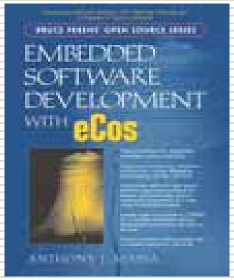
Embedded Software Development with eCos – Chapter4


BRUCE PERENS' OPEN SOURCE SERIES
EMBEDDED SOFTWARE DEVELOPMENT WITH eCos
ANTHONY J. MASSA

中興大學資訊科學系
指導教授：張軒彬
學生：李佳翰

Outline

- Virtual Vector
- Virtual Vector Configuration
- Virtual Vector Table Initialization
- Communication Channels
  - Console and Debugging Channels
  - Mangling
  - Controlling the Console Channel

Virtual Vector

- eCos defines a group of pointers to service functions and data called ***virtual vectors***. Virtual vectors are contained with in the ***Virtual Vector Table* (VVT)** .This table contains 64 vectors.
- The fact that the vectors are always placed at the same location in the address space means that both ROM and RAM startup configurations can access these and thus the services pointed to.
- **The primary goal** is to allow services to be provided by ROM configurations (ROM monitors such as RedBoot in particular) with *clients* in RAM configurations being able to use these services.
- Without the table of pointers this would be impossible since the ROM and RAM applications would be linked separately.

  - in effect having separate name spaces - preventing direct references from one to the other.

Virtual Vector(cont.)

- In general, a loose policy for governing the VVT is that the ROM monitor or the standalone application initializes all vectors in the table.
- The RAM application can then reinitialize any services it needs to provide.
- The default configuration is that the ROM monitor provides the console and debugging I/O services, and the RAM application initializes all other services.

VVT Service Funcation and data

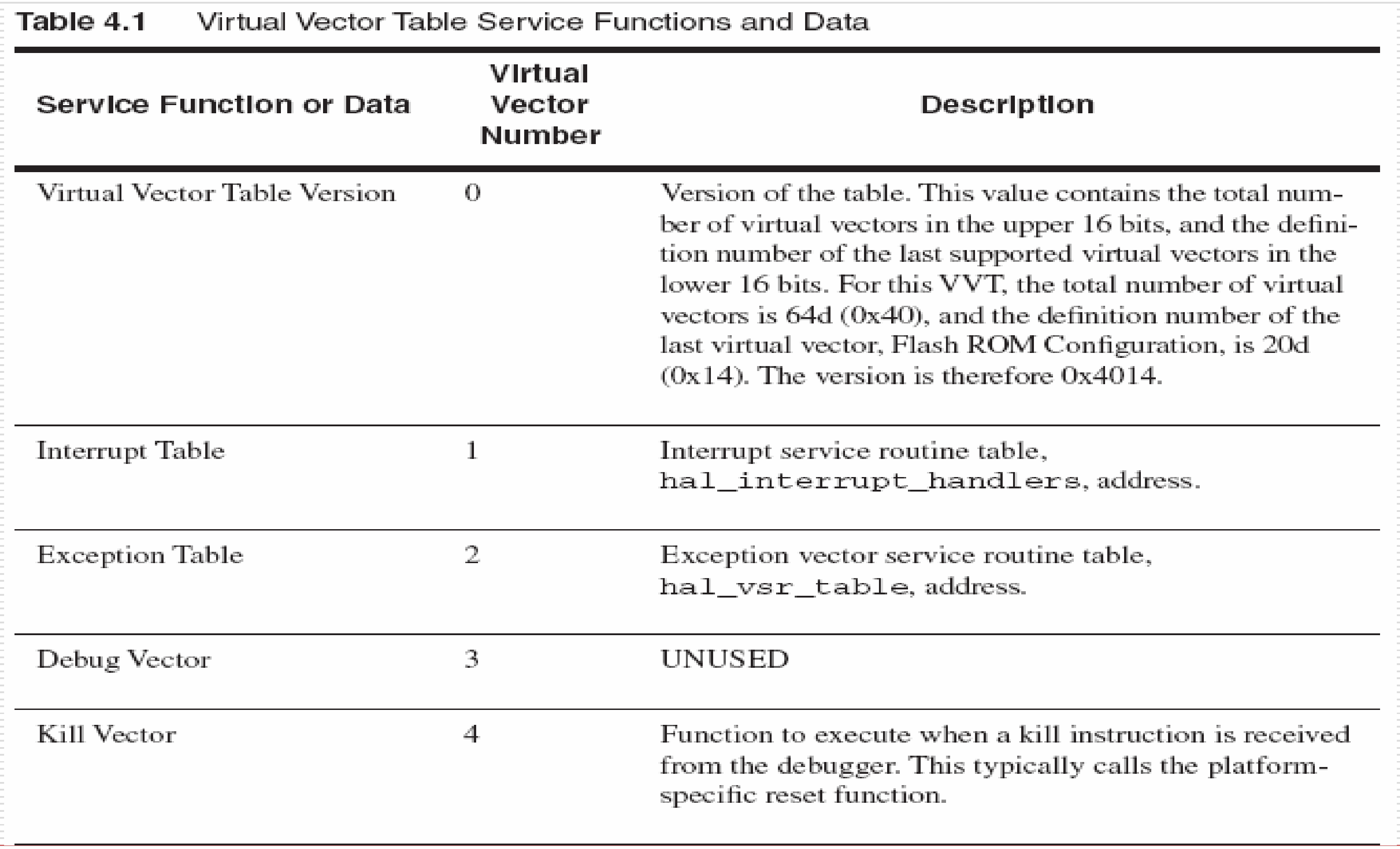
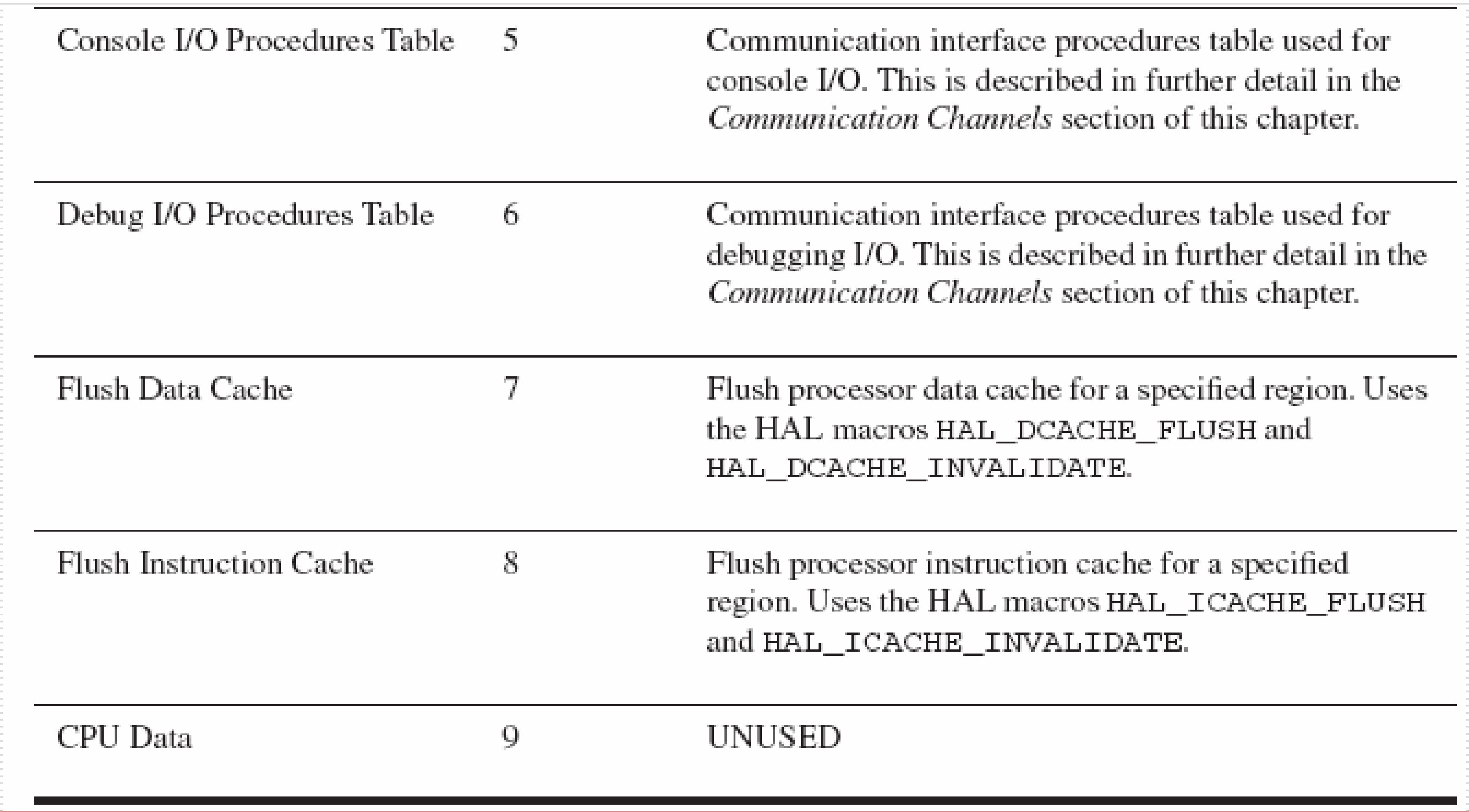


Table 4.1 Virtual Vector Table Service Functions and Data

Service Function or Data	Virtual Vector Number	Description
Virtual Vector Table Version	0	Version of the table. This value contains the total number of virtual vectors in the upper 16 bits, and the definition number of the last supported virtual vectors in the lower 16 bits. For this VVT, the total number of virtual vectors is 64d (0x40), and the definition number of the last virtual vector, Flash ROM Configuration, is 20d (0x14). The version is therefore 0x4014.
Interrupt Table	1	Interrupt service routine table, `hal_interrupt_handlers`, address.
Exception Table	2	Exception vector service routine table, `hal_vsr_table`, address.
Debug Vector	3	UNUSED
Kill Vector	4	Function to execute when a kill instruction is received from the debugger. This typically calls the platform-specific reset function.

VVT Service Funcation and data(cont.)



Console I/O Procedures Table	5	Communication interface procedures table used for console I/O. This is described in further detail in the *Communication Channels* section of this chapter.
Debug I/O Procedures Table	6	Communication interface procedures table used for debugging I/O. This is described in further detail in the *Communication Channels* section of this chapter.
Flush Data Cache	7	Flush processor data cache for a specified region. Uses the HAL macros `HAL_DCACHE_FLUSH` and `HAL_DCACHE_INVALIDATE`.
Flush Instruction Cache	8	Flush processor instruction cache for a specified region. Uses the HAL macros `HAL_ICACHE_FLUSH` and `HAL_ICACHE_INVALIDATE`.
CPU Data	9	UNUSED

VVT Service Funcation and data(cont.)

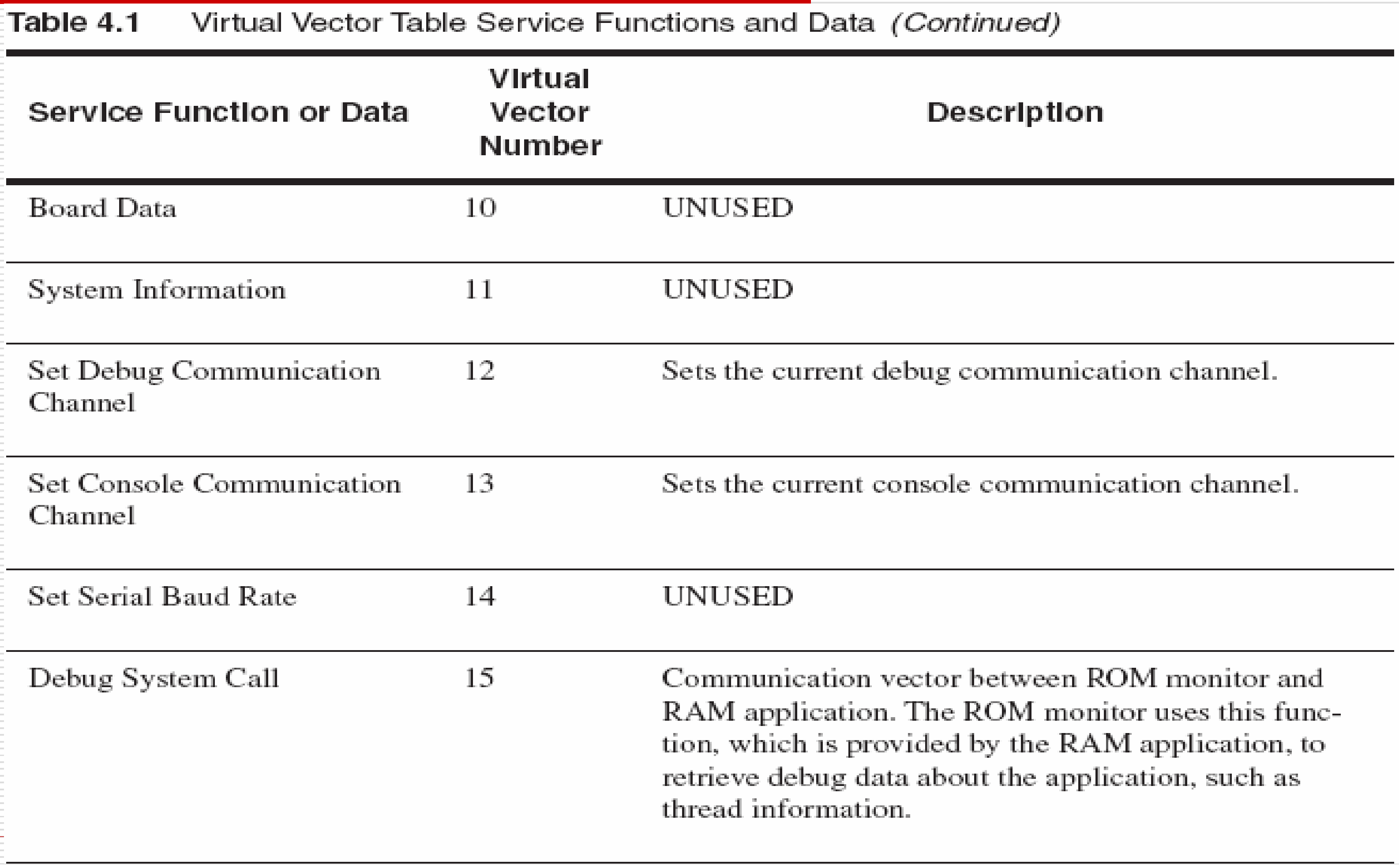
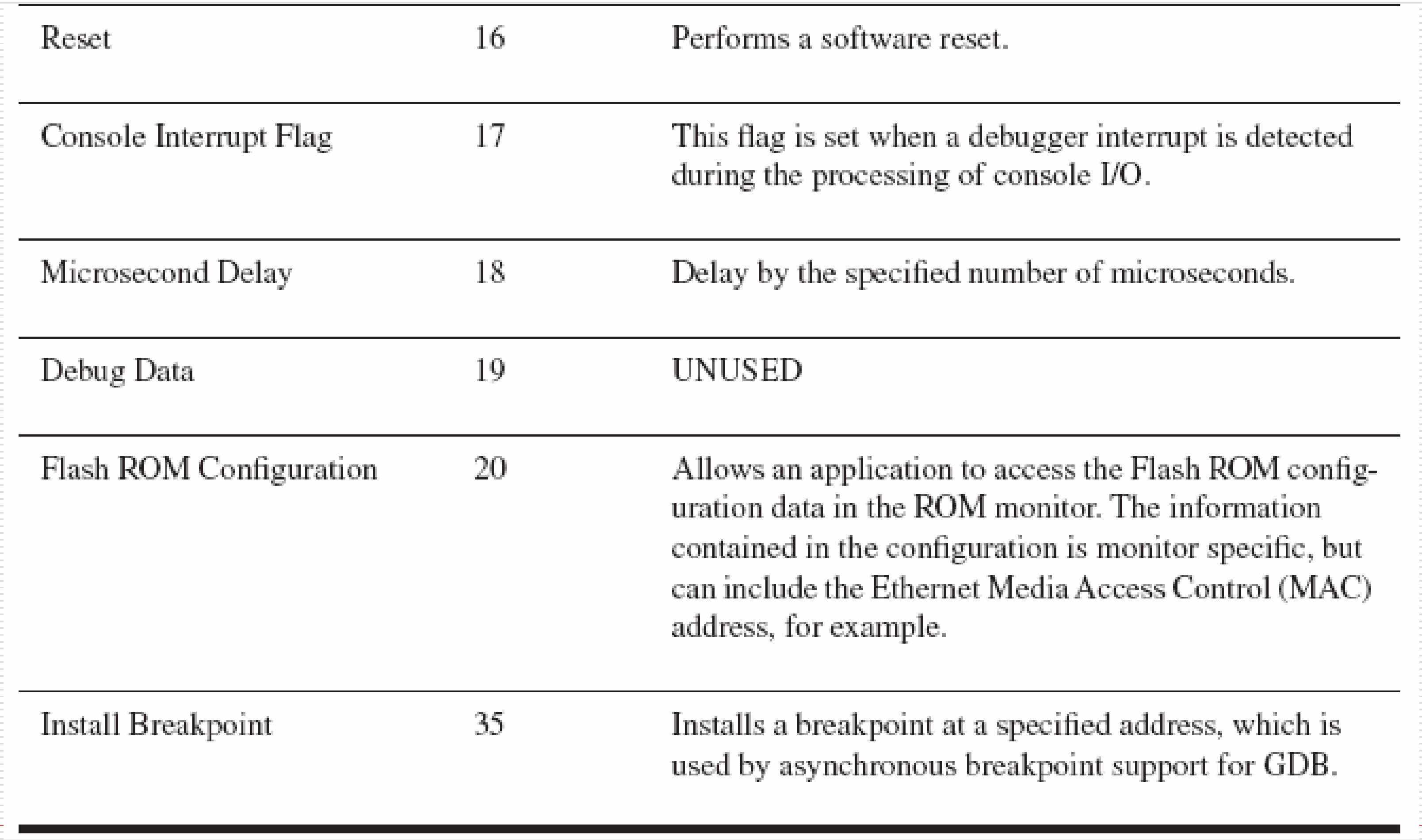


Table 4.1 Virtual Vector Table Service Functions and Data *(Continued)*

Service Function or Data	Virtual Vector Number	Description
Board Data	10	UNUSED
System Information	11	UNUSED
Set Debug Communication Channel	12	Sets the current debug communication channel.
Set Console Communication Channel	13	Sets the current console communication channel.
Set Serial Baud Rate	14	UNUSED
Debug System Call	15	Communication vector between ROM monitor and RAM application. The ROM monitor uses this function, which is provided by the RAM application, to retrieve debug data about the application, such as thread information.

VVT Service Funcation and data(cont.)



Reset	16	Performs a software reset.
Console Interrupt Flag	17	This flag is set when a debugger interrupt is detected during the processing of console I/O.
Microsecond Delay	18	Delay by the specified number of microseconds.
Debug Data	19	UNUSED
Flash ROM Configuration	20	Allows an application to access the Flash ROM configuration data in the ROM monitor. The information contained in the configuration is monitor specific, but can include the Ethernet Media Access Control (MAC) address, for example.
Install Breakpoint	35	Installs a breakpoint at a specified address, which is used by asynchronous breakpoint support for GDB.

Virtual Vector Configuration

- The virtual vector configuration options are located under the ***ROM Monitor Support*** (CYGPKG_HAL_ROM_MONITOR) configuration option within the HAL Common Configuration Components.
- The main virtual vector configuration option is ***Enable Use of Virtual Vector Calling Interface*** (CYGSEM_HAL_VIRTUAL_VECTOR_SUPPORT).

```
cdl_component CYGSEM_HAL_VIRTUAL_VECTOR_SUPPORT
           display     "Enable use of virtual vector calling interface"
           active_if   CYGINT_HAL_VIRTUAL_VECTOR_SUPPORT
           calculated  1
           description "
               Virtual vector support allows the HAL to let the ROM
               monitor handle certain operations. The virtual vector table
               defines a calling interface between applications running in
               RAM and the ROM monitor."
           compile     hal_if.c hal_misc.c
```

Virtual Vector Configuration(cont.)

- The two configuration options：

  Behave as a ROM Monitor (CYGSEM_HAL_ROM_MONITOR) and ***Work With a ROM Monitor*** (CYGSEM_HAL_USE_ROM_MONITOR), determine the type of image being built.

  - For RAM application debugging, typically ***Work With a ROM Monitor*** is enabled.
  - Enable the option ***Behave as a ROM Monitor*** if this program is to be used as a ROM monitor.

Virtual Vector Configuration(cont.)

- **Item List 4.1** Virtual Vector Configuration Suboptions：

Option Name： **Inherit Console Settings From ROM Monitor**
CDL Name： CYGSEM_HAL_VIRTUAL_VECTOR_INHERIT_CONSOLE
Description： Allows the RAM application to inherit the console setup by the ROM monitor using the configured channel and text encoding style.

Option Name： **Debug Channel Is Configurable**
CDL Name ：CYGPRI_HAL_VIRTUAL_VECTOR_DEBUG_CHANNEL_CONFIGURABLE
Description ：Allows the HAL startup code to make use of the debug channel configuration.

Option Name ：**Console Channel Is Configurable**
CDL Name： CYGPRI_HAL_VIRTUAL_VECTOR_CONSOLE_CHANNEL_CONFIGURABLE
Description ：Allows the HAL startup code to make use of the console channel configuration.

Virtual Vector Configuration(cont.)

Option Name ：**Initialize Whole Virtual Vector Table**
CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_INIT_WHOLE_TABLE
Description ：Causes the entire VVT to be initialized a default service function, nop_service. This is performed in hal_if_init.

Option Name ：**Claim Virtual Vector Table Entries By Default**
CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_CLAIM_DEFAULT
Description ：Allows the image to provide all services in the VVT, except Debug and Console Communication services, which will be provided by the ROM monitor. This option enables or disables the claiming of the individual virtual vector configuration options.

Option Name ：**Claim Reset Virtual Vectors**
CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_CLAIM_RESET
Description： Allows the image to provide the Reset and Kill Vector services.

Virtual Vector Configuration(cont.)

Option Name ：**Claim DELAY_US Virtual Vector**

CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_CLAIM_DELAY_US

Description ：Allows the image to provide the Microsecond Delay service.

Option Name ：**Claim Cache Virtual Vectors**

CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_CLAIM_CACHE

Description ：Allows the image to provide the Instruction and Data Cache Flush services.

Option Name ：**Claim Data Virtual Vectors**

CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_CLAIM_DATA

Description ：Allows the image to provide the Data services, which are currently unused in the VVT.

Virtual Vector Configuration(cont.)

Option Name： **Claim COMMS Virtual Vectors**

CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_CLAIM_COMMS

Description ：Allows the image to provide the Debug and Console Communication Channels.

Option Name ：**Do Diagnostic I/O Via Virtual Vector Table**

CDL Name ：CYGSEM_HAL_VIRTUAL_VECTOR_DIAG

Description ：Allows all HAL-level I/O to be performed using the configuration settings in the VVT.

Virtual Vector Table Initialization

- The common HAL defines macros, in the file **hal_if.h**, to **execute** and **set** the services with in the VVT (hal_virtual_vector_table).
- There are two macros defined for each virtual vector：
  - **Call macro** which executes the services in the VVT, has the form CYG_ACC_CALL_IF_XXX.
  - **Set macro** which sets the services in the VVT, has the form CYG_ACC_CALL_IF_XXX_SET
  - xxx is the defined name of the virtual vector that gives its loaction in the VVT.
- Common HAL VVT call and set macros for Reset virtual vector：

```
#define CYGACC_CALL_IF_RESET(_p_, _n_) \
(hal_virtual_vector_table[CYGNUM_CALL_IF_RESET])((_p_), (_n_))

#define CYGACC_CALL_IF_RESET_SET(_x_) \
hal_virtual_vector_table[CYGNUM_CALL_IF_RESET]=(CYG_ADDRWORD)(_x_)
// Install async breakpoint handler into vector table.
hal_virtual_vector_table[35] = (CYG_ADDRESS)install_async_breakpoint;
```

Virtual Vector Table Initialization(cont.)

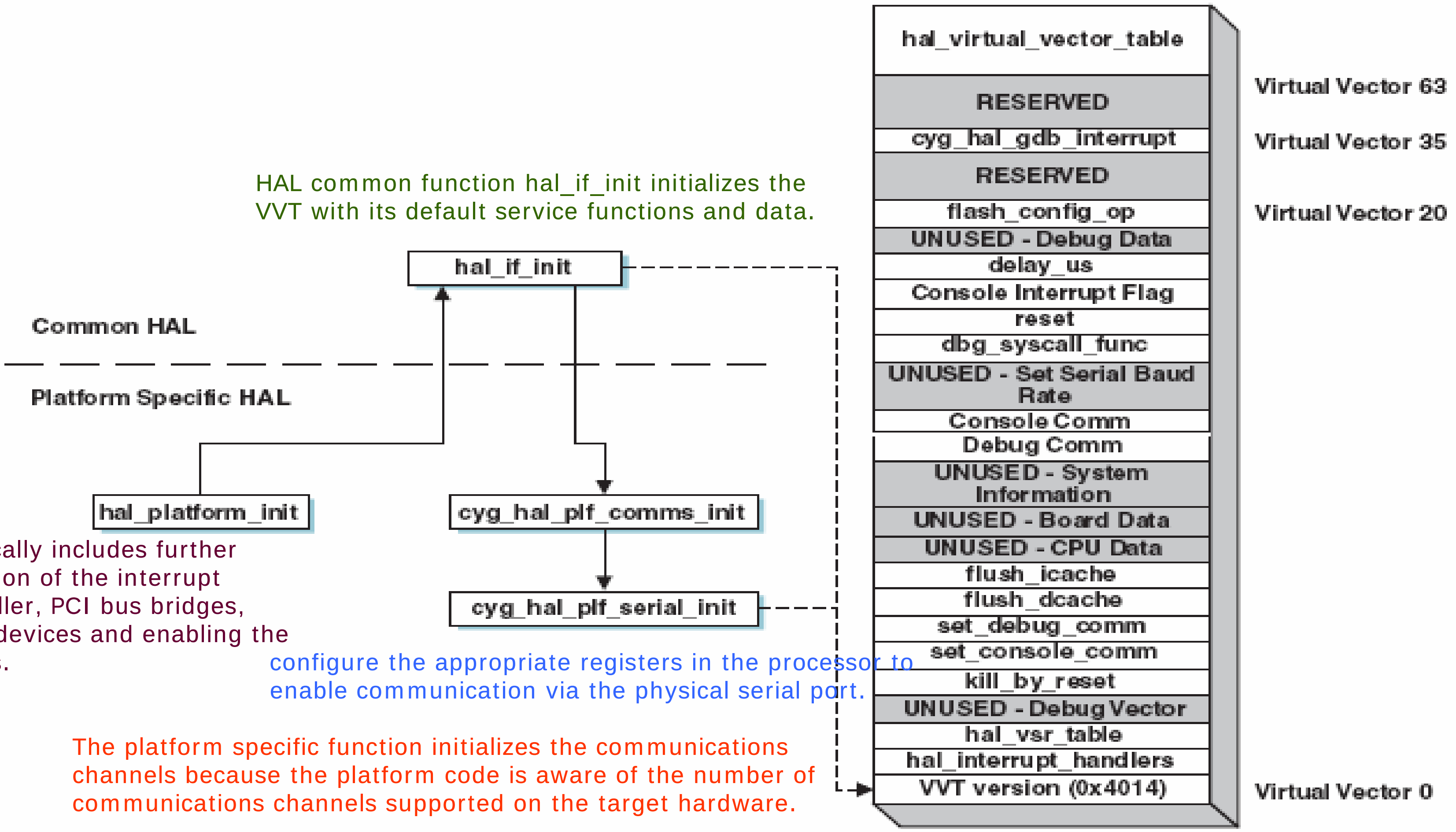
- Steps involved in the initialization of the VVT are:

1. ROM monitor is booted, causing the HAL built into the ROM monitor image to initialize the VVT with its own default service vectors according to the virtual vector configuration suboption settings.

2. The RAM application is loaded into memory via the debug channel in the ROM monitor.

3. Next, the ROM monitor is given a command to execute the RAM application. This turns control over to the RAM application; however, the VVT still contains function and data services provided by the ROM monitor.
4. Finally, the RAM application HAL executes, reinitializing the VVT according to the virtual vector configuration suboption settings. Any services that the RAM application provides are set into the VVT.

Virtual Vector Table Initialization(cont.)



HAL common function hal_if_init initializes the VVT with its default service functions and data.

This typically includes further initialization of the interrupt controller, PCI bus bridges, basic IO devices and enabling the caches.

configure the appropriate registers in the processor to enable communication via the physical serial port.

The platform specific function initializes the communications channels because the platform code is aware of the number of communications channels supported on the target hardware.

Figure 4.1 Virtual vector table initialization sequence.

Communication Channels

- All HAL I/O happens via the *communication channels*, also called ***COMMS* channels**.
- Since the HAL controls the low-level I/O functions for diagnostic and debug communication.
- There are two types of COMMS channels within the HAL, **console** and **debug**.
- Each channel type can be individually configured to use any physical port, such as serial or Ethernet, on the target hardware.

Console and Debugging Channels

- **The console channel** is used for any I/O initiated by calling the diag_*() functions.
  - Console channels are used for diagnostic I/O during the debugging process.
  - Note that these should only be used during development for debugging, assertion and possibly tracing messages.
  - All proper I/O should happen via proper devices.
- ***Debug channels*** are used for communication between the host debugger,such as GDB, and the ROM monitor.
- Having two separate channels allows.

Communication Interface Table

- Option ***Number of Communication Channels on the Board*** (CYGNUM_HAL_VIRTUAL_VECTOR_COMM_CHANNELS), under the HAL architecture-specific components, defines how many channels are present for a particular platform.

- There is a ***Communication Interface Table (CIT)*** associated with each COMMS channel in the system.

- ***CIT*** is an array that contains pointers to procedures or data relevant to the specific COMMS channel.

Console and Debug Communication Interface Table Procedures

•Table 4.2 lists the supported COMMS CIT procedures：

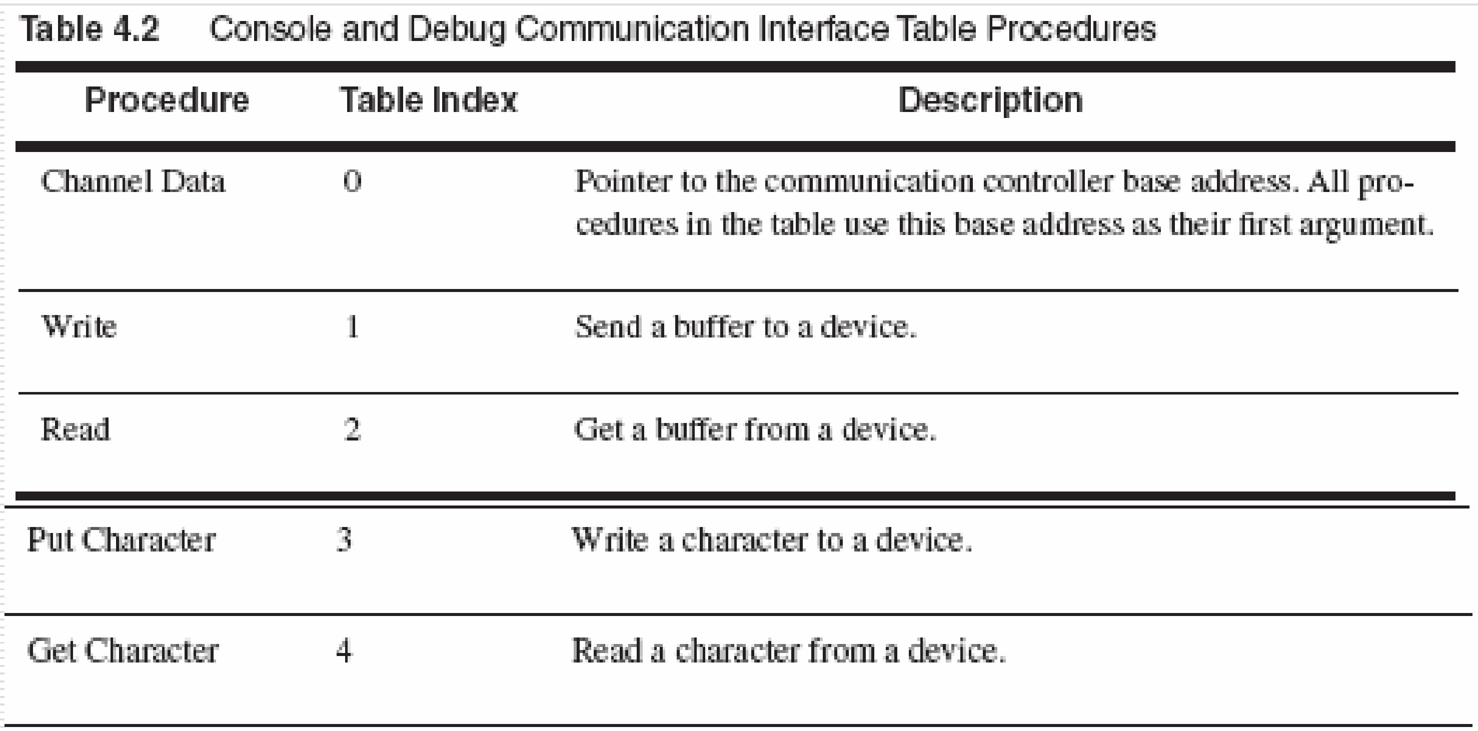


Table 4.2 Console and Debug Communication Interface Table Procedures

Procedure	Table Index	Description
Channel Data	0	Pointer to the communication controller base address. All procedures in the table use this base address as their first argument.
Write	1	Send a buffer to a device.
Read	2	Get a buffer from a device.
Put Character	3	Write a character to a device.
Get Character	4	Read a character from a device.

Console and Debug Communication Interface Table Procedures(cont.)

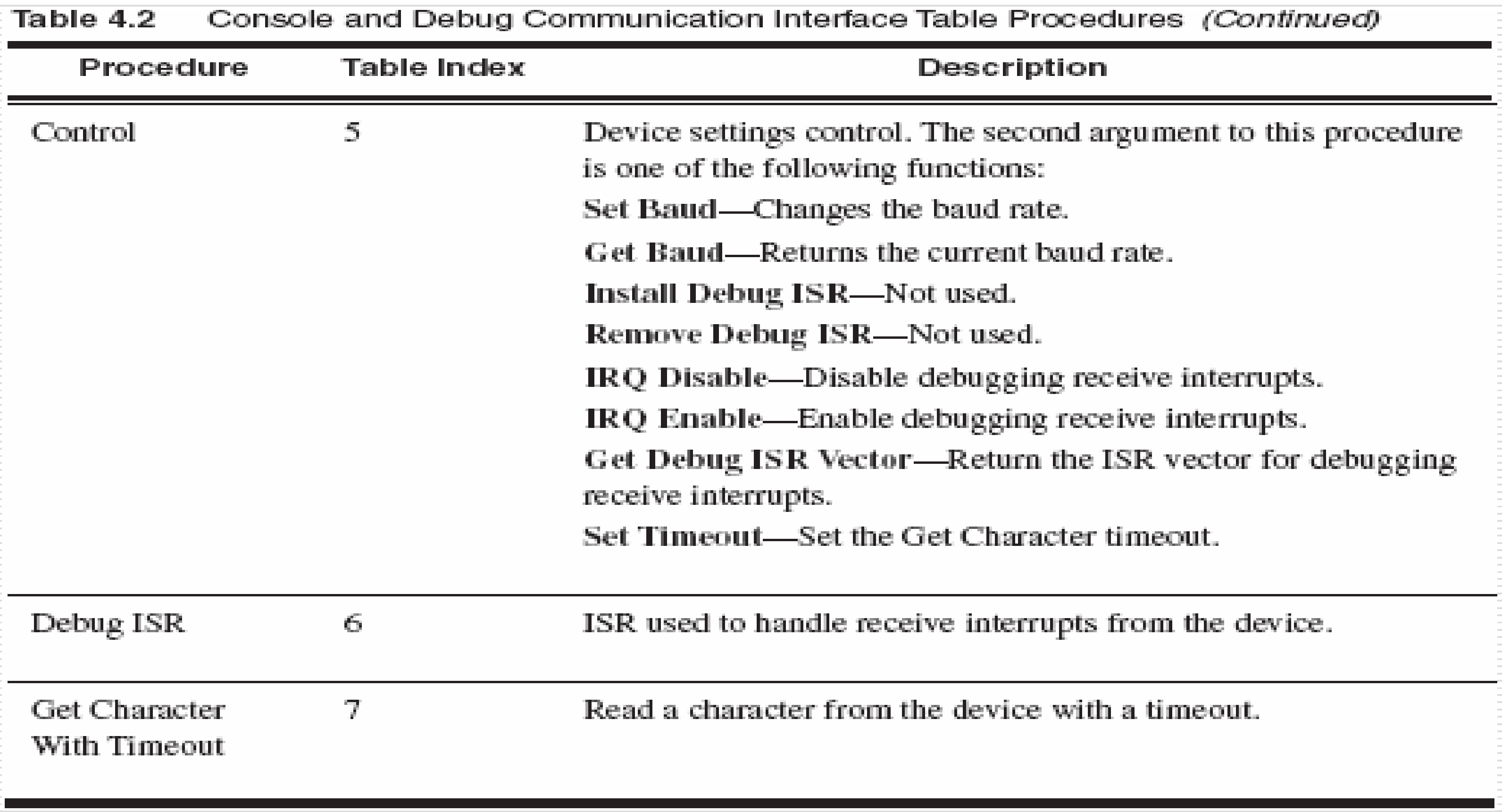


Table 4.2 Console and Debug Communication Interface Table Procedures *(Continued)*

Procedure	Table Index	Description
Control	5	Device settings control. The second argument to this procedure is one of the following functions: **Set Baud**—Changes the baud rate. **Get Baud**—Returns the current baud rate. **Install Debug ISR**—Not used. **Remove Debug ISR**—Not used. **IRQ Disable**—Disable debugging receive interrupts. **IRQ Enable**—Enable debugging receive interrupts. **Get Debug ISR Vector**—Return the ISR vector for debugging receive interrupts. **Set Timeout**—Set the Get Character timeout.
Debug ISR	6	ISR used to handle receive interrupts from the device.
Get Character With Timeout	7	Read a character from the device with a timeout.

Mangling

- To allow diagnostic messages to use the ***debug COMMS channel***, it is necessary to wrap the message with the protocol so that it can be properly displayed by GDB.

- This wrapping of the message with the protocol is called *mangling*.

- The HAL provides functions to encapsulate messages according to the selected mangler.

- Debuggers, such as GDB, typically use some type of protocol to encode the commands exchanged between the target hardware and the host debugger machine.

Controlling the Console Channel

- Console output configuration is either inherited from the ROM monitor launching the application, or it is specified by the application.

- This is controlled by the new option ***Inherit Console Settings From ROM Monitor*** (CYGSEM_HAL_VIRTUAL_VECTOR_INHERIT_CONSOLE). This option is enabled by default when using a ROM monitor.

- The configuration option ***Mangler Used on Diag Output*** (CYGSEM_HAL_DIAG_MANGLER),under the HAL common configuration components, allows the selection of a mangler. The possible values for this option are

Controlling the Console Channel(cont.)

- GDB
  - the GDB protocol is applied to text messages.
- None
  - which outputs raw text messages.
- The mangler procedures are contained in a communication interface table supporting the same functionality shown in Table 4.2.
- The use of the CIT procedures for diagnostic I/O is enabled by the configuration option ***Do Diagnostic I/O Via Virtual Vector Table*** (CYGSEM_HAL_VIRTUAL_VECTOR_DIAG) . This allows the console used for diagnostic I/O to be changed during run time.

Last Week

- Interrupt Service Routine Management, controls the attachment and detachment of interrupt service routines within the three HAL ISR tables ***(handlers, data, and objects).***

- ***hal_interrupt_handlers***—contains the addresses of the interrupt service routines installed by the application.

- ***hal_interrupt_data***—contains the data to be passed into the ISR.

- ***hal_interrupt_objects***—contains information that is used at the kernel level and hidden from the application layer.

The End