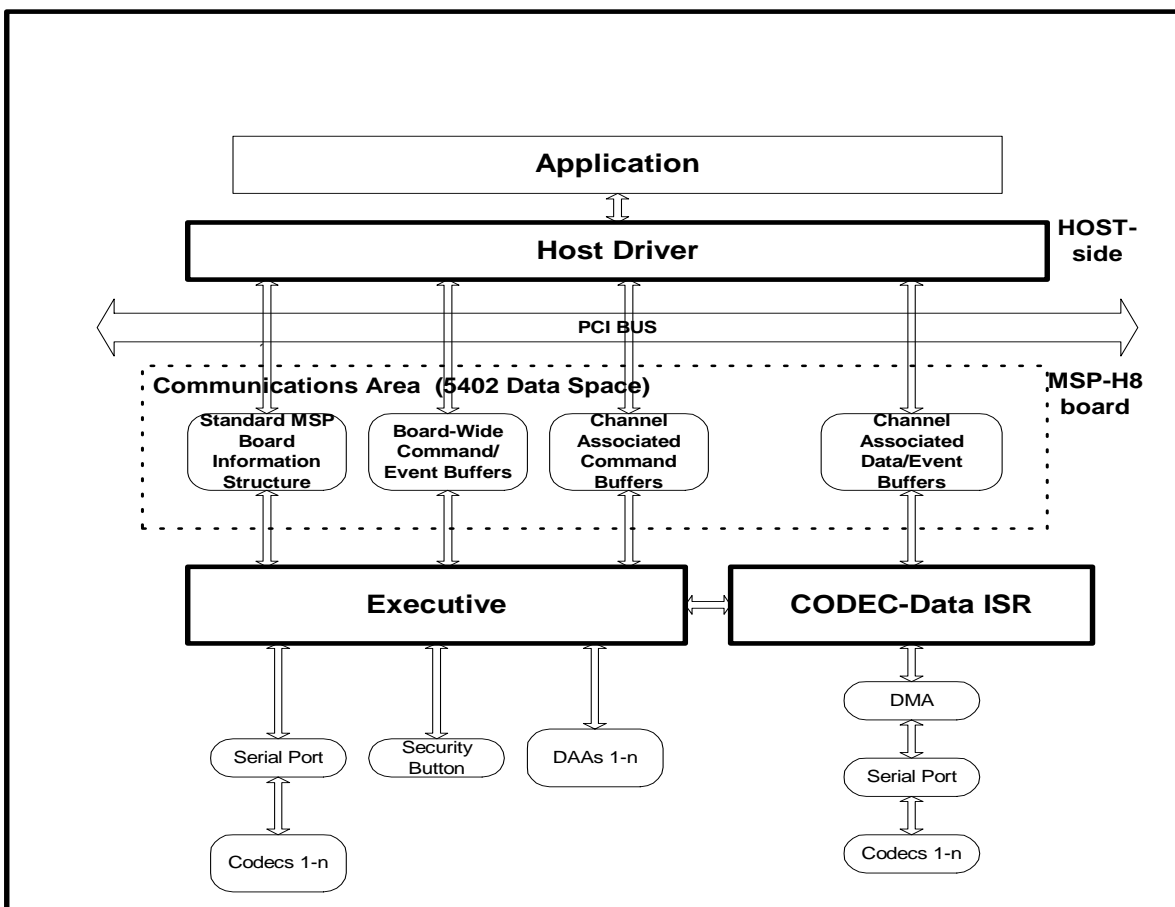
Overview

The developer of a communications system requiring proprietary call-stream media processing and the associated network interface can either make or buy the hardware platform. The MSP-H8 and its development tools provide a strong argument for the buy decision for low-density analog applications. When bundled with the MSP-H8 hardware, the Embedded Developer's Kit gives the designer all the MSP-H8-specific software tools needed to continue the platform development into the signal-processing software phase. The Kit includes the same software tools used by Commetrex' development engineers to test and

driver. These intermediate drivers may perform manipulations of the data streaming to/from the low-level driver. For example, a soft-modem might be an intermediate driver. Interrupt Request Packets are created by the I/O Manager and sent from driver to driver, either causing specific action by a given driver or being passed through to the driver below.

The MSP-H8 driver performs the following tasks:

- Enumerate all boards in the system,
- Associate an ISR with each board,



verify the MSP-H8 embedded software.

NT Driver

The MSP-H8 Windows WDM driver, included in the Kit as source code, is a “lowest-level” driver as discussed in Windows 2000 DDK, Kernel-Mode Drivers, Design Guide, Part 1, Section 4 – Basic Driver Structure. This architecture allows layering of kernel-mode “intermediate” drivers above the low-level

- Provide a DPC to process an interrupt,
- Verify 'C5402 Host Port access,
- Download code based on information in the registry,
- Maintain board state information,
- Move commands, events, and data using the MSP Low-Level API (MSPLL).

IOCTL Code
IOCTL_MSP_DEVICE_SET_LOOPBACK
IOCTL_MSP_DEVICE_GET_ALL_EVENTS
IOCTL_MSP_DEVICE_READ_MEMORY
IOCTL_MSP_DEVICE_WRITE_MEMORY
IOCTL_MSP_DEVICE_GET_PARAMETERS
IOCTL_MSP_DEVICE_SET_PARAMETERS
IOCTL_MSP_DEVICE_GET_SERIAL_NUMBER
IOCTL_MSP_DEVICE_GET_DRIVER_REVISION
IOCTL_MSP_DEVICE_GET_DOWNLOAD_REVISION
IOCTL_MSP_DEVICE_RDWR_ALL_CHANNELS
IOCTL_MSP_DEVICE_PLACE_BOARD_OUT_OF_SERVICE
IOCTL_MSP_DEVICE_PLACE_BOARD_IN_SERVICE
IOCTL_MSP_CHANNEL_SEND_CCSM_COMMAND
IOCTL_MSP_DEVICE_QUERY_CHANNEL_STATE
IOCTL_MSP_DEVICE_QUERY_DEVICE_STATE
IOCTL_MSP_DEVICE_SET_CODEC_GAIN
IOCTL_MSP_DEVICE_SPY

Tasking Executive

MSP_EXEC is the MSP-H8's execution controller. It maintains a list of tasks to be run and their periodicity. It allows tasks to be added or deleted from the list at run-time.

The MSP_EXEC maintains a list of tasks, each with an associated timer. For each real-time clock "tick", Timer0_ISR examines the task list and ages each node. MSP_EXEC executes an infinite loop, where it checks the aging of the tasks and calls the execution entry point for each ready-to-run task.

The executive does not preempt tasks. If the execution list is not completed before the next clock tick, the tick is queued and the next loop execution will begin as soon as the current task is completed. The executive monitors the backlog and signals the host when the backlog exceeds a preset threshold.

Prior to the executive running, individual tasks are added to the task list, with each task

detailing three entry points: an *open*, an *execute*, and a *close* entry point. The executive will call the *open* once per task before a call to the *execute* or *close* entry points. The *execute* entry point is called to perform whatever real work the task performs. A typical signaling state machine's *execute* entry point processes commands from the host, as well as signaling information gathered asynchronously by a hardware sampling interrupt service routine. The *close* entry point is not normally called unless the executive determines that the needed CPU cycles exceed the available cycles. The *close* entry point is responsible for returning the hardware to a known state (going on hook, for example) and for posting appropriate events to the host.

Tasks, when added to the task list, supply addresses to their *open*, *execute*, and *close* entry points. In addition, they supply an indication of the periodicity with which their entry points must be called.

A periodic ISR is used to gather inputs for the channel state machines. This ISR is always running, regardless of the state of the channel state machine. The ISR records inputs for each channel in a circular buffer. Channel state machines, which include loop-current status and hook control, consume these data and keep up with the ISR only if the tasks are not using more CPU cycles than are available. The ISR will detect overruns and will set a flag that indicates the overrun condition.

Diagnostics

MSP_DIAG is an MSP-H8 test and diagnostics tool. It is a Windows console application (see screen illustration on the next page) that allows monitoring of events (rings, on hook, off hook, etc.) from the MSP-H8. It also allows issue of Seize and Release commands, dialing of DTMF digit strings, as well as playing and recording binary audio files. Multiple MSP_DIAG threads can execute simultaneously. Multiple MSP-H8 boards are supported.

```

Command Prompt
D:\proj\msp-lib\msp_diag\Debug>msp_diag
msp_diag - 1 H8 devices found

mdp_diag board (0-0) command
command is one of:
  seize channel (0-7,0xff=all)
    sends a Seize command to the specified channel
  answer channel (0-7,0xff=all)
    sends an Answer command to the specified channel
  dial channel (0-7,0xff=all) DTMF
    Dials DTMF digits on the specified channel
    DTMF is a string consisting of: {0,1,2,3,4,5,6,7,8,9}
  release channel (0-7,0xff=all)
    sends a Release command to the specified channel
  play channel (0-7,0xff=all) playFile iteration (0-n)
    sends the the specified PCM playFile iteration times
    If iteration is zero (0), transmit indefinitely
  record channel (0-7,0xff=all) recordFile time
    records samples of PCM data for a given channel
  playNrecord channel (0-7,0xff=all) playFile recordFile time
    seizes then plays and records to/from the specified files
  loopback time
    Turns on loopback and transmits test buffers.
  monitor
    Monitors all messages from the board.

D:\proj\msp-lib\msp_diag\Debug>_

```

JTAG

The JTAG interface allows the control and debug of the on-board Texas Instruments TMS320C5402 processor. JTAG emulation is fully supported by Texas Instruments' [Code Composer Studio](#)™ when used in conjunction with TI's [XDS-series](#) emulators. JTAG emulation lets data be moved on and off chip non-intrusively, without interrupting the executing device. TI then augments this capability with additional emulation logic on the DSP to provide even greater visibility and access into registers and other internal functions such as on-chip cache memories.

Power-On Self-Test & Loader

Source code necessary to perform power-on testing and cold-boot downloads is included.

Security Button Controller

The MSP-H8 includes a small removable EEPROM security device (security button). It

provides a low-cost facility for storing and controlling access to sensitive information, such as software license administration.

The module is a state machine designed to perform the following functions:

- Reset the Security Button
- Read a data bit from the Security Button
- Read a data byte from the Security Button
- Write a data bit to the Security Button
- Write a data byte to the Security Button

Ordering Information

- MSP-H8 Embedded SDK, PN 70005
- MSP-H8, PN 70001
- MSP-H8 Trunk interface, PN 70011 (-1 LS, -2 GS, -3 DID, -6 FXO)

Commetrex, Open Telecommunications Framework and PowerFax are registered trademarks of Commetrex. All other trademarks are the property of their respective holders. Specifications subject to change without notice. Copyright © 2003.

Commetrex Corporation

1225 Northmeadow Parkway, Suite 120
 Roswell, GA 30076
 (770) 449-7775, Fax: (770) 242-7353
 www.commetrex.com τ sales@commetrex.com