OpenVMS Terminal Driver
Port Class Interface
for Itanium

Version 1.0
November 26, 2003

Author: Forrest Kenney

Legal Notice

Preface

The purpose of this document it to provide a tool to help modify terminal port drivers and MID drivers to work on OpenVMS for Itanium. The changes described in this document are intended to make porting your driver to Itanium as simple as possible. It was not possible to make it as simple as recompile and re-link.

Intended Audience

This document is intended for system programmers who are already familiar with the OpenVMS operating system. It provides the information to modify an existing terminal port driver to function correctly on OpenVMS for Itanium processors. It does not provide you with instructions on how to compile or link your drivers.

Document Structure

The document is divided into four major sections:

- Introduction that explains why the change is needed and, at a high level, what the change is.
- Terminal Class driver routines – what their old interface was and what the new interface is.
- Port driver routines – what their old interface was and what the new interface is.
- Posix and Asian Terminal Driver extension sections describe our plans for Posix as well as Asian extensions. The Asian section also describes all the Asian terminal driver extensions.

Revisions:

Version	Date	Description of change
X0.2	4/25/03	Fix incorrect data type in function prototype for port_fdt. Complete incorrect function prototype for port_set_line
X0.9	5/14/03	Update to X0.9 minor edits change page footers
1.0	11/26/03	Add HP warranty information and minor edits

Introduction

As part of the port of the terminal driver to the Itanium architecture, the decision was made to use the Intel object language and calling standard. This calling standard means that the assumptions that the compilers made about which registers are preserved and which are scratch registers had to change. This affects code written in Macro that calls or is called by programs written in other languages.

The compilers and the linker have been modified to help code find potential problems. But this does not help the terminal port and class drivers, as they are never linked together into a single image. Knowing that these changes would force the terminal class driver and every terminal port driver to have to change, we had to decide what that change should be. There were two clear alternatives:

1. Provide cookbook instructions that could be applied for every port driver. But that would require creating at least three sets of instructions – one each for C, Macro, and Bliss. It was likely that these instructions sets would need to be tuned until they were correct.
2. Switch everything to have a call-based interface. This had the advantage of being straightforward in the case of drivers written in C, simply conditionally compiling the linkage definitions. Drivers written in Bliss should be equally as easy. Unfortunately, drivers in Macro will require a bit more work.

It did not take long to decide that option 2 was in many cases faster and in every case simpler. If possible, at some point in the future we will make the Alpha and Itanium terminal driver use call interfaces.

The work to change the Terminal driver was on the order of 3-4 hours and was entirely mechanical. Porting YTDRIVER took 10 minutes to add needed conditionals for IA64, and FTDRIVER took a couple of hours.

Terminal Driver Routines

CLASS_DISCONNECT

Port drivers call CLASS_DISCONNECT to indicate to the terminal class driver that the terminal is no longer connected to the system. This is the preferred way of disconnecting a process from a terminal on a non-modem line.

VAX/Alpha Interface

Inputs

R5 - UCB

Outputs

None

Scratch

R4

Itanium Interface

void class_disconnect(UCB *ucb)

CLASS_DS_TRANS

This routine handles data set transitions. The inputs to CLASS_DS_TRANS include a type code indicating what type of transition this is. If it is a transition of modem signals, the changed signals are also provided.

It is important to note that this routine should be called with the MODEM$C_INIT transition type from the unit init routine of the port driver if the unit is capable of data set transitions.

VAX/Alpha Interface

Inputs

R1 - Transition type (one of the following

MODEM$C_INIT	Initialize modem control
MODEM$C_INIT_NORESET	Start modem protocol but does not set DTR/RTS
MODEM$C_SHUTDOWN	Shut down the line and disconnect the process
MODEM$C_SHUTDOWN_NOHANGUP	Stop modem protocol but do change DTR/RTS

MODEM$C_DATASET	Data set signal changes

R2 - Modem signals mask
R5 - UCB address

Outputs

None

Scratch

R0 – R4

Itanium Interface

void class_ds_trans(int type, int signals, UCB *ucb)

CLASS_FORK

CLASS_FORK is the routine a port driver calls if it needs to start a driver fork process that would normally use the UCB's built in fork lock. The port driver must never initiate a fork directly using this fork block – it must always call this routine. CLASS_FORK, using the UCB, will set up the fork block and follow other necessary protocol on the port driver's behalf. When the fork has taken place, the class driver will call the port driver at the port driver's port service routine PORT_FORKRET.

VAX/Alpha Interface

Inputs

R5 - UCB address

Outputs

None

Scratch

R3, R4

Itanium Interface

void class_fork(UCB *ucb)

CLASS_GETNXT

This routine returns with the next character to be output on the unit. It should be called whenever the terminal port driver has completed the current character or burst. If data is returned by CLASS_GETNXT, a time is set up (unless explicitly disables) and the interrupt expected bit is set.

VAX/Alpha Interface

Inputs

R5 - UCB address

Outputs

R2 - Number of characters if UCB$B_TT_OUTYPE is negative
R3 - Character to output if UCB$B_TT_OUTYPE is 1

Figure 1 UCB$B_TT_OUTYPE

Zero	No data to output
One	One character to output returned in R3 for VAX and Alpha. It is returned in R0 for Itanium
Negative	Burst of characters to output UCB$L_TT_OUTADDR is address of first byte to output. UCB$W_TT_OUTLEN number of characters to output

Scratch
R1, R4
Preserved
R0

Itanium Interface

unsigned char class_getnxt(UCB *ucb)

CLASS_PUTNXT

This routine is called by port drivers to pass input characters. Characters received on non-passall units are filtered for immediate control sequences. Slave mode (no unsolicited input) units must have outstanding reads, otherwise the character, after control character filtering, is ignored.

If the input characters will be echoed, CLASS_GETNXT is called to notify the port driver. This routine may or may not return output data depending upon the setting of interrupt expected. If the UCB$V_INT bit in UCB$L_STS is set calls to CLASS_PUTNXT will not return data.

If data is returned from CLASS_PUTNXT it should be assumed that more data may follow, so the terminal port driver should be coded to call CLASS_GETNXT when the data that was returned has been output.

VAX/Alpha Interface

Inputs
R3 - Input character
R5 - UCB address
Outputs
R2 - Number of characters if UCB$B_TT_OUTYPE is negative
R3 - Character to output if UCB$B_TT_OUTYPE is 1

Figure 2 UCB$B_TT_OUTYPE

Zero	No data to output
One	One character to output returned in R3 for VAX and Alpha. It is returned in R0 for Itanium
Negative	Burst of characters to output UCB$L_TT_OUTADDR is address of first byte to output. UCB$W_TT_OUTLEN number of characters to output

Scratch
R1, R4
Preserved
R0

Itanium Interface

unsigned char class_getnxt(unsigned char in_char, UCB *ucb)

CLASS_SETUP_UCB

This routine is called at unit init during both system startup and power failure. All terminal related fields in the UCB are zeroed except for the speed and fill counts. The cursor is set to 1 to force a CR-LF. The holding tank is invalidated, and the fork block is initialized.

The write queue may be initialized if the list head is empty.

VAX/Alpha Interface

Inputs
R5 - UCB
Outputs
None
Scratch
None
Preserved
R0, R5

Itanium Interface

void class_setup_ucb(UCB *ucb)

CLASS_POWERFAIL

This routine is called the port driver's unit init routine when a powerfail is detected.

VAX/Alpha Interface

Inputs

R5 - UCB address

Outputs

None

UCB$W_STS	UCB$V_INT is cleared UCB$V_TIM is set
UCB$L_DUETIME	cleared

Scratch

None

Preserved

All preserved

Itanium Interface

void class_powerfail(UCB *ucn);

CLASS_READERROR

CLASS_READERROR is called when the terminal port driver detects a parity, data overrun or framing error on the terminal line. CLASS_READERROR completes the read with error if a read is active, or just returns if no read is active.

VAX/Alpha Interface

Inputs

R3 - Character that triggered error bits 12-14 set to indicate error type

Bit 12	parity error on the given character
Bit 13	Framing error on the given character
Bit 14	Data overrun

R5 - UCB address

Outputs

R3 - Character to output based on UCB$B_TT_OUTYPE

UCB$B_TT_OUTYPE	0 - if no character to output 1 - if valid character to output

Scratch

R1, R2, and R3

Preserve

R0, R4, R5

Itanium Interface

unsigned char class_readerror(unsigned int character, UCB *UCB);

Port Driver Routines

PORT_DISCONNECT

This routine notifies the port driver of last deassign on the UCB. A call to this routine means that there are no longer channels associated with the device. If the delete bit is set in the UCB$L_DEVSTS field in the UCB then the UCB will be deleted by the system. Note: As long as the device name is known to the system, broadcasts and assign channel requests may occur on this device. (Broadcasts, however, will not occur if the DEV$V_NET bit is set.)

VAX/Alpha Interface

Inputs

R0 Flags

Bit 0 nohangup	if set then no hangup was requested I.E. the user requested that the UCB not be deleted.

R5 UCB

Outputs

None

Scratch

None

Preserve

Any used

Itanium Interface

void port_disconnect(unsigned int flags, UCB *ucb);

PORT_DS_SET

The PORT_DS_SET routine outputs modem signals to the specified unit. Modem Jmasks are defined in $TTDEF. Signals defined include the following:

TT$M_DS_CARRIER	Carrier signal detected
TT$M_DS_CTS	Clear to send
TT$M_DS_DSR	Data set ready
TT$M_DS_DTR	Data terminal ready
TT$M_DS_RING	Ring indicator
TT$M_DS_RTS	Request to send
TT$M_DS_SECREC	Secondary receive
TT$M_DS_SECTX	Secondary transmit

VAX/Alpha Interface

Inputs

R2 Set & Clear signals

Byte 0	Signals to set
Byte 1	Signals to clear

R5 UCB

Outputs

None

Scratch

R1, R2, R3

Preserve

Any used except R1, R2, R3

Itanium Interface

void port_ds_set(unsigned int signals, UCB *ucb);

PORT_FDT

When the QIO function code is IO$_TTY_PORT, the terminal class driver passes control to the PORT_FDT routine. It is the responsibility of the port to do whatever processing a FDT routine would normally do. This includes validating function modifiers, checking the P1 - P5 parameters, verifying access to buffers, and terminating with a call to EXE$QIORETURN, EXE$FINISHIO, or EXE$ABORTIO.

This mechanism allows a port driver to implement function modifiers, which are device specific. The port driver is thus not dependent on extensions to the port/class interface for new functionality. Note that if the PORT FDT request is not completed attempts to cancel the request may place the process in RWAST State. Drivers that provide a PORT FDT routine should also support a PORT CANCEL routine that take care of canceling PORT FDT requests.

On Alpha and VAX the port driver returns control to the $QIO dispatching code. For IA64 they return control to the terminal class driver FDT routine which returns control to the $QIO dispatching code.

VAX/Alpha Interface

Inputs

R3 Address of the IRP for this request
R4 Current PCB
R5 UCB address
R6 Assigned CCB
R7 Function code
AP Address of first function dependent QIO parameter (P1) "**VAX ONLY**"

Outputs

R0 SS$_FDT_COMPL **"Alpha and IA64"**

Scratch

R2

Preserve

Any used except R0 and R2

Itanium Interface

```
int     port_fdt(IRP *irp, PCB *pcb, UCB *ucb, CCB *ccb,
                 unsigned short int fcode);
```

PORT_FORKRET

This entry vector is provided as a return address to the port driver when a fork is requested by the port. The fork returns no context other than the UCB.

On IA64 the terminal class driver does a CALL to this routine rather than a JSB like on Alpha.

VAX/Alpha Interface

Inputs

R5 UCB

Outputs

None

Scratch

None

Preserve

Any used

Itanium Interface

```
void    port_forkret(UCB *ucb);
```

PORT_MAINT

This routine is called whenever a SETMODE QIO with the maintenance function is issued. The parameters to the IO$M_MAINT function are placed into the location UCB$B_TT_MAINT. Each port driver must decide which functions it needs to support. Possible maintenance functions are listed in section 5.4.3 of the OpenVMS I?O User's Reference Manual.

VAX/Alpha Interface

Inputs

R5 UCB address (UCB$B_TT_MAINT - functions to be performed)

Outputs

None

Scratch

None

Preserve

Any used

Itanium Interface

void port_maint(UCB *ucb);

PORT_SET_LINE

PORT_SET_LINE changes the terminal line parameters. It is called whenever any terminal characteristic in UCB$L_DEVDEPEND or UCB$L_DEVDEPND2 is changed or when speed, parity, and the enabling or disabling of DMA and automatic flow control are affected.

This is the only port routine that is allowed to write the fields UCB$L_DEVDEPEND and DEVDEPND2.

VAX/Alpha Interface

Inputs

R5 UCB

UCB$B_TT_MAINT	Maintenance parameters
UCB$B_TT_PARITY	Parity, stop bits and frame size
UCB$W_TT_SPEED	Low byte transmit speed High byte receive speed or 0
UCB$W_TT_PRTCTL	DMA and AUTOXOFF enable flags
UCB$L_DEVDEPEND	First device dependent long word
UCB$L_DEVDEPND2	Second device dependent long word

Outputs

None

Scratch

R0

Preserve

Any used except R0

Itanium Interface

void port_set_line(UCB *ucb);

PORT_SET_MODEM

A call to this routine informs the port that this line has been enabled for modem

signal input transitions. Ports implementing modem functions must insure that the hardware is ready to detect changes in input modem signals. This function is implemented by timer based polling when the hardware does not provide this capability.

VAX/Alpha Interface

Input:

R5 UCB

Output:

None

Scratch:

None

Preserve

Any used

Itanium Interface

void port_set_modem(UCB *ucb);

PORT_STARTIO

This routine is called to start up output on a inactive unit. It will always be called with either a character or a burst of data. PORT_STARTIO is not called unless the line is IDLE (UCB$V_INT is clear in UCB$W_STS). The INT bit is used as an interlock to signify that the port output logic is busy. INT is always set by the class driver when PORT_STARTIO is called. If the port requests timers to be set up (NOTIME clear in UCB$W_TT_PRTCTL word) then an output timer is computed for the burst or character and the TIM bit is set.

VAX/Alpha Interface

Inputs:

R3 Character to output

R5 UCB

UCB$B_TT_OUTYPE	Zero	No character to output
	One	One character to output
	Negative	Burst to output
UCB$L_TT_OUTADR		Address of burst to output if outype is negative
UCB$W_TT_OUTLEN		Length of burst

Outputs:

None

Scratch:

R1, R2, R3, R4

Preserve:

An used except R1, R2, R3, R4

Itanium Interface

void port_startio(unsigned char out_char, UCB *ucb);

PORT_ABORT

A call to this routine commands the port to abort any currently active output activity. This usually means the last burst of output sent to the port. This routine may be called at any time from the class driver and will invalidate the contents of the data in UCB$L_TT_OUTADR.

VAX/Alpha Interface

Inputs

R5 UCB

Outputs

None

Scratch

R0

Preserve

Any used except R0

Itanium Interface

void port_abort(UCB *ucn);

PORT_RESUME

This vector informs the port to resume any previously stopped output. The port must tolerate this routine being called at any time (even if output is active or has previously been stopped). This routine should always insure that the hardware is enabled for output.

VAX/Alpha Interface

Inputs:

R5 UCB

Outputs:

None

Scratch:

R0

Preserve:

Any used except R0

Itanium Interface

void port_resume(UCB *ucb);

PORT_STOP

This routine is called when the terminal class driver wishes to halt the output data stream. The data stream should be stopped as soon as possible. STOP is normally called in response to input flow control.

VAX/Alpha Interface

Inputs:

R5 UCB

Outputs:

None

Scratch:

R0

Preseerve:

Any used except R0

Itanium Interface

```
void    port_stop(UCB *ucb);
```

PORT_XOFF

A call to XOFF signifies that the class driver is approaching or has reached its input limit. The port should take steps to stop the input data stream. For character oriented controllers the port is commanded to insert the flow control character in the output data stream as soon as possible.

VAX/Alpha Interface

Inputs:

R3 Flow control character to be inserted in the data stream

R5 UCB address

UCB$W_STS UCB$V_INT may be set

Outputs:

UCB$W_STS UCB$V_INT should be set

Scratch:

R0

Preserve:

Any used except R0

Itanium Interface

```
void    port_xoff(unsigned char flow_char, UCB *ucb);
```

PORT_XON

XON is called when the terminal driver has cleared up its input path and is now ready to accept data. For character oriented controllers the port should insert the flow

control character in the output data stream.

VAX/Alpha Interface

Inputs:

R3 Flow control character to be inserted in the data stream

R5 UCB address

UCB$W_STS UCB$V_INT may be set

Outputs:

UCB$W_STS UCB$V_INT should be set

Scratch:

R0

Preserve:

Any used except R0

Itanium Interface

void port_xon(nsigned char flow_char, UCB *ucb);

PORT_CANCEL

Port cancel routine is called when either the $DASSGN or $CANCEL system service is called. It is used to allow a port driver to cancel PORT FDT requests that cannot be cleaned up the TERMINAL driver. This routine is optional for all port drivers that do not support PORT FDT routines.

VAX/Alpha Interface

Inputs:

R4 PCB

R5 UCB

R6 Negative of the channel number

R8 Reason either CAN$C_CANCEL or CAN$C_DASSGN

Outputs:

None

Scratch:

R0, R1

Preserve:

Any used except R0 and R1

Itanium Interface

void port_cancel(PCB *pcb, UCB *ucb, int chanel, int reason);

PORT_START_READ

Start read is called when the TERMINAL driver has made a read active. This

routine is only called if TT$M_PC_SMART_READ is set in UCB$W_TT_PRTCTL. If a read has a prompt string, this routine is called before the prompt is output.

VAX/Alpha Interface

Inputs:

R3 IRP
R5 UCB

Outputs:

None

Scratch:

R0

Preserve:

Any used except R0

Itanium Interface

void port_start_read(IRP *irp, UCB *UCB);

PORT_MIDDLE_READ

Port middle read is called wither when TTDRIVER get the first input character for an active read or when the prompt string is output. Like PORT_START_READ it is only called if TT$M_PC_SMART_READ is set in UCB$W_TT_PRTCTL.

VAX/Alpha Interface

Inputs:

R4 Address of TTY read buffer
R5 UCB

Outputs:

None

Scatch:

R0

Preserve:

Any used except R0

Itanium Interface

void port_middle_read(void *tty_read_buffer, UCB *ucb);

PORT_END_READ

PORT_END_READ is called just before the terminal driver forks to complete the read. Like PORT_START_READ it is only called if TT$M_PC_SMART_READ is set in UCB$W_TT_PRTCTL.

VAX/Alpha Interface

Inputs:

R3 IRP
R5 UCB

Outputs:

None

Scratch:

R0

Preserve:

Any used except R0

Itanium Interface

```
void    port_end_read(IRP *irp, UCB *ucb);
```

Posix and Asian Terminal Driver Extensions

Posix

POSIX is not supported for OpenVMS in Itanium Processors, no changes will be made to the code in TTDRIVER that calls the POSIX terminal driver.

Asian Terminal driver hook

We are not changing all the places in TTDRIVER that call the Asian Terminal driver. The existing code is written in MACRO and should not have any issues with linkages and register usage. The section below describes all routines and their interfaces.

TTY$A_ASIAN_CHECK_ODLSEQ - Check for ODL request

Check whether the input is part of an ODL request and initiate ODL request parsing if necessary.

Inputs:
- R2 ADDRESS OF THE UNIT STATE VECTOR
- R3 CHARACTER TO BUFFER
- R5 UCB ADDRESS

Outputs:
- R2, R3, R5 ARE PRESERVED
- R1 0 signal TTDRIVER to continue as usual
  1 signal dismiss

ASIAN_BEGIN_ECHO - start output if necessary

Call TTY$GETNEXTCHAR to start output if the device is not busy and we are not recovering from ODL request sequence error.

Inputs:
- R2 ADDRESS OF THE UNIT STATE VECTOR
- R3 CHARACTER TO BUFFER
- R5 UCB ADDRESS

Outputs:
- R2, R3, R5 ARE PRESERVED
- R1 0 Signal OK to start output
  1 Signal not OK to start output

ASIAN_PRELOAD

Inputs:
- R0 GETNEXTCHAR1 routine address

R5 UCB ADDRESS
if UCB$B_TL_A_MODE preload not set
R3 0 AND CC = ZERO - NO CHARACTER TO OUTPUT
CHAR AND CC = PLUS - SINGLE CHARACTER TO OUTPUT
ADDRESS AND CC = NEG - BURST (R2 = LENGTH)
UCB$B_TT_OUTYPE = -1 BURST
ADDRESS IN R3 AND UCB$L_TT_OUTADR
LENGTH IN R2 AND UCB$W_TT_OUTLEN
0 NO CHARACTER TO OUTPUT
1 SINGLE CHARACTER TO OUTPUT IN R3
Outputs:
R1 0 SIGNAL THE CALLER TO RETURN TO ITS CALLER
1 SIGNAL THE CALLER TO GO TO TTY$GETNXTCHAR
2 SIGNAL THE CALLER TO GO TO GETNXTCHAR1
R3 0 NO CHARACTER TO OUTPUT
CHAR - SINGLE CHARACTER TO OUTPUT
ADDRESS - BURST (R2 = LENGTH)
(ADDRESS AND LENGTH ALSO IN UCB)
UCB$B_TT_OUTYPE = -1 BURST
ADDRESS IN R3 AND UCB$L_TT_OUTADR
LENGTH IN R2 AND UCB$W_TT_OUTLEN
0 NO CHARACTER TO OUTPUT
1 SINGLE CHARACTER TO OUTPUT IN R3
R5 UCB ADDRESS

ASIAN_CURSOROVERF - Format for Carriage Return

This routine sets up the proper fill for a carriage return on the target unit.

Inputs:
R2 ADDRESS OF THE UNIT STATE VECTOR
R3 TTY$C_CR
R5 UCB ADDRESS
Outputs:
R2 ADDRESS OF THE UNIT STATE VECTOR
R3 TTY$C_CR
R5 UCB ADDRESS
R0 #AS$C_GD_GETNXT (Set dispatch code)

TTY$A_ASIAN_MOVEREADATA - move character from type-ahead buffer to read buffer

This routine moves a character from the type-ahead buffer and starts the echo.

Non-immediate action control sequences are handled here.

Before returning a character for echo it is converted to its multiple echo string if appropriate. In this case, the character returned is the first of the multiple echo characters.

Inputs:
R5 UCB ADDRESS

Outputs:

R3 CHARACTER IF ANY (CC = EQL)
R5 UCB ADDRESS

ASIAN_FIND_BOL - Find the beginning of this line

Given a string this routine will find the offset to the character that will end up in the first character position of the bottom line of the screen

Inputs:

R2 ADDRESS OF THE UNIT STATE VECTOR
R4 ADDRESS OF THE READ BUFFER
R5 ADDRESS OF THE UCB

Implicit inputs:

TTY$L_RB_TXT
TTY$L_RB_LIN
TTY$L_RB_PRMLEN
TTY$L_RB_TXTOFF assumed non-zero
TTY$A_RB_PRM
TTY$W_RB_LINOFF
TTY$W_RB_LINREST assumed zero

Implicit outputs:

TTY$L_RB_LIN address of the first character in this line of data
TTY$L_RB_LINOFF offset from LIN to the end of the line
R3 is destroyed

TTY$A_ASIAN_RDVERIFY - Read with verification

Read verify allows programs that wish to do character validation to issue one IO rather than a QIO per character as was previously the case.

Inputs:

R2 Unit state vector
R3 input octet from the type-ahead buffer
R4 address of type-ahead buffer
R5 address of the UCB

Implicit inputs:

UCB$L_SVAPTE	The address of the read buffer.
TTY$L_RB_TXT	The address of the first character of the initial string
TTY$W_RB_TXTOFF	Offset to the last character in the initial string
TTY$W_RB_TXTSIZ	Length of the data buffer.
TTY$W_RB_LINOFF	Offset to the end of the field, initial offset.
TTY$L_RB_PIC	The address of the picture string.

Outputs:

None

ASIAN_JISCON - JIS conversion

Translate DEC kanji 1983 keisen code to DEC extended area keisen code for supporting DEC kanji 1978 terminals. Also some escape sequence is parsed to identify processing substring

being kana or kanji.

Inputs:
R2 destination address
R3 IRP address
R5 UCB address
R6 source address
R7 source length
Outputs:
R2 end of destination string + 1
R5 UCB ADDRESS
R0-R4 destroyed

ASIAN_UPPER - Translate a string to upper case

Given an input string it will take all of the lower case ASCII characters in it and change it to upper case (characters in escape sequences are not bothered).

Inputs:
R2 DESTINATION ADDRESS
R5 UCB ADDRESS
R6 SOURCE ADDRESS
R7 LENGTH
Outputs:
R2 END OF DESTINATION STRING +1
R5 UCB ADDRESS
R0 -R4 DESTROYED

ASIAN_FDTSENSEM - SENSE MODE

ASIAN_FDTSENSEC - SENSE CHARACTERISTICS

This routine passes the current characteristics for sensemode and the permanent characteristics for sensechar.

Returns a LONGWORD buffer

P1 buffer address (length is assumed to be 1 longword)

Inputs:
R3 I/O PACKET ADDRESS
R4 CURRENT PCB ADDRESS
R5 UCB ADDRESS
R6 CCB ADDRESS
R7 FUNCTION CODE
AP ARG LIST FROM QIO
Outputs:
Control is returned to TTDRIVER.
Status returns:

SS$_NORMAL SUCCESSFULL

SS$_ACCVIO BUFFER NOT ACCESSIBLE

ASIAN_CURSOROVERF - Format for Carriage Return

This routine sets up the proper fill for a carriage return on the target unit.

Inputs:
- R2 ADDRESS OF THE UNIT STATE VECTOR
- R3 TTY$C_CR
- R5 UCB ADDRESS

Outputs:
- R2 ADDRESS OF THE UNIT STATE VECTOR
- R3 TTY$C_CR
- R5 UCB ADDRESS

ASIAN_FDT_SETM - Set Mode

Modify terminal characteristics according to the user buffer. The function code is set for a fast case on type

Inputs:
- R3 I/O PACKET ADDRESS
- R4 PCB ADDRESS OF CURRENT PROCESS
- R5 UCB ADDRESS
- R6 CCB ADDRESS FOR ASSIGNED UNIT
- AP ADDRESS OF ARGUMENT LIST AT USER PARAMETERS

Outputs:
THE FUNCTION IS COMPLETED HERE BY "EXE$FINISHIO".

Implicit Outputs:
R3,R5 ARE PRESERVED.

ASIAN_FDT_SETC - Set Characteristic

This routine is the function decision routine for terminal set characteristics.

Inputs:
- R3 I/O PACKET ADDRESS
- R4 PCB ADDRESS OF CURRENT PROCESS
- R5 UCB ADDRESS
- R6 CCB ADDRESS FOR ASSIGNED UNIT
- AP ADDRESS OF ARGUMENT LIST AT USER PARAMETERS

Outputs:
The function is completed here by "exe$finishio" or by queuing it to for follow on processing by TTYSTRSTP.

ASIAN_START_READ - READ operation startup

Initialize UCB data structure for READ operation.

Inputs:
I/O PACKET FORMATTED AS DESCRIBED IN TTYFDT.
R3 I/O PACKET ADDRESS
R5 PHYSICAL UCB ADDRESS
Outputs:
R0 return status
R1 I s destroyed

ASIAN_DO_SETM - SETMODE operation

Set the Asian terminal driver specific characteristics.

Inputs:
I/O PACKET FORMATTED AS DESCRIBED IN TTYFDT.
R3 I/O PACKET ADDRESS
R5 PHYSICAL UCB ADDRESS
Outputs:
R0 return status

ASIAN_DO_SETC - SETCHAR operation

Set the Asian terminal driver specific characteristics.

Inputs:
I/O PACKET FORMATTED AS DESCRIBED IN TTYFDT.
R3 I/O PACKET ADDRESS
R5 PHYSICAL UCB ADDRESS
Outputs:
R0 return status

ASIAN_DELETE_ASC - Delete the ASC

Delete the UCB's ASC if the ASC is owned by the UCB.

Inputs:
R5 UCB address
Outputs:
R0 destroyed

ASIAN_SETUP_UCB - Reset UCB's Asian terminal driver fields

Reset UCB fields specific to ASDRIVER.

Inputs:
R5 UCB address
Outputs:
None

ASIAN_FONTFORK - Deliver the glyph request

Delivers glyph request.

Inputs:
None
Outputs:
None

ASIAN_PRELOAD_FORK - Deliver the preload request

Delivers glyph preload request.

Inputs:
None
Outputs:
None

ASIAN_DEL_CACHE_FORK - Deliver the Soft-ODL Del cache request

Delivers glyph preload request.

Inputs:
None
Outputs:
None

ASIAN_CRE_CONTROL - Fixup for cloned UCB

Fix up cloned UCB and the ASC.

Inputs:
R5 LOGICAL UCB ADDRESS
R6 PHYSICAL UCB ADDRESS
Outputs:
R0 destroyed

ASIAN_PRELOAD_CLEANUP - Clean up TQE and GCB

This routine deallocates TQE conditionally, and also deallocates GCB. The routine is called while holding DEVICE LOCK

Inputs:
R0 THE ADDRESS OF ASC
Outputs:
ALL THE REGISTERS ARE PRESERVED

ASIAN_CLONE_UCB - Fixup for cloned UCB

Fixup cloned UCB and the ASC.

Inputs:
R5 LOGICAL UCB ADDRESS
R6 PHYSICAL UCB ADDRESS
Outputs:
R0 destroyed

ASIAN_ABORT - Abort Asian Driver operation

Called from TTY$ABORT to make Asian specific abort operations

Inputs:
R5 UCB address
Outputs:
R0 destroyed