Format strings for iscanf()

The library defines these formats in addition to the ANSI C formats:

%b - Read in a data block.

%c - Read in a character.

%d - Read in an integer (any format).

%f - Read in a float or double (with l).

%F - Store data from device into the given file.

%s - Read in a string without white space.

%S - Read in an IEEE 488.2 quoted string.

%t - Read in a string until an END indicator.

Optional field width supported on %b,%c,%t,%s and %S.

Optional Flags:

w, l, z, Z - See iprintf().
* - Format, but don't save results.
, - Comma separated list for %d and %f. Integer array size after the ,
- Read field width or array size from parameter (int for %d, %f, %s, %t, %c, %S; long for %b).

Data types used in reference:

u_char ->	unsigned char	*u_short* ->	unsigned short
u_int ->	unsigned int	*u_long* ->	unsigned long

Summary of Constants

Termination reasons for use by iread():
- **I_TERM_CHR** - *Termination character read.*
- **I_TERM_END** - *END indicator arrived.*
- **I_TERM_MAXCNT** - *bufsize bytes read.*

Formatted I/O buffer choices for use by iflush() and isetbuf():
- **I_BUF_READ** - *Use read buffer.*
- **I_BUF_READ_SZ** - *Default read buffer size.*
- **I_BUF_WRITE** - *Use write buffer.*
- **I_BUF_WRITE_SZ** - *Default write buffer size.*

Trigger lines used by ixtrig() and VXI trigger routines:
- **I_TRIG_ALL** - *All trigger lines supported.*
- **I_TRIG_ECL0/_ECL1** - *ECL0 trigger line.*
- **I_TRIG_EXT0/_EXT1** - *External BNC or SMB triggers.*
- **I_TRIG_STD** - *Standard trigger mechanism.*
- **I_TRIG_TTL0/..._TTL7** - *TTL0 to TTL7 trigger lines.*

Interface session interrupt types for isetintr():
- **I_INTR_INTFACT** - *Interface becomes active.*
- **I_INTR_INTFDEACT** - *Interface becomes deactive.*
- **I_INTR_TRIG** - *A trigger occurred(secval is mask).*
- **I_INTR_GPIB_IFC** - *Interface clear occurred.*
- **I_INTR_SERIAL_MSL** - *A modem status line changed.*
- **I_INTR_SERIAL_BREAK** - *BREAK received.*
- **I_INTR_SERIAL_ERROR** - *Parity/Framing/Overflow error.*
- **I_INTR_SERIAL_DAV** - *Receive buffer no longer empty*
- **I_INTR_SERIAL_TEMT**- *Transmit buffer not empty.*
- **I_INTR_VXI_SYSRESET** - *SYSRESET occurred.*
- **I_INTR_VXI_VME** - *VME interrupt occurred.*
- **I_INTR_VXI_UNKSIG** - *Unknown VXI signal*
- **I_INTR_VXI_LLOCK** - *Lock/clear lock WS command.*

Device session interrupt types for isetintr():
- **I_INTR_SERIAL_DAV** - *Receive buffer no longer empty.*
- **I_INTR_VXI_SIGNAL** - *VXI signal arrived.*

Commander session interrupt types for isetintr():
- **I_INTR_DEVCLR** - *Commander sent device clear to this interface.*
- **I_INTR_GPIB_GET** - *GET message received.*
- **I_INTR_GPIB_PPOLLCONFIG** - *PPOLL config has changed.*
- **I_INTR_GPIB_REMLOC** - *Remote/Local message received.*
- **I_INTR_GPIB_TLAC** - *Device addressed to (un)talk/(un)listen.*
- **I_INTR_SERIAL_MCL** - *Specified modem control line changed.*
- **I_INTR_STB** - *Commander requesting status from this interface.*

Any session interrupt types for isetintr:
- **I_INTR_OFF** - *Turn off all interrupt conditions.*

Standard error handlers for use by ionerror():
- **I_ERROR_EXIT** - *Print diagnostics and exit.*
- **I_ERROR_NO_EXIT** - *Print diagnostics.*

Standard map_space values for imap() and imapinfo():
- **I_MAP_A16** - *VXI A16 address space (entire space).*
- **I_MAP_A24** - *VXI A24 address space (64k pages).*
- **I_MAP_A32** - *VXI A32 address space (64k pages).*
- **I_MAP_EXTEND** - *VXI devices A24/A32 memory.*
- **I_MAP_SHARED** - *VXI shared local memory.*
- **I_MAP_VXIDEV** - *VXI device registers (64 bytes).*

Refer to the Reference Manual for additional interface-specific constants (**I_GPIB_BUS_***, **I_VXI_BUS_***, **I_SERIAL_***, etc.)

HP Standard Instrument Control Library

Quick Reference Guide for C Programmers

HEWLETT® PACKARD

E2090-90024 Printed in U.S.A. E0995

Mapping and Register-based I/O

char ***imap** (INST *id*,int *map_space*,u_int *pagestart*,u_int *pagecnt*,char ***suggested*);
Maps in a section of an interface's memory. See I_MAP_*.

int **imapinfo** (INST *id*,int *map_space*,int ***numwindows**,int ***winsize*);
Returns the number and size of the hardware map windows available in the given map_space address space.

int **iunmap** (INST *id*,char ***addr**,int *map_space*,u_int *pagestart*,u_int *pagecnt*);
Unmaps a section of an interface's memory.

void **ibpoke** (u_char ***addr**,u_char *val*);
void **iwpoke** (u_short ***addr**,u_short *val*);
void **ilpoke** (u_long ***addr**,u_long *val*);
u_char **ibpeek** (u_char ***addr**);
u_short **iwpeek** (u_short ***addr**);
u_long **ilpeek** (u_long ***addr**);
Writes val to or reads from mapped addr, with byte swapping.

Error Codes

Normal Completion:
I_ERR_NOERROR - *Successful completion.*
Invalid Parameters:
I_ERR_BADFMT - *Invalid format for iprintf/iscanf/ipromtf.*
I_ERR_BADID - *The specified INST id is invalid.*
I_ERR_BADMAP - *Invalid map request.*
I_ERR_PARAM - *Invalid parameter.*
iopen errors:
I_ERR_BADADDR - *Invalid address.*
I_ERR_INVLADDR - *Invalid address.*
I_ERR_SYMNAME - *Invalid symbolic name given.*
I_ERR_SYNTAX - *Syntax error occurred parsing address.*
Unsupported or unallowed operations:
I_ERR_NOPERM - *Permission denied.*
I_ERR_NOTIMPL - *Not supported on this implementation.*
I_ERR_NOTSUPP - *Operation not supported.*
Unavailable errors:
I_ERR_NOCMDR - *Commander not active or available.*
I_ERR_NOCONN - *No connection to remote.*
I_ERR_NODEV - *Device is not active.*
I_ERR_NOINTF - *Interface is not active.*
Data I/O errors:
I_ERR_DATA - *Data integrity violation.*
I_ERR_IO - *Generic I/O error.*
I_ERR_TIMEOUT - *Timeout occurred.*
Locking errors:
I_ERR_LOCKED - *Resource already locked by another process.*
I_ERR_NOLOCK - *An iunlock() on device that wasn't locked.*
O.S. and Resource errors:
I_ERR_BADCONFIG - *Invalid configuration identified.*
I_ERR_BUSY - *Interface not available for use by the library.*
I_ERR_NESTEDIO - *Task attempted function call before previous call completed (WIN16).*
I_ERR_NORSRC - *Out of system resources.*
I_ERR_OS - *Generic O.S. error.*
I_ERR_OVERFLOW - *Arithmetic overflow.*
Miscellaneous:
I_ERR_ABORTED - *Function call aborted by iabort() or external means.*

Error Handling

NOTE: For WIN16 programs on Microsoft Windows platforms, handler functions used with **ionerror()** must be exported and declared as *_far _pascal.*

void **icauseerr**(INST *id*,int *errcode*,int *flag*);
Causes simulated error to occur and optionally force handler call.

int **igeterrno**();
Returns the last error that occurred (unpredictable result if last library call succeeded). Per thread on multi-threaded O.S.

int **igetonerror**(void (***proc*)(INST *id*,int *error*));
Returns a pointer to the current error handler.

char ***igeterrstr**(int *errcode*);
Returns error string for given error number.

int **ionerror**(void (**proc*)(INST *id*,int *error*));
Installs the given procedure as an error handler. This procedure gets called when any library error occurs. This can be a user-defined procedure or an I_ERROR_* procedure.

Format strings for iprintf()

The library defines these formats in addition to the ANSI C formats:

%b/%B - Send an IEEE 488.2 arbitrary data block.

%C - Send a character with an END indicator.

%c - Send a character.

%d - Send an integer.

%f - Send a float.

%F - Read data from the given file and send to the device.

%s - Send a string.

%S - Send an IEEE 488.2 quoted string.

%t - Toggle END indicator on \n.

%% - Send ASCII "%" character.

Optional field width supported on %c,%d,%f,%s and %S.

Optional precision supported on %d,%f,%s and %S.

Optional Flags:

@1 - Use NR1 format (eg. 357), for %d or %f.

@2 - Use NR2 format (eg. 357.5), for %d or %f.

@3 - Use NR3 format (eg. 3.575E2), for %d or %f.

@B - Use IEEE 488.2 BINARY (eg. #B101110), for %d or %f.

@H - Use IEEE 488.2 HEX (eg. #H2F34), for %d or %f.

@Q - Use IEEE 488.2 OCTAL (eg. #Q2777), for %d or %f.

\- - Left justify the result. Use with field width.

\+ - Always output the sign character. Use with @1, @2, @3.

0 - Pad field with leading zeros for numeric conversion. Ignored if - is used, and also if precision is specified with @1, @H, @Q or @B.

\- Prefix with a blank if signed and positive. Use with @1, @2, @3.

, - Comma separated list for %d and %f. Integer array size after the ,.

w - Word width for %b.

l - Long word width for %b. Long for %d. Double for %f.

L - Long double for %f.

h - Short integer for %d.

z - Float width for %b.

Z - Double float width for %b.

* - Read field width

Session Control

INST **iopen** (char **addr*);
Opens a device/interface/commander session. See the Reference Manual appendix for a session support summary.

int **iclose** (INST *id*);
Closes a device/interface/commander session.

Formatted, Buffered I/O

NOTE: For WIN16 programs on Microsoft Windows platforms, if not compiling with compact, large or huge memory models, make sure that formatted I/O pointer/address parameters are passed as *_far*.

int **ifread** (INST *id*,char **buf*,u_long *bufsize*,int **reason*,u_long **actualcnt*);
Reads non-formatted, buffered data from device/interface. See I_TERM_* for reason parameter values.

int **ifwrite** (INST *id*,char **buf*,u_long *datalen*,int *end*,u_long **actualcnt*);
Writes non-formatted, buffered data to device/interface.

int **iflush** (INST *id*,int *mask*);
Flushes the formatted I/O read or write buffers. See I_BUF_*.

int **iprintf** (INST *id*,char **format*,... *arg1*,*arg2*,...);
Writes formatted, buffered data to device.

int **ipromptf** (INST *id*,char **writefmt*,char **readfmt*,... *arg1*,*arg2*,...);
Atomically writes/reads data to/from device.

int **iscanf** (INST *id*,char **format*,... *arg1*,*arg2*,...);
Reads formatted, buffered data from device.

int **isetbuf** (INST *id*,int *mask*,inst *size*);
Sets size of formatted I/O read and write buffer. See I_BUF_*.

int **isetubuf** (INST *id*,int *mask*,int *size*,char **buf*);
Supplies the buffer(s) used for formatted I/O.

The following are identical to the above counterparts except that the *va_list* parameters list provides the parameters:

int **ivprintf** (INST *id*,char **format*,*va_list ap*);

int **ivpromptf** (INST *id*,char **writefmt*,char **readfmt*,*va_list ap*);

int **ivscanf** (INST *id*,char **format*,*va_list ap*);

The following are identical to the above counterparts except that the data is written to or read from the buffer *buf* instead of the device:

int **isprintf** (char **buf*,char **format*,... *arg1*,*arg2*,...);

int **isscanf** (char **buf*,char **format*,... *arg1*,*arg2*,...);

int **isvprintf** (char **buf*,char **format*,*va_list ap*);

int **isvscanf** (char **buf*,char **format*,*va_list ap*);

Application Cleanup

Sub **_siclcleanup** ()
Cleans up SICL resources used for an application. This routine should be called before a WIN16 Windows application terminates.

Non-Formatted, Non-Buffered I/O

NOTE: Do not intermix these non-buffered I/O calls with the formatted buffered I/O calls in the same device, interface, or commander session.

int **igettermchr** (INST *id*, int **tchr*);
Gets current termination character.

int **iread** (INST *id*,char **buf*,u_long *bufsize*,int **reason*,u_long **actualcnt*);
Reads non-formatted, non-buffered data from device/interface. See I_TERM_* for reason parameter values.

int **itermchr** (INST *id*, int *tchr*);
Sets iread() termination character, -1 means no termination character.

int **iwrite** (INST *id*,char **buf*,u_long *datalen*,int *endi*,u_long **actualcnt*);
Writes non-formatted, non-buffered data to device/interface.

Interrupt and SRQ Control

NOTE: For WIN16 programs on Microsoft Windows platforms, handler functions used with **ionsrq()** and **ionintr()** must be exported and declared as *_far _pascal*.

int **igetonsrq** (INST *id*,void (***proc*)(INST *id*));
Returns a pointer to the current SRQ handler for the device/interface.

int **igetonintr** (INST *id*,void (***proc*)(INST *id*,long *reason*,long *secval*));
Returns current interrupt handler for this session.

int **ionintr** (INST *id*,void (**proc*)(INST *id*,long *reason*,long *secval*));
Installs an interrupt handler for this session, doesn't enable any interrupt events (see isetintr()).

int **ionsrq** (INST *id*,void(**proc*)(INST *id*));
Installs an SRQ handler for the given interface/device.

int **isetintr** (INST *id*,int *intnum*,long *secval*);
Enables interrupt events for this session, doesn't install an interrupt handler procedure (see ionintr()) See I_INTR_*.

int **iintroff** (); int **iintron**();
Globally disables/enables all interrupt & SRQ handlers (per process).

int **iwaithdlr** (long *timeout*);
Waits for an event (interrupt or SRQ handler executing) on any session in current process. Ignores state of iintroff()/iintron().

Locking and Timeouts

int **ilock** (INST *id*);
Locks the device/interface to this session.

int **igetlockwait** (INST *id*,int **flag*);
Returns value set by isetlockwait().

int **igettimeout** (INST *id*,long **tval*);
Returns current timeout value.

int **isetlockwait** (INST *id*,int *flag*);
Sets action when a function is locked out by another process. Non-zero flag (default), function will block, else return an error.

int **itimeout** (INST *id*,long *tval*);
Sets the timeout (in milliseconds). Default is 0 (infinite).

int **iunlock** (INST *id*);
Unlocks the device/interface.

Device/Interface Control

int **iabort** (INST *id*);
Aborts current function call in another thread. Valid on multi-threaded operating systems; otherwise no action.

int **iclear** (INST *id*);
Performs a device or interface clear.

int **ihint** (INST *id*,int *hint*);
Sets the preferred data transfer mechanism. See I_HINT_*.

int **ilocal** (INST *id*);
Puts device into local mode (turns on front panel).

int **ireadstb** (INST *id*,u_char **stb*);
Reads status byte from the device.

int **iremote** (INST *id*);
Puts device into remote mode (turns off front panel).

int **isetstb** (INST *id*,u_char *stb*);
Sets status byte for this controller (commander sessions only).

int **itrigger** (INST *id*);
Sends a trigger to the given device.

int **ixtrig** (INST *id*,u_long *which*);
Performs an extended trigger. See I_TRIG_*.

Miscellaneous Functions

int **ibeswap**(char **addr*,u_long *length*,int *datasize*);
Converts data between big-endian and platform-native byte order.

int **igetaddr** (INST *id*,char ***addr*);
Returns pointer to address string passed to iopen for this session.

int **igetdata** (INST *id*,void ***data*);
Returns user-defined data structure stored in the INST id.

int **igetdevaddr** (INST *id*,int **prim*,int **sec*);
Returns the device address of this device

int **igetintftype** (INST *id*,int **pdata*);
Returns interface type. See I_INTF_* in the Reference Manual.

int **igetintfsess** (INST *id*);
Returns an interface session for given device/commander session.

int **igetlu** (INST *id*,int **lu*);
Returns the logical unit for this INST id.

int **igetlulist** (int ***lulist*);
Returns a list of all valid logical unit numbers (interfaces) on this system. The list is terminated with a -1.

int **igetluinfo**(int *lu*,struct *lu_info *luinfo*);
Returns pointer to information for given logical unit.

int **igetsesstype** (INST *id*,int **pdata*);
Returns session type. See I_SESS_* in the Reference Manual.

int **ileswap**(char **addr*,u_long *length*,int *datasize*);
Converts data between little-endian and platform-native byte order.

int **isetdata** (INST *id*,void **data*);
Stores a pointer to a user-defined data structure into INST id.

int **iswap**(char **addr*,u_long *length*,int *datasize*);
Swaps byte order of data.

int **iversion** (int **siclversion*,int **implversion*);
Returns version level of the library, and software revision of the library that the application was linked against. See also I_SICL_REVISION.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HP-IB Specific Functions

NOTE: Using these interface-specific functions may reduce the portability of your program.

int **igpibatnctl** (INST *id*,int *atnval*);
Controls the state of the ATN line. Does not affect iwrite, iread, iprintf, iscanf, ipromptf, or igpibsendcmd operation.

int **igpibbusaddr** (INST *id*,int *busaddr*);
Changes the bus address of the HP-IB interface.

int **igpibbusstatus** (INST *id*,int *request*,int **result*);
Returns the status of the HP-IB interface. See I_GPIB_BUS_*.

int **igpibgett1delay** (INST *id*, int *delay*);
Retrieves the T1 delay setting on the HP-IB interface.

int **igpiblo** (INST *id*);
Performs HP-IB local-lockout function (see iremote()).

int **igpibpassctl** (INST *id*,int *busaddr*);
Passes active controller role to specified bus address.

int **igpibppoll** (INST *id*,u_int **result*);
Performs parallel poll and return (8-bit) result.

int **igpibppollconfig** (INST *id*,int **cval*);
Configures device/interface to respond to ppoll. Disable if val is -1.

int **igpibppollresp** (INST *id*,int **sval*);
Sets ppoll response for the HP-IB interface.

int **igpibrenctl** (INST *id*,int *ren*);
Controls the state of the REN line.

int **igpibsendcmd** (INST *id*,char **buf*,int *length*);
Sends data bytes over the HP-IB interface with ATN set.

int **igpibsett1delay** (INST *id*, int *delay*);
Sets the t1 delay on the HP-IB interface.

GPIO Specific Functions

NOTE: Using these interface-specific functions may reduce theportability of your program.

int **igpioctrl** (INST *id*,int *request*,u_long *setting*);
Controls various lines and modes of the GPIO interface.

int **igpiogetwidth** (INST *id*,int **width*);
Returns the current data width (in bits) of a GPIO interface.

int **igpiosetwidth** (INST *id*,int *width*);
Sets the data width (in bits) of a GPIO interface.

int **igpiostat** (INST *id*,int *request*,u_long **result*);
Determines the current state of various GPIO modes and lines.

Serial Specific Functions

NOTE: Using these interface-specific functions may reduce the portability of your program.

int **iserialbreak** (INST *id*);
Sends a BREAK.

int **iserialctrl** (INST *id*,int *request*,u_long *setting*);
Sets characteristics of serial port. See I_SERIAL_*.

int **iserialmclctrl** (INST *id*,int *sline*,int *state*);
Controls the modem control lines.

int **iserialmclstat** (INST *id*,int *line*,int **state*);
Gets current state of modem control lines.

int **iserialstat** (INST *id*,int *request*,u_long **result*);
Monitors characteristics of serial port. See I_SERIAL_*.

LAN Specific Functions

int **igetgatewaytype** (INST *id*, int **gwtype*);
Indicates whether the session is via a LAN gateway.

int **ilangettimeout** (INST *id*, long **tval*);
Returns LAN timeout value.

int **ilantimeout** (INST *id*,long *tval*);
Sets LAN timeout value.

VXI Specific Functions

NOTE: These interface-specific functions may reduce program portability.

int **ibblockcopy** (INST *id*, u_char **src*,u_char **dest*,u_long *cnt*);
Copies data byte (8 bits) from one device's memory to another.

int **ilblockcopy** (INST *id*, u_char **src*,u_char **dest*,u_long *cnt*, int *swap*);
Copies long word (32 bits) from one device's memory to another.

int **iwblockcopy** (INST *id*, u_char **src*,u_char **dest*,u_long cnt, int *swap*);
Copies data word (16 bits) from one device's memory to another.

int **ibpopfifo** (INST *id*,u_char **fifo*,u_char **dest*,u_long *cnt*);
Reads data byte (8 bits) from a FIFO into memory.

int **ilpopfifo** (INST *id*,u_char **fifo*,u_char **dest*,u_long *cnt*, int *swap*);
Reads long word (32 bits) from a FIFO into memory.

int **iwpopfifo** (INST *id*,u_char **fifo*,u_char **dest*,u_long *cnt*, int *swap*);
Reads data word (16 bits) from a FIFO into memory.

int **ibpushfifo** (INST *id*, u_char **src*,u_char **fifo*,u_long *cnt*);
Copies data byte (8 bits) from device's memory to other device FIFO.

int **ilpushfifo** (INST *id*, u_char **src*,u_char **fifo*,u_long *cnt*, int *swap*);
Copies long word (32 bit) from device's memory to other device FIFO.

int **iwpushfifo** (INST *id*, u_char **src*,u_char **fifo*,u_long *cnt*, int *swap*);
Copies word (16 bit) from device's memory to other device FIFO.

int **ivxibusstatus** (INST *id*,int *request*,u_long **result*);
Returns the status of the VXI interface. See I_VXI_BUS_*.

int **ivxigettrigroute** (INST *id*,u_long *which*,u_long **route*);
Returns in 'route' the trigger lines that the 'which' trigger line is routed to. See I_TRIG_*.

int **ivxirminfo** (INST *id*,int *laddr*, struct vxiinfo **info*);
Returns resource manager information about any VXI device.

int **ivxiservants** (INST *id*,int *maxnum*,int **list*);
Returns list of all VXI servants in system.

int **ivxitrigoff** (INST *id*,u_long *which*);
Turns off the specified trigger lines. See I_TRIG_*.

int **ivxitrigon** (INST *id*,u_long *which*);
Turns on the specified trigger lines. See I_TRIG_*.

int **ivxitrigroute** (INST *id*,u_long *in_which*,u_long *out_which*);
Routes trigger line specified by inwhich (one only) to the trigger lines specified in outwhich (ORed mask). See I_TRIG_*.

int **ivxiwaitnormop** (INST *id*);
Suspends current program until resource manager completes.

int **ivxiws** (INST *id*,u_short *wscmd*,u_short **wsresp*,u_short **rpe*);
Sends a VXI word-serial command and grabs the response.